Lambert Spaanenburg
Hendrik Spaanenburg

Cloud Connectivity and Embedded Sensory Systems

Springer

Cloud Connectivity and Embedded Sensory Systems

Lambert Spaanenburg · Hendrik Spaanenburg

Cloud Connectivity and Embedded Sensory Systems

Springer

Lambert Spaanenburg
Department of Electrical & Information Technology
Lund University
S-221 00, Lund, Sweden
lambert@eit.lth.se

Hendrik Spaanenburg
Heterogeneous Computing, LLC
Durham, NH 03824, USA
henk@heterogeneouscomputing.com

ISBN 978-1-4419-7544-7 e-ISBN 978-1-4419-7545-4
DOI 10.1007/978-1-4419-7545-4
Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2010937673

To Henk Sr.

Preface

Computers, great or small, can be found in many digital data producing and processing system nodes. Their performance is steadily increasing, and some have already become miniature supercomputing nodes. More and more of them are connected to the Internet, thereby physically forming very large networks.

The classical networks contain computers that are cabled together. History shows two developments here. On the one hand, we see mainframes becoming personal computers and then notebooks, getting more personal all the time. On the other hand, we see mainframes turning into supercomputers and then server farms into the cloud, getting more into a shared service all the time.

There are many stories about the origin of "cloud computing." The most popular one is that people were diagramming an information-processing architecture on a whiteboard. In order to sketch out the magical process that was finally to do everything, some wrinkles were drawn in the sky and they called it a cloud. This is not a rigid definition and consequently the name "cloud" is used for a lot of things.

A cloud provides shade against the sun. To be in the shade, you have to move regularly. If you do not want that, then take an umbrella. It does not give shade to a lot of people, but it will help you. More or less the same is true for the rain cloud. You can walk around with a pot to water your plants, but it is just a bit of water at a time. Having a cloud makes it much easier.

In the industrial age, we have become accustomed to mass fabrication, mass transport, and mass delivery. Already the Romans had structures to gather and transport water, in order to make it massively available in the city. Otherwise you need to live near the well, and when the well goes dry you have to move. Later the same principle was applied for electricity, again with the main benefit that massive use becomes possible without the need to live next to the generator. In the information age, we want essentially the same, but this time for information and specifically sensor-obtained information.

The Internet has become the aqueduct of modern times, bringing information from the large server farms to each household. At the origin, it is not really a cloud, but let us not be too picky. The point we want to make here is to take the cloud as part of a shared resource in a cyclic bio-like system. This would bring us closer to

the concept of "sensory networks." In talking about such networks, we can easily draw up the network nodes, but to indicate the overall processing function some wrinkles in the sky have to be drawn. Just like a cloud!

The point is that a function can be part of a node, can use some nodes, and can also be handled at a server farm. The function is executed on some resources and we do not want to specify immediately where these resources are located and how much of them are needed. It should be moving dependent on the need, and we do not want to know that and where it moved. In other words, *the network becomes the cloud*!

The nature of the network has also changed, though the backbone is still cabled. However, the introduction of wireless technology has caused the computer to become mobile. And from there, we see other computerized mobile equipment coming into fashion, notable the mobile telephone. Yearly 1.5 billion camera phones are sold!

The rich set of user interactions is currently being augmented with inter-device interactions, making the camera phone a flexible node in a community network. Being based on preferential attachments, its function and construction changes with the whim of the participants rather than on pre-composed orchestration.

But the story does not stop with phones. All kind of products are turning to wireless, if not only to get rid of the cables. The doorbell connects wirelessly to the phone and allows you to answer the doorbell, even when you are not at home. The car seat for the children warns for dehydration of the baby. And in all these new devices, we have computing platforms ranging from simple controllers to graphic accelerators.

Similar networking trends such as in wireless consumer communities are visible in smart energy grids, homeland security systems, and environmental networks. These sensory networks are meant to create awareness in space and time. They may be measuring the presence of an object or a condition, characterizing an object stream or a situational pattern, or even detect abnormalities that are to occur. Different types of sensors can be used for similar purposes, allowing for enhanced system reliability and safety, but they tend to be preferentially attached to the network.

For the network functionality, one can distinguish between cloud, and even swarm and flock computing, the difference being the distribution of the decision control. Just plugging sensors into the cloud is one thing, but the question remains how the extended network itself can be made more intelligent and therefore less dependent on the quality of the individual sensors. We call this *embedded sensory systems*.

Sensory systems are populated by embedded functions that safely combine migrating software on virtualized hardware. It brings the connectivity mechanisms from the global cloud into local clouds of sensor-activated resources. This emphasizes safety for society through ownership of data by a rigorous awareness for the users and their intentions. Such systems will leave a clear mark on society in the coming years.

The Book

It provides conceptually a one-stop entry into the world of intelligent sensory networks, as we envision them to exist in relationship to the computing cloud(s). This is the "ambient intelligence" equivalent of what "cloud computing" means to business intelligence, and therefore the new generation of large-scale sensory networking systems. Cloud computing usually refers to the accessibility from the network of a well-orchestrated data warehouse. Clearly, cloud computing concepts also apply to networks of embedded sensors. The book discusses concepts, problems, and initial solutions for the latter, while also providing illustrations from a wide field of practice. Becoming aware of the basic problems and solutions helps the readers to prepare for innovation based on distributed intelligence in their own area of technological expertise.

This book provides new theory on the design of wireless sensory networks. It provides a step-by-step discourse on building case studies to capture the requirements, taking into account practical limitations to creating ambient intelligence. Notably, it pays attention to topological and communication constraints, while adding intelligence, security, and safety to the overall desired system functionality. Such aspects will be highlighted by reviewing applications in a variety of typical sensory/cloud network scenarios.

The reader will not only achieve a better understanding of such concepts as sensory clouds but will also be guided by examples on how to design such networks, taking the typical characteristics of diverse application areas into account.

In general, knowledge on new technologies is communicated in new conferences and journals, and in addition needs to be condensed regularly in books. This book will give the readers an overview of the current directions in theory and practice and will therefore allow them to stand on the shoulder of "giants"/peers. Our emphasis in writing this book has been on developing the new concepts, especially the disruptive technology aspects, for use in the establishment of new businesses.

The book is divided into four parts. The first part explores the opportunities that will come about from the further developments of cloud computing into the outlying domains of sensory systems. It also includes a description of a new software model for large-scale networks. The second part reviews system design aspects needed for the understanding of the subsequently illustrated potential sensory network applications. The emphasis is on system applications that would benefit from the connection into the Cloud. The third part addresses concerns for security and safety that will need to be supported in order to result in successful cloud-centric sensory networks. Lastly, we focus on details of a futuristic sensory/display-instrumented home environment completely in touch with the Cloud.

This book aims to be one of the first to bring the sensor-networking-into-the-cloud state-of-the-art together with a perspective on the near future to stimulate further research. It provides a selection of material from courses at Lund University (Sweden) and the Daniel Webster College in Nashua (New Hampshire, USA). It brings a vision that has gradually evolved from the experience with

a number of students. Especially our gratitude goes to Wen Hai Fang, Peter Sundström, Lars Lundqvist, Erik Ljung, Erik Simmons, Barbara Olmedo, Sajjad Haider, Joe Evans, Dalong Zhang, Miao Chen, Mona Akbarniai Tehrani, Shankar Gautam, Olle Jakobsson, Lijing Zhang, Wang Chao, Shkelqim Lahi, Nabil Abbas Hasoun, Dongdong Chen, Bintian Zhou, Cheng Wang, Deepak Gayanana, Markus Ringhofer, Vince Mohanna, Johan Ranefors, Suleyman Malki, Jos Nijhuis, and Ravi Narayanan.

Lund Sweden — Lambert Spaanenburg
Durham New Hampshire, USA — Hendrik Spaanenburg

Contents

List of Figures

List of Tables

Part I
Clouds into the Sensors' Edge

"The Economist" called it the "Death of Distance": Time for communication is dwindling under the upcoming network technology, making the notion of a global village as posed by the Canadian philosopher/artist Marshall McLuhan a virtual reality. The consequence is the "Martini" feeling: Everything is happening at all places, at all times, and all around. The Internet has realized this for the PC era, causing the single user to cross physical limits in connecting to others, but all largely with the same functionality.

The explosive increase in computer count allows for specialization. Instead of the single computer with many virtual processes, we see many different computers interfacing to the real world at different locations, but acting together as a single process. The combination is more than the sum of the parts, as the autonomous intelligence adds a layer of awareness that, after the development of mainframes and personal computing, leads to the *third paradigm of computing*.

The first part of this book explores the opportunities that will come about from the further developments of cloud computing into the outlying domains of sensory systems. It also includes a description of a new software model for large-scale networks.

Highlights

Sensors-in-the-Cloud

The notion of cloud computing will be extended downward into the domain of intelligence sensory networks. Cloud computing will introduce new capabilities and opportunities for sensor intelligence gathering application and management, resulting in more efficient integrated systems. The processional relationship between sensors/displays and the Cloud will be expanded.

Software Migration

From the sensors point of view, it will be not important to know where the processing application software is run, in the cloud or even somewhere in the sensor network. This location process can be applied at start-up, as well as on regular

intervals (or as a result of a significant processing requirement event) during the data collection process.

Software Is Virtual Hardware!

The line of demarcation between hardware and software will disappear as a result of virtualization, of the increased use of multi-core/tile processors, as well as of the introduction of reconfigurable and reprogrammable FPGAs in the network nodes. Multiplicity of alternative resources will make system execution optimization feasible and very attractive.

System Optimization

Heterogeneity and centralization in the cloud will facilitate the use of optimized processing choices, as well as the use of up-to-date technology. Technology upgrades and updates will be available in the greater cloud domains. The cost of processing resource ownership at the sensor network level can be dramatically reduced.

Chapter 1
Introduction to Cloud Connectivity

In this chapter we introduce the world of networked computing nodes, each embedded within a particular environment and together building a sensory world. We will coarsely go through a number of principles and concepts to set the stage for the discussion on cloud-centric processing that will be further detailed and illustrated in the chapters to come.

Computer technology has developed into different directions. In one direction, computers are used to run ever larger data sets on ever more complex software systems. Where the monolithic supercomputer has bumped into technological and economic walls, the move has been from distributed servers to farms of servers. Simultaneously, the in-house service developed by consulting experts has moved to the outsourced use of universal services by experts, made available on the virtual market.

The ruling word here is "Cloud." With a growing number of server farms being available all over the world and the software being run on an arbitrary one (just as you don't know where your Google search request is serviced), one may say that "it is all out there in the clouds." Currently, the Cloud is the business model of choice and everyday there are workshops on this subject somewhere in the world. In this book, we will talk about the Cloud, but in a similar, yet different way.

The other direction in which computer technology has developed has to do with actions rather than data. Though there has always been a prosperous 86xx market for control applications, characterized by real-time rather than by high-performance software, the real breakthrough occurred when the platforms became strong enough so that software could execute in real time and with better accuracy the real world problems that before were deemed to be the sole domain of analog or electromechanical devices.

Initially, we find in the applications the classical controller with a lot of I/O being supplemented with computational power. This milestone was called embedding, and actually we have seen many other names surface. But this was not the end. The race went on. Already in the car we have seen the embedded processors being networked into a single overall functionality, called auto. In 20 years, computer technology has entered, for instance, the automotive domain, where despite the lower cost of the technology, it still amounts to more than 30% of the production costs of a car.

L. Spaanenburg, H. Spaanenburg, *Cloud Connectivity and Embedded Sensory Systems*, DOI 10.1007/978-1-4419-7545-4_1,

Next to improved functionality by one or more embedded processors, we envision the impact of large collectives operating with a single mind like a legion or with the single purpose like a swarm. The added difficulty in those cases is that the system components are not necessarily fixed to a moment in time or to a place in space. There is not even a single synchronized sequence of events, as each of the parts is primarily acting in the local environment.

Behind these developments, hidden in the background, is the need for ever more complicated algorithms. For the same reasons that brought server farms into existence, there is also no reason why every embedded node should not be capable to handle the most complex algorithm. For such situations, cloud support will be meaningful, but we will argue that this may take different shapes than a mere implementation of a server farm.

1.1 Embedded Systems

The first microelectronic computing engine, the INTEL4004, is a typical component developed for an embedded system. In this case the target was to replace a mechanical chronometer. Though not the immediate success hoped for, it did set a mark in the history of the digital watch market. INTEL opened the digital control market but left it later to pursue the higher and more lucrative goal of scientific computing.

Microcontrollers, as soon became the name for the INTEL4004 and its derivatives, have always been the bigger market. The 4-bit version has been the digital components' market leader for years, drawing a lot of competitors around the base model to optimize the part for all kinds of applications. It only started to lose market share when it became more efficient to produce 8-bit versions. The market did not really ask for this next generation. For example, an electric shaver will not be more accurate when controlled by an 8-bit rather than a 4-bit machine!

The microcontroller is a computer with nondigital interfaces. The peripheral variety created a diversity that supports many players. The use for real-time control forced the need to have fixed and dependable timing for the software execution. This has eventually developed into parts that provide a single clock cycle execution for almost all instructions. Having such a guaranteed execution time allowed for virtual peripherals, I/O that can be digitally configured into a range of previously dedicated nondigital interfaces.

Simultaneously, the amount of on-chip memory was increased, giving way for larger programs. Such programs encompass basic signal processing in addition to the interface circuitry, raising the quality of the sensory data extraction. Already around 1995, it was possible to code in 100 lines a lattice filter, with a performance that 5 years before required an artful ASIC. Steadily pushing the boundaries of digital sensing and actuating, it has become possible to complement nondigital functions in the digital domain to achieve a functionality that outperforms the cleverest nondigital realization. Today, we find products like cars answering environmental demands by high-performance real-time signal processing in a way that cannot be

achieved by just optimizing the mechanics of the engine. And in digital cameras, we already encounter super-computing components to provide the best picture quality ever!

Finally, the increasing number of applications where the installed computing potential created products that otherwise would not be possible – at least not with the same efficiency and at similar costs – have kindled the fire and a new technological area arose, the embedded systems. Its applications are everywhere and in amazing quantities around us, giving it names like pervasive, ubiquitous, and ambient, but the common thread is the embedded technology.

1.1.1 Characteristics of Embedded Systems

The main characteristic of embedded systems is that they operate in conjunction with their environment, the so-called embedding system. This makes them often specific for defined use cases, even for high-volume consumer applications. By (re-) configuration they can still cover entire application domains. As such domains can be quite extensive, involving billions of mobile telephones or television sets, the economy of scales is there to provide the necessary specificity (Fig. 1.1).

As embedded systems are contained within their embedding environment, they are hard to access in real time. Unlike scientific systems where a keyboard and a monitor are present to give the developers direct feedback on their work, an

Fig. 1.1 Embedded systems will be everywhere, but mostly unnoticeable or invisible, to enhance the functionality of the devices and equipment shown

embedded system needs to be developed off-line based on a model of the operational situation and provided with enough test extensions that the in-line usage can still be adequately diagnosed [1].

Embedded systems can be characterized by the following properties [2]:

1. They are an information-processing subsystem of their embedding systems.
2. They provide specific and highly configurable information-processing services to their embedding systems.
3. They are reactive, i.e., they interact with their physical environment often in a continuous mode at a speed imposed by the environment.
4. They provide usually a complex functionality to their embedding system with a combination of hardware and software specialized to meet a wide variety of nonfunctional constraints.
5. They are mostly not visible or directly accessible by the users of the embedding system although they are often used to increase the user-friendliness and awareness of an embedding system.

Embedded systems not only interact with their environment, but are nowadays also equipped with communication interfaces. For a long time, this digital communication was domain- or even supplier-specific, but the standards developed in the classical computing world are rapidly getting more popular. This allows embedded systems to be combined into networks, both to the external world and within the embedding world (Fig. 1.2). Consequently, an embedded system can serve

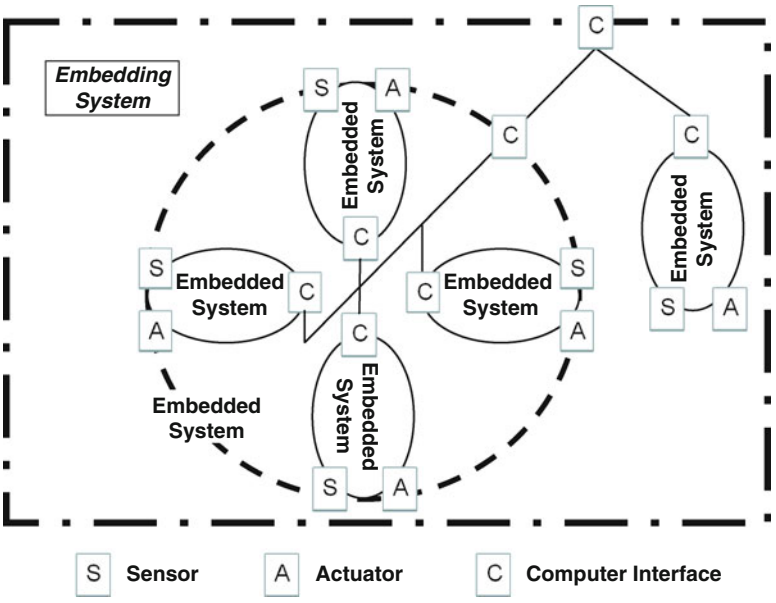

Fig. 1.2 Embedded systems in their environment

as embedding for other systems, creating an entire embedded infrastructure for complex products as cars or entire production lines.

1.1.2 What Makes Embedded Systems Special

It is the embedding system that regulates the operating speed of the embedded parts. Such parts are required to react immediately and in-time on stimuli. This is in sharp contrast to interactive systems where the result can be supplied at any time. For instance, user interaction places the users at the receiving end of the communication, biding their time to get an answer. A reactive system demands the embedding system to be in full control. This results in the need for special capabilities for the hardware and software architecture of the platform to be used.

Where the embedding system is in control, the designer of the embedded system is confronted with a major problem. He/she has to develop for reactions on largely unknown stimuli. Therefore, a model is required for the embedding, but this model is not necessarily complete. As a consequence, the system must be either safe enough to handle unknown situations or adaptable to a changing environment. To create a safe embedded system takes much effort, involving lengthy simulations and above all many real-time tests.

As the system is embedded, there are usually restrictions on its size, cost, and energy consumption. Also the envisaged location plays a role, as it is usually meant to function at a hard-to-reach place. A sensor to monitor environmental conditions is usually far away, and therefore switching the battery is not an option. This is also true for maintenance. In general, the embedded system has to function for years without any attention.

The embedding system is usually nondigital, and therefore the sampled stimuli may be highly distorted. Well known are the problems with electromagnetic interference (EMI), which cause undesired signals and thereby invoke undesired behavior. Any location with generators such as factories and cars is hard to handle. Still we want the parts to be low cost, as embedded systems are not visible to the user. Often being used in mobile or wearable appliances, low power is also a standard constraint for embedded systems.

Special design-related challenges come from the specialization and customizing of target platforms based on detailed application know-how. The challenge is how to maintain some degree of flexibility with the related wish to increase the reuse of hardware and software components. Reliability, robustness, and safety constraints derive from situations where restart is impossible and a certain degree of autonomous behavior should be possible. The use of many different disciplines and the heterogeneity of applied technologies are the last, but not the least, important factors in making the design of embedded systems special.

Add to all this the trend to link embedded systems into networks, creating complex infrastructures that should survive despite failing parts. Safety will therefore be a major concern, but such should be part of the overall

architecture, even in such situations where the parts have been independently designed and developed. More often than not, embedded infrastructures will go unattended[1] and any lack in security and safety can be catastrophic. We can discern three main categories: (a) encapsulated, (b) dichotomized, and (c) expanded architectures.

1.1.3 Encapsulated Architecture

The computing portion in the encapsulated architecture[2] is totally immerged in the physical reality. Lee [3] is basically addressing this issue when discussing cyber-physics. The reality is enhanced by adapting physical behavior to the desired one; in other words, nature is compensated for its frivolity. This architecture has stimulated futuristic thinking a lot, and underlies such technophile studies as by Ray Kurzweil [4] and Hans Moravec [5]. The popular example of an anticipated practical application is illustrated in the famous "7 of 9" from Starship Enterprise (the Voyager), whose brainpower is enhanced by implanted supercomputing. Nevertheless, the bio-related concept remains heavily debated [6] and issues in system reliability may make it impractical.

The state of the art contributes already in situations, where a model of desired behavior can be enforced on the (non-living) physical reality, such as the automotive power train [7]. From the overall needs of the car, it can be deduced what the power train needs to deliver. Unfortunately, this behavior cannot be built in conventional electromechanical technology at acceptable cost. The experienced technician can probably make an acceptable part, given enough time and permanent dedication. But the salary costs make this impossible to pursue, even if sufficient amounts of such technicians would exist to support mass production. The alternative is to add a super-computing part that will control reality such that it will perform as desired. It is clear that this can be done only by virtue of modern microelectronics and will only perform as it overrules the mechanical shortcomings.

The main characteristic of the encapsulated architecture is local autonomy. The additional computing is devoted to enhance the physical reality. For this purpose, it has a model of the reality and a set of targets to achieve through adaptation. All it does is match these two to the benefit of the product. That product may again be part of a larger product, like the power train which is a part of an automobile.

---

[1] Many sensor networks will go under the name "unattended sensor networks" or USN. Unless otherwise stated, we will assume this to be the usual case.

[2] Definition of Encapsulated (Merriam-Webster Online Dictionary): 1. surrounded by a gelatinous or membranous envelope <encapsulated water bacteria>, 2. condensed.

1.1.4 Dichotomized Architecture

An alternative is the dichotomized architecture[3] (Fig. 1.3). The system is only partially embedded; or rather the computing portion is, on one hand, attached to the physical portion and, on the other hand, to a conventional digital network. Typically, the local microcontroller is meant solely to provide some support for data logging, while a central computer receives the data from all the respective sources and provides incentives for the actuation. One may say that we merely have an elaborate digital/analog interface here, because all the intelligence lies within the central computer.

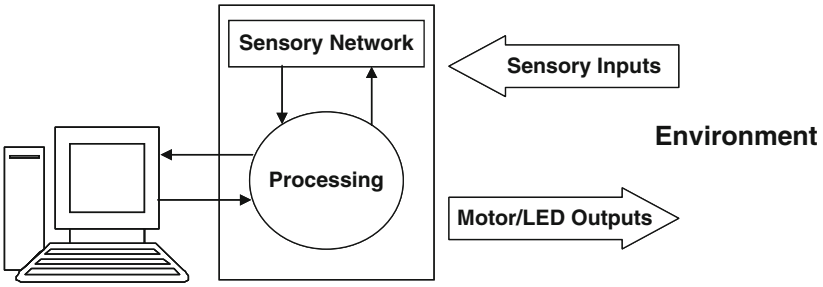

Fig. 1.3 A computing node in a dichotomized architecture

The main characteristic of this architecture is the total lack of local autonomy. There is little intelligence in the local node, and a central controller makes most (if not all) of the decisions. The name "sensor network" is used where the local nodes are simple sensors that just provide measurements on distinct physical phenomena.

1.1.5 Expanded Architecture

The expanded architecture is a compromise between the former ones. The physical parts are twinned up with computing facilities to create what seems to be a physical network using digital communication. Where the parts are fully autonomous, the architecture has just a dichotomy between the parts. Actually, for a sequence of strict object-oriented processes, it seems irrelevant for all except the first and the last function on which platform they are performed. Admittedly having data compression before transmission makes a difference on the efficiency of the sequence of processes in the network, but it does not fundamentally change the architecture.

Expansion becomes different when the network becomes noticeable. For a distributed data collection system, transmission effects will hardly play a role. But when interaction occurs through the network, this may not be true anymore. The

[3]Definition of Dichotomized (Merriam-Webster Online Dictionary): to divide into two parts, classes, or groups.

classical example is in the history of Energy Grid breakdowns, where the central processor received the warning that a node is collapsing, but could not react in time [8]. This makes it necessary to look closer into these problems during the design of the overall system.

The main characteristic of this architecture is the level of collaboration of the nodes. The main problem of the Expanded Architecture arises when the physical reality does not provide for a simple measurement but needs interpretation. This needs to be served by collaboration over a network, expanding the architecture for extending its reach. We will see this theme re-occur throughout this book.

1.2 Network Processing Architectures

In the previous sections we have described embedded nodes. Such nodes come seldom alone. In dedicated architectures, it is possible to connect them in a unique and not reusable manner, but it has become good practice to use a more universal and extendable means. This results in a network of processing nodes.

There is a world of difference between having many small computers and a few big ones in a processing network. Though the network connectivity brings an amount of reuse and scalability, the real innovation comes about when some additional parameters such as variability and accessibility are added.

In this section, we will argue that the processing network will become the new design unit, the next abstraction beyond the computer. At this level, the software and hardware components on which the processing functionality is implemented will become so integrated that they will not be separately discernable again. It will make the network a cloud of embedded computers, similar to the computer-farm being a cloud of servers.

1.2.1 Statically Structured Network Processing

Before cloud computing, other large-scale networked computing approaches have been proposed and attempted, such as cluster computing, a homogeneous collection of processing blades, to grid computing, with its communication and distributed network management mechanisms, ultimately all the way to Legion, envisioned as an integrated global computer.

1.2.1.1 Cluster Computing

As the microprocessor is a general-purpose device, it is good for anything but not for something in particular. For this reason, it is often supplemented with an attached processor that is specifically designed for a special purpose. An example is in graphics processing. The single computer with graphics software is notoriously in need of high performance, just for that purpose. A direct improvement is to attach a

special-purpose graphics processor to the computer. A next speed-up comes when several computers are clustered together with a fast interconnect. This allows building a supercomputer from a number of standard computing blades, each containing multiple processing nodes. Where the single computing node has gradually been equipped with a number of threads in which the processing node performs a program partly in parallel with the others, the cluster allows many more simultaneous threads.

The cluster concept is limited by the scalability of the interconnect network and the memory architecture. A shared-memory architecture is commonly applied to facilitate that all nodes may work coherently on the same problem. If the coherency requirement is easy to fulfill, the architecture seems reasonably scalable and has gradually found its place on the chip level as well. A necessary, but not sufficient, condition is well-behaved timing on the internal memory bus. In this sense, it helps to have all nodes of the same technology, for instance, on one board or on one chip. A typical example is the IBM Cell cluster of Synergistic Processing Elements. Extending this architectural concept over more than one Cell chip adds variability to the memory access timing and consequently provides coherency problems.

Adding a next interconnection network to orchestrate the nodes, each in turn being a multicore arrangement of multithreading processors results in the classical supercomputer. As timing (and especially coherency) is crucial, it is not easy to program such an idolized computing concept. So gradually such machines are being replaced by server farms, where the fast provision of service to many clients is the goal rather than the long-awaited fast execution for each client in the waiting line.

1.2.1.2 Grid Computing

With the coming along of the Internet, the role of the clustering interconnect network can be taken over by a general-purpose structure. But what the Internet contributes is a large number of already connected computers that can each independently perform a part of a large data-processing task, but not necessarily at the desired moments in time. The main characteristic is the loose coupling of dissimilar and geographically widely distributed nodes. The concurrency is therefore on the job level, each computer working on a part of the large problem and its related database.

The basic philosophy behind grid computing is to bring together the massive resources of the Internet and to make them available for a desired purpose that is not defined as a single parallel program, but rather by task-by-task volunteering. A typical example is the SETI (Search for Extra-Terrestrial Intelligence) project, which invites people from all over the world to run a downloadable program and a new stellar map to find new life forms. Reportedly, more than 3 million people have volunteered and found nothing.

Grid computing technology has been supported by several user-based organizations, such as The Globus Alliance (www.globus.org) and The Global Grid Forum (www.ggf.org).

1.2.1.3 Legion Processor

The Legion processor is an experiment to create the ultimate virtual grid computer: one that covers the entire world [9]. It would be assembled out of geographically distributed, heterogeneous collections of workstations and supercomputers. Legion provides the software infrastructure layer, so that the system can interact seamlessly. No manual installation of binaries over multiple platforms would be required, Legion does it automatically by providing this additional overlay layer. Legion aims to extend on grid computing by an "open software" organization. It is a work-in-progress project at the University of Virginia, aiming for a testbed in the design of highly parallel software systems. It was started in 1993 and is not supposed to end at any time near.

1.2.2 Dynamically Structured Network Computing

An important milestone in the history of computing technology is the coming of age of mobility. Mobile devices do not have a fixed connectivity, as its environment will change on the move. Before cloud computing came about, many related forms of computing and related networks have been researched and analyzed.

The original words of Thomas J. Watson (the founder of the International Business Machine company) that there will be a worldwide market for at most six computers is still passed on among computer scientists. Today, this saying has become almost hilarious as history has passed the crucial point already decades ago. And while passing from a state of scarcity to a state of abundance, the world has changed too. Watson's opinion is typical for the first wave of computing. Based on mainframes, each shared by many people, and housed in special "places of worship," called Computing Center, there was no reason at that time to expect the machine to be used for others than a few devotees. But the technology has shifted, first from tubes to transistors, then from bipolar to field-effect devices. It produced a magical decrease in power consumption and enabled a drastic level of miniaturization.

The second wave of computing started out as a typical disruptive technology. While the mainframe market went to service its customers with faster, more complex machines, experiments with microcontrollers were commencing in a large number of start-up companies. Hand in hand with advances in microelectronic fabrication, the microcontroller became a processor to enable the Personal Computer, the PC. This placed the single computer at the disposal of the single person. With the success of the personal platform came the movement to bring more luxury to the user, both in friendliness as in transparency. The Internet opened up a world-wide web of computers in a technology that unifies all mechanical means of communication.

1.2.2.1 Ubiquitous Computing

The third wave of computing was already named by Mark Weiser as Ubiquitous[4] Computing [10]. The network of a myriad of computing devices was to diffuse the personality of the individual node and make all functionality available at all places and at all times.[5] For every user, there will be a couple of hundreds of computing devices available. Where in days past, additional capacity was swiftly filled by the new operating system or latent user needs, the ubiquitous computing society will have enough redundancy in space and capabilities to isolate users from such technical details.

Reeves and Nass [11] have coined a different name to reflect that the future computer will support its humanoid environment: ambient computing. It stresses the presence of intelligent social interfaces on a networking technical infrastructure of embedded computers with personalized, adaptive, and anticipatory behavior. Where the late Mark Weiser prophesized the third wave to flow from 2005 until 2020, later researchers expected faster progress. In 2001, the European Community has predicted that ambient computing may already be mature and socially accepted within a decade![6] Such a technology will best be characterized as conscious or aware. This emphasizes the inversion over time in the societal function of the computer: from a unique machine to a mass service.

1.2.2.2 Nomadic Computing

Nomadic Computing[7] received a wider interpretation in the course of time. Initially, it was just meaning that a person traveled and then attempted to work from another location, say the hotel. The concept became more meaningful with the introduction of the portable computer. The users did not only travel, but take their network nodes with them. At the other location, the users plug their computers into the new network, aiming to use the local facilities as if they were at their home.

Nomadic computing became feasible with the introduction of the Internet as the network topology was liberated from the physical location by the network address. The addition of wireless communication created the flexibility that made nomadic computing in its mobile computing variety almost the normal mode. Though going unnoticed, its acceptation is a paradigm shift, causing a drastic change in ways we think about information processing.

---

[4] Definition of Ubiquity (Merriam-Webster Online Dictionary): presence everywhere or in many places especially simultaneously: omnipresence.

[5] Jokingly this has been dubbed the "Martini feeling" in reference to the 1990s commercial where the song quoted that Martini could be drunk "Any Time, Any Place, Anywhere" because "It is a wonderful world you can share."

[6] "Scenarios for Ambient Intelligence in 2010," ISTAG Report (European Community), 2001.

[7] Definition of Nomadic (Merriam-Webster Online Dictionary): 1: of, relating to, or characteristic of nomads <a nomadic tribe>, 2: roaming about from place to place aimlessly, frequently, or without a fixed pattern of movement.

With nomadic computing, one is never sure about the available bandwidth. When connectivity is lost, work should continue as good as possible. A telephone call will halt it (but should it break off?) and e-mail handling will continue, but synchronized again later. Nomadicity provides a transparent virtual network, making the accidentally connected computer to operate (almost) independent of the available bandwidth.

1.2.2.3 Pervasive Computing

Embedded computers are pervasive[8] by nature. Seemingly unnoticed, the embedded market has always been dominant, but using the more primitive hardware components it lacked appeal. For a long time, the 4-bit controller was the main seller. But meanwhile the product got better streamlined in timing and programming possibilities, adding real-time operation and support, as well as software transparency. Sensors and actuators have become flexible, configurable peripherals that can be networked as thin clients to bind the outside world into the server process.

Where the PC era brought the technology of the virtual world in which persons and artifacts were represented on the single platform as interacting software bots mimicking human behavior, the pervasive technology brings a real world of interacting natural and artificial processes communicating over virtual links to create a designed behavior. Every player has its own set of devices that characterize the contribution offered to the ensemble. Replacing the collection of simulated parts by the simulation of collected parts creates an artificial awareness and turns the computation upside down. The system operates directly and in real-time on what happens in the outside world or, in other words, is aware of what happens.

The discipline of computing science becomes rapidly multidisciplinary with the advent of computational awareness. The functions by which human behavior aspects are communicated within the simulated world are psychological and sociological of nature. Modeling approaches that underlie such pervasive systems will hardly be of the mathematically precise, deterministic kind that are used for the more chemical and physical problems of old.

Pervasive computing takes place in networks of loosely distributed elemental computers. The size of the network will be dynamically changing through generation/attrition of compute nodes. The nodes are wirelessly connected through relatively low-bandwidth communication paths. The individual nodes represent integration of ultra-low-power technology. They are intended to be low-cost disposable elemental computers. The Web-like nature of the communication network allows then for the executing of a larger problem on a cluster of elemental

[8]Definition of Pervade (Merriam-Webster Online Dictionary): to become diffused throughout every part of.

computers, especially if the problem can be partitioned into a set of not-necessarily-homogeneous subproblem computations.

1.2.3 The Network Is the Computer"TM"

For many years companies such as Sun Microsystems, and also Oracle, have used in their marketing philosophy the notion of "the Network is the Computer." This was reflected in their emphasis on "dumb" terminals, the thin clients. Sun Microsystems during their recent comeback has created systems highly supportive of cloud computing, in their BlackBox (containers for server farms) and SPOTS (sensors for monitoring the low maintenance BlackBox). Most recently, Oracle has acquired Sun Microsystems.

The point of network-based pervasive computing is that the question on the nature of the computer seems actually moot. Anything in the embedded world is a computer, even and including a network. And there are so many nodes that most of them could be inactive at a given moment. Given further that through nomadicity the network structures will constantly change and we see that any network is a computer. Ad hoc networks use this point of view to their advantage. For instance, where buoys are drifting through the Great Barrier Reef [12], their wireless intercommunication network to reach eventually the support center on the mainland uses an organization based on the current location of the individual buoys.

But the ad hoc network does not always have to be homogeneous. Embedded products come with a variety of resources, and this will not stop them from having ad hoc arrangements. Where the nodes can communicate, they do not have to be general-purpose anymore. Any local functionality can be extended by neighbors when the communication is close enough. In case of redundancy, this is a mere matter of decision-making.

A typical example is in the camera array, where sensors may look at the same object from different angles [13]. It remains to be decided which camera has the best view (for instance, has the least shadow). Another extension is when a node is found to be defective, upon which its role in the network can be taken over by another one [14].

For instance, in a vision network for home living monitoring, the role of the leading camera can be passed around, ensuring that the local net can always communicate to the home network. All such considerations make that the network is not just a computer, but that the network can be any computer it needs to be.

1.3 Intelligent Sensor Networking

More and more intelligent sensor subsystems will become available and cost-effective although initially they could be somewhat esoteric. An example of the latter could be hyperspectral cameras that currently are expensive and extremely useful, but that could evolve into future more ubiquitous use.

1.3.1 Various Sensing Principles

Sensors can be used to measure shock (accelerometer), sound (microphone), temperature (thermometer), location (GPS), and various elements of the spectrum (Infrared, Video, MicroWave), among many others. More intelligence can be had by integration (fusion) of multiple sensors. In addition, integrated neural net-based learning can be applied. The following are some sample sensor subsystems used in security applications:

- Infrared/Multispectral Sensors – Uncooled infrared sensors provide valuable information, particularly in harsh environments. More information can, however, be obtained by analyzing responses from sources in other frequency ranges too. For instance, finding body heat while the visible spectrum shows a nonliving object can reveal a person in hiding.
- Hyperspectral imaging systems [15] – Such systems with dramatically more frequency responses are generally found in space and survey applications. In spite of their applicability, they have not yet been deployed in volume in industrial vision markets such as security, medicine, biotechnology, machine vision, or other image-related spin-off markets.
- X-Ray/Radar/Microwave Sensors – Radar has been used for protecting borders, detecting people and moving vehicles, among others. Hand-held radar systems for through-walls-visibility are being developed for the detection of people in buildings. In addition, three-dimensional representations of the inside of the building [16] will be of interest to emergency responders. Backscatter radar van is used for inspecting vehicles and containers in transit. X-ray systems have been used for checking persons, hand-carried luggage, checked baggage, and other cargo.
- Acoustic Camera's/Sound Visualization – A small vibration sensor module can be replicated and applied in quantities and arrangements appropriate to a conveyance at issue. Combining signals from the multiple vibration sensor modules and accounting for their respective positions will result in a topographic map. This "acoustic camera" image [17] can be displayed for human inspection or manipulated by suitable 2-D signal processing algorithms, such as Cellular Neural Networks (see details in Section 4.5.4).

Multiple small sensors can be distributed over an area of interest. Again information, such as vehicle passage, can be gained by fusing data from individual sensors. The sensors will be communicating over an amorphous distributed network.

Issues concerning the tradeoff between processing and communication, especially in regards to power consumption, will need to be addressed. Some form of power harvesting from the environment will be required for long-term deployment.

1.3.2 Sensory Networks

Sensors provide information on a physical effect. By proper installation, the sensor becomes related to the object under measurement, but it is not confined to that. Even more, sensors may give information that overlap and complement. Therefore, sensory signals may be correlated and provide redundant information about the world in which it is placed rather than on the product to which it is attached.

Such structural redundancy can also be applied to entire networks. The major difference is that actuation can usually be observed to find whether the control in question is in error. For a sensory network, such a simple fault determination is not possible. The obvious way to improve that is by voting between redundant results. Sensors of different technology but equal potential quality can view the same event. When all sensors agree, then the observation is probably true; a faulty sensing can be concluded for a specified amount of disagreement. These two concepts can be fused by noticing that (1) the requirement on the sensors to be different but equally capable is too stringent (as long as they see the same event), and (2) a pure "line-of-command" layering is too discriminatory.

Neural networks can be used to extract the relevant information from the sensors. Combining such information over a neural network allows making decisions based on the combined contributions. Overall, this provides quantification for a next layer of abstraction, as described in Chapters 3 and 4.

Looking back, the microelectronic computer has gradually moved from a product by itself to an embedded system part. Simultaneously, we can see that value added in one application domain sets the moment to enter into another domain. This gradual movement can be seen in Fig. 1.4. Together, they have created the case for intelligent sensors allowing for distribution over a sensory network. It is anticipated that wireless communication will lead to a further loosening of the system cohesion and spawns the federated network on which application domains such as the

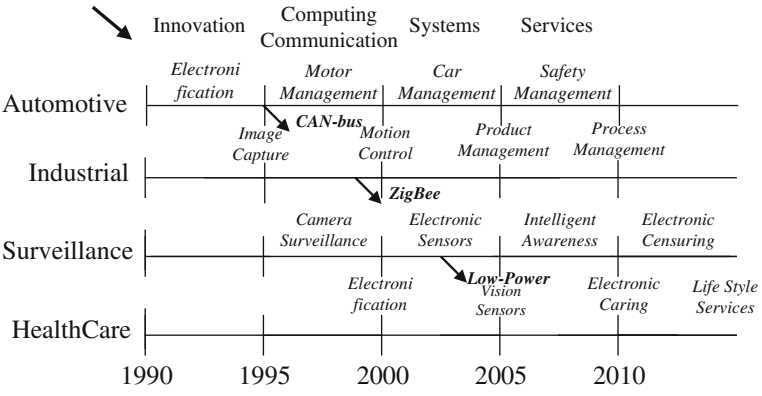

Fig. 1.4 The influx of sensory networks [14]

Caring, Comforting and Concerned (C3) Home, Home Automation and Healthcare will boom (see also Section 4.3.2).

A network, also one comprising of sensors, is a collection of cooperating objects. Some division of labor has to be introduced to bring structure in the seeming chaos. The guiding rule to divide a network into parts emphasizes one of more of the outstanding system characteristics. Such a choice is made during the historical development of computer networks in a continuous stream of small improvements. But a network of appliances will have to be designed in the presence of a wide variety of device functionalities, taking all into account at once.

Within the sensor network, each sensory node is equipped with a network interface and will therefore become part of a digital network, indistinguishable from other computing elements with a network interface. Such could be another Web-based appliance, but also a Personal Computer. Some of the elements have a special task. They perform a service for the network community and are therefore called Servers. Examples are database servers and e-mail servers. Others just perform their limited duty for the outside world and are being served for the network functionality. They are called Clients. When they need little help, one rather speaks of Thick Clients; when they are almost helpless with respect to the network, the term Thin Client[9] is used. One or more servers and a number of clients are linked together into a Local Net.

Two directions can be discerned in the wide variety of network protocols. The token-passing protocol assumes that a single packet moves from node to node. At each node, it is decided whether the destination is reached or whether the message must be passed to the next node. So the packet serves as an activity token, unique within the network. Such a mechanism is especially suited for expensive, high-speed connections. The alternative is the collision-detect protocol. The concept assumes a fully connected structure without any consciousness of activity. Therefore, every node may put a packet on the network at any time. When two packets are present simultaneously, this is detected at the receiving side as a bit mutilation. Both messages are then withdrawn and put on the penalty bench for a random period. For a low traffic density, such poses no problem. Problems arise with densities over 60%; 100% is only a theoretical possibility.

At least one of the servers provides access to other nets by means of a Gateway. The gateway knows what is included in the local net. So when a message arrives, it can easily determine whether to guide the message into the local net or not. The gateway also recollects for some time, where outgoing messages went. When it knows the required path, it does not have to find the path again. The gateway can communicate directly to a neighboring net or to a router. Routers can be arranged in a hierarchical manner to ensure that every node can connect to another node in maximum two times the depth of the hierarchical tree. If the topology is flat, routers need large address tables; if the topology is not exactly flat, more routers are needed. The optimal system is based on a judicious balance between the number of routers

[9] Similar to Sun Microsystems' original notion of a "Dumb Terminal."

and the size of the address tables. Specialized, high-speed routers are required to shift the optimum.

Clearly, such hardware devices are part of the grand scheme of universal world-wide communication and therefore not required in personal-area, body-area, or tiny-area sensory networks. Web-based appliances are often extremely thin, in the sense that they have only a basic web support and need the help of the gateway to provide a fully functional presence on the net. As a consequence, the networks need to be designed on basis of the anticipated information flow within an architecture that is meant to provide a functionality which is more than the sum of the parts.

1.3.3 Sensor Networks Design Methodology

Currently, unattended sensor networks (USN) are developed from the bottom-up [18]; first the sensor technology is developed, then its compute capabilities within its power limitations, followed by its network principles/protocols and ultimately its actionable interface to the world, including its access to the Internet (Fig. 1.5).

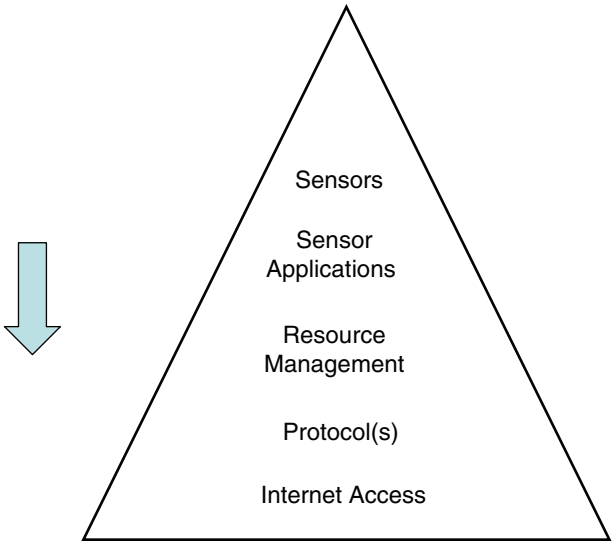

Fig. 1.5 Current bottom-up network design

This preferential attachment worked nicely in the past for the Internet, where third parties competed on the market to develop ever better network structures and services. In other words, the growth of the network implies an expensive, closed, and labor-intensive process of continuous renewal, which can only be paid from the exponential increase of commercial value of the resulting network.

As such is not the case for USNs, a different, rather top-down enforced, development plan (Fig. 1.6) is needed. We suggest [19] that this alternative should be based

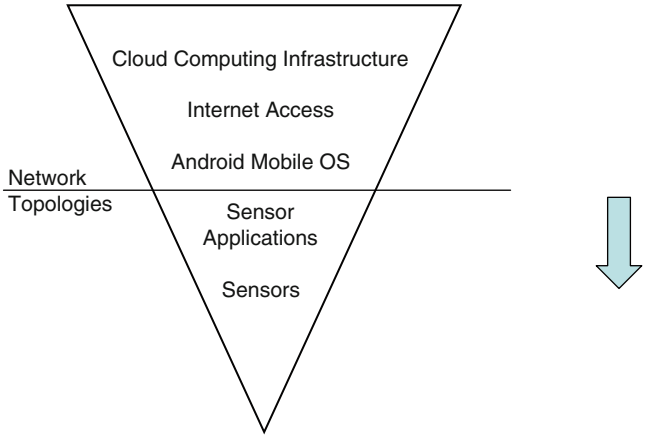

Fig. 1.6 Future top-down network design

on "cloud computing" as recently announced for advanced large-scale distributed computing environments. We have envisioned the implementation of "cloud computing" as a large "mainframe"-like centralized enterprise-computing facility with an Internet-connected large set of thin-client sensor nodes.

Sensory devices are domain transformers. Events in different physical or physiological domains are sampled and turned into an electrical signal. Sensors are of a wide technological variety, though microelectronic sensors are rising in popularity, as they promise a further integration of the sensory element and the signal-processing system. For instance, a temperature sensor can be based on the temperature sensitivity of electron/hole pair generation. This flow of charge carriers constitutes a current that can be handled as an analog value and therefore easily converted into the digital domain.

The presence of some level of digital signal processing is gradually moving from remote computer boards into the sensors. The driving force behind this move is the need for programmability, that is, the structured reuse in time of scarce resources. The basic notion of Instruction Set Architecture (ISA) arose from the desire to perform different computations on the same hardware. At first sight, this does not seem applicable to a sensor. A specific sensor is meant to be dedicated in performing a specific measurement. Though such may be true, the environment in which to measure may be constantly changing. To accommodate for such dynamics is often beyond the scope of the analog signal processing. An early example is the software defined radio (SDR (http://www.wirelessinnovation.org), where the need for Hi-Fi in a moving vehicle necessitates a (re-)programmed device.

The sensor output is often representative of the measured phenomenon rather than in direct correspondence. The transformational processing often takes the primary component out and might therefore loose some of the available information. Such is not always to the benefit of any subsequent detection. To compensate for such effects, good sensors tend to be extremely dedicated and/or expensive.

Compared to nature, this seems strange. Even the most brainless creatures have an amazing capacity to sense. It seems therefore natural that biology will inspire a sensor designer to do it differently. Biology exploits redundancy in a grand style and for many different purposes. On first sight, it lowers the demands on dependability as the desired information is present for as long as at least one sensor is communicating with the embedding world. Where the active sensors perform the same measurement, this already provides a level of robustness. But the example of the Ariane 5 disaster in June 1995 [20], where a buffer overflow caused a computer to shut off and pass control to a spare device that was suffering from the same problem, shows how limited such security can be. A more natural way of majority voting occurs when the sensors have each a different, but partially overlapping, interest (as for collision avoidance described in Section 4.5.1). An example is the replacement of a single expensive but error-prone sensor in the rocket chamber test by a large set of cheap sensors distributed over the area [21].

Apparently, *n*-version-measuring can only be trusted (and even then only to a degree) when applying diversity in sensor placement and function. In other words, full independence of the data source is required at all times [22]. Within the context of a single grand world model, the sensors should give readings that are in full agreement or risk being overridden by the models interpretation of what the other sensors have found.

This brings us essentially to what biology shows in experimental proof: a large number of cheap, dumb sensors can be better than a single, expensive one. An example is the collision avoidance experiment from the early 1990s EU-Prometheus project [23] (see more details in Section 3.2.3), where a single neural network can be trained to balance the influence of many sensors.

The network of sensors is based on digital technology. A simple communication standard (such as ZigBee, more in Section 4.1.1) makes the sensors mutually aware and creates a collective reading. The central controller needs not to be present, as most primitive creatures have no brain but are still able to have coordinated sensory reading. This illustrates that the old-fashioned central controller paradigm may be keeping us away from simple and still efficient solutions!

1.4 Expansion of the Ambient Intelligence Approach

Current notions of "ambient[10] intelligence" in sensor networks are based on the development of information/awareness spread over an ad hoc distributed sensors environment. This involves a large amount of computation and communication in the network in order to uncover the global information/awareness content.

[10]Definition of Ambient (Merriam-Webster Online Dictionary): existing or present on all sides: encompassing. With some small differences, we see a lot of similar techniques under different names. It seems to show the firm preferences of some companies: Pervasive Computing (IBM), Ubiquitous Computing (Microsoft, Intel, Xerox), Sentient Computing (AT&T), Active Badge (HP), Cooltown (HP) and Ambient Computing (Philips).

Subsequently, this information/awareness will have to be communicated outside of the network to become actionable. Though the workload will already be high for a simple peer-to-peer data collection scheme, it will rapidly become overbearing and not realistic where timely reactions are needed. In Chapter 4, we will review examples of the following two situations:

- Awareness Integration – In a typical gesturing system, multiple cameras observe the body from different angles. Some of them look at a right hand and collectively learn to understand its movements; others look at the face or the mouth. In a subsequent layer, such image details are combined into an overall understanding. Many image-understanding applications can be supported by this architecture, but it is often intractable to optimally allocate tasks on sensors while being both geometrical- and functional-specific [24].
- Network Modeling – A misleadingly simple behavioral modeling example is in capturing the motions in the Sinai billiard system [25]. Again the ground knowledge of the physical elasticity laws can be designed into a basic set of nodes, elementary motions trained into a next layer to finally produce motions. As has been published [26], many simple nodes in selected topologies have been found to perform better than a few complex nodes in an ad hoc arrangement.

1.4.1 Situational Computing

Where the computer was in the hot spot when it was still a scarce resource, the current proliferation of computers places the user in the center of the virtual world. One can view this as an essential inversion of the computing paradigm, which will therefore demand new solutions. The reliance on human behavior in situational awareness sensing stresses the flexibility to handle any situation in which the person is placed. One or more cameras can be used for this purpose, but that requires a major centralized computational effort and results in a cloudless architecture still limited in data quality and completeness. The alternative is to add many intelligent sensors to the person's body, either directly or through clothing. This wearable technology is by nature a mobile application though stretching far beyond the usual portable communicator. Gershenfeld [27] illustrates this with the story of the invisible cello.

When the situation is location-dependent, some measures are required. In days past, this was often realized by dedicated transponders; in aviation this has been the case for a long time, while in sailing the lack of geographical marks has quickly spurred the search for other means. The use of low-frequency radio transponders placed on land was popularized in the LORAN system. Recently, a more generic means was created by the installation of the Global Positioning Satellites. Originally designed for military purposes, GPS facilitates a careful measurement of the location on earth by a triangular communication to the geostationary GPS satellites. It

has quickly gained in popularity and produced location-based computing. All kind of routing services, either mobile or fixed into a mobile environment, can be devised using such data.

Vandevelde of the former Belgium company StarLab has introduced the notion of "Situational Computing" in analogy with the similar notion in psychology to describe the behavior/bias interface [28]. It brings elements like Self and Relation into play to support interaction between actors. Similar relations can be described between the persons and the space they act in. Hence, we enter the world of bots and avatars.

But we have to draw a line here between the virtual world with real functions situated within the single computer as was usual in the desktop area, and the real world with virtualized functions created out of the communication of available Web- and Cam-tops. In the real world, we have to augment the cognitive functions to enrich the interaction and collaboration between the actors in the present situations (Augmented Reality), and this is what situational computing is all about. The theory is nice, but in order to get this really working some radical breakthroughs are required.

- Energy – As most of the situational applications are (partly) of a mobile nature, energy comes at a premium. We need nonsupervised behavior, batteries with a long life-time, power consumption extremely low, or energy passed as part of the message;
- Maintenance – With so many computers invading society, the parts need to be high quality and low maintenance as the users will generally not to be trusted with fault diagnosis, isolation, and repair;
- Emotion – The device must be transparent to the user and answer to all his/hers emotional hang-ups. It should be more than a best friend: it should be supportive and yet not visible;
- Security – As the devices are highly networked, intrusion prevention is a major issue.

1.4.2 Autonomic Computing

Autonomic Computing[11] has been touted as a bio-oriented distributed networking realization. The goal was to create computing networks and systems that – like the biological analog – hide complexity from the user and deliver much greater value than today's systems can. These new systems need to be self-managing, self-configuring, self-healing, self-protecting, and continuously self-optimizing.

[11] Definition of Autonomic (Merriam-Webster Online Dictionary): 1. acting or occurring involuntarily <autonomic reflexes>, 2. relating to, affecting, or controlled by the autonomic nervous system or its effects or activity <autonomic drugs>.

To manage a large distributed system requires a stable blue-print. A detailed analysis of the blue-print will then provide the control for the system. Already for large software systems, this will be hard to reach, and – even if one could get there – the maintenance is hard to do. Things get only more complex for large, geographically distributed systems. Therefore, the ideal is to get away from all these things and on bio-inspiration to come up with an architecture that can manage itself by configuring around the failures to achieve a high degree of system health.

Most of large systems develop by autonomous growth and preferential attachment. For instance, a production line is based on machines that are individually optimized in the expectation that this will bring a healthy overall system. The practice of large-scale fabrication shows a phenomenon called "alarm flood." The many nonideal situations give rise to many signals that indicate potential failures, but often are not that drastic. Consequently, such alarms are generally ignored together with the occasional realistic alarm, subsequently causing a real disaster (see also Section 6.3). So these ideals seem really desirable.

Unfortunately, autonomic computing has been found to be too challenging, one bridge too far. It has encountered a recent revitalization as a result of cloud computing conferences [29] and other developments.[12]

1.4.3 Organic Computing

From a further bio-inspiration comes "Organic[13] Computing." Next to ease of maintenance, it stresses the adaptive, goal-directed reconfiguration of a large number of simple components. In this sense, it is a learning mechanism where the system slowly adapts to its environment and to the tasks it has to perform.

An organic system has to be robust and self-organizing. A strategic goal can be supplied, but how to reach this goal is open to gradual adaptations. Global control is absent as such would imply the presence of a scheduling mechanism. Similarly, synchronization has to be based on triggering events rather than on fixed clocks, as a master clock is a global control mechanism.

---

[12] International Research Institute for Autonomic Computing, www.irianc.com.

[13] Definition of Organic (Merriam-Webster Online Dictionary): 1. archaic: instrumental, 2. a: of, relating to, or arising in a bodily organ b: affecting the structure of the organism, 3. a (1): of, relating to, or derived from living organisms <organic evolution> (2): of, relating to, yielding, or involving the use of food produced with the use of feed or fertilizer of plant or animal origin without employment of chemically formulated fertilizers, growth stimulants, antibiotics, or pesticides <organic farming> <organic produce> b (1): of, relating to, or containing carbon compounds (2): relating to, being, or dealt with by a branch of chemistry concerned with the carbon compounds of living beings and most other carbon compounds, 4. a: forming an integral element of a whole: fundamental <incidental music rather than organic parts of the action – Francis Fergusson> b: having systematic coordination of parts: organized <an organic whole> c: having the characteristics of an organism: developing in the manner of a living plant or animal <society is organic>, 5. of, relating to, or constituting the law by which a government or organization exists.

Fault-tolerance is required to avoid a too high sensitivity to model aberrations that can easily invoke an alarm flood. Instead, it is assumed that the nodal functionality evolves toward a given global target, but the way this happens is not predefined. Consequently, nodes may starve when going in the wrong direction, but given enough nodes enough healthy nodes will survive. Though again too ideal to become ever fully realized, it is important to keep organic functionality in mind. If pervasive computing will ever become an accepted reality, it is because it will never draw attention. For this goal organic-like behavior is a must. However, a mixed approach where the physical hardware reality is supplemented by a virtual copy "in the Cloud" is demonstrated to be most promising. An example is the CloneCloud, as discussed in Section 2.1.1.

1.5 Cloud Computing Concepts

Cloud computing[14] is based on a large "mainframe"-like centralized enterprise-computing facility, with an Internet-connected large set of thin clients [30]. Characteristic of such a "cloud computing" system [31] is its virtual nature (location and infrastructure details of the "cloud" are transparent to the clients), its scalability (ability to handle complex client workloads across an incrementally expandable "cloud" infrastructure), its efficiency (services-oriented "cloud" computing resources), and its ultimate flexibility (serving a wide variety of workloads). In the "cloud computing" architecture, the data and services reside in massively scalable data centers that can be universally accessed from any connected devices over the Internet.

The emphasis in our cloud-computing-based use of sensor network processing is a top-down approach to sensors network development. In principle, the high-performance centralized (directing/collecting) computer and its Internet communication/ infrastructure capabilities will come first; the thin-client sensor elements at first only collect and forward data. The resulting division of labor will maximize compute capabilities, minimize communication and total system power requirements. In order to support the development of such extremely large sensor networks, we suggest the development of a processing infrastructure for the collection, analysis, and visualization of large data sets from sensor network based on the cloud computing paradigm. This development will include many

[14]Definition of Cloud (Merriam-Webster Online Dictionary): 1: a visible mass of particles of condensed vapor (as water or ice) suspended in the atmosphere of a planet (as the earth) or moon, 2: something resembling or suggesting a cloud: as a: a light filmy, puffy, or billowy mass seeming to float in the air <a cloud of blond hair> <a ship under a cloud of sail> b (1): a usually visible mass of minute particles suspended in the air or a gas (2): an aggregation of usually obscuring matter especially in interstellar space (3): an aggregate of charged particles (as electrons) c: a great crowd or multitude: swarm <clouds of mosquitoes>, 3: something that has a dark, lowering, or threatening aspect <clouds of war> <a cloud of suspicion>, 4: something that obscures or blemishes <a cloud of ambiguity>, 5: a dark or opaque vein or spot (as in marble or a precious stone).

attributes available "as-a-Service" in the cloud computing environment, such as Software-as-a-Service, Security-as-a-Service leading up to Analysis-as-a-Service and Visualization-as-a-Service realizations (see Section 1.6.1).

Cloud computing will lead to a more effective processing networking environment and infrastructure for defense (UAV, fire control), commercial (surveillance, security), medical (body-area-networks, patient monitoring), gaming (MANET paintball, geo-caching, etc.), and other markets.

In Section 1.4, we have reviewed various networking architectures and their allusions to the concept of Cloud Computing. In Section 1.6, we will summarize these developments toward cloud computing, starting with the old mainframe technology.

1.5.1 The Road to the Cloud

The dinosaurs are not with us anymore. The species are extinct and nobody knows exactly why. It could be a meteor; it could also be the ashes from a volcano outburst. Room enough for speculation! However, in computer engineering nothing gets really extinct. Even COBOL is still with us. But we can see some developments and try to extend on that in order to see the future. What we are especially concerned with are disruptive changes that come to pass when the gradual improvements get too complicated. We will review these developments by looking at Fig. 1.7, while pointing out that real surprises are not there. In fact, most of the innovation is based on existing things that suddenly get the chance to have more meaning.

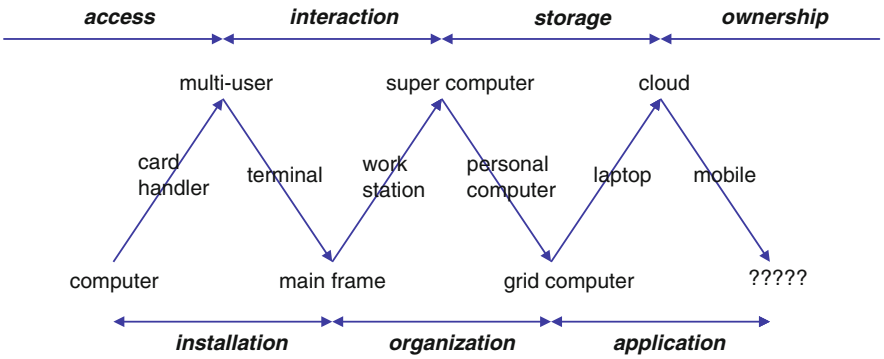

Fig. 1.7 Bouncing against the barriers

1.5.1.1 Where Are/Were the Barriers?

Computing features a steady improvement in terms of the potential of the workhorse in question. The most basic original form of computing is the single, primitive engine with a single point of entry and capable of doing a single job at a time.

The point of entry was a card station or a paper tape reader, where the medium was off-line prepared and the job was executed in batches.

With maturing technology, the batches are mixed so that more users can be seemingly handled at the same time. This works especially well when the batches do not require to be active at the same time, for instance, when handling input. This went nicely hand-in-hand with the coming of the teletype, where the in-line entering of the input had to wait regularly for the next key-stroke. Taking this one step further, we can combine the teletypes with an I/O station that allows for buffering and thereby editing while the main computer focuses on the real executive tasks.

Low-cost, in-line program editing explodes the number of users. But users do not only want to run programs but also want to see results. And while input is preferably textual, output is better understood in its graphical form. This urges more power with the user to support his/hers needs for rendering and graphical manipulation. Having graphical output also leads to more and more complicated programs and gradually the main computer turned into a supercomputer.

Having to wait for the batches to finish is no fun. Already at the time of the multi-user computer, it was possible to continue editing through the I/O station while the main computer was down, as long as one was already logged in. So with the workstation, it was even better. Once the output was transferred, one could locally work on it. But still one had to go to the station. Then making the workstation personal allows working at one's own time and place. In time, such personal computers became so powerful that they were a computer by themselves.

While turning into supercomputers, mainframes became too big to simply serve personal computers. So the mainframes are still there, but this time called "servers." Each mainframe handles a group of smaller computers, providing expensive and/or elaborate facilities to be shared. For the supercomputer they have become what the I/O station was in the past. But with a lot of local computer power, the next step was to bring them together into a virtual entity, the grid. This is the supercomputer, but facilitated by a large number of personal computers at various locations.

While the personal computers got more and more roaming and became portable or lap computers, the servers got assembled to optimize the use of the resources. This is conventionally called the cloud. Though size is still an issue, service provision has become a major worry. Service is the ultimate "program + data." So the management of digital rights is ruling the use of programs on data sets. In the meantime, the laptops have grown into mobile device of a varying structure with the potential to be grouped into units with a common goal.

1.5.1.2 What Are the Growth Dimensions?

There are three obvious architectural dimensions: speed, data storage, and physical size. With the advance of microelectronic fabrication, we see a clear increase in speed at both the user and the engine side. This is a clear "push" effect; things are just made smaller and faster. Disk technology has also a clear impact, raising the amount of locally stored data by factors. Though different in details from what is used for the processor, it is still a microelectronic technology and therefore uses the

advances in production techniques in this area. Again it seems that the fabrication technology is pushing the state-of-the-art in computer engineering. And together with this comes physical size. Where the paper tape reader had to be tightly integrated in the computer frame, current networks may perform a single computational task over a very wide area.

If this is just a series of continuing improvements, what then makes for the regular surprises? It seems that microelectronics is a major driving force, but not the only one! On reflection, we see that the historical review described before is one of changing emphasis. In the first instance, the interest was in creating access to the platform for as many users as possible. Then the focus shifts to interaction, allowing the user to work without being aware of the other people at the same system. Subsequently, the issue of data storage arises to have the user work in real time with real data. We pose here that as a direct consequence the current focus is on ownership.

This has everything to do with the way the parts hold together. The computer of old was completely designed by the manufacturer with unique cables and assumptions on the closeness of the frame(s). With the coming of the Internet, the composition of the system is defined by the organization in which it is used. They design the location of the cables in line with the available locations in the building. The wireless technology destroys all such beautiful concepts, and a new arrangement can and will be made according to the needs of the specific application.

What we also see is that new directions are not necessarily coming out of the blue. Interactive displays were already in existence on the single-user computer. But they were bulky and expensive, only to be bought for specific purposes. The case of large-scale storage is similar, according to the seminal book by Christensen [32]. This leaves the mobile area where it is clear how many ownership problems we still have to solve. More about that later in Chapters 5 and 6.

1.5.2 Commercial Clouds

The cloud will require access to networked high-performance computing systems. Current clouds generally are blade-based servers and will require high-speed interconnection communication paths. It is possible for users to pursue potential opportunities with Google, in particular relative to their Google Code Labs and their open Android software. Similar activities exist within Microsoft, Yahoo!, Intel, IBM, and Amazon.

In our notion of the cloud, in addition to data collection, analysis, and reporting, the emphasis needs to be on providing increased safety and trustability to the various sensory network data-collecting elements. Programs would need to introduce new elements to cloud computing, notably the need to assess the integrity of the collected data elements from the sensory network. Up to now the assumption has been that all data are reliable. When making real-time decisions from collected data, this will not always be the case, just like every Internet website is not always trustable.

Key to the notion of cloud computing, in addition to the presence of server farms, will be the availability of low-cost thin-client processors. In this case, thin-client is defined in the sense of low capabilities, memory, and application software. A thin-client will also be sufficient in a sensory network, since most of the compute power will be expected to be in the cloud. Of course, the cloud is not confined to the server farms, as any thickening of the clients may in due course lead to local clouds as well. This comes to bear when we discuss software migration issues in Chapter 2.

Let us take the mobile phone as a simple, though extreme, example. This device has grown from a simple audio-processing unit to a unit that also handles video, text, and data. Though the applications are growing in complexity, the question may be asked whether this is the only reasonable development path. We suggest here that the mobile phone needs not be more than a connectivity manager. Reaching other mobiles in a timely and well-behaved manner is a primary concern. Where data transfer is of little importance, tools can run everywhere and on-line server support becomes an option.

- Lemma 1: With improving technology, more complex tools can be embedded locally.
- Lemma 2: With improving connectivity, less complex tools can be admired in the cloud.

These lemmas give the design conflict. With more local power, better algorithms can be applied locally. However, the "more than Moore" observation clearly tells that the algorithmic needs grow faster than the processor improvement. On the other hand, where connectivity improves it becomes more attractive to have such algorithms on another platform, optimized for speed rather than for power.

The situation of interest is where Tool_1 is embedded on one mobile device and communicates with another (not necessarily mobile) device where Tool_2 provides additional computational power. The end result will always be communicated back to the mobile device for final viewing. Or as an equation:

$$\text{Execution} = \text{Tool}_1 + (2 \times \text{transfer}) + \text{Tool}_2$$

The tool execution time is exponentially dependent on the algorithmic complexity, while the basic speed is given by the platform clock. Let us assume a factor α difference in basic execution time between the platforms (server $= \alpha$.PC $= \alpha^2$.mobile with $\alpha < 1$) and a factor β for the scaling of algorithmic complexity. Then our equation becomes for a task that is divided in a part γ for Tool_1 and a part $(1 - \gamma)$ for Tool_2:

$$\text{Execution} = \alpha.\gamma^{\beta}\,\text{Task} + (2 \times \text{transfer}) + (1 - \gamma)^{\beta}.\,\text{Task}$$

In this model we take transfer as a given, though the size of the data need is clearly dependent on the quality of the generating tool. The question when a job

can become faster to executing some part on a server can be answered from the equation:

$$\alpha \text{ Task} < \alpha\gamma^{\beta} \text{ Task} + 2\text{transfer} + (1-\gamma)^{\beta} \text{ Task}$$

If we assume that $\beta = 2$, then we find $\gamma = 1 - 2\alpha/(1+\alpha)$, which checks that for α approaching zero, γ approaches 1. For a reasonable value of α, say 0.1 for hosting on a notebook, the equation finds a cross-over at $\gamma = 0.8$, or for hosting on a server with $\alpha = 0.01$, the crossover is at $\gamma = 1$. This means that the notebook already promises a speed-up, while for the server any load reduction more than the overhead in transfer will work.

1.5.3 Server Farms

Central, at the highest level, to the cloud computing concept is the establishment of strategically distributed server farms [33] optimized for compute density and power consumption. Large server farms have been established by companies such as Google, Microsoft, and IBM. Since power consumption is an important issue for the large computing facilities, proximity to cooling opportunities such as rivers, as well as to cheap power such as by hydro-electric dams, has been crucial.

The server farms are generally constructed out of homogeneous clusters of commodity processors packaged in racks. To minimize the cost of facilities and their infrastructure, the racks have been assembled in containers. The containers are self-contained in power, cooling, and cabling. Sun Microsystems' Blackbox is a good example of this kind of packaging. Curiously, the box also contains a cadre of sensor nodes (Sun SPOTs) used for measuring acceleration (in transport), temperature, humidity, light, and location.

The use of containers facilitates also the quick and efficient assembling of a server farm; no specific building of facilities needs to take place, except for cooling, power, and cabling. The original use for the server farms was to speed-up web searches, the use of the same facilities for on-demand computing was a consequence of its capabilities and also of its utilization levels.

1.5.4 Network Extenders

In the wireless communication scheme of operations, some level of concentration of data (Fig. 1.8) from the respective sensor end-leaves might need to take place. The concentration takes place in low-power radio systems, known as femtocells (or hotspots). They plug into a residential broadband connection and enable mobile subscribers to use their existing mobile cellular handsets to access both data and voice services. Femtocells would enable optimum quality mobile services inside buildings.

Fig. 1.8 Femtocell becoming mainstream within 3G[15]

At this level of processing, use can be made of multicores/tiles processing elements. High core/tile-count systems have been developed by IPFlex, Ambric, Tilera, Stretch, MathStar, PicoChip, ElementCXI, PACT, IBM, Intel, ClearSpeed, SUN, AMD, Stream Processors, Stretch, Broadcom, Cavium Networks, CISCO Systems, and ElementCXI. It is of interest that in addition the boundary between "chunky" (at the ALU level) Field-Programmable Gate Arrays (FPGAs) and multicores/tiles systems has becoming less clear, so much so that they both are being marketed as multicore processing devices.

PicoChip[16] is a fabless semiconductor company developing high-performance chips for wireless infrastructure applications. Devices from picoChip contain roughly 300 processor cores and are targeted for 3G cellular and WiMAX base stations. In the picoChip design approach, the individual processor cores are kept relatively simple, to simplify the programming and the distribution of the processes.

The assignment of processes to processors and the patterns of the interprocessor communication are fixed at compile time: no bus contention issues remain to be resolved during the actual execution. Each processing element relies primarily on its local memory to implement the functionality mapped on to it; when local memory is insufficient, access to the external SRAM is under the control of the programmer.

In addition, the debugging tools provide graphical visibility into the entire chip. These simplifying assumptions and factors make developing and debugging of its complex applications manageable, despite the large number of processors involved. PicoChip's picoArray technology has been applied to the development of the picoXcell local network extender base station.

---

[15] Baines, R., "The Need for WiMAX picocell and Femtocells," www.wimaxworld.com.cn/picoChip-WiMAX-Femto.ppt.

[16] "PicoChip: Defying the Odds," BDTI (Berkeley Design Technology, Inc.), December 13, 2006.

1.6 Cloud-Sourcing

After the cloud computing paradigm becomes common practice, new business models and techniques will be introduced. They generally will benefit from the centralized aspects of cloud computing, including the database and software issue centralization, thereby creating high customer efficiencies. In addition to data collection, analysis, and reporting, the emphasis needs to be on providing increased safety and trustability to the various sensor network data-collecting elements. In addition, the new business models that will be created will have to take into account new aspects such as responsibility for cloud tools update, application version control, the physical and secure location of data and computation, etc.

1.6.1 Business Models

In the current business model, users acquire application software packages and maintain all data locally. The new cloud-based model makes the user into a "thin-client," only creating data and consuming reports. This change will greatly influence the business model for many current software packages vendors.

So in principle, the suppliers would have to move from a "transactional" relationship with the customers to a tailored "service" relationship [34]. In cloud computing many forms of service can be provided from centralized locations. Included would, of course, be updates and upgrades. The ultimate benefits to be gained from cloud computing depends whether or not one or more of the following concepts can be translated into viable commercial propositions.

1.6.1.1 Hardware-as-a-Service

Virtualization (as described in Section 2.1.1) represents a form of Hardware-as-a-Service. Virtualization could also include notions of load balancing and reconfiguration of resources. Grid computing has that same flavor of making resources available specifically for the requested computation.

It would be advantageous for a particular set of critical computations, to be able to request say 200 processing nodes, charge the expense to your credit card, and to have the nodes available and accessible in say five minutes. The computation will still need to be properly laid out for the 200 processing nodes for maximum benefit. Cloud computing will make HPC-like (High-Performance Computing) capabilities economically accessible from the home, the industry, and the university.

1.6.1.2 Enterprise and Utility Computing

Enterprise computing is a pre-cloud computing notion of on-demand utilities and computer system usage. Many current CAD tools have been provided on a per-use basis. Again one of the benefits is an automatic access to fully supported and up-to-date software. The billing will be provided for the time used; there will also

be charges for data base storage. In particular for CAD tools configuration and generation, control will need to be provided.

1.6.1.3 Software-as-a-Service

The initially defined and obvious benefit to be gained from cloud computing was the use of software to be executed on a third-party (in the cloud) computer system. No software for local execution would need to be purchased; the required software would be centrally maintained for correct operation. Centrally, a form of version control would also need to be provided.

The advantages of software-as-a-service will dramatically increase when the specific computations requires specific configurations and/or high performance, performance not necessarily available at the user's node or network. It would be unrealistic, in the long run, to expect increased local capabilities at lower price and lower power consumption.

1.6.1.4 Games-as-a-Service

A natural extension would be to make the locally popular software for games to become centrally available. In addition, the games could then also have multiple participants. The individual gaming nodes would be more like thin-clients instead of little supercomputers. Some level of graphics processing might still be done locally at the gaming node, possible in a multi-core/tile GPU (Graphics Processing Unit) configuration.

1.6.1.5 Systems-as-a-Service

When working in the cloud, subnets could be isolated for subsystem execution, not necessarily fixed in size and/or location. This approach would also benefit from any heterogeneity of alternate computational configurations present in the cloud. In addition to updating/upgrading of software in the cloud, it would also be possible to expect a variety of computational architectures to be available in the cloud.

An example is the neuromorphic simulation engine as designed as part of the 6th Framework "Fast Analog Computing with Emergent Transient States" Integrated Project FACETS. It is based on mixed-signal ASICS [35] brought together on wafer-scale to provide realistic simulation of biological neural systems, taking full redundancy into account.

1.6.1.6 Networks-as-a-Service

In addition, subclouds could become an available network, kind of in the distinction of an Intranet from an Internet. A corporate Intranet or private cloud could be set up in this fashion. One of the reasons for an Intranet or a private cloud would be the need for security and safety, assuming that those have not as yet be developed for generic global cloud computing systems.

1.6.1.7 Infrastructure-as-a-Service

The infrastructure[17] would need to have the desired configuration, as well as the application software that would be domain specific. The infrastructure will include all aspects of computation, as well as for communication and storage.

1.6.2 Cloud Economics

The described integration, in addition to many benefits, will also introduce new requirements. We need to deal with those new requirements (such as safety, security, and integrity), as well as develop new "disruptive" business concepts (software-, security-, gaming-, every-thing-as-a-service). These developments constitute a completely new business paradigm, gone are the days of sole-source software packages, e.g., Office.

The economical model for cloud computing is based on the actual use of third-party resources. This represents a change in regards to the ownership and responsibilities for the resources. Currently, the users generally use their own (acquired) resources. The user also buys the application software package and makes themselves available for future corrections and updates of the software. Already it is hard to enforce the corrections and updates of the acquired software, the presence of maliciously obtained botnets illustrates that fact. A fully corrected and up-to-date updated user community would have prevented this undesirable situation.

The notions of local "ambient intelligence" will transition to a more global notion of "collective intelligence." That also means that the data will be in the cloud. When the collective data resides in the cloud, the responsibility for the data will also be in the cloud. Cloud management will be responsible (at a charge of course) for the resiliency of the data by the proper protection mechanisms, as well as by the proper physically distributed redundancy.

When the management of application software clearly resides in the cloud, as a consequence, the responsibility for version control (i.e., the timely adaption of the new version and therefore also the support of older versions) will also be in the cloud. This is one of the price(s) to be paid for as a result of cloud-centeredness. Access to the data in the cloud will need to be carefully managed. Again this is one of the price(s) to be paid for. Also once the database is centrally located, synchronized access control needs to be provided. The cost model needs to contain charges for the computation, as well as for some of the price(s) indicated above. That is it should contain charges for provided maintenance, security, and access control, among others.

---

[17] Definition of Infrastructure (Merriam-Webster Online Dictionary): 1: the underlying foundation or basic framework (as of a system or organization), 2: the permanent installations required for military purposes, 3: the system of public works of a country, state, or region; also: the resources (as personnel, buildings, or equipment) required for an activity

As a result of the open nature of cloud computing, and of its centralized locations of computing, collocation of work facilities will no longer be required. IT (Information Technology) will become a local repair of thin-clients and the support of version-control in the cloud. More work can actually be done from home or work centers. Pure work and innovation will be performed at the various work centers or locations, support in tools will be provided in the cloud. This will also mean that additional work will be provided in the cloud in support of the tools.

1.7 Internet-of-Things

The Internet-based IP address for a device needs to be a unique number. The development of sensor-based systems, in which each sensor will have an individual IP address, will greatly benefit from the forthcoming upgrade of the underlying Internet protocol from IPv4 to IPv6.

The next-generation Internet will connect to thermostats and lights, cars, environmental monitors, factory machines, medical devices, and billions of other digital-ready things (Fig. 1.9). The Internet of Things could contain mobile phones that read bar codes on everyday objects; businesses that track goods via radio frequency identification (RFID) tags, and building or home automation systems on smart-utility grids that cut energy use.

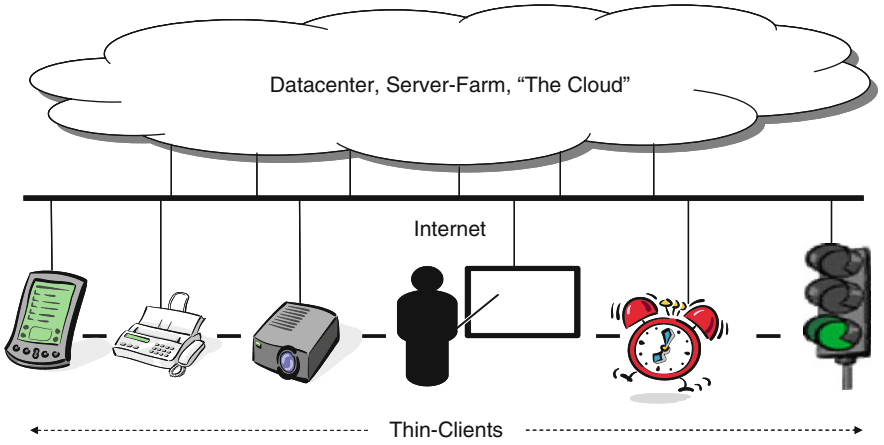

Fig. 1.9 Datacenter is the computer

The European Commission has announced plans to support the development of interconnected networks formed from everyday objects with RFID tags embedded in them, the so-called “Internet of Things.” Again, the change from IPv4 to IPv6 will facilitate this dramatical increase in separately addressable items.

Companies such as IBM, Cisco Systems, and Google have realized that for "Internet of Things" to be successful, standards for objects linked to the web need to be agreed upon. The CISCO-led IPSO (Internet Protocol for Smart Objects) Alliance has been formed to speed up these developments.

The starting point for the cloud notion, as described before, is a large "mainframe"-like centralized enterprise-computing facility (the server farms) connected through the Internet to a large set of thin clients (the things). In this "cloud computing" architecture, the data and services principally reside in massively scalable data centers that can be universally accessed from any connected devices over the Internet. Most services (Apps) will be provided in the cloud, some (see Chapter 7) will be most efficiently provided in a local "sensory" cloud.

1.7.1 Very-Thin-Clients

Plug computing is an example of thin-clients architectures, such as are envisioned in the ultimate cloud computing environment. They have enough capabilities to handle the local users' interface operations, as well as the interfacing into the cloud. Again at this level multi-cores/links processing elements can be applied, however, only if within power limitations of the design.

The Marvell Semiconductor's SheevaPlug is a recent low-cost entry into the world of so-called plug computing. It features a Kirkwood Series SoC with an embedded Marvell Sheeva^{TM} CPU core running at 1.2 GHz. This device connects to the network using Gigabit Ethernet, offers desktop class performance, and can be used to replace a PC-based home server for many applications. Peripherals connect using the included USB 2.0 port.

Thin-clients [36], such as the described SheevaPlug and the Hot-E device from ThinLinX, are second-generation devices similar to the first generation of so-called dumb terminals, as originally sold by Sun Microsystems (now Oracle). Interestingly, one of Oracle's products is their Sensor Edge Server, which provides for the channeling of sensor-based (such as RFID) information back to the higher-level infrastructure for management of the data.

1.7.2 Android-in-the-Cloud

Android is a software platform and operating system for mobile devices based on the Linux operating system, developed by Google and later the Open Handset Alliance. The Open Handset Alliance is a consortium of 34 hardware, software, and telecom companies (including among others Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, Sprint Nextel, and NVIDIA) devoted to advancing open standards for mobile devices.

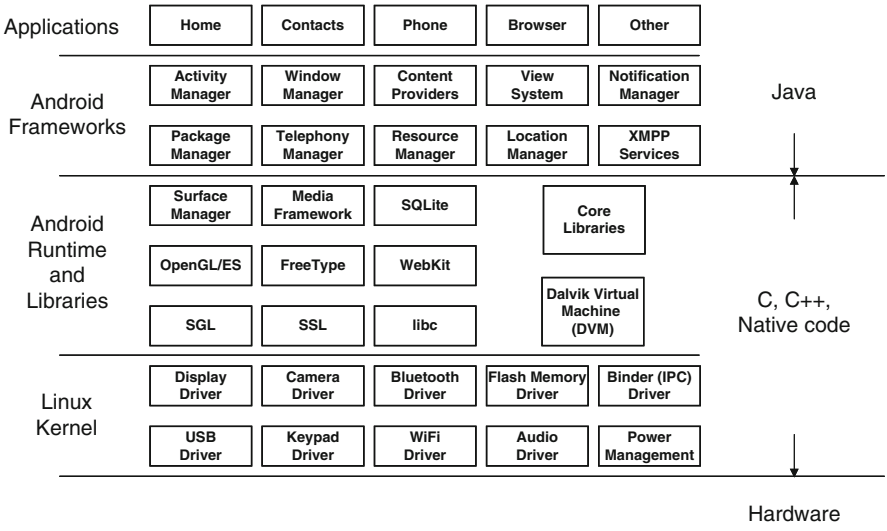

Fig. 1.10 Android architecture [37]

Software radios with the Google's Android operating environment (Fig. 1.10) have implemented the thin-client for sensor data access. The beauty of the Android-in-the-Cloud approach is that the communication infrastructure all the way to the Internet is freely provided, which leaves only the (sensor-based) application development to the user.

Android is open software and comes with a library (http://code.google.com/android/what-is-android.html) of functionality. Its Android Core C/C++ Libraries contain in principle the following elements:

- System C library – a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices
- Media Libraries – based on PacketVideo's OpenCORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG
- Surface Manager – manages access to the display subsystem and seamlessly composites 2-D and 3-D graphic layers from multiple applications
- LibWebCore – a modern web browser engine, which powers both the Android browser and an embeddable web view
- SGL – the underlying 2-D graphics engine
- 3-D libraries – an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3-D acceleration (where available) or the included, highly optimized 3-D software rasterizer
- FreeType – bitmap and vector font rendering
- SQLite – a powerful and lightweight relational database engine available to all applications

1.7.3 Sensing-into-the-Cloud

The introduction of IPv6 and the subsequent avalanche of IP-identified devices will result in a vast richness of sensors, data, information, and intelligence. The already maturing "cloud computing" concept allows for the integration of data collection networks into a combined cloud computing/ambient intelligence environment. The notion of "ambient intelligence" is based on the development of information/awareness spread over a distributed sensor environment. Classically, this is performed on the mainframe; but in the ubiquity of the sensors and especially their networks, this can also be supported by swarming, the forming of a local cloud. To bring this point to the fore, we will introduce "polytopol computing" [19], a concept whereby the notions of "cloud computing" and "ambient intelligence" sensor networks are merged (Fig. 1.11).

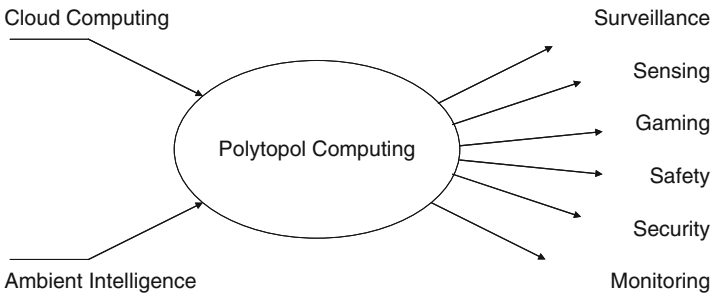

Fig. 1.11 Integration of "cloud computing" and "ambient intelligence"

In polytopol computing, the relationship between the sensors and the central computers will be similar to those in "cloud computing": a set of thin clients (sensors) and a "software-as-a-service" focal point (the directing/collecting central computer). Another way to look at the concept is from the computational perspective. The network takes a large number of nodes that can each contain one or more cores, supporting multiprocessing both in the nodes and in the sensor network.

In order to accelerate innovation in the area of multicore computers, the UC Berkeley RAMP [38] program has developed an FPGA-based standard research platform (the BEE3, (Berkeley Emulation Engine version 3)) for evaluation and testing of large multicore systems and especially its software development environment. The program originally started by the definition (after analysis [39]) of the initial seven so-called "dwarfs," a form of Model of Computation. After that initial study, the list was augmented with six more for a total of thirteen computational structures, somewhat domain-specific. A new approach to code optimization and generation, so-called Auto-tuners, is to produce optimized C code to be compiled for that particular computing structure. Similar as in the 1960s "Fixed-Plus-Variable (F + V) Structure" studies [40], the RAMP ParLab studies program will determine its computational structures from analysis. In the RAMP case, the high-value

application domains of personal health, image retrieval, hearing, music, speech, and parallel browsing.

Our concept of "polytopol computing" is derived from the notion that both in the multicore central processor (the cloud) as well as in the distributed sensors (the thin-clients) network topological aspects have been pursued though with differing orientations. In multicore system development, software is related to SystemC-like "Models of Computation," hiding details of the procedural interconnect from the application code. In distributed sensor networks, such as found in ZigBee-based networks, a relatively small set of basic interconnect patterns (star, mesh, and cluster tree) are generally applied. Figure 1.12 illustrates that both in the multicore central processor (the cloud) as well as in the distributed sensor (the thin-clients) network topological aspects have been pursued though with differing orientations.

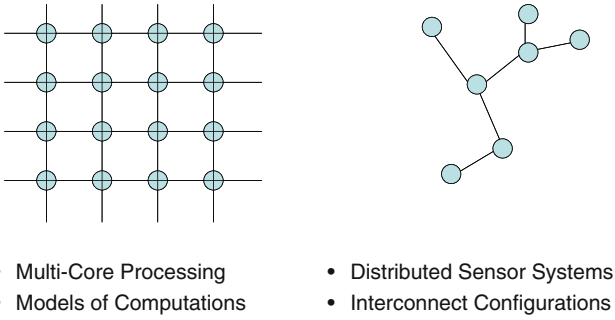

Fig. 1.12 Dualism in polytopol computing between multi-core processors and distributed sensor systems

We expect research steps [41] similar to the RAMP program to be performed in order to select a practical set of network topologies, starting with and including the three sensor-collector networks described by Spaanenburg [19].

The notion of polytopol computing is derived from the area of sensory networks. Multiple unattended sensors can be distributed over an area of interest in a subset of operational (poly-topological) configurations, while a (directing/collecting) computer system will (manage) traverse the sensor network to harvest the results. The central (multicore) computer will collect and analyze the data for subsequent updates of the network and/or transmission of the external actionable data items. Integral to this concept of "polytopol computing" is the partitioning of labor between "collector" and "sensor" nodes.

- "Collectors" provide functions such as a knowledge integrator, awareness collector, situation displayer/reporter, clue communicator, and an inquiry-interface provider. Knowledge integration will provide functionality as generally provided in high-performance processing systems including sophisticated artificial intelligence, multidimensional analysis, genetic algorithms. This capability generally is deemed not to be realizable on a network of sensor-processing nodes or at least not within its compute/power restrictions. Still the question remains how much

can already be done locally. The collector provides integration of global awareness to the "touching-the-elephant" sensor nodes. The collector also provides inquiry/display interfaces for out-of-sensor-net communication.
- "Sensors" provide functions such as anomaly detection (only communicating singularities, not continuous observation). They are generally powered or self-powered, amorphous (not on a grid) with generation-and-attrition, field reprogrammable, and sensor plug-and–play-able. The sensors, in principle, should provide no continuous "Big Brother" observations; they should, as much as possible, be limited to anomaly/change detection.

In addition to data collection, analysis, and reporting, it is important to emphasize the need for providing increased safety, integrity, and trustability to the various sensor network data-collecting elements. Oberheide [42] describes an innovative technique for virtualized in-the-cloud security services for mobile devices, a Security-as-a Service approach for sensor-based thin-clients (see Section 5.5 for more details).

The concept of "polytopol computing" is highly applicable in security applications where actionable intelligence is required. Technology will need to be included to access working status of network elements, including self-test, tampering, adversity spoofing, and maliciousness all the way to "Byzantine Generals" network analysis [43]. It is extremely important to access the integrity of the collected data, in addition to collecting the data itself. This level of security technology is a missing element in many currently applied sensor networks (see Section 6.2 for more details).

In polytopol computing the abstraction comes from the use of topology. This separates the partitioning of labor between collector and sensor from design and construction of them. Future polytopol computing applications stress the point that, by topology, programming of over 1000 cores, over 1000 sensors, and over 1000 computers needs to become the same. It points out that what becomes important is not the location but the relationship between the elements, and this means division of labor. Since it is a top down approach, it needs the functions of the software to become migratable [44]. In addition, aspects such as organization, topology, software distribution, client–server relationships will be more important than the individual node or central processor programming.

1.8 Summary

In this chapter we have traveled from embedded sensors to large-scale networks. Where the past is governed by a limited amount of and therefore application-specific parts, we have argued that in current days ubiquity and generality rules. We are rapidly moving toward swarms of sensory devices, where the computing power may be local but where a supporting (server) farmer may also be at hand. The consequence of all this is that a function is not closely coupled to a platform. In fact,

the hardware is becoming invisible and the software is becoming elastic. This is the microelectronic equivalent of Heisenberg's Law: a microelectronic function has no place to stay and no time to loose.

In the remainder of this book, we will further elaborate on this theme. We will look at clouding concepts to provide freedom in the system, and related measures to enhance safety and security at the same time. But first we will look in the next chapter further into this notion of migratable software in mobile (sensory) based network systems.

References

1. Collins RR (September 1997) In-circuit emulations: how the microprocessor evolved over time. Dr Dobbs J. Available at www.drdobbs.com
2. Eggermont LDJ (ed) (March 2002) Embedded systems roadmap 2002, STW Report 2002 06 06-03, Technologie Stichting STW, Utrecht, The Netherlands
3. Lee EA (October 2006) Cyber-physical systems: are computing foundations adequate? NSF workshop on cyber-physical systems: research motivation, techniques and roadmap, Austin TX
4. Kurzweil R (1999) The age of spiritual machines. Viking, New York, NY
5. Moravec H (2000) Robot: mere machine to transcendent mind. Oxford University Press, New York, NY
6. Joy B (2000) Why the future does not need us. Wired 8(4):238–246
7. Hammerschmidt C (April 2007) Car designs on fast track. Electronic Engineering Times Europe (online), 16 Apr 2007.
8. Amin M (September/October 2003) North America's electricity infrastructure: are we ready for more perfect storms? IEEE Secure Private 1(5):19–25
9. Waugh A, Williams GA, Wei L, Altman RB (January 2001) Using metacomputing tools to facilitate large scale analyses of biological databases. In: 6th Pacific symposium on biocomputing (PSB2001), Kamuela, HI, pp 360–371
10. Weiser M (September 1991) The computer for the 21st century. Sci Am 265(3):94–104
11. Reeves B, Nass C (1996) The media equation: how people treat computers, television and new media like real people and places. CSLI Publications, Stanford, CA
12. Bondarenko O, Kininmonth S, Kingsford M (October 2007) Coral reef sensor network deployment for collecting real-time 3-D temperature data with correlation to plankton assemblies. In: International conference on sensor technologies and applications (SENSORCOMM2007), Valencia, Spain, pp 204–209
13. Zhang L, Malki S, Spaanenburg L (May 2009) Intelligent camera cloud computing. IEEE international symposium on circuits and systems (ISCAS), Taipei, Taiwan
14. Ljung E, Simmons E (2006) Architecture development of personal healthcare applications, M.Sc. Thesis, Lund University, Lund, Sweden
15. Gomez RB (April 2007) Hyperspectral remote sensing: physics, systems, and applications. In: 14th Annual mapping, imaging, GPS and GIS user's conference (GeoTech07) lecture notes, Silver Spring MD
16. Baranoski E (July 2006) Visibuilding: sensing through walls. SAM 2006, Waltham, MA
17. Kercel SW, Baylor VM, Labaj LE (June 1997) Comparison of enclosed space detection system with convential methods. In: 13th Annual security technology symposium, Virginia Beach, VA
18. Stankovic JA (October 2008) Wireless sensor networks. IEEE Comput 41(10):92–95
19. Spaanenburg H, Spaanenburg L (August 2008) Polytopol computing: the cloud computing model for ambient intelligence. In: 1st international conference and exhibition on waterside security (WSS2008/Structured Session F), Copenhagen, Denmark

20. Gleick J (December 1996) A bug and a crash: sometimes a bug is more than a nuisance. New York Times Magazine, 1 Dec 1996
21. Guo T-H, Nurre J (July 1991) Sensor failure detection and recovery by neural networks. Proc IJCNN Seattle WA I:221–226
22. Kalinsky D (August 2002) Design patterns for high availability. Embed Syst Program 15(8):24–33
23. Nijhuis JAG, Höfflinger B, Neusser S, Siggelkow A, Spaanenburg L (July 1991) A VLSI implementation of a neural car collision avoidance controller. Proc IJCNN Seattle WA I: 493–499
24. Simmons E, Ljung E, Kleihorst R (October 2006) Distributed vision with multiple uncalibrated smart cameras. ACM workshop on distributed smart cameras (DSC06), Boulder, CO
25. Sinai Ya.G (1970) Dynamical systems with elastic reflections: ergodic properties of dispersing billiards. Russ Math Surv 25(2):137–189
26. Therani MA (December 2008) Abnormal motion detection and behavior prediction. MSc thesis, Lund University, Lund, Sweden
27. Gershenfeld N (1999) When things start to think. Hodder and Stoughton, London
28. Schmidt A, Vandevelde W, Kortuem G (2000) Situated interaction in ubiquitous computing. In: Conference on human factors in computing systems. The Hague, The Netherlands, p 374
29. Dobson S, Sterritt R, Dixon P, Hinchey M (January 2010) Fulfilling the vision of autonomic computing. IEEE Comput 43(1):35–41
30. Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, Lee G, Patterson DA, Rabkin A, Stoica I, Zaharia M, (February 2009) Above the clouds: a Berkeley view of cloud computing, Electrical Engineering and Computer Sciences. University of California at Berkeley, Technical Report No. UCB/EECS-2009-28
31. Lohr S (2007) Google and IBM join in cloud computing. New York Times
32. Christensen CM (1997) The innovator's dilemma, when new technologies cause great firms to fail. Harvard Business School Press, Boston, MA
33. Katz RH (February 2009) Tech titans building boom. IEEE Spectr 46(2):40–43
34. Prahalad CK, Krishnan MS (2010) The new age of innovation. McGraw-Hill, New York, NY
35. Schemmel J, Meier K, Muller E (2004) A new VLSI model of neural microcircuits including spike time-dependent plasticity. In: Proceedings international joint conference on neural networks (IJCNN2004) Budapest, Hungary, pp 1711–1716
36. Vance A (2008) Revived fervor for smart monitors linked to a server. New York Times, October 13, 2008
37. Thompson T (September 2008) The android mobile phone platform. Dr Dobb's J. Available at www.drdobbs.com
38. Patterson DA, (February 2006) Future of computer architecture. Berkeley EECS Annual Research Symposium (BEARS2006), University of California at Berkeley, CA
39. Asanovíc K, Bodik R, Catanzaro B, Gebis J, Husbands P, Keutzer K, Patterson D, Plishker W, Shalf J, Williams S, Yelick K. (December 2006) The landscape of parallel computing research: a view from Berkeley, Electrical Engineering and Computer Sciences, University of California at Berkeley, Technical Report No. UCB/EECS-2006-183
40. Estrin G (May 1960) Organization of computer systems – The fixed plus variable structure computer. In: Proceedings Western Joint IRE-AIEE-ACM Computer Conference (WJCC), San Francisco, CA, pp 33–40
41. Spaanenburg H (May 2008) Multi-Core/Tile polymorphous computing systems. In: 1st International IEEE conference on information technology, Gdansk, Poland, pp 429–432
42. Oberheide J, Veeraraghavan K, Cooke E, Flinn J, Jahanian F (June 2008) Virtualized in-cloud security services for mobile devices. In: Proceedings of the 1st workshop on virtualization in mobile computing (MobiVirt'08), Breckenridge, CO, pp 31–35
43. Lamport L, Shostak R, Pease M (July 1982) The Byzantine generals problem. ACM Trans Program Lang Syst 4(3):382–401
44. Spaanenburg H, Spaanenburg L, Ranefors J (May 2009) Polytopol computing for multi-core and distributed systems. In: SPIE symposium on microtechnologies for the New Millennium, Dresden, Germany, 736307, pp 1–12

Chapter 2
Software-from-the-Cloud

Connectivity has become a major concern in this time where computing platforms move from executing scientific algorithms to providing services through consumer products. Ideally, data may become a commodity that can be tapped anywhere from a communication infrastructure, just like water and energy come "out of the wall." But unlike in the case for transport systems for water and energy, there is for data communication already more than a limited amount of dedicated suppliers to serve all customers. Anybody can be a supplier and anybody can be a consumer. This poses problems in establishing a quality of service that cannot be pushed into the future.

The dedicated supplier in the classical transport system sets the upper limit for the network capacity. Knowing the amount of clients and their nature (house, office, and so on), the required capacity can be calculated for each (fixed) topology of the network. This situation has drastically changed with the introduction of alternative energy sources, where consumers can make themselves independent of the general supplier. Problems have arisen when such consumers are allowed to deliver back to the energy grid at unknown times in unknown quantities. Today, the centralized control model has started to crack and the need has come for the "intelligent energy grid" to pro-actively handle the local transport problems. This is not a new problem for digital communication.

As pointed out by Carr [1], the development of computer systems has started off in a similar manner as that of electricity distribution. When the first computers were introduced, computing was a scarce resource that needed to be shared. The mainframe at the heart of the system allowed in-line access through dumb terminals using dedicated wires and dedicated outlets, serving hundreds of users. Then, in a continuous decrease of size and power dissipation, microelectronics created more intelligent terminals and eventually almost autonomous workstations. Subsequently, workstations became processors that could be easily moved, transported, and embedded, and the network connecting the equally intelligent devices grew in importance. In the meantime, the mainframe became the server, and servers became networked into clouds, creating a situation that clearly deviates from the classical generator-centered distribution model (Fig. 2.1).

L. Spaanenburg, H. Spaanenburg, *Cloud Connectivity and Embedded Sensory Systems*, DOI 10.1007/978-1-4419-7545-4_2,

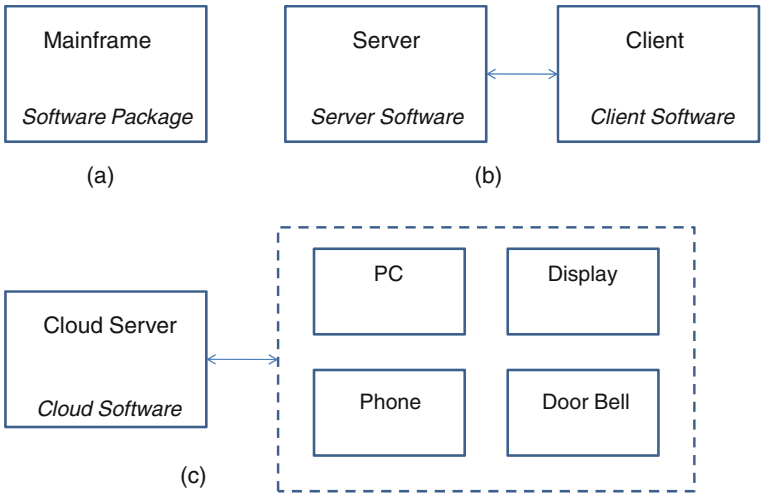

Fig. 2.1 From (**a**) Mainframe over (**b**) Client/server to (c) The embedded cloud

It seems, however, that the communication issue is not confined to data only. In audio, microelectronics has caused a similar movement. The dumb telephones have been replaced by intelligent ones that subsequently went mobile and started to collect more and more functions. We see the same trend in vision. The studio of old where video programs were made and broadcasted to dumb television sets is on the way out. The massive camera has grown into a consumer product and subsequently integrated into the mobile telephone and many more products. People are creating their own video snippets and make these available over the Internet.

The three ingredients of computer communication (data, video, and audio) are not only different in nature but also different in typical traffic density. On top of that, there are different means of transport. Connections between digital products can be made over many media: by air (OTA), by Bluetooth (OTB), by cable (OTC), and so on. The large variety comes from the history of microelectronic development, where the communication devices have originally been using analog circuitry, then moved to digital circuitry and are currently migrating to software.

This gradual transition is directly reflected in the history of wireless devices, where the presence of many standards to be handled by the same device spurred a multitude of standard-specific chips. The cost of this is especially killing for the baseband stations, where every new standard caused the need for a new infrastructure and therefore heavy investments. Moreover, it has caused customer churning, as a new infrastructure forces clients to buy new devices, and so on. Gradual transitions are needed as facilitated by configurable products.

The connections do not come at the same capacity and/or price. Wireless is by nature slower than wired, and a power-line is not a coax. Much depends on what is available, whether it is private or public, and so on. This is not a major issue for the basic interaction architecture, but the cost factor unmistakably creeps in when more

devices are connected into a distributed implementation of multimedia dissemination. Consequently, an application router is required to plan the message paths for the single interactive application.

We will describe how we will deal with software distribution and execution in sensors-in-the-cloud systems. Subsequently in Chapters 5 and 6, we will describe how to deal with new requirements such as safety, security, and integrity in cloud-centric sensory system applications. These cloud-based software developments, in particular in its location-of-execution aspects, constitute a completely new business paradigm.[1]

2.1 The Nature of the Cloud

Technically speaking, Cloud Computing is the historical consequence of the need to share software on a single platform for such purposes as:

- System Fault Sensitivity – running the application as a single instance of a SMP (Shared Memory Processor) Operating System can easily become a single point of failure.
- System Health – monitoring system properties like energy consumption and memory access conflicts can indicate the need for dynamic reconfiguration.
- System Maintenance – where most of the software costs is in maintenance, it pays to be seamlessly forward and backward compatible on the process level.

The enabling technology for Cloud Computing is Virtualization. From a software point of view, it all started in the sixties when IBM developed time-shared, multi-user computers. The aim was to let many users share unconsciously the resources of a single platform at a single time. All the software still had to be compiled for the target machine, and therefore had to be recompiled for each different platform. Emulation was a first step to make a machine look different, but did not work nicely in a multi-user environment because software could still not be scheduled freely. It helped, however, to update the firmware to fight firmware issues, and thereby sanitize the platform while still being able to run legacy applications in a separate batch.

With advances in CPU architecture came the on-chip support to simultaneously run processes with different rights on accessing the resources, for instance, system maintenance and user applications. The growing availability of software libraries forced the need to have differently compiled software processes collaborate on a single platform. From there the step was made toward process containers, each having their own flavor of operating system, and the cloud technology was born (Fig. 2.2)!

[1] "How to Plug into the Cloud," Special Report: Business Innovation, Information Week, December 8, 2008.

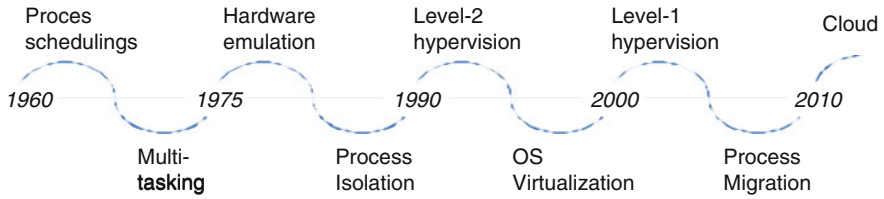

Fig. 2.2 Virtualization concept development timeline

The direct consequence of the ability to run software that has its own operating system support is that is has become IP (Intellectual Property). As it runs without need for recompilation or adjustments of the underlying hardware, it can be dynamically relocated to where it is needed. In the typical cloud, where many servers are available but each runs a separate energy bill, it pays to consolidate the system by moving it to an already active server with sufficiently remaining capacity. Every time an application is run, it may therefore be running on a different server with different resource qualities and thereby having different run-time characteristics.

We are concerned with the actual location of the software. Where the mainframe was the only place to execute software, in a network the focus has shifted. First the heavy computation has remained in the server, while the workstations did their share to support the users in short turn-around interaction. Gradually, supercomputing came in the hands of the user, and the mainframe moved into the cloud to do even larger tasks [2]. The user interaction has moved on from the workstation to any device that happens to be in the hands of a user. In other words, software got split into parts that communicate over the network while each being located at a convenient location. For the moment, this is still a static division and it can be argued that technology exists to realize a more dynamical configuration.

Two aspects are important for the cloud of servers to be meaningful. The first thing is the high-speed interconnect. Supercomputing qualities can also be achieved on the grid, where a number of processors can be put to work in parallel. But when the program cannot be fully parallelized, or when data have to be moved in large quantities, and/or at a high speed, the performance will rapidly drop. Bringing them all together with high-speed interconnect and a single locally integrated database will bring clear advantages. The second thing has to do with the fact that the cloud has to service everybody, and can therefore not be dedicated to a single operating system. Let alone that the server hardware cannot be allowed to impose constraints on the software that may be innocuously offered.

2.1.1 Virtualization

In the pursuit of flexible software organization, a number of phases can be discerned. The original form of flexible connectivity is the interrupt structure. An interrupt handler was placed at the top of the memory space, containing a table with pointers to

the routines that handled the specifics of data passing for each individually connected external process. It was a first attempt to create connectivity through the definition of addresses in the single integrated memory space and still serves nicely to connect arbitrary devices to a computer.

From the need to support multiple processes on a single computer has come the distinction between the local and the global memory space. The global memory is a single integrated view on the system, composed from the number of processes that each views only a local part. The distinction is hardware-supported through a memory management unit as it is called to virtualize the memory space: through a page table, local memory can be placed anywhere in the global memory. Still all processes are equally capable, which is changed when one of them is given the power to supervise the organization. In the supervisor mode the management can be changed, while in the user mode the management is operated (Fig. 2.3a).

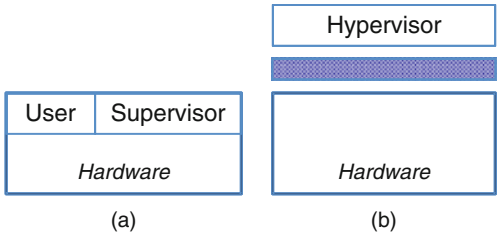

Fig. 2.3 The coming of age of the hypervisor

On top of the hardware lies conventionally the operating system. It operates directly on the underlying hardware, but masks the details away for the application programmer. A number of platforms can be supported using an abstraction of the underlying hardware, the virtual machine. This Physical-to-Virtual transformation can be done in software as well as hardware. Popular examples of a software technique on the compiler level are the P0 layer for Pascal compilation and the Java Virtual Machine (JVM) for executing Java program.

The original virtual machine was compiler-driven and did not provide a clear transparent view on the underlying hardware. It was therefore severely limited in its application. Already as early as 1967, IBM has shown how also the supervisor mode could be handled. When around 2000, Sun Microsystems used this potential to support multiple operating systems on the same hardware, it became known as the hypervisor or the Virtual Machine Monitor VMM (Fig. 2.3b). The hypervisor will check all fetched instructions and separate all that are either non-native or privileged for special treatment, such as emulation or simulation.

In the monolithic hypervisor all guest operating systems refer to the same hypervisor. This requires standardized interfaces between the guest and the underlying VMM like:

- A framework API for virtual management
- API for communication between the virtual machines
- Para-virtual operations to reach the underlying layers
- Mechanisms for load balancing and maintenance.

All ingredients of the operating system concerning the network communication, the storage hierarchy, the process scheduler, and the device drivers are placed in the supervised area. In other words, the hypervisor can in itself already be a real-time operating system (RTOS).

The monolithic hypervisor lacks flexibility. Because all the device drivers are in the supervised area, it cannot be extended for specific requirements of a guest. Therefore, the console-guest hypervisor moves the drivers upward, on top of the virtualization layer, so functionality can be added. This is the mechanism that allows adding drivers to Linux. The overall name for such variations with large and complex OS software underneath as shown in Fig 2.4 is hosted or type-2 hypervisor.

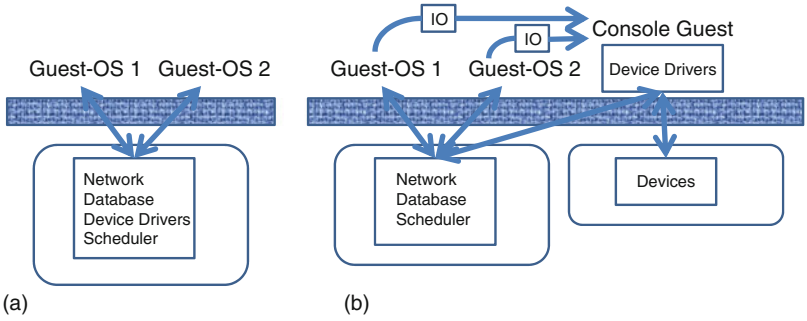

Fig. 2.4 (**a**) Monolithic and (**b**) Console-Guest hypervisor [29]

This also states the fundamental problem of the hosted architecture: the virtualization program is complex and has a large footprint. Hence it is difficult to maintain, or even worse easy to hack. Therefore, the next move is to limit the supervised part to the essentials, a microkernel rather than a kernel.

This is characteristic for the native or fully virtualized or type-1 hypervisor (Fig. 2.5). There is a small common part in the privileged area (called "bare metal" when it is considered to be absolutely minimal) and the rest is a virtualization process in the user area that will be instantiated per guest. The added advantage is that virtualization can be easily (and even dynamically) tuned for individual applications.

When such individual needs are brought in, the mechanism is called para-virtualization. This allows building virtualization on top of one another, each interface being optimally suited for optimal performance. The state of the art supports even real-time performance in a transparent platform. This being the case, hardware is by itself not a differentiation and a distributed sensory network is also multicore in sharing a software distribution problem. The static solution balances the load by optimally pre-assigning tasks to nodes. Such is known to work for parallelizable computational problems, but that is where it stops. The dynamic solution allocates processes to nodes when and where this comes to pass.

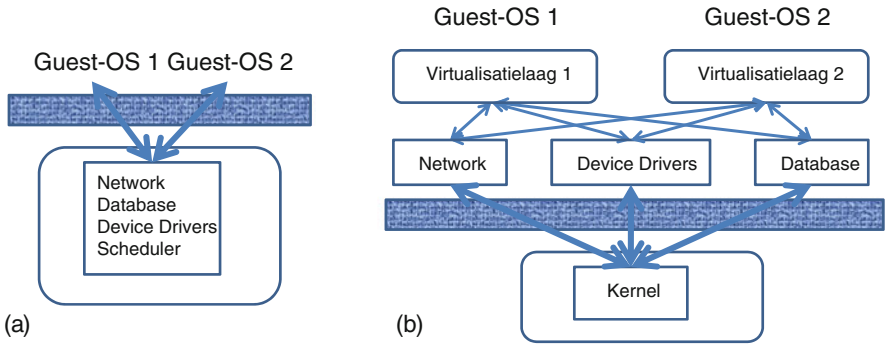

Fig. 2.5 (**a**) Monolithic and (**b**) Native hypervisor [28]

The primary use of virtualization is currently in the cloud of servers, where it facilitates the location-free execution of software and/or services. Customers need to be serviced on the available rather than the planned servers, while software will have differing needs of resources. On the other hand, customers will need to test and develop software with a guarantee on scaling the application onto the cloud later on. Apparently, there is more to a cloud than just a computer park.

2.1.2 Looking into the Cloud

Frequently, many people have the same question. Therefore, it pays to have the answer stored for future use. Some people have an occasional, specific question. Saving the answer for eventual use is a waste of storage space. It is much more efficient to re-compute the answer when the need arises. This dichotomy does not take the role of data into account. Where a question is frequently asked but with constantly new data, it still brings little to store the answer. This is, for instance, the case with the face recognition for border inspection. The question is simply: can this person be admitted to the country?

Though the question is asked constantly, the person to check is steadily different. Related is the fact that for complex situations the question cannot always be asked adequately. So it is required to support the users in their navigations through the universe. Long answers are not what they want. Needed are short adequate directions that do not need much reading or long education. This poses the cloud as an information market where many servers with very large data storage systems and a high-speed, high-capacity interconnect allow for non-everyday shopping at the comfort level of the shop-around-the-corner.

A typical workload for a server has both the application and the supporting operating system in a single "container" to allow a maximal freedom of dynamic configuration [3]. In general, three types of reconfiguration can be seen (Fig. 2.6). Workload isolation is the oldest variety and aims to have different containers to securely interact on a single platform. Consolidation occurs when workloads from

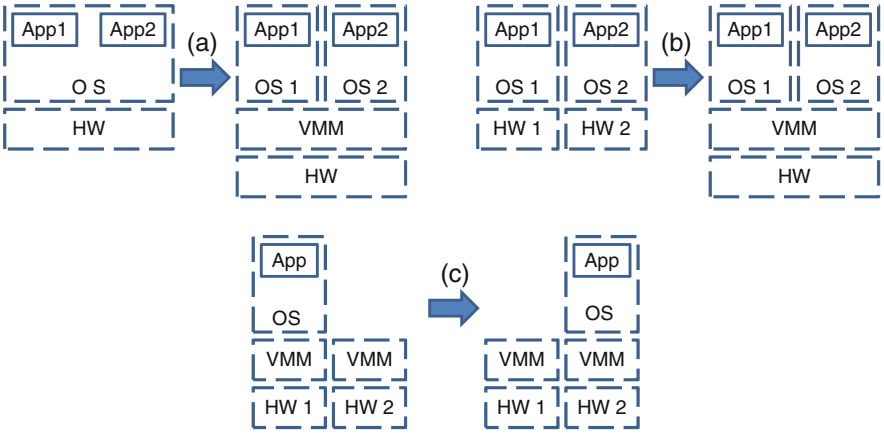

Fig. 2.6 Workload (**a**) Isolation, (**b**) Consolidation, and (c) Migration

different servers are first contained and then optimally configured onto the smallest amount of servers as possible. Of late, we also see migration, where a container is reconfigured to a different server because of utility considerations such as being faster, having better peripherals, and so on.

The access to the Cloud has a relatively low bandwidth. Similarly, the computational energy is centered on the cloud, not on the access. This gives the Cloud unlimited power in storage and computation, but is by the same reasoning hard to access. This should not be a problem if the Cloud means to provide a service, not just the result of a computation. This means that the Cloud reduces the required bandwidth by providing an abstraction. The user may query in loose terms on a concept. The cloud then computes several meanings of that query and returns some initial results. In turn, the user may improve his query on which the clouds terminates some searches and narrows down the potential meaning to continue computing in more promising directions. This continues till user and cloud start to agree on the rightly formulated query at the same moment as the right answer becomes available. This constitutes an "asymmetric function" similar to the ones that are applied in creating secure transmission.

A typical example is the CloneCloud (Fig. 2.7). In Chun and Maniatis [4], it is discussed how a mobile phone can be augmented by having a mirrored version in the cloud. Both operate at the same time; but when the phone has to perform a complex operation, the mirrored version will be considerable faster to get a result and send it back to the real phone. Consequently, the real phone can be kept light because it looks much faster than it actually is. The added advantage is that it will still work when cloud access is not available (or when one is not in a hurry).

Embedded systems in general cover a wide area. On one hand, there is a need for real-time response, and therefore the distances in the computational system need to be short; on the other hand, there is a need to be close to the application. In terms of virtualization needs, this means that we have a need for both hosted and native

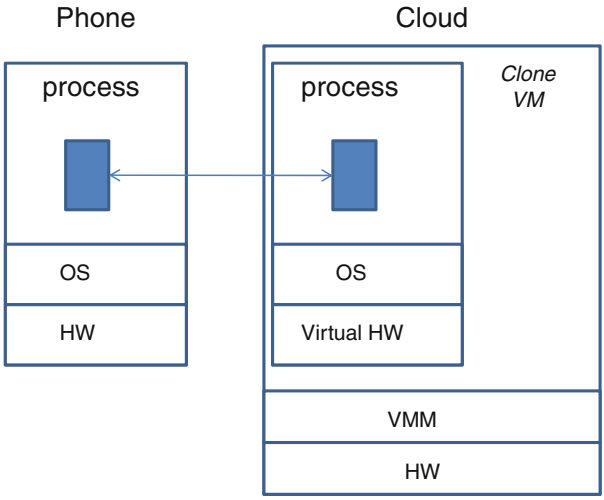

Fig. 2.7 The augmented phone

arrangements. Therefore, sensory networks will have to compromise in their system architecture.

Separating such needs to different abstraction layers, we arrive at sandwich structures as depicted in Fig. 2.8. In this way operating systems can run on varying platforms, but likewise applications can run on different operating systems. Orthogonal to this can libraries and non-CPU interface be virtualized.

A typical example is in Face Recognition, where the cloud can perform a typical database search well beyond the capacity of the real phone. Modern digital cameras have GPGPU (General-Purpose computing on Graphics Processing Units) hardware to handle such applications. The suggestion here is that the typical feature phone does not have to copy this example, but can become smarter by a proper cloud connection.

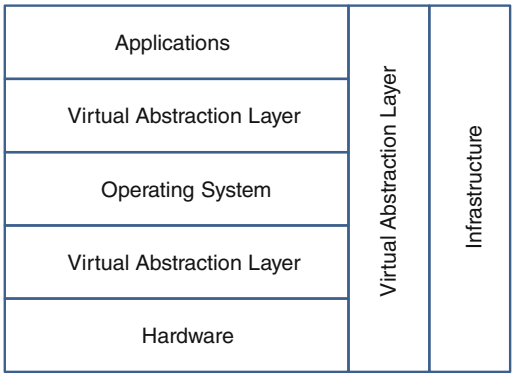

Fig. 2.8 Abstraction layers built through virtualization [29]

2.1.3 The Other Cloud

Another way to look at the history of computing is from the role of software. In the scarcity of resources, hardware and software are naturally together to create the (system) function. That does not really change in the server/client architecture. The server and the client software are still self-contained and an exchange format is defined to facilitate communication. The crucial change comes when computer peripherals split off from the workstations, taking their own processor with them. Gradually making them more intelligent creates the embedded computer: a device, which can compute but looks different. Most of the available computing power is not needed for the basic device operation, but can be used for applications using that device. Consequently, software is not an installed program that brings the hardware to execute the locally intended function, but any program that happens to be there.

Function migration makes programs look like hardware. In the past, hardware was the scarcely available component of a system taking one CPU, one mainframe, and several locally furnished programs acting on local data. Today while we are moving into the ubiquitous and pervasive hardware era, bandwidth has become the limiting factor and programs and/or data need to be moved to where they can be applied most effectively from the networked system perspective. One says that software is becoming "harder" and a connectable provider of system functionality. In other words, software is taking over the role of hardware as system component. This is the basic essence of the era that we are moving into, while combining cloud, multicore, and FPGA technology.

This puts emphasis on the network. It not only connects hardware but moreover connects software into functions. This connection-centric view is part of the more general scheme of polytopol computing, where functions are created over the network in varying topologies [5]. This is already being done to some degree. Harvesters go through a network while searching for information on the nodes. In ad hoc networks, data searches its way through the network to find a consuming server. Of late, the online game Guild Wars introduces a minimal bandwidth delivery system that allows clients to get content as they need it. Or more importantly, prior to the need for new content, it will already be pushed to the nodes during normal game time to be finished with downloading when the client needs it. This is a clear step away from the scheduler type of monolithic install and distribution scheme.

This brings us to the situation where we have the potential of large sensory nets with some hardware variety based on the available ports to the embedding world, but largely personalized by software. For instance, there are already 600 M telephones in use with broadband access. They exist in a situation where all kind of computing and display devices coexist. Freely connecting such devices to retrieve, transform, and display information such as pictures seems a reasonable next step [6]. Such can be seen as a next cloud structure, and it is of interest to see whether they can use the same technology. In Want et al. [6], this is illustrated with the following case description.

Fred and Sally are visiting their friend Joe's house when the topic of Sally's recent vacation comes up. Instead of just showing them pictures on her mobile device, Sally displays a collection of her favorite picture of Joe's wall-mounted flat-screen TV, using her mobile device to advance slides.

2.2 Connection-Set Architecture

The cloud grows toward a distribution network around a domain-specific (Software-as-a-Service) supercomputer. For the cloud of mobile devices the trend is to build applications on such platforms by moving software to where it is needed instead of having it pre-installed. Both make the Connection-Set Architecture (CSA) to a key ingredient, assuming that measures are present to allow migrating software to "glue onto" judicious nodes.

The CSA works on and with abstract connections with the goal of making functionality in software movable. For this to work, we see the need to replace the scheduler way of moving software to that of the skeleton injector. The skeleton injector works on abstract connections between software, and is a master in the sense that it tells the nodes what they should do and not the other way around. The injector can be a slave to another injector, and it takes this role regularly when spreading onto the nodes in a network. In the spreading of the injector and functionality onto a network, complexity is lowered by its abstraction of the networks topology.

2.2.1 Software Connects

In the mainframe era it was important that the source and caller knew, where and how, the topology and protocol of the networks was arranged. With limited resources, software had to be ported taking the requirements for format conversion into account. The goal was to connect two points, the network was a means to accomplish this, and hence the act of network routing has been of high importance. Today the situation is considerably different: the network is everywhere, connecting two particular points is not special, and formats have been standardized. As the network goes into software, the hardware gets abstracted away. It does not matter for the function whether it is executed at the source, the caller, or somewhere in-between, nor does it matter what route and/or protocol it took or followed.

For example, a cell-phone might seem to make software mobile; however, it is just a container for software. For a user in a ubiquitous and pervasive environment, this behavior does not infer what should move along with the user. Is the mobile phone really the container for software or a platform to support incoming "*functionality + data?*" Apparently, we do not need to differentiate between data or computation, but can look primarily at the flow of information and its static/dynamic properties. We are considering the dimensions of the function migration problem as

it exists on the cloud, sensor, software, or (multi)-core level in a CPU. However, first we will focus on the connections to make software movable and only later on the network itself.

A connection is not that of a wire, or between two nodes as in hardware; instead, it is finding the equivalent in software. The goal here is to make software movable by introducing a dynamic connection. For this to happen, there are two things that have to change in the software. The first is that the connection cannot be fixed at compile time. The second is that how the software is divided and structured have to become realized from a perspective on being able to move.

The Connection Set Architecture abstracts the connection with software attached to it. The CSA has the effect of separating the functional intention of the software from its practical use on an unknown source. This means that software can be handled as packages, while the placement of packages forming a complete system is taken care of on a network level. On the individual connection level, what is sent and what is presented, can be controlled without regard to individual design and implementation.

This is the same relation as collector and sensors (see Section 1.7.3) in supervised sensory networks [7]. The sensors give information, but do not know how that information is going to be used by a collector. In software, the collector may be the user interaction interface and the sensors the functionality behind it. Today the sensors not only provide information but also how the information should be presented and interacted with. That functionality should be separated and in a world of migration come from the user, attached as a sensor.

We intend to break this (collector-sensor) dependency by introducing a higher level of autonomy, reuse, and maintainability. In the CSA approach the users (sensor) prescribes how things are to be presented to the user interface (collector) and what is presented is something the functionality (sensors) contributes. In a software example, the user is the control of flow of a program, the user interface is the API, and the functionality is the logic and information. This means that the API does not have to be static in terms of functionality and interaction. In addition, when using an API or standard, parts of the standard can be dynamically overridden without side effects.

This is in a way reminiscent of the “injection” concept introduced in the Java environment for how a virtual machine moves program to other hardware (Fig. 2.9). It moves source programs to be compiled and executed elsewhere. This was an attractive technology for Internet purposes to begin with. Remember that the basic point is to force all the software onto one node, but there are more nodes and better ones to inject into.

2.2.2 Skeleton Injector

In the classical multicore software development, software parts are statically analyzed and assigned to a node, whereafter the parts assigned to a node are brought

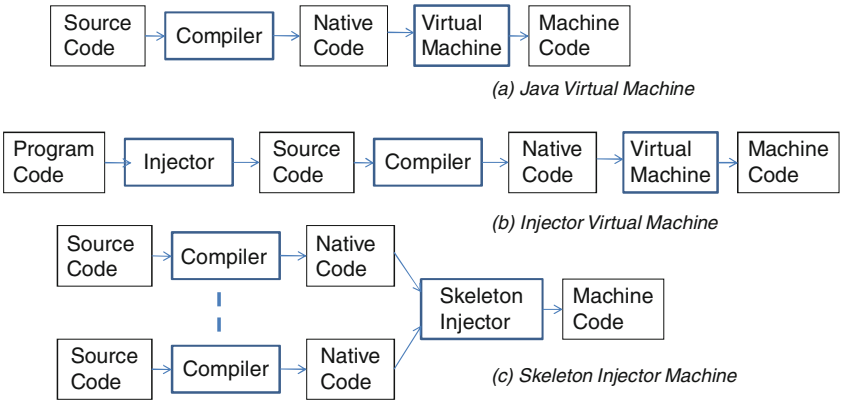

Fig. 2.9 Java versus skeleton injector

into a scheduled program order. In one embedded CPU with one core or sensor, there is a small performance penalty, not much, for the scheduling. As we scale to more cores and sensors, the penalty is rapidly increasing and the quality gets less. Handling smaller program parts would help, but the scheduling time rises even more. On the other hand, handling larger program parts decreases the scheduling time, but at the expense of quality. Evidently scheduling is not a good thing for many cores, and we move over to allocation.

We propose to replace the scheduler with an injector type of resource handler (Fig. 2.10). The injector encapsulates the parts of the software that are still handled by the scheduler. These packages we call compartments, and the injector technology works on the level on how to place and connect these. The compartments are created during design time to the level as needed. Compartments can be of many different sizes, but they do not split computations and functions. This means a slight performance loss for a single-core CPU, but at the gain of shorter development time and lower maintenance cost.

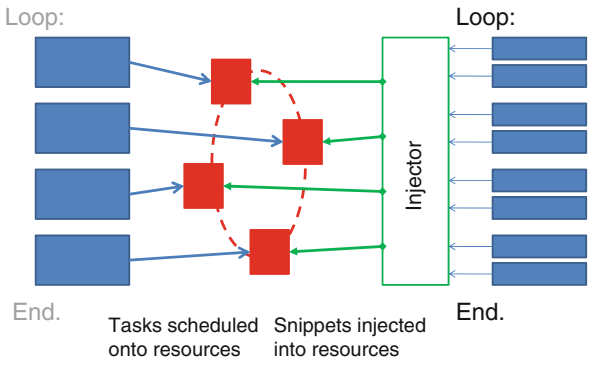

Fig. 2.10 Scheduling versus injection

Connection Set Architecture makes it possible to move compartments of software. This sets the focus on a method for describing and configuring, a. k. a. a language of how to connect these compartments and set them up from storage to run-time usage. The skeleton injector, on the other hand, is the supporting technology that makes it possible not to have to think on a point-to-point basis of how things are connected or allocated onto resources. In the Instruction Set Architecture (ISA) the register was the container, which made it possible in an abstract way to handle, describe, configure, and move information. In the CSA, the compartment is that kind of container.

The skeleton injection system when arranging work can be compared to that of IPFlex' DAPDNA and that of the TRIPS multicore architecture [8], where a large computational part is put onto several cores and, when a call is made to the outside, a new computational part is loaded onto the cores. Here a statically binding regarding size of cores and computational part is made. However, the outside call from a computational block is where an injector connection point is inserted. Note, however, that it is not from the computational block that the call will come in the injector system. A core or a cluster of cores does not decide themselves what package they compute.

The injector is a master and, if added in a modular fashion, an optimizer. An injector is not called from the resource to ask for more data. This is logical since how can a computer resource know what to ask for unless it is tied into the software and consequently would not be movable software. What to send to a core may not be related to the logic of the program, but to that of the system, such as running on low-power versus normal-power level.

2.2.3 Synchronization

The interaction between the user and the machine is built from intentions that are incepted and interpreted to create actions, making the cause and effect understandable. We give this more attention because it shows the cognitive aspects of system utilization. Such creates the state-based behavior that mimics an intelligent interplay between man and machine and hereby allows to model intuitive reactivity. We use here the name "3I model" (Intention, Inception, and Interpretation).

The interaction between a user and his/hers gadget usually involves text, audio, and video. Conventionally, they are handled by separate applications, but the human user does not differentiate. Commands and directions can be incepted from all these information sources. But the simultaneous handling of these sources over synchronized channels gives the opportunity to direct in a natural way, by "point and tell." So telling "call," while pointing at the familiar person on the screen, has the same effect as telling "Pierre," while pushing the call button.

Apparently, this discussion still leaves wide open what the means of pointing is: a cursor, a finger, a stick, or even a remote. As the medium is the message, pointing reveals the intention, but revealing the intention is by itself already a form

of pointing. Hence, "gazing" would avoid the need to point by stating implicitly the focus of attention. On the classical desktop, all icons are shown on one level in two dimensions. Focus will need to raise the amount of levels (by sliding a beam or a loupe) or of dimensions (mapping on a corridor or globe).

This leaves room for interpretation. Words do not make a sentence, though a sentence uses words. When the man stumbles and cries "hell," his telephone should not make a connection with his mother-in-law. The context of the sentence and especially the spatial and temporal annotation gives meaning. Consequently, the telephone should be situational-aware, both in time (I just woke up) and in space (am I home?).

The 3I's span the control space of the basic "Interaction Architecture" [9] (Fig. 2.11). The popular mouse click (left to point; right to operate) is gradually being reconsidered. The fixed order of manipulations on a single-entry device requires training as it does not allow for variations; in case of the mouse it even gives risk to stress-based health risks.

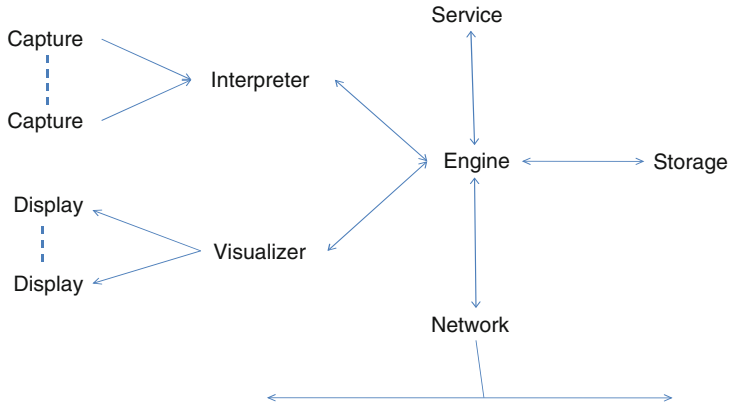

Fig. 2.11 Basic interaction architecture

There is evidently much to be gained by providing flexibility over multiple-entry devices, but it also raises new problems in channel synchronization. Streams come together into a single cognitive expression that can be checked on correct syntax and meaningful semantics. This seems overdone for the usual "show-and-tell" application [4], but is required on entering the cloud.

2.3 Software Migration Concept

In advanced computing on clouds, and even swarms or flocks, increased abstraction comes from the use of network topology. This separates the partitioning of labor between data collector and sensor from their design and construction. Future cloud computing applications stress the point that, by topology, programming of over 1000 cores, over 1000 sensors, and over 1000 computers needs to become the

same. It points out that what becomes important is not the location but the relationship between the elements, and this means division of labor. Since it is a top down approach, it needs the functions of the software to become migratable [5].

Function migration makes software movable and the effect of this is that functionality becomes more stable. Function migration is a top down approach and can be applied to the degree of gradualism needed. At the same time, function migration is bottom up on the scale global/local. This duality from the program point of view comes from the fact that we are not working on the network nodes, but with the network onto a topology. This means that location is a concern of the network and function that of the program.

Software migration started when the amount of hardware grew explosively and software had to migrate. Function migration need comes in at a time where the devices attached to hardware are growing in number, ubiquitous, and pervasive. What this means is that the hardware is forming a network around us and this changes the demands on software. It also blurs the conventional difference between hardware and software, and questions what should be fixed or movable (somewhat similar to Estrin's Fixed-Plus-Variable Structure [10] notion). A view of the future may be that of a stable user moving through a changing community.

This calls for a new view on software and hardware, and what we propose is a Connection Set Architecture (CSA). The connections set architecture works with abstracts connections with the goal of making functionality in software movable. For something to move some things need to be more concrete, "harder," more stable, and the "roads" and placeholders need to become more flexible and abstract.

2.3.1 Software Components

Program structure is an important factor in programming. A good structure makes for a readable, and therefore less faulty, program. Especially persons like Edsger Dijkstra have emphasized this aspect, notably from the viewpoint of being mathematically clean. The early programs had little structure, being dominated by the lack of program space and the fact that the division of the program space in pages had little to do with the individual algorithm. Edsger Dijkstra has fought a lifelong war against the "goto" statements that were more or less forced by the paging structure.

Lack of structure has also led to global variables. Especially languages like FORTRAN advocated a general variable space (usually called IAUX) that could be used for different purposes through the "OVERLAY" construction. Some disciples of Dijkstra still have his famous first commandment written on the wall: "Thou shall not overlay!" Later this became an accepted statement in the interpretation that the global variable is the root of most (if not all) evil.

A first attempt to bring some structure in the program that reflects the algorithm rather than the platform is the Object-Oriented Programming (OOP). The mental model is that objects are defined as assemblies of data and operations on such data with relations between the objects just like sentences are composed of verbs and nouns in a language. Such programs tend to have problems in the real-time area.

Component-based programs (CBP) do not make such assumptions on relations between the objects. In a sense they look more like hardware, where well-defined modules (say components) are glued together as demanded by efficiency considerations. Similar to hardware, the ruling paradigm is "separation of concerns." The software component is a software package that encapsulates a set of related data plus programs. All software parts are created as components, and are therefore modular and cohesive.

Similar to hardware, components communicate through ports defined on its interface. When it provides functionality to another component, it is given a provider interface, and consequently the provider component becomes encapsulated into the consumer component. On the other hand, the consuming component is given a consumer interface, stating which provisions are required to be encapsulated.

From the reliance on interface specifications comes the potential for components to be substituted. Components that deliver functionality according to the same provider interface can be substituted in the consuming component to update and/or otherwise improve functionality. As a general rule of thumb, component B can immediately replace component A, if component B provides at least what component A provided, and requires no more than what component A required.

Reusability is an important characteristic of a high-quality software component. A software component should be designed and implemented so that it can be reused in many different programs. It takes significant effort and awareness to write a software component that is effectively reusable. The component needs to be:

- fully documented
- more thoroughly tested –
  - robust – with comprehensive input validity checking
  - able to pass back appropriate error messages or return codes
- designed with an awareness that it *will* be put to unforeseen uses

Today, modern reusable components encapsulate both data structures and the algorithms that are applied to the data structures. It builds on prior theories of software objects, software architectures, software frameworks, and software design patterns, and the extensive theory of object-oriented programming and the object-oriented design of all these. It claims that software components, like the idea of hardware components, used for example in telecommunications, can ultimately be made interchangeable and reliable. Consequently, the practice of component-based programming seems extremely well suited to develop networks of software running on arbitrary hardware, e.g., in AUTOSAR (more about that in the next section).

One of the most significant benefits of the application of component software is that it allows for the separation of application programming ("assembling applications from parts") from component programming ("creating reusable building blocks"). These activities require different skills and different tools. It also allows for alternative implementations of components without changing or rebuilding applications. It therefore allows flexible and dynamic deployment and upgrades of

software at granularities smaller than applications. Application can also run without acceleration, with no middleware present.

As an example we refer here to SystemC[2]: a language to describe hardware and software designs at multiple levels of abstraction, with a C++ class library to standardize system modeling: clocks, concurrency, interrupts, hierarchy, hardware data types. In addition, it is a lightweight simulator kernel and actually an executable parallel system programming environment.

In SystemC, application functionality is expressed as a collection of computational components performing computations on locally based data. These components are conceptually inherently parallel in that they can be run in parallel along with any other component. The correct order of operation for different components is managed through signaling and data communications. In SystemC, both signaling between applications and data exchange are managed through expressive communications primitives that can emulate a broad variety of communication architectures, both abstract and real.

SystemC allows for a broad variety of Models of Computation (MOC) to be expressed. Abstract MOCs [11] capture high-level application functionality and promote compact program expressiveness and portability. Physical MOCs have the expressiveness to capture efficient algorithmic instantiations that have excellent performance on the target platform. SystemC supports, as a minimum, distinct Models of Computation based on the view of time, type of process activation and process communication methods such as Kahn Process Networks (KPNs), Static Multi-Rate Dataflow, Dynamic Multi-Rate Dataflow, Communicating Sequential Process (CSP [12]), and Register Transfer Level (RTL).

The UC Berkeley RAMP program [13] (see Section 1.7.3) has defined a total of 13 computational structures; they are somewhat domain-specific forms of Models of Computation. In addition, they envision the development of so-called Auto-tuners, which will produce optimized C code to be compiled for the respective computing structures.

Still other work is continuing, often in a direct binding of software and hardware. A typical example is the Singularity project [14]. Where previous attempts are based on a careful management of the memory space, this project attempts to remove such restrictions and allows arbitrary chunks of "Data + Program" to be located at arbitrary locations in the virtual memory. It seems that such initiatives still have a long way to go.

2.3.2 Introducing AUTOSAR

The Automotive industry is the flagship of industrial automation [15]. With the rising number of complex functions, the development of in-vehicle electronics is becoming increasingly broader in scope and more complicated. The automotive

[2] http://www.systemc.org.

market has produced the Controller–Area Network (CAN-bus) that subsequently revolutionized industrial automation. The CAN-bus is a vehicle bus standard designed to allow microcontrollers and devices to communicate with each other within a vehicle without a host computer.

Technologies like CAN have been originally developed for automotive applications, where the demands on real-time behavior, reliability, and (of course) life-long cost-of-ownership are not only high but also have to be achieved in a complex supplier hierarchy. Competition is intense and the cost of research is high. This has made the automotive industry the first to embrace a precompetitive collaboration model. AUTOSAR (Fig. 2.12) is such a project.

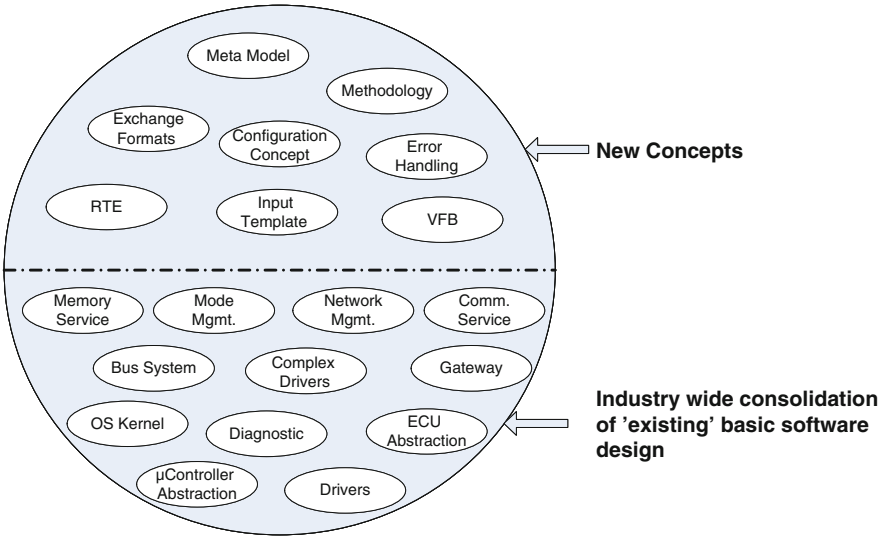

Fig. 2.12 AUTOSAR technical scope

Typical for the automotive industry today is the concept of co-development. Where in the classical industry the contractor sets out contracts for the parts it needs, co-development calls for a shared responsibility. This has come together with a sharp decrease in the number of suppliers for a typical car. So a car is developed between the car manufacturer and a small number of major suppliers, who together take the responsibility of a safe car.

Founded in 2003, AUTOSAR (AUTomotive Open System ARchitecture) is an open and standardized automotive software architecture, jointly developed by automotive OEMs, suppliers, and tool developers, with the vision to develop and establish a de-facto open industry standard for automotive E/E architecture, which will serve as a basic infrastructure for the management of functions within both

future applications and standard software modules.[3] The partnership consists of the following core members: the PSA Peugeot Citroën, Toyota Motor Corporation, Volkswagen, BMW Group, Daimler AG, Ford Motor Company, General Motors and automotive suppliers Bosch, Continental, and Siemens VDO. Volvo is a premium member of the AUTOSAR partnership.

The AUTOSAR partnership was set up with the objective of defining an open standard for an automotive E/E (electrical/electronic) architecture. The motivations behind this architecture are:

- Management of E/E complexity associated with growth in functional scope
- Flexibility for product modification, upgrade, and update
- Scalability of solutions within and across product lines
- Improved quality and reliability of E/E systems,

and its goals are:

- Fulfillment of future vehicle requirements, such as availability and safety, software upgrades/ updates, and maintainability
- Increased scalability and flexibility to integrate and transfer functions
- Higher penetration of "Commercial off the Shelf" software and hardware components across product lines
- Improved containment of product and process complexity and risk
- Cost optimization of scalable systems

The consortium has a vision of open systems that are "plug-and-play" compatible, where components from one supplier can easily be replaced by those from another supplier. The potential benefits of this methodology are clearly also attractive in the domain of embedded system: reduction in development time and cost, improved quality, and specialization of expertise.

2.3.3 The AUTOSAR Case

The following gives a short introduction to AUTOSAR standard to illustrate some of the concepts treated before (Software migration aspects will be treated in Section 2.4.2). The basic idea of AUTOSAR is simple: have a standard frame for the communication between hardware platforms, shield it by an appropriate Run-Time Environment with some utilities and placeholders for memory usage and process connectivity and you have an open system, where application software from different suppliers can be arbitrarily mixed.

[3] AUTOSAR Documentation and Specifications: www.autosar.org.

The AUTOSAR[4] concept is based on a type-2 hypervisor using modular components with defined interfaces.[5] The current system in AUTOSAR comprises the following elements (Fig. 2.13):

- ECU: Electronic Control Unit, the physical hardware
- RTE: the Runtime Environment
- MSW: the Main Software, a combination of
  - BSW: the Basic Software, building on RTE, to provide some general utilities
  - SWCs: the Software Components, base entities of any software assembly
- CSW: the Complementary Software, a fabricator- and model-specific SWC lifted out of the MSW

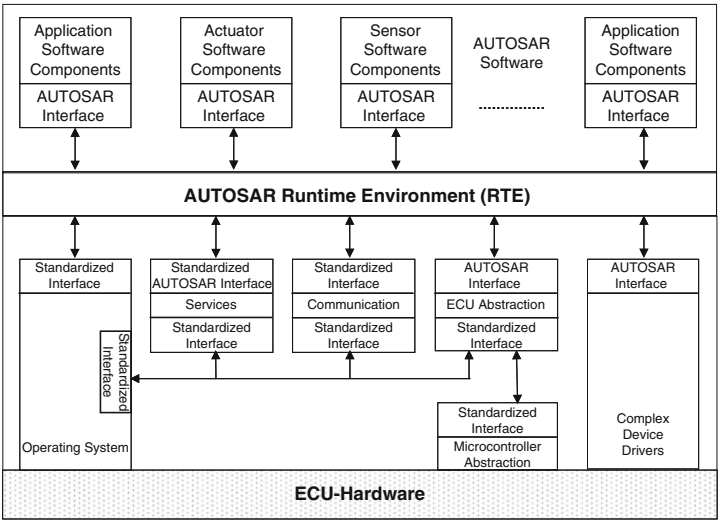

Fig. 2.13 The AUTOSAR ECU software architecture

2.3.3.1 Basic Software (BSW)

BSW is a standard layer, which provides the overall functionality of the AUTOSAR infrastructure (SWCs and RTE) on an ECU and is essential for running the functional part of the software; however, it does not fulfill any functional job itself. The SWCs do that. The BSW component is divided into the two sectors (Fig. 2.14): standard and ECU-specific.

[4] AUTOSAR Documentation and Specifications: www.autosar.org.

[5] Volvo Internal documents on current project and AUTOSAR, 2008.

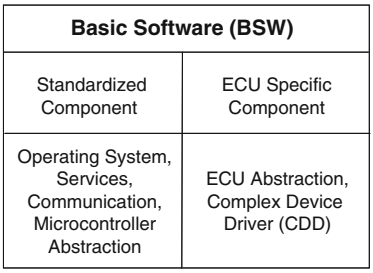

Fig. 2.14 BSW Components

The Operating System (OS) is an essential factor as it provides a common architecture for all the vehicles domains. The standard OSEK OS (ISO 17356-3) is used as the basis for the AUTOSAR OS,[6] containing features such as priority-based scheduling, protective functions at run-time, real-time performance, static configuration, scalable and hostable on low-end controllers without dependence of external resources. It provides the system services and covers diagnostics protocols, flash and memory management, and non-volatile random-access memory (NVRAM). It consists of communication, memory, and system services.

Such services are a group of modules and functions, which can be used by modules of all layers. Examples are Real-Time Operating System (which includes timer services), Error Manager and Library Functions (like CRC, interpolation). It is the system communication and covers the communication framework, e.g., CAN, LIN, FlexRay, MOST, and also I/O management and network management. Basic Software components or other higher software interacts with the hardware through a special hardware-specific lower software layer known as Microcontroller Abstraction, which restricts the direct access to the microcontroller register from the software. It manages the microcontroller peripherals and provides the components of BSW with microcontroller-independent values. It includes Digital I/O, Analog/Digital Converter, Flash, Watchdog Timer, Serial Peripheral Interface (SPI), Pulse Width Modulator/Demodulator (PWM, PWD), EEPROM, and Inter-Integrated Circuit Bus (IIC).

The ECU-specific component provides a simple software interface to the electrical values of any specific ECU to higher-level software. The reason for this is to make it independent from all underlying hardware. It is a loosely coupled container, where specific software implementation can be placed.

2.3.3.2 Software Components (SWCs)

AUTOSAR SWCs, also known as the "Atomic Software Component" is the implementation of the parts of the functionality of the automotive application and are the fundamental building blocks of AUTOSAR systems. Atomic here means that the

[6] AUTOSAR Documentation and Specifications: www.autosar.org.

SWC cannot be distributed over several AUTOSAR ECUs. Each SWC encapsulates part of the functionality of the application. It may be a small, reusable piece of functionality (such as a filter) or a larger block encapsulating an entire application. The "SWC description" describes both external and internal behavior of the SWC in standardized XML format.

There are two types: (a) Application and (b) Infrastructural components (Actuators and Sensors – hardware-dependent) (Fig. 2.15). The internal behavior of SWCs is mainly RTE events, runnables, exclusive areas, inter-runnable variables, per-instance memory sections, multiple instance support, etc. The application SWC consists of interconnected SWCs, which encapsulates certain functionality of the application. The point to be noted is that these SWCs (in source code only) are fundamentally independent from the ECU, the memory location, etc.

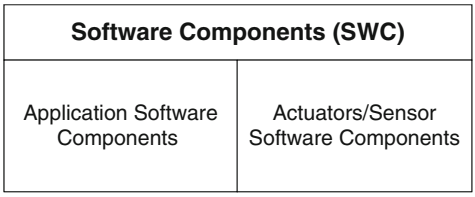

Fig. 2.15 Types of SWCs

An actuator/sensor SWC is a specific type of SWCs used for sensor evaluation and actuator control, and it is dependent on hardware components; here hardware implies specific sensors or actuators, independent of the designed μController and ECU HW. The Sensor/Actuator SWCs are located above the RTE for integration reasons (standardized description). Relocatability is restricted because of their strong interaction with raw local signals. Examples of tasks of a Sensor/Actuator component are switch debouncing, battery voltage monitoring, DC motor control, lamp control.

The structure of an SWC is described on three distinctive levels:

- SWC type describes the external behavior (Fig. 2.16) by means of
  - Provided interfaces (What the component gives: P-Ports)
  - Required interfaces (What the component needs: R-Ports)
- SWC internal behavior that describes how the software component behaves internally.
- SWC implementation, e.g., resource usage.

Communication models describe the AUTOSAR communication patterns. It supports two kinds of communication patterns: Sender Receiver and Client Server:

- A sender–receiver interface defines one (or more) data elements that are sent/received, which can be simple types (integer, floating-point) or complex (array, record) and each data element can be sent (and received) independently. Components can use "1:n" and "n:1" communication.

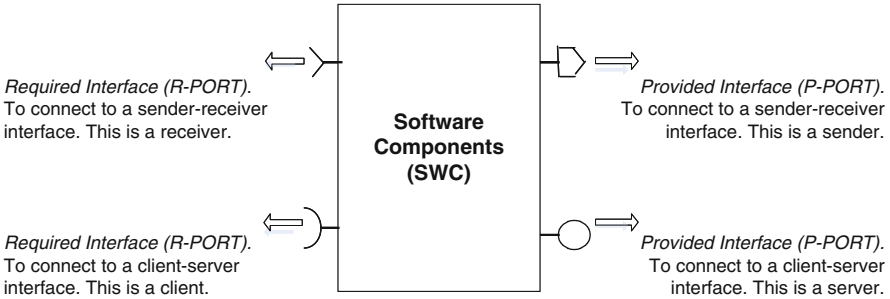

Fig. 2.16 Overview of SWC with different interfaces

- A client–server interface defines one (or more) operation and parameters for operations plus their type and direction. The functions are provided/used for each operation. Each operation can be invoked independently. Components support "n:1" communication. So, clients can not have multiple servers for the same call.

The RTE provides the implementation of these communication patterns. For inter-ECU communication, the RTE uses the functionalities provided by COM (Communication).

Each software component contains one or more runnables, which are threads of execution/control inside the software component and the place where programmer writes the code for implementation. Runnable entities are triggered by events, for example, periodically, when data are received, when a server is called (Fig. 2.17). Interaction between events and runnables is captured using an internal-behavior configuration in the AUTOSAR XML.

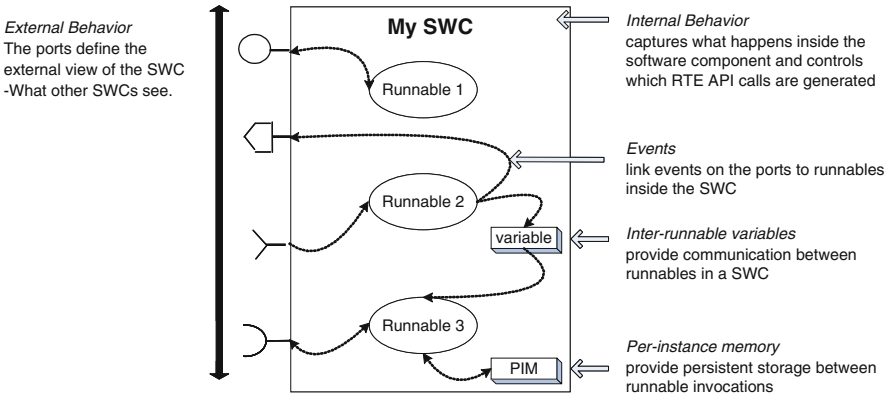

Fig. 2.17 Description for internal and external behavior

2.3.3.3 Runtime Environment (RTE)

The RTE is the heart of the ECU architecture. All the communication between SWCs and with BSW including the OS and communication services is carried out through the RTE layer. More precise, RTE is the run-time implementation of the VFB (Virtual Functional Bus) for a specific ECU, which is responsible for providing the independency from communication mechanisms and channels for SWCs. Each ECU contains an RTE. The communication interface of an SWC can consist of several ports (which are characterized by port-interfaces). An SWC communicates through its interface with other SWCs (whether that component is located on the same or on a different ECU) or with BSW modules that are located on the same ECU. Communication can only occur via the component's ports.

A port is categorized by either a sender–receiver or client–server port-interface. A sender–receiver interface provides a message-passing facility whereas a client–server interface provides function invocation. The RTE supports both types of SWCs where the source is available ("source code SWCs") and SWCs where only the object code ("object code SWCs") is available, the latter being important for IP protection purposes. The binary code of each SWC has a specific .xml file describing its interface.

The RTE is an implementation of the AUTOSAR VFB. The VFB provides the abstraction that makes SWCs portable and reusable. The RTE provides the encapsulation of the mechanisms required to make the abstraction work.

According to the specification of the RTE documentation,[7] it supports two modes of communication:

> *Explicit* – A component uses explicit RTE API calls to send and receive data elements.
>
> *Implicit* – The RTE automatically reads a specified set of data elements before a runnable is invoked and automatically writes (a different) set of data elements after the runnable entity has terminated. The term "implicit" is used here since the runnable does not actively initiate the reception or transmission of data.

The RTE header file defines fixed elements of the RTE that do not need to be generated or configured for each ECU. The relationship between the RTE header file, application header files, the lifecycle header file, and AUTOSAR SWC is illustrated in Fig. 2.18. It also shows how those files are included by modules implementing SWCs and, by general, non-component code.

The application header file defines the "interface" between a component and the RTE. The interface consists of the RTE API for the component and the prototypes for runnable entities. The definition of the RTE API requires that both relevant data structures and API calls are defined. The data structures required to support the API are defined in the RTE Types header file. This enables the definitions to be available to multiple modules to support direct function invocation. The data structure types

[7] AUTOSAR Documentation and Specifications: www.autosar.org.

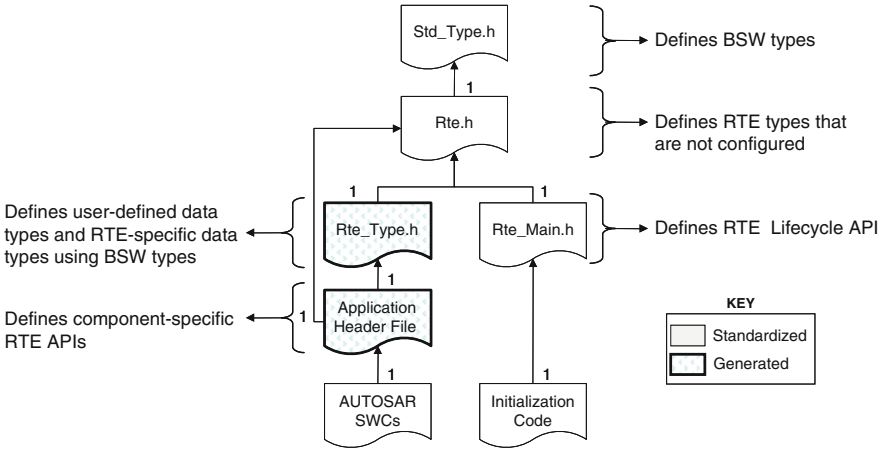

Fig. 2.18 Relation of AUTOSAR components to header files

are declared in the RTE Types file, where as the instances are defined in the generated RTE. The application header file defines the mapping from the RTE API to the generated API functions that are generated/configured for the component. Only RTE API calls that are valid for the particular SWC type are defined within the component's application header file.

The AUTOSAR types header file defines RTE-specific types derived either from the input configuration or from the RTE implementation. The generated RTE can include zero or more AUTOSAR data types created from the definitions of AUTOSAR meta-model classes within the RTE generator's input. The available meta-model classes are defined by the AUTOSAR SWC template and include classes for defining integers and floating-points as well as "complex" data types such as records. The RTE generator shall create the AUTOSAR types header file defining the AUTOSAR data types and RTE implementation types. Then the RTE implementation types include the Component Data Structure.

2.4 Topological Impact

Microelectronics has led to a continuous decrease in size and power dissipation, allowing for more intelligent terminals and eventually almost autonomous workstations. It then moved on in miniaturization and workstations became processors that could be easily moved, transported, and embedded. Simultaneously, the network connecting the equally intelligent devices grew in importance. This is the situation we currently find in sensory networks. In the meantime the mainframe became the server, and servers became networked into clouds. It is this situation that we address with polytopol computing.

In polytopol computing, the state-of-the-art is extended by distributing and connecting software to create functions, which bring nodes together over the network. This has a counterpart in multicore computing. In order to do this, we have to conceive a scheme that allows software to become connected on the network. The following case gives some illustration of what we envision.

> While watching television, the telephone rings. My wife calls. Which of the children's photo I like to have framed for the grandparents? As the home network trusts her, she accesses our photo repository and shows her selection in the corner of the TV screen. With my telephone I point to my favorite photo. My wife agrees and says that she will take it from here. This is the typical situation where the entry, storage and display devices in a communication are all different, and have to take on different functions. Where the wife starts to call, everything is on her device. Then she sends gesture interpretation to the TV and repository search to the Home Directory. Gesturing leaves a copy on her phone that follows my movements, so she sees what I select.

In other words, the virtual caller moves into the home network, communicates with me and leaves again. Clearly, such a scenario requires a careful design and a raised level of security.

With the requirement to move software in place comes the need for distribution. In the network we have a multitude of sensors and devices, and in hardware we get a network of cores. Polytopol computing brings these areas together by introducing an abstraction of the network topology, enabling computation to be executed on the network rather than on the nodes. It is here that the new field of function migration comes in and provides together with polytopol computing the infrastructure that enables flow of functionality on a topology. This hides the network structure and therefore the location of software functionality. Enabling technologies are the skeleton injector as a replacement of the task scheduler, the polytopol computing topology abstraction of the network, virtualization applied on the connection of software to software, and the introduction of a connection set architecture.

2.4.1 Homological Sensor Networks

An issue of effective coverage pops up when awareness is extended from the device to the network level. Let us take for example a network of vision sensors to supervise an area. Then it is required that the sensors cover the intended area without any holes. This is a maintenance issue, as the mechanical placement of cameras cannot be relied on. The ideal is then self-calibration. Two main situations can be discerned.

When all sensors should be able to see exactly the same object but from different angles, we have to ensure that all angles are covered. The object will be nonlinearly transformed in a way that is angle-dependent. Coverage can be ensured by analyzing this perspective.

In a typical WiCa system [16] (more details to be provided in Section 4.5), the foundation will be handled by the 8051 controller while the processing is performed by the IC3D intelligent vision processing chip. The application layer is available

on each camera, but only one of the cameras in the dynamically configured network will function as the cloud server. The cameras are coupled over a wireless ZigBee network and communicate observed data instead of images, therefore making an efficient use of the available bandwidth. The ZigBee-based operating system allows moving the cloud serving functionality to perform even in the presence of nonfunctioning cameras.

Similarly when all the sensors cover each a part of an area, their view has to be guaranteed to overlap to be sure that there are no holes in the observation. Again we have a maintenance problem and the same principle can be used, but this time we measure the view angle of each camera, given the location. Fig. 2.19 shows the principle of operation [17]. When an object passes by or when multiple cameras take a picture, the light source casts different shadows of the object. These shadows are measured and the values are sent to the server, where the Light-Object-Camera-Screen system is nonlinearly validated as a redundant number series. Based on this, we can determine whether the intended coverage is reached.

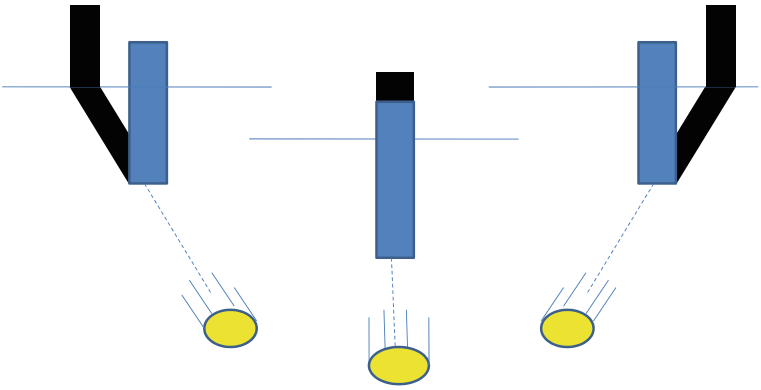

Fig. 2.19 Shadow changes with light

This still leaves the question of how much cameras will be enough. When location data in the individual sensors becomes cost-prohibitive, it becomes attractive to analyze/control the global view (coverage, minimal power requirements) of the network from topological relative data obtainable from the respective sensors [18].

This homology analysis [19] is illustrated by Fig. 2.20, which shows that, given a particular sensor network (top), mathematicians can analyze a theoretical shape known as the Rips complex (center) to reveal which sensors (bottom, darker dots) must be active for full coverage of the area and which may be turned off. In this example, 101 of the network's sensors are essential and 111 may be put into sleep mode.

Alternatively, this approach allows us to judge what needs to be done when one or more cameras are failing. Especially in public places, there is an appreciable amount of damage done to surveillance cameras. In such cases, it is necessary to

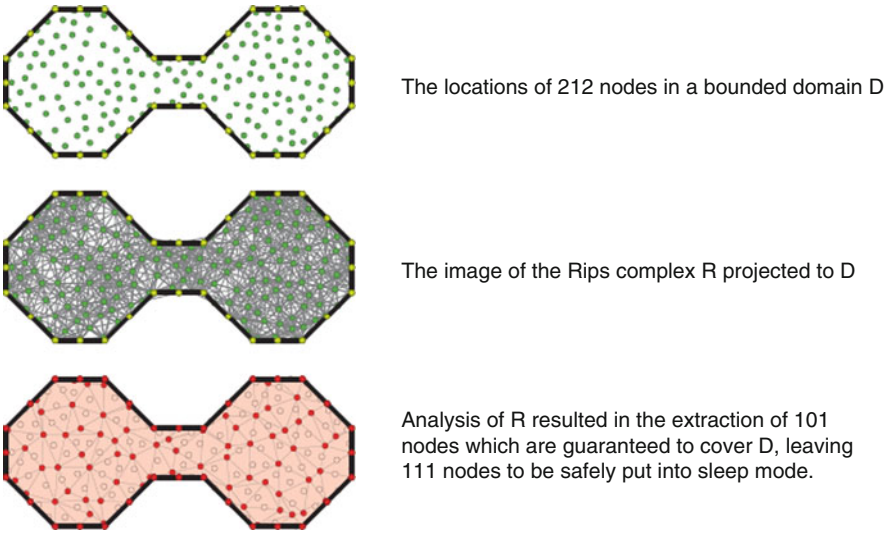

Fig. 2.20 Homological sensor networks [30]

come quickly with a plan which cameras to wake up in order to restore the system functionality.

2.4.2 Configuration of Embedded Software

Configuration of embedded software can be done in two ways: static and dynamic. Static configurations are done at compile- or link-time while dynamic configuration is done in post-build time. However, dynamic configuration has many constraints, for example, limited memory, real-time requirements, and several other OS-related constraints. In this text, the word dynamic configuration and post-build configuration is used interchangeably. The processes are shown in Fig. 2.21 [20].

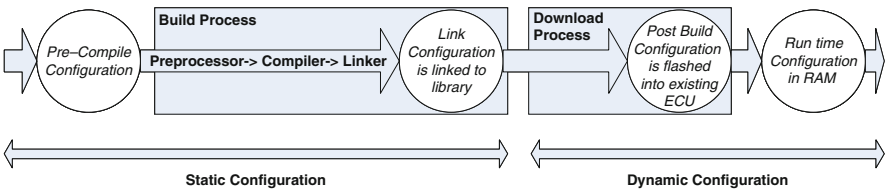

Fig. 2.21 Configuration phases

- **Static configuration** is applied before the source code is put into the build process. This allows for the implementation of some configuration settings (e.g., handling of variants) by macros or C pre-processor switches. It tends to be more efficient at run-time, but it is less flexible than dynamic configuration. It is limited in that the type of specialization needed may not be known until run-time. For instance, a memory management system can take advantage of information about page usage collected at run-time to alter the page replacement policy, in order to reduce paging overhead [21]. Link-time configuration is typically used to link tables with ROM constants to a library.
- **Dynamic configuration**, alias Post-Build Configuration is performed at run-time, either through external input, for example, via user interaction, or automatically as the result of the application requesting a reconfiguration. Dynamic Configuration is proposed as a means to extend the basic system, or in simple term the post-build configuration start after the software is flashed into an existing platform.

The question then remains how this can be achieved by dynamic configuration. In principle, there are two change strategies: extension and replacement. Replacement is probably the easiest option. A function is replaced by a new version. Room must exist for the new version; otherwise replacing other parts must create room. A minimal requirement will be that the new version is not an enlargement but rather a limitation of the original functionality.

Extension requires the existence of a point of adhesion. A classical way to do this is by replacing a "stub." The standard system is equipped with a number of functionless stubs, acting as placeholders. When needed, a stub can be taken out and replaced with a functional part. The alternative is the software bus, i.e., a compiled port structure. Connecting a CSW to this bus will connect it to the system. It is possible to marry these concepts by having a stub that creates a sub-bus, so that CSWs are attached to a proprietary extension.

Both strategies have been applied successfully for the post-loading of AUTOSAR functions. The Hörner paper [20] basically refutes the use of post-loading as it would make the system unmanageable. He argues that it will destroy the benefits of a component-based architecture, both in terms of the internal data structures as in the system reliability. The current version of AUTOSAR 3.0 has RTE as a static layer, which prohibits the addition of signals, and runnables in post-build. Figure 2.22 shows that all the communication between SWCs and BSW is handled via the RTE layer.

The assumption for the post-build configuration in AUTOSAR is to keep the same interface file (.xml) for the new SWC as the previous version, while maintaining the same memory location as in the previous version will result in a working system. For the solution of this problem, the old component is removed from the compiled and linked project and the new component is added and linked to the project.

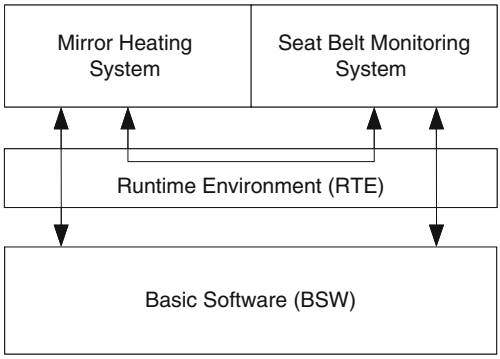

Fig. 2.22 Transparent BSW/SWC communication architecture

2.4.3 Adaptive Reconfiguration

It is a far stretch from the central server with client satellites to the fully autonomous collaboration desired for an electronic network. Looking for inspiration, one usually seeks refuge to biology as it clearly pictures that large amounts of "stupid" creatures can perform together an "intelligent" function. The dictionary is not going to help us as it usually considers words like "herd," "flock," and "swarm" to be equivalent. The only differentiation is with respect to the type of animal, like "cow herd," "bird flock," and "insect swarm." But still some further differentiation can be made:

- [One can be and is controller] – The herd has a common goal, like grazing, but is otherwise relatively unstructured. There may be an effect of imitating the so-called control animal. This implies a kind of centralized control, dictating the herd behavior. For instance, if the control animal starts to run away mindlessly, there will be a stampede. Otherwise, there are little meaningful relations within the herd.
- [Many can be and one is controller] – The flock has common tactics to reach the common goal. A certain structure is usually assumed, like the famous V-shape of the geese flock to minimize wind resistance. This implies a degree of local relations whereby gradually global synchronization is achieved, but there is no statically fixed central control. Rather each bird can fly upfront, each cyclist can take the lead, and each node can take control.
- [Many can be and are controller] – The swarm has a common strategy to reach the common goal. There are no assumptions on the structure and central control is absolutely absent. Swarms are fully adaptable and maybe even learn under hazardous conditions. In short, swarms are the ideal and optimal networks, changing dynamically at maximal performance by collaboration between the nodes, but its consequences in society need careful consideration.[8]

[8]Crichton, M., "Prey," Avon Books, 2002.

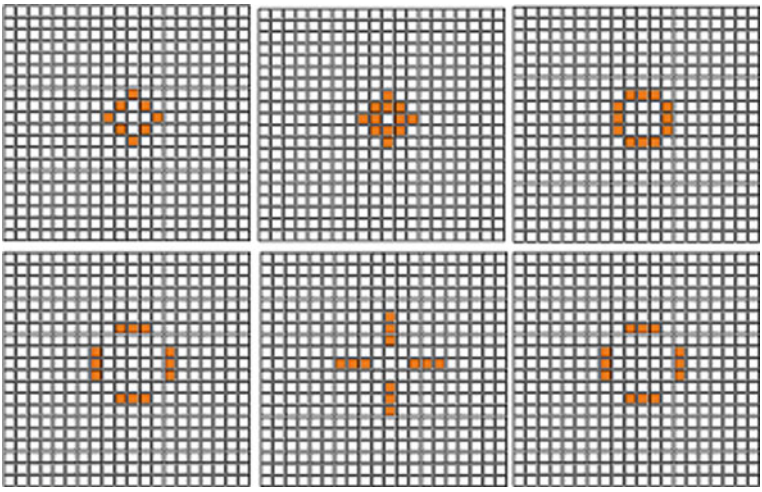

Fig. 2.23 Game of life examples

The prototype of a collaborative system is the Cellular Network. This is a theoretical notion that has become known from such plays like the Game of Life. An even more complicated play is the Predator/Prey Game. Such games can be run on the electronic implementations of Cellular Neural Networks and more (Fig. 2.23) [22].

Such examples show the existence of networked intelligence. In sensory networks, we raise such concepts to a higher abstraction level, involving not mere exchange of data but also exchange of software. This covers both a data reduction process that reduces the bandwidth requirements as well as a collaboration process that has the opposite effect. Modular neural and machine intelligence techniques are proposed, which allow the provision of an efficient balance while keeping the platform constraints of the sensory nodes in check.

The promise of networked interaction is the emergence of distributed control from the basic controllers in the network nodes [23]. The mutual adaption to raise the quality of the task learns the nodes to work together in a strategically sound way. This plays an important role in the effort to optimize the maintainability of the system.

2.5 SoftWare Is Virtual HardWare!

Dwarfs (as described in Chapter 1), in general Models of Computation, form (in our opinion) the after-2007 new wave segment in Makimoto's wave [24] (Fig. 2.24). It will lead to truly application-specific computations, realizing personal (multicore) supercomputers.

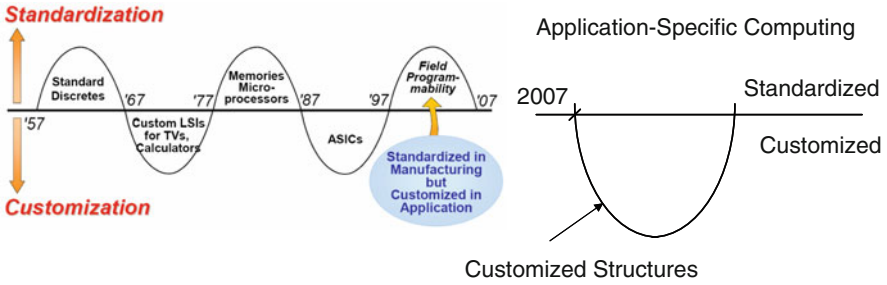

Fig. 2.24 Makimoto's wave extended

After defining a "dwarf" model of computation, a "virtual middleware" architecture can be defined and can be carefully mapped, e.g., onto an FPGA architecture (such as the BEE3). This approach will result in effective performance of the "virtual" architecture, with maximum parallelism and throughput. To the system programmer the "virtual" (middleware) machine will become its programming environment. Programming and code generation of the actual "virtual" machine will make use of conventional software tools, such as compilers and assemblers, specific to the "dwarf" in question. An example of this approach, although limited to Data Flow (its Virtual Processor Architecture), can be found in the Mitrion-C or Vector Fabrics toolset. The Mitrion Virtual Processor lets the user run software in their FPGA-based "Virtual" Middleware Concept.

Some years ago, in an interview with *Electronics Weekly* (9 May 2005[9]), Wim Roelandts, President and CEO of Xilinx, made the following observation:

> The next step is really to make FPGAs disappear. Today our customers are hardware engineers. But FPGAs are programmable devices. If we can create a level of abstraction that appeals to software engineers, we can increase our customer base by at least 10x. That is really where our future is. As long as you have a set of interfaces that you can program to, you don't have to know what the hardware looks like.

The application of virtualization in multicore systems, specifically based on a set of models of computation (as in SystemC), will lead to a highly effective programming development environment for system designers as well as for model (of computation) developers. The combination of software component technology, SystemC, and virtualization middleware will eventually lead to the realization [25] that "SoftWare is Virtual HardWare!"

2.5.1 Elasticity

The problems of software engineering lie notably in the creation of complex systems. To keep such problems in check, structure must be introduced. That starts by stressing the principle of local computation by having (many) well-contained

[9]Electronics Weekly (2005) http://www.electronicsweekly.com/Article39564. Accessed 9 May 2005.

processes. The next step is the limitation of the variety in the procedure parameter lists. This leads to the concept of the virtual bus, an approach that facilitates the construction of extendable systems. A complex system is extendable when it is constructed of a minimal (core) system that provides a basic but bare functionality, while the internal buses can be externally accessed by procedures to enrich or bridge the core functions. As exemplified by the AUTOSAR project, a minimal system can be released much sooner than the complete one and also be repaired much easier.

The component programming dogma states that, faced with the complexities of a heterogeneous, distributed world, the right thing to do is to break the applications into pieces that can be developed, shipped, and upgraded independently; can inter-operate with other pieces; and can be separated from their original application context. Such pieces, software components, are often likened to hardware-integrated circuits. Each integrated circuit provides a discrete, well-defined set of functions that can be used by any number of printed circuit boards. Similarly, each software component provides a discrete, well-defined set of functions that can be used by any number of applications.

In the ultimate sensor network, software components can be floating through the network, hence the notion of elasticity.[10]

2.5.1.1 Hot-Swapping

At first we need a technique for "hot swapping." This is where connectors come into play again. Hardware connectors are entrapments of cables to machines of a specific shape such that no wrong connection can be made. Similarly, as the hardware counterpart has pins, the software connector has parameters on the pre-assigned addresses. But it misses the shape to preclude an inadverted application. We propose here the "call-by-trailer" approach. The idea is to have a snippet of code on both sides of the connection. If two compatible processes are connected, there will be no (or hardly any) signal error because of simultaneous access to the shared memory. If two compatible processes are linked, there will be a symphony of signal errors, so there is a simple detection that is fully in-line with the need to autonomously "detect and connect."

To be complete, the classical connection between a function and a call is through the parameter list. As the call parameters are simply listed on the addresses following the call, they are simply copied to the function parameters on the addresses following the function start (call-by-place). This is fast but requires establishment and maintenance of the correct list and order. As this is cumbersome for ad hoc lists, writing software is more efficient when values are not tied to a place but called

[10]Definition of Elasticity (Merriam-Webster Online Dictionary): 1: the quality or state of being elastic: as a: the capability of a strained body to recover its size and shape after deformation: springiness b: resilience 2 c: the quality of being adaptable, 2: the responsiveness of a dependent economic variable to changes in influencing factors <elasticity of demand> <price elasticity>

by name. Values that are not named can be defaulted (call-by-value). Such measures are of less importance when the debug phase has passed. As the software is coded, virtual buses have been defined, which create the networked architecture of the software processes.

This comes to bear in the sensory network to cloud connection. The cloud offers itself as a software "plug" in the wall. The network calls the IP service, but it may depend on the circumstances whether the service is full-size (and therefore in the cloud), just a limited version (and therefore local), or even a test version. Similar to shown in the AUTOSAR case, practical use requires a fluid change-over between such services, not requiring recompilation of the application.

2.5.1.2 Agile Software

Agile software is widest known because of the development strategy of that name where requirements and code evolve through iterative collaboration between self-organizing teams.[11] But the technology is not confined to this, nor does it always appear under the same name. We feature it here because the definition clearly states a network behavior.

Iterative collaboration is defined by a recurrent structure, maybe even cellular. From this analogy, one may deduce that contrary to popular belief agile operation is not necessarily stable. A well-known example is the Hopfield neural network. This set of nodes with global feedback is notorious for its lack of convergence while learning, unless properly designed. This is even more so in the case of unsupervised learning, so our case will be confined to adaptation schemes, such as using a Hebbian rule.

2.5.1.3 Opportunistic Computing

In opportunistic [26] networks when one or more device comes within reach of another device, close enough to communicate, we will have the beginning of an ad hoc network. The network will then provide the opportunity to share resources and data. In addition, it provides for executing tasks remotely and to forward messages farther into the network.

In so-called MANETs (Mobile Ad Hoc Networks), the location of the various network elements is not fixed; in addition, the number of participating network elements will vary. Since the network elements are mobile, it is hard to pre-assign functionality to the elements. A MANET will be set up incrementally by growing the net and distributing software accordingly.

In opportunistic computing, this growing of the net takes place two-nodes-at-a-time between nodes that find themselves close enough for reliable communication. The realized network is amorphic and expresses the mobile nature of the underlying (in particular social) networks. Some of the benefits of this approach are that the

[11] "Manifesto for Agile Software Development," www.agileManifesto.org.

communication can be low power, and elements become MANET members only if they are in close vicinity of most.

2.5.2 Heterogeneity

Heterogeneous embedded multiprocessing systems generally include one or more processing subsystems, such as traditional general-purpose processors, digital signal processors (DSP), adaptive processing systems (based on FPGAs), and/or fixed resources (ASICs). The availability of these alternative processing realizations creates additional degrees of freedom in which to implement a particular function, as well as tradeoffs in realization in each of the alternatives individually.

For instance, the interface to a 2-D FFT component can be supported by several implementations: one using the runtime host, another using a single PowerPC, a third using 4 PowerPCs, and a fourth using 16 PowerPC 750s. These four implementations are compatible with any application that uses the interface without changing, recompiling, relinking, or otherwise handling the existing application. This leaves component authors free to scale an implementation without breaking applications that rely on the implementation's interface.

A typical example is in power-aware functions. Given that a processor runs continuously a process that determines the available energy in the local battery, then the scheduler can determine which of a set of functionally equivalent processes to run. Such difference can be in hardware versus software, in numerical accuracy, or even in location.

The notion of power-efficient instructions has to do with the use of cache memories and IP blocks. This makes that the same function can be available in a number of power ranges. Usually, the merits of the diverse implementation have been measured and made available to the scheduler. But this appreciation may change when the function is migrated to another node. Furthermore, the energy spent on the communication has to be taken into the equation.

For a typical cloud architecture server consolidation seems called for, but in a sensor network the number and variety of neighboring nodes that are likely candidates for software migration is much higher. Hence the diversity may become too large to be handled by a pre-assembled table and the network has to be able to adapt to such changing circumstances.

2.5.3 Optimization

Even for a relatively simple type of computation, more than one form of networked solution configuration can be effective. In other words, a system having only a single kind of multiprocessing configuration, be it a parallel, pipeline, or recursive doubling structure or a combination of these structures, will not, in general, be universally effective.

When we attempt to evaluate alternative configurations for a particular program, there will be several realizations possible. In general, we might ask ourselves how

one would decide what configurations to use for a certain algorithm and how one would secure the effectiveness and, if possible, optimality of such a program when transformed for execution on those system configurations.

A key and unique aspect of evaluating multiprocessor computing environments is that, in them, the constitutive functions and subsystems can be implemented at various points along their respective design space tradeoff curves. In other words, each element and/or function would have to be evaluated along a design continuum. After that, the total system solution can be evaluated for the right combination of design space points for their constitutive elements. This assignment/selection procedure is similar to procedures used in Operations Research for scheduling of assembly lines with resource constraints [27].

This question is not only important for the initial design but also for subsequent technology updates and upgrades of the particular implemented application, especially when they have to be implemented in the same constrained real estate. Truly heterogeneous processing systems are the basis of obtaining optimized total performance per system, throughput, latency, size (weight and volume), power dissipation, and cost.

Elasticity adds new aspects to the optimization opportunities. Most optimization tradeoff opportunities are of a parallelism or pipelining nature. With elasticity in the cloud, optimization becomes a 2-D, or even 3-D space tradeoff evaluation. Local optimization becomes one of the most natural heuristic applicable. Small organizational perturbation can be evaluated in a hill-climbing approach.

2.6 Summary

In this chapter we have looked into the technology that separates the cloud from the grid: virtualization. Software is aimed to run on any hardware, provided it has the handles to incorporate one or more hypervisors. This allows software to be (1) coded in the programming language of choice, (2) compiled at the station of preference, (3) linked with libraries of quality rather than of source, (4) verified in the desired embedding, and (5) outsourced to the cloud, all without alterations. There was a need to do this on servers to ease life for the clients, but there was no way to do this on simple sensors with at best a 86xx-type controller. That has all changed and sensors are rapidly equipped with virtualizable hardware. Where sensory networks can be "clouded," the question remains how that can be turned to new advantages.

References

1. Carr N (2007) The big switch. W.W. Norton, New York, NY
2. Lohr S (2007) Google and IBM join in cloud computing. The New York Times, October 8, 2007

3. Uhlig R, Neiger G, Rodgers D, Santoni AL, Martins FCM, Anderson AV, Bennett SM, Kägi A, Leung FH, Smith L (May 2005) Intel virtualization technology. IEEE Comput 38(5): 48–56
4. Chun B-G, Maniatis P (2009) Augmented smart phone applications through clone cloud execution, In: Proceedings 12th workshop on hot topics in operating systems (HotOS XII), Monte Verita, Switzerland
5. Spaanenburg H, Spaanenburg L, Ranefors J (May 2009) Polytopol computing for multi-core and distributed systems. In: Proceedings SPIE microtechnologies for the new millennium, Dresden, Germany, 736307, pp 1–12
6. Want R, Pering T, Sud S, Rosario B (2008) Dynamic composable computing. In: Workshop on mobile computing systems and applications, Napa Valley, CA, pp 17–21
7. Spaanenburg H, Spaanenburg L (August 2008) Polytopol computing: the cloud computing model for ambient intelligence. In: 1st international conference and exhibition on waterside security (WSS2008/Structured Session F), Copenhagen, Denmark
8. Sankaralingam K, Nagarajan R, Liu H, Kim C, Huh J, Ranganathan N, Burger D, Keckler SW, McDonald RG, Moore CR (2004) TRIPS: a polymorphous architecture for exploiting ILP, TLP and DLP. ACM Trans Archit Code Optim 1(1):62–93
9. Ranefors J (2007) Integrated development environment for user interface development and customization of mobile platforms, M.Sc. Thesis in Computer Science and Engineering, Lund University, Lund, Sweden
10. Estrin G (May 1960) Organization of computer systems – The fixed plus variable structure computer. In: Proceedings of Western Joint IRE-AIEE-ACM Computer Conference (WJCC), San Francisco, CA, pp 33–40
11. Lee EA (2003) Overview of the Ptolemy project, Technical Memorandum UCB/ERL M03/25 July 2, 2003
12. Hoare CAR (1985) Communicating sequential processes. Prentice-Hall International Series in Computer. Science, Prentice-Hall International, UK
13. Patterson DA (February 2006) Future of computer architecture. In: Berkeley EECS Annual Research Symposium (BEARS2006), University of California at Berkeley, CA
14. Hunt G et al (October 2005) An overview of the singularity project. Microsoft Research Technical Report MSR-TR-2005-135
15. Gautam SP, Jakobsson O (April 2010) Usage of CSW in AUTOSAR, Master's Degree Thesis, Department of Electrical and Information Technology, Lund University, Lund, Sweden
16. Kleihorst R, Abbo AA, Choudhary V, Broers H (2005) Scalable IC platform for smart cameras. Eurasip J Appl Signal Process 2005(13):2018–2025
17. Zhang L, Malki S, Spaanenburg L (May 2009) Intelligent camera cloud computing. In: Proceedings ISCAS, Taipei, Taiwan
18. Klarreich E (2007) Sensor sensibility. Sci News Online 171(18), Week of May 5
19. de Silva V, Ghrist R (January 2007) Homological sensor networks. Notices Am Math Soc 54(1):10–17
20. Hörner H (2007) Post-build configuration in AUTOSAR. White Paper, Vector Informatik GmbH. Available at www.vector-informatik.de/autosar
21. Beyer S, Mayers K, Warboys B (December 2003) Dynamic configuration of embedded operating systems. In: WIP proceedings of the 24th IEEE real-time systems symposium (RTSS2003), Cancun Mexico, pp 23–26
22. Spaanenburg L, Malki S (July 2005) Artificial life goes In-Silico, CIMSA 2005 – IEEE international conference on computational intelligence for measurement systems and applications, Giardini Naxos – Taormina, Sicily, Italy, pp 267–272
23. Zhirnov V, Cavin R, Leeming G, Galatsis K (January 2008) An assessment of integrated digital cellular automata architectures. IEEE Comput 41(1):38–44
24. Manners D, Makimoto T (1995) Living with the chip. Chapman and Hall, New York, NY
25. Spaanenburg H (May 2008) Multi-Core/Tile polymorphous computing systems. In: First international IEEE conference on information technology, Gdansk, Poland, pp 429–432

26. Conti M, Kumar M (January 2010) Opportunities in opportunistic computing. IEEE Comput 43(1):42–50
27. Moder JJ, Phillips CR (1970) Project management with CPM and PERT, 2nd edn. Van Nostrand Reinhold Company, New York, NY
28. Hoogenboom P (November 2009) Architectuur bepalend voor beveiliging hypervisor (in Dutch) Bits & Chips, 3 Nov 2009, pp 37–39
29. Dunlop J (September 2008) Developing an enterprise client virtualization strategy. White paper, Intel Information Technology
30. De Silva V, Ghrist R (December 2006) Coordinate-free coverage in sensor networks with controlled boundaries via homology. Int J Robot Res 25(12):1205–1222

Part II
Cloud-Centric Systems

When all nodes in the network are connected to all others, a Gordian-knot-like maintenance problem can result. When all nodes are chained into a single file, communication will easily saturate. The optimal network will have a topology, which has made compromises between the various cost drivers. Overall one finds myriad pico-nets that combine hierarchically into the larger structure in the same way as geographical maps show many small streets on the local level, and at a higher level a few major highways to provide for fast national transport.

This situation is in sharp contrast to the early days of radio communication. Radio was introduced to provide a fast message transport over long haul, where telephone bridged the shorter distance. In other words, a local fixed net with a global wireless connect. Today, radio is applied at the short to very short distance to allow for mobility and is supported with a fixed glass-fiber net for the fast connect over very long distances. This characteristic inverted situation is what Negroponte in his ground-breaking book *Being Digital* (Vintage Publishing, 1995) has called the RADIO SWAP.

The second part of the book reviews system design aspects needed for the understanding of the subsequently illustrated potential sensory network applications. The emphasis is on system applications that would benefit from the connection into the Cloud.

Highlights

Every Network Tells a Story

A network plays a role in everyday life. Every moment in life, the network takes on a role as part of an ongoing story. Therefore analyzing the story brings out the requirements on which the software/hardware system design will be based.

Story-Teller Configures the Network

Neural Networks are well suited to provide strategy to the network. Distinct use cases are conventionally created in a unified modeling language (UML). Here, basic

aspects of neural control are applied to bring the scenes from the use cases on the story board together.

Through the Looking Glass

Where persons go, the network will follow. The person will influence the network, and the network will observe the person. This makes the sensory network an indistinguishable part of society as it will come to pass.

Everything-as-a-Service

In order to support the development of extremely large sensor networks, we envision the development of processing infrastructures for the collection, analysis, and visualization of large data sets from sensor network based on the cloud computing paradigm. This development will include many attributes available "as-a-Service" in the cloud computing environment.

Chapter 3
System Requirements, Understanding, and Design Environment

The typical embedded system design starts with listing the requirements. But there will be little to go on, when the system is to be a loosely coupled network on which a number of functionalities are to be created. First, the design space has to be explored to find the practical operating limits despite the fact that not everything is clearly defined at the beginning. Applying wisdom is one approach to solve the dilemma, but a more structured development scheme is advisable.

Model-based design assumes that reality can be mathematically described. It is the fundamental base of engineering as the model allows calculating and proving system properties without extensive experimentation. This is based on the assumption that the building blocks of the model are a correct reflection of reality.

In history, the development of physics has gone hand-in-hand with the development of mathematics [1]. Assumptions made about the existence of elementary particles lead to a mathematically defined hypothesis. Based on this model, an experiment is defined that will either show the correctness or refute the hypothesis. Such an experimentally supported model can then be formulated by a physical formula, such as for Ohm's Law.

Physical laws are used in electronics to develop higher-order models, which are assumed to be correct on the basis of mathematical construction. Given infinite computational power, such an assumption seems reasonable. However, as one cannot do infinitely many computations in an infinitely short time, model-reducing assumptions have to be made.

A typical model reduction limits the solution dimensions to the ones that are deemed relevant to the problem. The system that converts a given problem to the desired solution assumes a number of internal states. As the behavior of the system can be described on the basis of these states, they span the dimensionality of the system. Reducing the complexity of the system means reducing the number of states. So a range of specific solutions is modeled, one for each specific aspect of the problem at hand.

L. Spaanenburg, H. Spaanenburg, *Cloud Connectivity and Embedded Sensory Systems*, DOI 10.1007/978-1-4419-7545-4_3,

3.1 System Design Methodology

In order to develop a product, a certain amount of planning must be made. This ensures organization with minimal waste of resources and usually results in fewer errors, better value, and shorter production times. A product development life cycle is the set of procedures and processes together with the corresponding documentation that is used to proceed from the idea to the final product.

Product management is therefore essential in a product development project. Good management involves reducing risks, planning, and coordination. A special case is formed by the technology development projects. In these projects the final product is a new technology, technical functionality, or even just new knowledge. This characteristic makes them different from normal new product development projects. Traditional management techniques cannot be applied as they are defined for more predictable projects, while technology development is uncertain and risky.

Many large projects have been developed outside the area of microelectronics and therefore much can be learned from other areas of expertise. In the typical multimedia setting of sensory networks, we find that the development of a movie script is such a source of inspiration. The main difference will be that though the story as to be told still has to be developed, the technology by itself is clear. In our case, we are not even sure which aspects of microelectronics will flow into our product. Actually, we will develop only the technological base on which the film will be shot and many movies can make use of it.

Another aspect of embedded system design is its sensitivity to faults and failures. We have to create a platform with a minimal requirement on maintenance, as otherwise the production costs for the movies will become too large. What we want is scenery (or a set) that can easily support a lot of movies despite which of the applied instruments will be used and for which qualifications. To reach this goal we first have to find out what is achievable and how we can mix the potentially interesting features to the benefit of the production company.

The better approach to manage a project is its division into phases. This division can be as detailed as desired, ranging from just phases to a layering comprised of phases, stages, tasks, and steps. Between phases it is common to have program reviews called milestones. In these reviews deliverables are produced and decisions about the future of the project have to be made. The decisions involve requirements review, strategy changes, tests result analysis, and updated estimations of time and costs, among others. There are several ways to structure the phases of a project. Here three of such methodologies are shortly presented.

3.1.1 The Traditional Ways

The traditionally most used methodology is called “waterfall,” as shown in Fig. 3.1. First comes the analysis phase, then the design phase, followed by the implementation phase, with testing completing the process. The team that does each

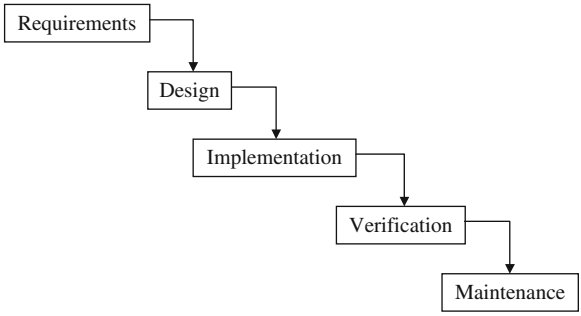

Fig. 3.1 The waterfall approach

phase is different and there may be a management decision point at each phase transition.

This methodology is called the waterfall methodology because each phase flows naturally into the next phase like water over a series of falls. The waterfall methodology assumes that all the work from the previous stage is done prior to beginning the next stage.

This model fulfills some principles of good methodology; the requirements are thoroughly studied and defined in advance, work is done gradually, and there are reviews between phases. However, it was soon proved that such ordered structure was not suitable for products with critical market urgencies.

The waterfall model evolved to adapt to projects with multiple deliveries or versions, typically software development projects. The result was the spiral model, which is based on iterative and cyclical development processes (Fig. 3.2).

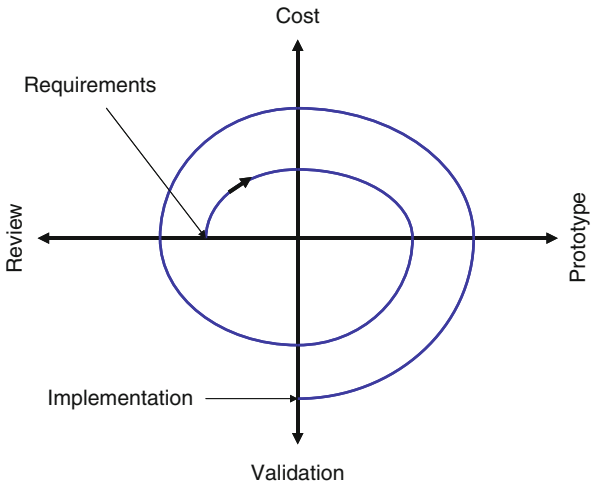

Fig. 3.2 Böhm's spiral development [2]

In this approach the stages that were consecutive in the waterfall methodology become iteratively repeated until the product is sufficiently satisfactory. There is a possibility of building a low-fidelity prototype or even get something in the market in a short period of time, while the development team can learn from early user experiences and trial/error strategies.

A prototype is a product sketch or representation that can be used with evaluation purposes despite its limited functionality. Prototyping is a very useful technique especially when a development team wants to involve users in the testing stage. Technical specifications become more visible when they can be tested for real, and small defects can be easily detected even though the prototype is just paper-based.

Prototyping is heavily linked to the iterative development concept. In order to make prototypes worthwhile, they should be low cost and rapidly made. They can be used to try design ideas in the initial stages, or to explore user behavior in more advanced phases. The detail level is expressed in terms of fidelity. A prototype is said to be low-fidelity, if it is far from the final product and it just represents a concept. To this type belong paper-based prototypes, mock-ups of the intended product. High-fidelity prototypes, however, are accurate representations of the complete system and are used to ensure that the final product meets all the desired requirements and the functionality needed to carry out the intended task. There are a few modalities of prototyping. Some of these are:

- Reusable prototyping: Some parts or the whole prototype can be used for the real final product.
- Incremental prototyping: New ideas are added to the initial prototype as the design is developed through the stages.
- Horizontal prototyping: A wide range of aspects is covered but most of them are not fully operative.
- Vertical prototyping: The prototype focuses deeply on certain functionality.

Two observations can be made regarding progress in computing architectures in gross lines. We will take a look at each of them individually. Note that the law and insight are of a rather intuitive nature. We will refer to the described notions in Chapters 5 and 6 to illustrate their common applicability.

The first “law,” illustrated in Fig. 3.3, is that of “conceptual continuity”: there is a clear evolution from the original mainframe approach via distributed computing, to recently, as a result of a similar evolution in communication technology as reflected in the Internet, the emergence of large-scale server farms (another form of mainframe!).

Similarly, technology innovation and development seem to happen in waves. For instance, in areas of research such as artificial intelligence and neural networks developments will go through a time of frenzy, followed by a time of introspection and reconsiderations, after which it will revive with new solutions to old problems.

In the second “law,” the one we would call the Law of the “Conservation of Trouble,” technology innovation result in benefits, but these benefits come at a price. For instance, development of multi-chip packaging has resulted in devices

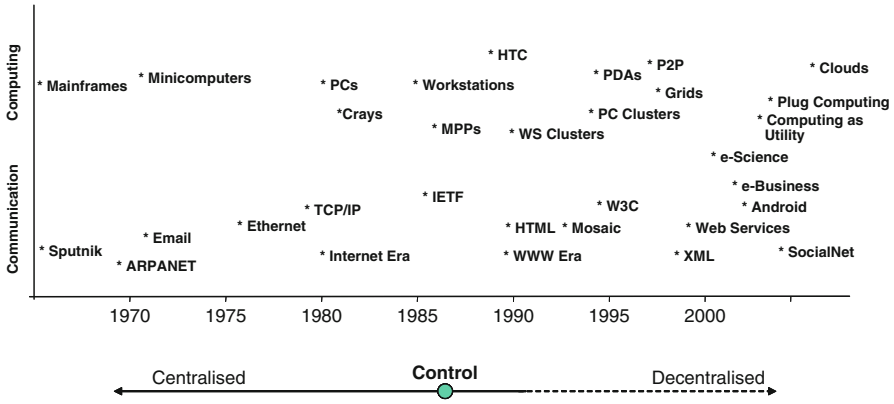

Fig. 3.3 Computing and communication technologies evolution: 1960–2010! [3]

with smaller size and increased compute density, but at the price of additional needs for observability for diagnosis and test. For cloud computing increased capabilities will come at the price of increased needs for security and safety. This increased level of safety and security technology is a missing element in many currently applied intelligent sensor networks. Chapters 5 and 6 will cover these issues.

3.1.2 Embedded System Design

The design of integrated circuits is no easy matter. Hardware complexity has grown over the years. In line with Moore's Law it has become necessary to add design abstractions to master the exponential growth in available design elements at a level: about one level is added every 8 years since 1970. It is not getting easier with embedded systems. The design covers multiple views and multiple abstractions, each to be handled by complex tooling. No wonder not a single person can master it all, and in practice a team of specialists is involved.

The act of designing has stepped away from the Gajski Y-chart by recognizing that testing is not a luxury. Focusing on synthesis followed by assembly, testing involves the verification of functional equivalence between the design to be synthesized and the result after assembly, on every level of detailing (Fig. 3.4).

The world of the system designer who develops a requirement set from a thorough understanding of the domain is totally different from the world of the person who develops a generic platform by which a range of specifications within the domain can be supported. And this is in turn different from the world of the chip architect who works on the domain-specific ASIC to make such a platform efficient. All these persons (and more) are called system architects, which is confusing to say the least.

Where so many people meet with a different understanding of the problem, they tend to use words with a different appreciation. As we just said, the word "architect"

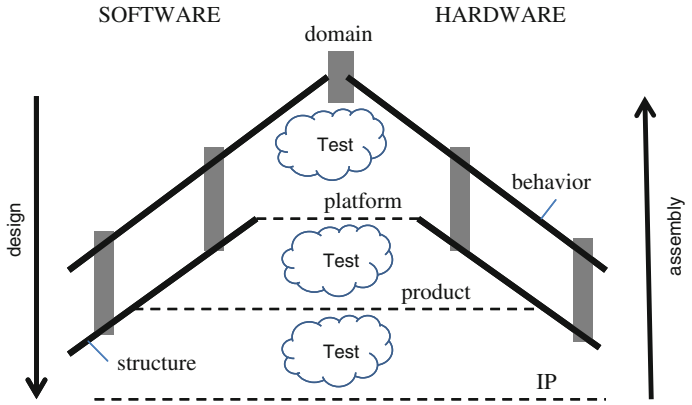

Fig. 3.4 The overall design flow for embedded system design

already gives rise to misunderstanding. Therefore, it is not uncommon to have some persons going with the project from the beginning to the end. A hardware designer usually takes at most two levels into account in his/her design: the level at which he/she specifies and solves his/her problems using the building blocks and their available communication mechanisms from one level below. The efficiency of lower levels is taken for granted by him/her from a perspective of design cost and time-to-market.

A notorious battlefield is where software and hardware meet. Software designers have similar complexity problems, but do not have similar physical forces leading to a shared view on how to deal with abstraction and hierarchy. This difference in visibility and structuring of hierarchy poses significant coordination problems in projects with combined hardware/software design. There exists a lot of re-invention by failure to communicate with "the others," sometimes based on arrogant depreciation of the scientific struggles that occur on the other side [4]. Nowadays, the different architects are likely to live in different parts of the world, or to have been raised in different corners. This adds a cultural problem, where for instance the word "yes" has different unspoken qualities. Let us look again at Fig. 3.4 for a more technical example. At the highest level one deals with concepts and strives to come up with a specification that can be simulated and/or prototyped. Somewhere at a lower level in Fig. 3.4, one starts from executable specifications that incorporate sufficient lower level information to simulate properly the actual product. Apparently, the term "executable specification" can be significantly different within the design process.

In conclusion, apart from expressing the iterative process of specification, construction, and verification for hardware/software co-design, a design flow for embedded systems represents also a communication and coordination flow between the different types of designers involved.

Every sensible system design starts from carefully considered requirements. Without requirements there are no quantifiable aims, and therefore one runs the chance of a long and expensive development of a product without a market. Or the

development is not tractable, and therefore not manageable, because there is little consensus on realistic and attainable goals, which as a consequence tend to drift over time. This is often found the reason for late or never completion of complex projects.

3.1.3 Model-Based Design

There are usually many ways to go through the system. To limit this complexity, it is common to develop a (UML[1]) Use Case: a typical walk-through (Fig. 3.5) that states a typical case of using it. It is then stated that the system can only be guaranteed to perform this Use Case.

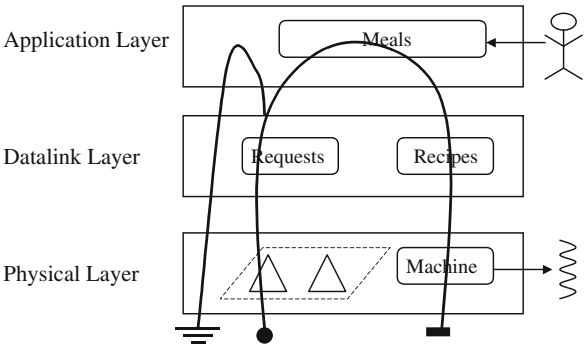

Fig. 3.5 A walk through the system

Usually, some typical cases are handled in more detail. For one or more cases, the typical activity is expressed by a sequence of events that brings the participating components in a certain state. Therefore, the next step is to picture these interactions between the components in time. This is done in the "Sequence Diagram" (Fig. 3.6).

Vertically in the sequence diagram we find for every component its story through time. It receives inputs, and the reaction on these inputs may be dependent on the input(s) it has received. In other words, it goes through a set of states, each characteristic of a history and a probable reaction. Therefore, the next step is to collect this life story and model it through a state diagram (Fig. 3.7).

Many things can go wrong during the above development, before we finally have derived a detailing of the design in terms of collaborating state machines. As a final check, we need to create a simulation of this design. As the focus is in the communication between the parts, it may suffice in the first instance to do a mock-up where the state machines are coded into separate Java classes, each showing their behavior in a separate window.

[1] Unified Modeling Language (see more in Section 3.3.1)

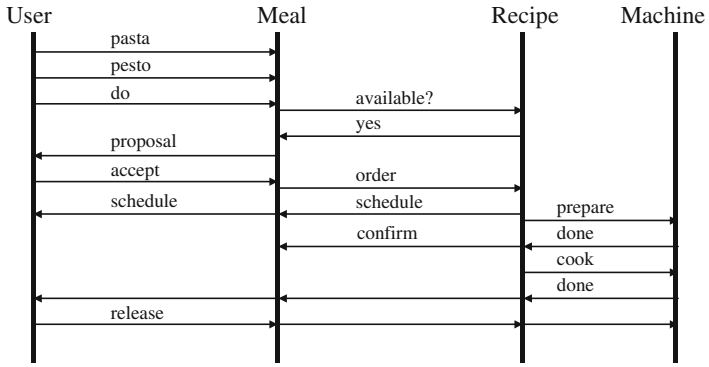

Fig. 3.6 The sequence diagram

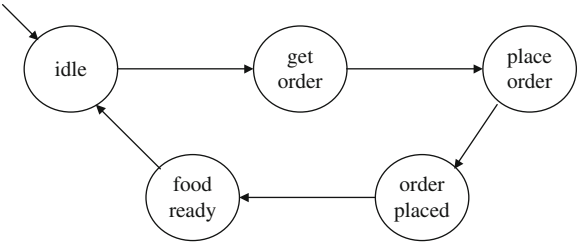

Fig. 3.7 Meal manager state diagram

A minimal state system is not necessarily the one with the least complexity. It appears that state minimization may under circumstances not simplify the computational process, but on the contrary make it more complex. An example from physical modeling is in the string model [5] where the reality is better described by an 11-state model than by using 10 states. Another example, but this time from system design, is in asynchronous normal-mode coding of digital sequential machines. Therefore, we need to be stricter and define the dimensionality as the number of states for which the computational process is least complex.

Computational complexity is not only a matter of the amount of computations but also of the required numerical accuracy. From computer arithmetic it is known that under circumstances the ordering and timing of the operations cannot be freely chosen without compromising the quality of the result. This may create a feedforward structure that influences the dynamics of the system similar to a feedback structure, and therefore has all the characteristics of a state.

There are even more reasons why the relevance of solution dimensions may change, for instance failures. Failures will easily lead to state transitions and therefore essentially change the behavior of the system. When Isermann [6] talks about the detection, isolation, and diagnosis of failures, he proposes to have a library of system models under failure.

Upon application of a stimulation sequence, the system will show a behavior that is unique for a model in the fault library; consequently, the system is supposed to

have that failure. Van Veelen [7] explains this from the intrinsic system dimensionality. System behavior as a whole is multidimensional, but the design limits such dimensions in the behavior of the realized system. When failures appear, they may easily extend into such hidden dimensions, bringing them to the foreground.

Most failures are not catastrophic; in fact, most alarms will just indicate discrepancies between the model and the machine, caused by maintenance, ageing, and so on. For a single factory this can already be an expensive experience. Alarms of this kind can be removed by a systematic update of the settings, if a causal relationship can be established. For a large-scale distributed sensory network, this will easily become too laborious. Therefore, we discuss here what else can be done to dam the alarm flood.

Hidden dimensions may re-appear in system assembly. When a number of systems s_i, each with a possibly unique set of states q_{ij}, are assembled into a single system S, the overall system will have $j!$ states from the Cartesian product of the states of the subsystems. Even when the number of subsystems is small, the overall dimensionality will soon become large: this is called state explosion. In practice a number of these states will be equivalent and can be reduce to a single one, but the algorithm to find such state equivalence is NP-complete. The equivalence will be easily disrupted by occurring failures.

In a typical production line, we find the situation where a number of individually developed machines are connected into a new, single system. This seems similar to a "divide-and-conquer" situation, but it is not. Dividing a system into parts assumes that the full system behavior is still represented in one or more of the parts. In the production line, it is not guaranteed that all aspects of the system are contained in the parts. Consequently, the occurrence by construction of new behavior cannot be excluded. This observation has also been made where two large-scale networks are combined, for instance, in adding a communication network to an energy grid (more details are found in Section 6.3.2).

Assembling parts can explode the number of states, sharply increasing the model complexity. The notion of "abstraction" is popular to indicate an effort to decrease the overall complexity of the system model. It is often claimed that abstraction is the removal of unnecessary detail, but we have argued before that "no fat is present." Therefore, the removal of detail to simplify a model is not abstraction, but actually a safety risk.

Real abstraction is a notational sublimation, taking concepts to a higher level by including the detail and more! As a consequence, redundancy becomes a benefit to be exploited for safety reasons.

3.2 Neural Network System Control

(Artificial) neural networks entered the electronic age as a means to simulate hypotheses in human brain research with the proposition of the McCulloch-Pitts circuit in 1943. The first steps into practice were made by Rosenblatt in 1958 with

the invention of the Perceptron, but Minsky swiftly pointed out how limited the first ideas were. Meanwhile, Hopfield argued that one should not imitate the human brain but learn from it. A new era broke when Rumelhart, Hinton, and Williams re-invented the back-propagation algorithm. Boosted by the need of supercomputers to run algorithms with high computational complexity and the need of analog design to broaden its horizon, it suddenly entered a new beginning in 1986. Since then a number of impressive applications have been demonstrated.

Though most of the really successful industrial applications have not been widely publicized, their occurrence has clearly been spurred by pre-decessing toy-like ones. The most widely spread story is the *truck–backer–upper* as demonstrated by Widrow [8]. Here a truck is controlled by a neural network that is trained by examples on how to back-up a truck. This task is a well-known control problem as it already poses a challenge to the novice human truck-driver. In the course of time, this experiment has been repeated several times and has eventually produced the disclosure of an almost trivial solution [9].

A similarly popular experiment is the *balancing pole* [10]. Here, a pole is placed on top of a car, and the car is controlled to move such that the pole is well balanced and will never fall. This experiment is already of a more complex nature because of its dynamics. Even before the first neural experiment has been performed, a closed-form analytic solution has been found from control theoretical analysis. These two experiments have clearly indicated to the community the attractiveness of using neural networks to prototype solutions for nonlinear control problems and have thus contributed to further control systems research.

3.2.1 Neural Network Theory

A neural network is in its most general form a system of connected elementary processing units, which for historical reasons are called neurons. Each neuron is associated with a real numerical value, which is the state of the neuron. Connections between neurons exist so that the state of one neuron can influence the state of others. The system operation can be programmed by weighting each connection to set the degree of mutual influence. A single neuron is depicted in Fig. 3.8. It shows when going from left to right the following operations: (a) the incoming signals x are collected after weighting, (b) an offset θ is added, and (c) a nonlinear function φ is applied to compute the state [11].

Creating a network of neurons can be done in numerous ways, each different architecture for different applications. The system gets a specific function from the structure and the weight setting: the structure allows for general functionality, while the weights set the specific function. In the following, these two aspects of a neural system will be coarsely reviewed.

The neural architecture represents the network function, both globally and graphically. The network function (network transfer function, mapping function, or answer are equivalent terms) accomplishes the main task of the network: adapts

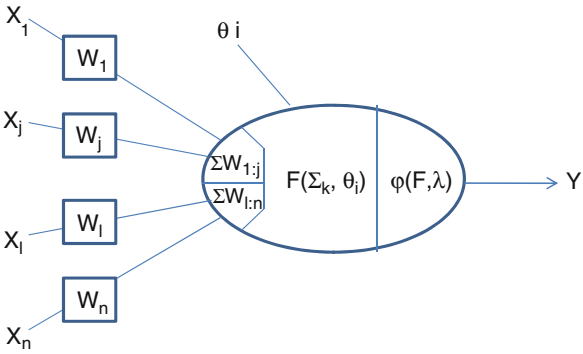

Fig. 3.8 The operation of a single neuron

to associate the questions, posed to the network, with their answers. The questions and answers are terms, borrowed from the human association process. The terms input and output "examples" are widely used: to the network are shown examples of how the problem to be solved behaves; the input example (the stimulus) is shown together with the adequate reaction to it – the response or the output example. This process is performed in the "forward pass" through the network: the outputs are obtained as a response of the neural system to the input (question) stimulation. Thus the forward pass through the network evaluates the equation that expresses the outputs as a function of the inputs, the network architecture, the nodal transfer, and the parameters, so that during the "backward pass" the "learning algorithm" can adapt the connection strengths to make the actual and desired output of equal value.

For the multilayer feed-forward neural network (Fig. 3.9), the input vector x and the scalar output y are connected via the network function f, as described by equation (1):

$$f(\vec{x}, \vec{w}) = \varphi\Big(\sum_i w_{ji}\varphi\Big(\cdots \varphi\Big(\sum_k w_{ik}x_k\Big)\Big)\Big), \tag{3.1}$$

where φ is the nodal transfer and w_{mn} denote the different weight connections within the network architecture, with indices according to the direction of information transport. The nested structure of the formula represents the steps that the feed-forward track of the learning algorithm has to pass.

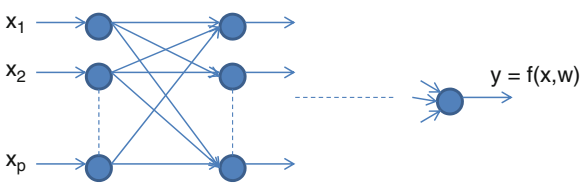

Fig. 3.9 A multilayer feed-forward network

The nature of the nonlinear neural transfer function has not been specified so far. Basically it could be anything, from a straight threshold to a complicated expression. As formula (1) already indicates, it is even possible to wrap an entire neural network to be used in another network as a single neuron with a complex transfer [12]. We will refrain from such and apply only an S-shaped type of transfer, especially a sigmoid function. More specifically, we will use the "logistic" and the "zero-centered" variety: two sigmoid transfer functions with an exemplary symmetry of which the effects on the overall learning performance are nicely suited to display the behavior as described. Their practical importance relies on the fact that the logistic sigmoid has found major application in digital circuitry, while the zero-centered sigmoid appears foremost in analog circuitry. The specification of these functions is shown in Table 3.1.

Table 3.1 Specification of two sigmoid transfer functions

	Logistic	Zero-centered
Function φ	$1/(1+e^{-x})$	$(1-e^{-x})/(1+e^{-x})$
Output range	$(0, +1)$	$(-1, +1)$
1st-order derivative	$e^{-x}/(1+e^{-x})^2$	$2.e^{-2x}/(1+e^{-x})^2$
$\varphi' = f(\varphi)$	$\varphi' = \varphi.(1-\varphi)$	$\varphi' = 0.5\ (1-\varphi)^2$

The Artificial Neural Networks (ANNs) adapt themselves to map the event-consequence evidences of behavior of the problem to solve on the known problem areas. In human terms it can be said that the network *learns* the problem. This learning is an optimization process, accomplished in a lot of the networks by a "gradient optimization" method. The gradient methods can be described as a hill-climbing (descending) procedure, which ends by finding an optimum (extreme) state in which the network models the problem best. At this state it is expected that every question from the problem area, presented to the network, will bring the right reaction (answer) at the output of the network.

Any optimization problem can be defined by a set of inequalities and a single cost function. In neural networks, this cost function is usually based on a measure of the difference between the actual network response and the desired one. Its values define the landscape in which an extreme must be found. By deliberately adapting the weights in the network, the actual and desired responses are brought together, which defines the desired extreme. In other words, a measure of the difference in responses for one or more inputs is used in an algorithm to let the network learn the desired weight settings: the learning algorithm.

As learning results from the mutual attraction of actual and desired responses, it will be based on the feedback principle. In *supervised learning* this is achieved by an external algorithm in which a new weight setting is computed by the supervisor; in *unsupervised learning* the algorithm is designed into the network structure as an internal competition between adjacent neurons. For this reason the usable network structure for the two approaches is fundamentally different. The operation of

the network is severely influenced by the relation between the number of independent parameters in the problem to be learnt (say, the intrinsic dimensionality of the problem) and the number of independent parameters in the network (say, the dimensionality of the proposed solution).

A network is said to be "under-parametrized," when the dimensionality of the proposed solution is too small to contain the problem; conversely, the network is "over-parametrized" when the dimensionality of the problem to be learned is much smaller than the capacity of the network. While in the former case the network will clearly never be able to learn, in the latter case the degree of freedom to learn the problem will usually be counter-productive. It is therefore clear that a proper parametrization of the network will be somewhat larger than what is required by the problem, but then again not too large. It is a fundamental network design issue to determine just the right value [13].

3.2.2 Neural Control

The use of set-points is popular to bring control applications into a desired state. As system parts may vary, each set-point needs to be adapted to the actual circumstances. To reduce such sensitivity, one usually allows set-points to operate in intervals. However, the process may also vary over time and will then be in constant need of adaptation. For such purpose, the process will be modeled. Consequently, the process model can replace the set-point as control reference. Instead of a single static value, the control now operates with reference to a value that is the ideal with respect to the circumstances. This is the typical situation for adaptive control (see Narendra for ground-breaking paper [14]). The basic structures are depicted in Fig. 3.10. In the direct arrangement, the neural controller is in series with the process. This way the control model can be adapted to the specific realization characteristics of the process.

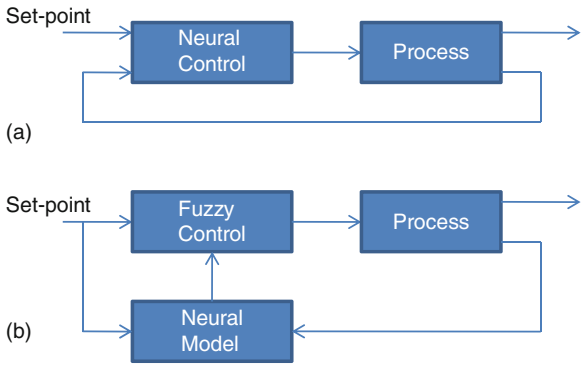

Fig. 3.10 Direct (**a**) and Indirect (**b**) Neural control

Having the neural model in series with the process brings a number of problems. Firstly, the controller is not fast and therefore has problems to react in real time. Secondly, one has to ensure that the learning is not too drastic as otherwise the process may overreact. Therefore, one usually has a simpler controller in series with the process. This leads to the second arrangement, the indirect one. Here the neural model is to learn the proper settings for the controller, which are consequently regularly updated.

One of the generally claimed disadvantages of a feed-forward neural network, as usually applied for control applications, is the ill-determined behavior outside the learning domain. For this reason, it is required to guard the neural model with the awareness of its valid domain. Consequently, the neural model in Fig. 3.10b needs an activation signal (telling when it should check the settings of the controller) and an exception signal (telling when the model steps outside the valid domain).

All this is based on a process with a single controller. In the case of multiple controllers, it is getting harder to adapt the production line to the individual set-points. This is similar to the situation with neural networks where many-level networks are hard to adapt in limited precision while recurrent nets tend to become unstable. As in the production line we adapt rather than learn and the situation is not as bad. But the effect is still there.

Let us take a side step and look at the rudder control for a ship (Fig. 3.11). Rudder control is important because the ship should never get out of control. So when the central control room looses contact, the machine room should take over. And when the machine room also does not manage to steer properly, then someone has to go down and do it manually. This is the basic hierarchical control as introduced by Rodney Brooks for robotics [15]. Such a back-up scheme provides structural redundancy, but only in a "line-of-command" way.

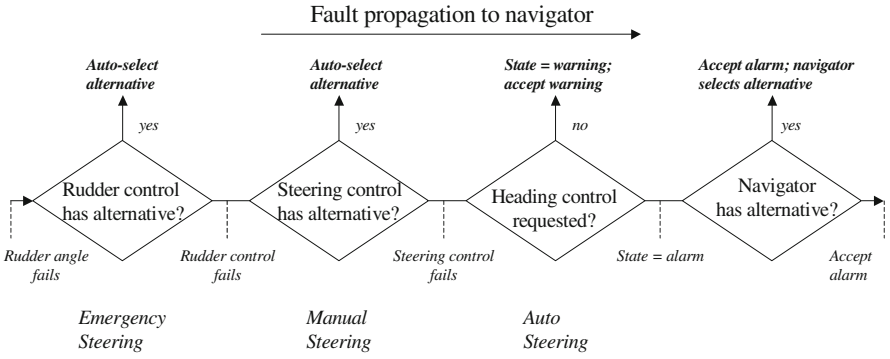

Fig. 3.11 Hierarchical alarm handling in rudder control [16]

3.2.3 Neural Controller Design

In 1986, the European automotive industry initiated the Eureka program Prometheus (Program for European Traffic with Highest Efficiency and Unprecedented Safety).

Autonomous vehicle guidance is one of the tasks that might be realized to increase safety and efficiency of future traffic. The Prometheus subproject PRO-GEN developed this so-called co-pilot function using image-processing techniques supported by extensive expert knowledge engineering. The VITA vehicle is an early example. The copilot covers a wide functionality, such as collision avoidance, lane switching, and convoy driving. A basic feature is lateral control of the car, which provides a safety resource for situations in which correct driver behavior is no longer guaranteed due to tiredness or sudden illness.

From a system-theoretical point of view, the human driver, the car, and the road form a closed-loop control system. Figure 3.12a represents this system generally. The actual dimension of the situation-dependent reference input $r_H(t)$ and the car/road system output $y_H(t)$ as perceived by a human are unknown. Most published efforts regarding the human steering behavior assume $r_H(t)$ to be equal to a zero lateral offset [17–19]. However, the goal directives of the human driver depend strongly on the current traffic situation (staying in a lane, overtaking another vehicle, keeping a safe distance) and will be more in line with vague linguistic statements like: "trying to keep the car on the road" and "trying to optimize driving comfort."

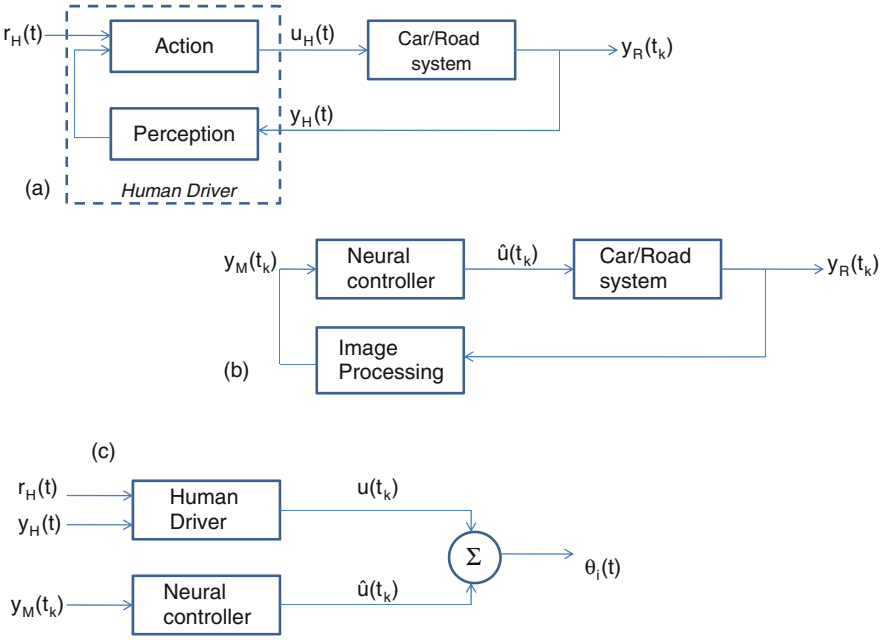

Fig. 3.12 The general human driver/car control system (**a**), the neural network/car control system (**b**), and a human driving behavior identification model

Figure 3.12b depicts the suggested neural control system. The data the neural network accepts are limited to the information delivered by the image-processing system mounted on the car. The image-processing system, designed by Daimler-Benz, is based on a small transputer network. The basic algorithms running on

this system are described elsewhere [20] and its reliability has been proven in combination with conventional state space controllers [21].

In our case, the number of used outputs of the image-processing system (and thus the possible number of feedback signals) is five, namely car speed $v(t_k)$, car yaw angle $\varphi_{\text{yaw}}(t_k)$, road curvature $c_{\text{ROAD}}(t_k)$, road width $y_{\text{ROAD}}(t_k)$, and the lateral deviation of the car on the road $y_{\text{OFF}}(t_k)$. The car yaw angle is the angle between the car direction and the road direction. The lateral deviation (or offset) is the distance between the cars position and the roads center line. The measured output $y_{\text{H}}(t_k)$ is assumed to be the neural function of these five sensory signals, although sensor uncertainty and quantization noise limit the data collection quality. The real-time image-processing system evaluates 12.5 images per second and computes the relevant parameters in less than 80 ms.

In our experiments [22], we concentrated on the basic task of staying in a lane. Therefore, the reference input to the neural controller is not explicitly necessary, but the control target is implicitly encoded in the internal net data. Obviously, the angle of the car's steering wheel is used to effectuate the lateral deviation of the car, that is, $\hat{u}(t_k) = \varphi_{SW}(t_k)$, where SW indicates the steering wheel angle.

Looking at both Figs. 3.12a, b, one can easily conclude that the problem of implementing driving behavior by a neural network can be treated as a system identification problem [14]. However, in contrast to general control approaches stated in literature, we do not identify the system to be controlled (the plant) but learn the closed-loop control task. Then, unlike normal system identification, object and model have different inputs (see Fig. 3.12c). Equation (3.2) describes the assumed human driver's action: controlling only the steering angle.

$$u(t_k) = P_{\text{HUMAN}}[y_H(t), r_H(t)]. \tag{3.2}$$

The objective of the neural network is to determine a function P_{NEURAL} such that $\forall t_k \in (0, N)$:

$$||u(t_k) - \hat{u}(t_k)|| = ||P_{\text{HUMAN}}\left[y_H(t), r_H(t)\right] - P_{\text{NEURAL}}\left[y_M(t_k)\right]|| \leq \varepsilon \tag{3.3}$$

for some desired $\varepsilon > 0$.

The values for $y_H(t)$ and $r_H(t)$ remain unknown as only $u(t_k)$ and $y_M(t_k)$ are recorded. Note that Equation (3.3) is a sufficient condition as long as the task is to imitate the human driver. For a stable controller, however, it is only a necessary one. In addition, one has at least to require that the net delivers an unbiased output $u(t_k)$. All networks trained during the entire development process have met this condition.

Several data sets from different human drivers have been available to train the neural network and validate its performance. They consist of 1,750–6,356 measurements recorded on a German federal highway with a total driving time of 140–580 s. In the first investigations all measured data are scaled to the range [0, 1]. These initial experiments are based on neural net structures with one hidden layer, as literature contains theoretical proof that these feed-forward networks are capable of

implementing any bounded continuous function f: $R^n \rightarrow R^m$ [13]. All neurons use the logistic sigmoid transfer function (Table 3.1). The classical error back-propagation algorithm adjusts the neuron weights with the learning rate and the momentum term taken at 0.7 and 0.5 to provide a reasonable compromise between stability and speed of training [23].

During these first, interruptible simulations, one can observe some correlation between the input data. A large neural net, containing 50 input neurons and 135 hidden neurons, learns to approximate human steering behavior. The input data contain all five measured quantities and their (up to ten) delayed values. After 100,000 learning cycles the neural net reproduces human driving actions with an average error of less than 1% and hardly a larger maximum error. Performing input component analysis by varying one quantity and keeping all others constant reveals some of the actual knowledge encoded in weight space. As expected, the actual output depends strongly on the road curvature, the lateral deviation, and the yaw angle of the car. Variations in car speed and lane width produce contrary effects to experienced driver knowledge, so they can be left out; their representation in the condensed learning set does not adequately reflect the physical dependencies.

With this pre-knowledge, the designer reduces the topology of the neural net in a second step to five input neurons and 15 neurons in one hidden layer. The input neurons correspond to the data signal yaw angle, road curvature, lateral deviation, and weighted time averages of road curvature and lateral deviation. This temporal memory ensures that the dynamic behavior of the vehicle can be taken into account by the neural net [24]. The neural net converges within 50,000 learning cycles to a stable solution, whereby the input neurons are fed with one frame of measured samples during each learning cycle.

This reduced neurocontroller approximates human driver actions with an averaged error of 5% (see Fig. 3.13). Looking at some parts of this diagram, we can

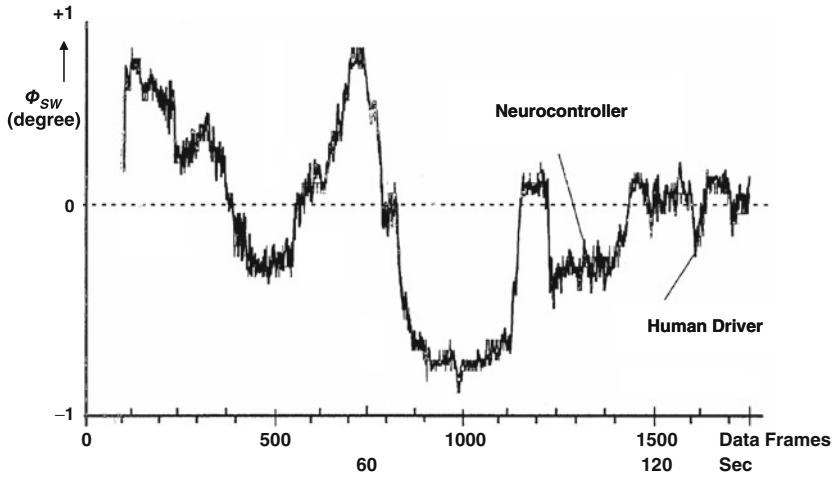

Fig. 3.13 The steering behavior of the human driver compared with the neurocontroller

see distinct differences in steering behavior, which might drive the vehicle off the road. Therefore, the lateral control capabilities of all trained neural nets need to be investigated in a closed-loop simulation with the neural network simulator NNSIM, a vehicle, and a road model. It turns out that the net uses the curvature $c_{\mathrm{ROAD}}(t_k)$ as command variable and $y_{\mathrm{OFF}}(t_k)$ as the variable to be controlled by the feedback loop. The values $\varphi_{\mathrm{yaw}}(t_k)$, $c_{\mathrm{ROAD}}(t_k)$ and $y_{\mathrm{OFF}}(t_k)$ control the dynamics of the car and dampen oscillations.

Experiments with different initial weight settings, numbers of hidden neurons, and human data sets indicate that the solution space of the network parameters is more sensitive to the learning set than dependent on the topology. But all potential solutions show an asymmetric behavior regarding left- and right-side offsets. Due to the scaling onto the interval between 0 and 1, 0 represents the maximum left offset. Multiplied with a static weight, 0 or values around 0 have no strong inhibiting or exciting influence on the neuron activity sum. A second reason is that in the training phase 0 input values prevent weight modification. Therefore, the net learns right offset deviations or curves more extensively than left ones.

Thus, in a third step we chose a symmetric output function, the logistic sigmoid. The standard sigmoid function is scaled and shifted to yield outputs in the range [–1, l]. This kind of function overcomes the problem just discussed.

To make use of the full value range, we have applied a semi-linear output neuron with saturation points at –1 and 1. The neurocontroller keeps the car on the road within 0.16 m, peak-to-peak offset drift. Like the human driver, the net shows a static offset of 0.17 m to the right-hand side. Further simulations on extreme situations reveal that the generalization capability of the net lets the controller handle offset deviations and curvatures much larger than those included in the learning set.

Using about 50,000 of the steering actions recorded for a “flawless” human driver on a public German highway, we captured the control task in a neural net. Subsequently, we selected a three-layer feed-forward neural network with 21 neurons for this model. Finally, we installed it in the Daimler-Benz Oscar (optically steered car) vehicle for practical validation in September 1992. The main result of our practical investigations is that the neural controller trained on human-driving examples exhibits an aperiodic behavior that does not vanish at higher speeds (tests performed at up to 130 km/h). It produces less lateral deviations than the linear state controller and gives a pleasant driving feeling [22].

3.2.4 The Modular Hierarchy

The core problem in developing neural networks is collecting, validating, and presenting the examples. Often it is quoted that more than 90% of the project time has been devoted to getting and preparing the data set to learn from Schuermann [25]. This is surprising, because usually a large amount of ground knowledge exists such as basic physical laws that can be incorporated in several ways.

Firstly, the networks may be initialized to approximate "ideal" behavior. The clean network has weight values that are randomly selected small values around zero. This is meant to create no built-in preferences, so that anything can be learnt. Larger and nonrandom weight values will introduce preferences, as desired, but it is not always transparent how they can be calculated.

Secondly, the problem can be described by more abstract examples. To facilitate this, the abstract notion must be found on reality by known principles, the so-called ground truth. Although the relation is well defined, the use of fixed knowledge can aggravate the learning problem [26]. Often it is cast into a neural network so its parameters can be slightly adapted to the circumstances under which the measurements are made.

For large networks, a monolithic solution tends to be hard (if not impossible) to train [27]. This is caused by conflicting data in the training set or by data that becomes conflicting due to the training itinerary. A solution is to compose the network from modules in the form of subnetworks that can be individually initialized or trained. A fortunate side effect is the structuring and reduction of the example set. Sometimes the composition can be derived from the problem at hand [28], but the convolutional network provides a generic framework as a hierarchy of layers with parallel operating subnetworks.

The layered hierarchy is based on the combined operation of many small fully connected feed-forward neural networks. The underlying idea is a 3-tier architecture, as popularized in many fields of telecommunication (Fig. 3.14), with the following meaning:

- The lower (foundation) tier expresses the basic technological ingredients. This may introduce facts of common knowledge to the neural system, abstract fundamental physical parameters to the digital realm, or unify different graphic packages into a single programming interface. In image understanding, we find here the low-level pixel operations for pre-processing, built from initialized subnetworks.

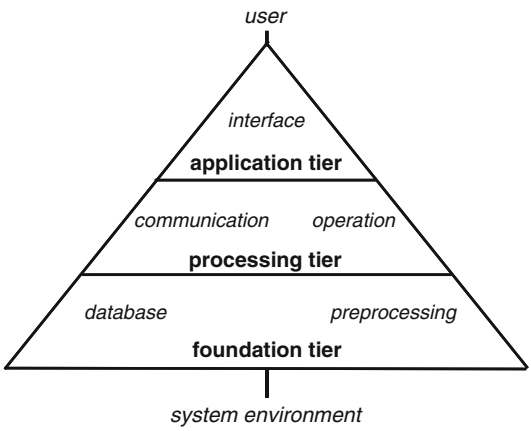

Fig. 3.14 The three-tier concept

- The middle (processing) tier contains the operational functions. It provides the transformations and operations to support a domain of applications in some related technological fields. This may offer a set of classes for the modeling in the envisaged domain and/or a set of algorithms to perform numerical support. In image understanding, we find here subnetworks individually trained for a choice of medium-level blob operations.
- The final (application) tier provides the interface to the application at hand. It personalizes the domain to provide a direct support to the user. The functionality is expressed in terms of the processing functions. As a consequence, any changes in the application will not induce a major effort as long as they can still be expressed in the available functions. In image understanding, we find here the high-level feature operations trained for the overall problem.

The introduction of three layers aims to decouple platform restrictions and hardware dependencies from the user application. In the software arena, the 3-tier architecture is directly related to concepts of reuse and maintenance; for hardware, the tiers image the integration stages between components and product. For the benefits harvested in applying the multi-tier architecture in neural systems, see Nguyen [29].

This method has been first applied at full length for the detection of surface defects on steel from a rolling mill [30]. Here, the defects appear as peaks in a flat area. By their reliance on domain rather than application properties, the lower tier can be built from one single network type. Basic physical laws on light reflection are turned into a set of artificial data. Samples of light reflections in all possible gradients are used, varying around the target value characteristic for the specific subnetwork. The outputs are the characteristic values corresponding to the applied physical laws, but in a manner that makes the subnetworks act as weak classifiers. As the upper networks combine these characteristic values to make the final decisions, it is required that the data to train the lower tier cover the entire problem space (Fig. 3.15).

The middle tier serves to provide parameter invariance. Rotation- and size-invariance is achieved by arranging the outputs from the lower tier into so-called peeling-layers [30]. Width invariance occurs as a large peak and will fill all the three peeling-layers, but a small peak will fill only the inner layer. We use the directions of the vectors that result from the lower layer and calculate three properties that reflect continuity in assembling these vectors into potential peaks. Additionally, a threshold is applied on the results to suppress any detection of a structure that does not adequately reflect a peak. The three output values from the middle networks are used as input for the upper network. The third tier of the network makes the final decision of whether the larger image is centered over a peak.

In this application, the neural system applies only to single images. It supports the detection of and measurements on defects by intelligently matching the expectations of light reflections to the actual observations. We call this “structural stitching” in contrast to the “situational stitching” to enforce continuity on the assembly.

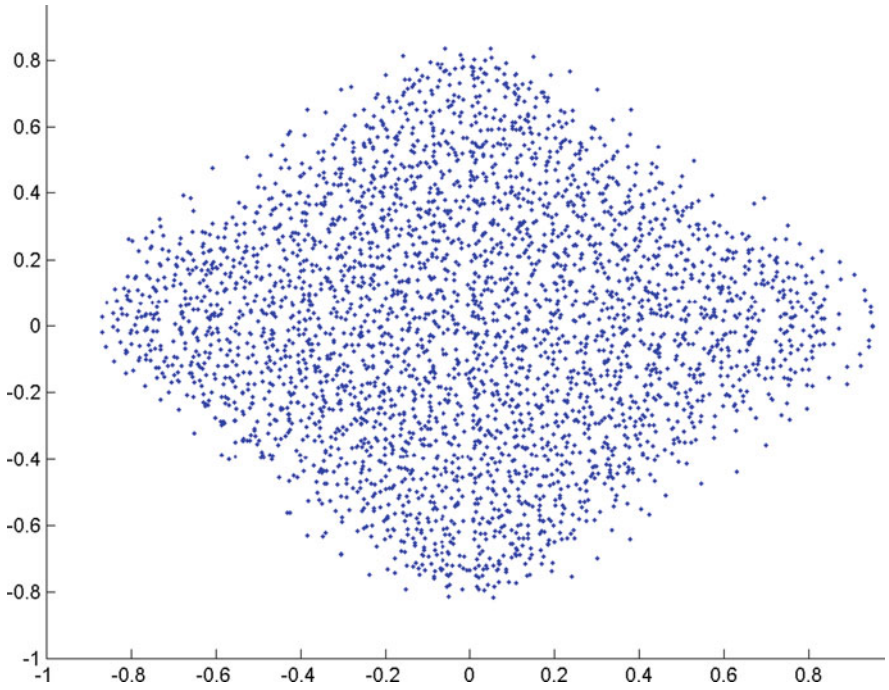

Fig. 3.15 Plot of target values for the lower networks (The training data represents the whole problem space.)

3.3 Networked System Design

Overall, we find that the underlying approach to design and exploration is generally based on "divide-and-conquer" strategy. This principle is founded on the observation that cutting it into parts, followed by solving the parts in isolation and then putting the partial solutions together, can solve a complex problem. This approach assumes that the problem can be cut into parts with a minimal amount of dependencies. This is not always the case and therefore it pays to explore the nature of the problem before the actual design starts (Fig. 3.16). For this methodology we will now introduce a technique familiar from writing a movie story.

A typical thing about multisensory stories is that they should appeal to the larger audience irrespective of physical handicaps. Blind should hear and feel; deaf should see what they cannot hear. Such a sensation will necessarily be composed from redundant inputs, as missing senses must be compensated to an acceptable degree. This will also benefit non-handicapped persons, as the origin of a handicap does not have to be a deficit in biological functionality. It can also be a momentary lack of interest, concentration, or direction. This is how you can hear or sense the car that you cannot see coming in your direction.

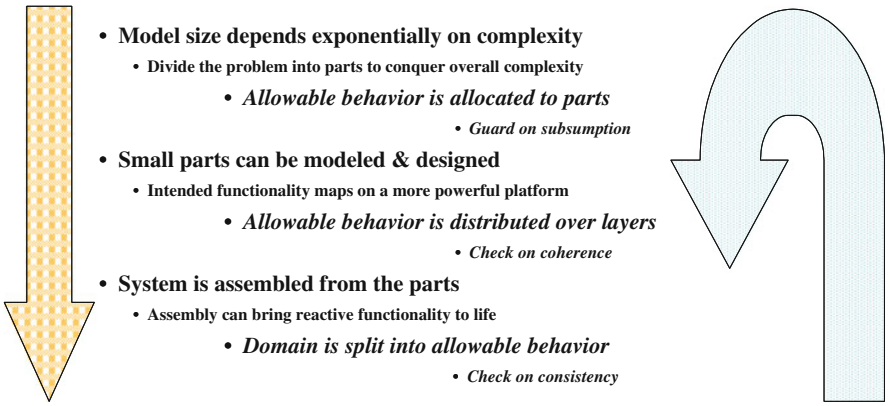

Fig. 3.16 Opposite level traversing in design and exploration [31]

The benefits of sensory redundancy are easily extended to the system itself. Handicaps relate to system failures that should not become errors. Some errors may be permissible if they are not crucial to the intended function. Errors may not only be caused by defects but may also be derived from delimited functionality. In that sense redundancy leads to extending the senses, in the interpretation of McLuhan [32].

Usually, sensors provide more than just a signal. For instance, a camera can receive signals from different spectra, one of them representing images as visible for the human eye. Where a camera is conventionally used to take a picture, that same camera can detect body heat, thereby identifying the presence of a living creature. Another example in water purification shows that expensive detectors for certain materials can be spared as their information is already observable as by-product through analysis of signals from other sensors [33].

3.3.1 Making a Story

A typical product outline will tell the purpose, the means, and the budget. Clearly, the means are not sufficient to implement the purpose. The means must be available to a degree that the purpose can be achieved. The numerical value of "to-a-degree" is given by the functional redundancy.

The typical story will start off by providing an outline, introducing the main idea and characters. This shows the generic functionality on which a number of products can be created. Both these aspects, functionality and requirements, are needed to capture the intended meaning and should provide a starting point for further development. Note that we will never be able to create all possible behaviors. But if we show that some typical ones can be implemented using the same architecture, then we can reasonably assume that the others are feasible too.

Most of the product development processes intend to be user-oriented. That is, they try to satisfy a user need in some way. Quality is a measure of the user satisfaction with the product. Depending on the kind of product, quality achievement can be difficult and become one of the major risks of the project. One of the best ways to ensure a certain quality is to understand how important the first stage of a product development process is: the understanding of customer needs and requirements. This is more difficult than what it may seem, as the tendency of industry is to separate users and producers.

In order to find a link between the user's needs and the development process, a concept called Quality Function Deployment (QFD™)[2] was created. Its purpose is the total understanding of the customer requirements by focusing on the positive quality that really adds value, i.e., the one that makes the user more satisfied. So we need to come with convincing cases of potential behavior. If these really add value and fulfill a customer need, then it is reasonable to expect that more can be done in subsequent product generations.

The Object Management Group[3] came with a language, the Unified Modeling Language, to help developers specify, visualize, and document models of software systems. It is based on a long history of Software Engineering Methods, which each attempted to solve the model capture problem and were partially successful.

The main advantage of UML is that it is technology-independent. UML is always suitable to express the results of an analysis, even with non-software systems. This has been the most important factor in determining the support that the language enjoys. Despite all these positive things, there have also been some negative observations to be made and subsequent versions have been released to address these points. In this book, we take the Software Engineering back-end for granted and are largely focusing on the means to capture a product model from a story outline and one or more episodes.

The reason for the split approach is the same that underlies Software Engineering methodologies: we need two independent views on the same reality to ensure that a comprehensive model can be created. We can call this tolerance to design-induced faults. In the case of embedded systems, the need for fault-tolerance also embraces the practical operation. Therefore, the tolerance needs to be extended, and we will attempt this by relating functions in the one view to processes in the other as a potential n-on-m mapping. We will start from the development scheme shown in Fig. 3.17 to provide for a structured and comprehensive approach.

In the story we outline the intended behavior and its system requirements. This has been detailed in an episode. The story as such takes a number of episodes, but by taking an exemplary one we can be able to provide the basic characteristics in such a way that all episodes are suitably represented. If not, we have to take a small number of widely varying episodes to provide for coverage. Doing this, we discover the fundamental processes that underlie the overall story.

---

[2] QFD Institute, http://www.qfdi.org.

[3] The Object Management Group, http://www.omg.org.

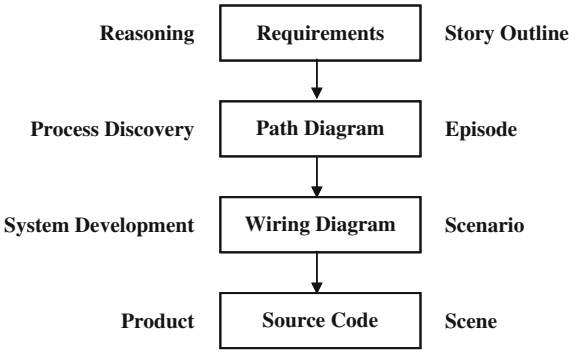

Fig. 3.17 Design path in UML

Once we have brought these processes into the open, we can continue to represent them by the components that can be wired into the system. This turns the behavioral view into a structural view, which in turn gives the information to generate hardware and software parts for the product. One way to do this is by splitting an episode into scenes. Every scene takes the ingredients on the path and the wiring diagram and illustrates different flows through the design.

Manually creating designs takes iterations. In the case of embedded systems, these iterations are not only due to getting the design right but also to optimize the requirements within the potential to develop a system based on the established processes.

The fundamental difference lies in the focus, and the notational sugaring comes only connected to that (Fig. 3.18). The issue is about data and control flows, how flows are separated into threads and recombine differently. Flows go from causes to effects; for example, pushing a button may cause an information packet to move effectuating a light to burn.

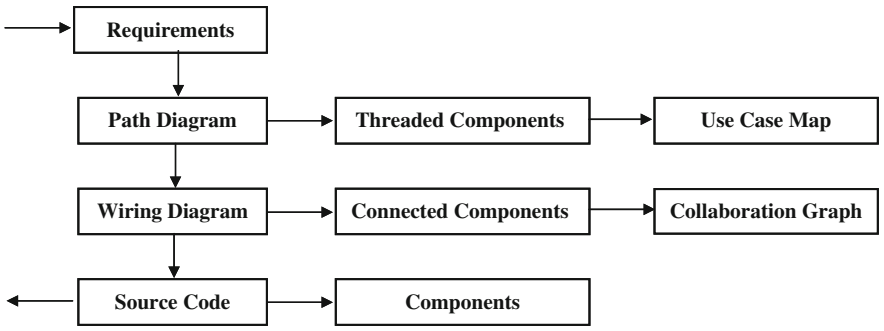

Fig. 3.18 Layer meaning

On the other hand, collaboration is based on data pools that are central to the operation of the components. In other words, where we first have searched for processes to support the data flows, the world gets subsequently inverted to the situation where processes operate on the interaction between data pools. Simultaneously, we move from a global to a detailed perspective.

Before we continue with a case study, we will introduce a more limited example for the mere purpose of illustrating the concepts in this and the next sections. Let us assume a virtual cooking machine in extension of the famous One-Minute-MicroWaver [34]. The futuristic goal is to have remote control of some machine that will pick up food from storage, put it in a place where it can be prepared, and produce it when either ready or called upon.

3.3.2 The Path Diagram

The primary purpose of the Path Diagram is to charter the desired functionality. Such functionality can be sketched as part of the Episode, but here we will simply let our imagination run using a generic structure (Fig. 3.19).

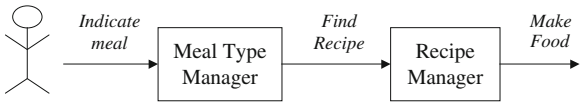

Fig. 3.19 Path diagram for the virtual food preparation

The first thing we need is to indicate what kind of meal we want. Such desires can be entered in a number of ways (such as speech, infra-red, wireless) and have to be handled such that a uniform access to the machine can be applied. To reach this we will have a repository of recipes from which a selection will be made on the basis of potentially limited information. From the type of meal, it can then be deduced what ingredients we need and how the food can be prepared.

Some person will select a meal type from a repository that is maintained by several means. One may expect some .com company to supply this information, or the information is gathered from history but it needs of course tuning to the current possibilities. Once a meal is selected, it needs to be ascertained whether the necessary ingredients are there and how they should be combined. This leads in turn to finding the proper setting for the machine and its parts to prepare the food. Then finally the food can be prepared and delivered

It is clear that this diagram is not an Entity/Relation diagram. It does not indicate concepts and relations, but just pleads the case of how usage will progress. There is a clear confusion possible between Use Graph, Use Map, Use Case graph, and so on. We will refrain here from such formality issues and just call it a Path Diagram.

3.3.3 The Wiring Diagram

The principle of requirements capture is to come with two orthogonal descriptions that can be checked versus one another. So where the path diagram shows the flow of data, the wiring diagram depicts the opposite: the passing of controls.

This short description suggests a three-layer structure, a typical way to separate an application from the real device in a flexible way. We see this in Fig. 3.20, where the persons expresses his desires on a meal in general terms, like "pasta but spicy and make it snappy." From this information a number of potential meals is selected, from which a selection will be made on the basis of what is in stock. Subsequently, the selected meal will be prepared at the right time to be available at the right moment.

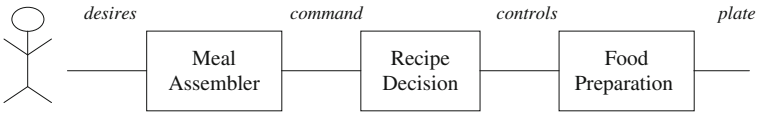

Fig. 3.20 Wiring diagram for the virtual food preparation

The Meal Manager allows the person to express his/her desires without having to check on the refrigerator. This allows for eventual extensions in which food orders will be made over the Internet. The actual decision of what needs to be prepared, and how, is of course not only based on what can or will be in store but also on the allowed time for preparation.

3.3.4 The Sequence Diagram

Once scenarios have been defined, each can in turn be partitioned into scenes. In each scene there is a direct interaction between the (passive/active) actors. On the horizontal axis are the actors, while time evolves on the vertical axis. In time, we see the different actors exchanging messages such that the flow in the path diagram will occur.

Once the sequence diagram (Fig. 3.21) is checked against the earlier path and wiring diagrams, the behavior of the actors can be synthesized. For this, we follow the vertical time axis for an individual actor and find the states. Then, on basis of the states, each vertical actor line can be redrawn as the state diagram for that actor. From there, the generation of supporting hardware and/or software is a well-defined piece of work.

Not every actor will be mapped on an individual platform. Where the scenes provide a link between the interpretation of path and wiring diagrams, the state diagrams need to be assembled into or partitioned over platform functionality. This brings us on the way up for Fig. 3.18. Hence we have now ended the design exploration and can start the actual technology mapping.

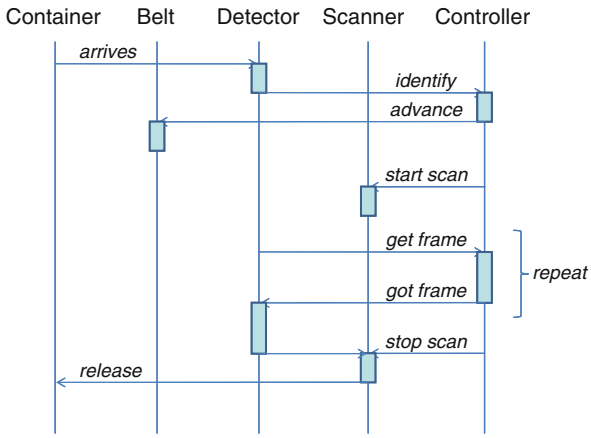

Fig. 3.21 Conveyor belt sequence diagram [35]

3.4 System Case Study

Though every picture tells a story, we have at best a mental picture. Therefore, the next phase is to elaborate the story line into one or more episodes. Each of them will tell what happens through the eyes of the noncommitted observer. Just as extreme values span a numerical domain, exemplary stories can define a behavioral domain. It is therefore crucial to select appropriate ones. Then rephrase each episode as one or more scenarios detailing the interactions of the product with its environment, and detail each scenario in one or more scenes and ultimately shots.

To exemplify the approach we present a case study. The problem we address is in Child Care. With the coming shortage of places at Child Care Centers, the question rises how the situation at the Home can be improved with pervasive ICT (Information and Communication Technologies). In the first stage the story line is created, introducing the main characters and the base line of the story:

> Daniel is not exactly what you would call a techno-freak. Job satisfaction was deemed to be more important than gadgets. Despite the fact that his wife Sara enjoys catching up with the state of the art of technology, they had always been the last on the block to adopt new technologies. But this changed when they started a family and moved to a children-rich community in a suburb. They decided to install a full intelligence system in their house. Victoria got born and brought them into contact with the other families in the street. These contacts intensified when the children grew up, and the families shared in turn the responsibility to look after the kinds in the street. By careful improvisation one managed to ensure that there was always a parent in charge.
>
> Still accidents did happen! One reason was of course that with every year the children got more mobile and inquisitive and one cannot foresee everything. But some part is also the lack of intuition that comes with experience. Things changed when one of the older kids got a nice robot for his birthday. He started to play with it, and tried things out. His father got interested too and together they went into more challenging things. Even Daniel and Sara got infected.

> With growing experience, more and more robots got installed in the houses and gardens, and they started to play a role in everyday life. A milestone was reached when the robots became part of a wireless network. Together they gathered and distributed the experience of handling the individual kids to the parents. This not only covered meals and diets, but also accident proneness, need for caresses and soothing words, tendency to drift away or simply needing some more attention than others.
>
> It is not that accidents did not happen anymore, but now there was always someone there to lend a helping hand.

3.4.1 Case Outline

The street where our story takes place can be turned into a single Wireless Network, but it seems better to arrange everything into one cabled Local-Area Network, providing triple play (television, data, and telephone) to all households from a single point of entry for both information and energy.[4] This is not only because having a single electricity meter for common services is cheaper and saves a lot of bureaucracy, but also it seems fair not to pay a surcharge for individual metering when you are collectively using it anyhow.

At the same time, sharing information streams like videos, phoning to the neighbors, listening to the sleeping children next door, and so on becomes much easier and cheaper. The added effect is the potential for redundancy in the energy supply, which ensure a decrease in vulnerability for power outage.

At the end-points of this network, we move into wireless because of its flexibility. The reach of each substation depends of course on the garden size, but we can assume that 10 m inside the house and 25 m outside will be enough to provide coverage of the area. The geographical distribution of the substations is important, as it should ensure that everybody in the area could be reached for whatever purposes. Overlap in the coverage ensures the required redundancy.

We do not want to create a "Big Brother" type of situation, but having some sensors in the house and in the garden can be useful. Sensors in the garden not only can help to supervise children in daytime but also can help to locate the children when dinner is being served. It seems out of the question to equip all the kids with RFID tags. The alternative is active search techniques using cameras, movement detectors, and heat detectors. This seems to make for a war game. But it may be good to see that we need the same sensory environment for burglar detection and neighborhood security.

In the house we need also surveillance. The sentient home will always look after its inhabitants, but probably some access control is sufficient with additionally the baby phone facility. Added value comes from providing a "good companion." This has three aspects. The perfect "secretary" is the mode in which appointments are

[4] Our faithful readers will easily recognize here the emergence of a local cloud.

made, messages are communicated, and agendas are maintained. This is approximately an office function, largely needed when people do not necessarily meet regularly. The "perfect friend" is the one that both amuses and irritates. It listens, contradicts, jokes, and teaches; in short, it provides direct communication and social interaction. The "perfect nanny" is somewhere in between.

Again we will find the need to create a degree of redundancy, though of less importance than before. When the child has not a virtual companion, there is a natural redundancy when more parents stay at home. This may disrupt the planned program for the day, but will be manageable with some improvisation. Let us now go one step further and make a typical episode, like:

> Daniel and Sara are a young couple with two small kids living in a suburban area. When the Smiths, their neighbors, first introduced an iCat™[5] home, they thought it was a useless purchase. But the kids always came home telling amazing stories about it, so they decided to give it a try.
>
> The first use they gave to it was as a game buddy for the kids. The cat was very easy-going and liked to play and chat, so it kept the kids entertained. It could play different games and, with connection to information, it could even be useful to teach them things in a funny way. It remembered also each one's favorite movies or music, and even gave advice on new releases. As they went for the more talkative version, it liked to tell jokes so it was a great entertainment in parties when other friend couples came to visit.
>
> It was soon after that Sara discovered that the cat was a very good help as organizer. It could keep track on everyone's calendars and deliver messages. She could now leave instructions for the kids when they came back from school and she was still working, or for Daniel, if she wanted him to do some shopping, for example.
>
> Sara had always been a real fan of technology, and despite Daniel's reticence they had installed an intelligent ambient controller for their house. Now everybody in the house is more than used to it, but Daniel is still complaining about how complicated it is to get things done exactly as you want. Since iCat came into the house, he is much more happy with the system, as he just need to tell the cat, step by step, how he wants the light or any room temperature. It is so much easier for him to talk, instead of pressing buttons!
>
> Brad, Daniel's father, had felt really lonely since Marta died. He could take care of himself and he did not want to be a burden for his son, so he had not moved out of his old house, where he kept so many memories of his beloved Marta. Daniel had thought about getting him a dog, but Brad was reluctant to that since a pet requires too much care. Three months after purchasing their first iCat, Daniel came up with the idea that maybe that could be a perfect solution for his dad. Of course it was not a real cat, but they could have conversations and play together some time, and, most of all, the iCat would be a great companion for him. Furthermore, it did not require any special care, which made it perfect!

Note here that currently sensors networks are developed from the bottom up: first the sensor technology is developed, then its compute capabilities within its power limitations, followed by its network principles/protocols and ultimately its actionable interface to the world, including its access to the Internet. From the

[5] Philips Research Robotics

development point of view, the open nature of "cloud computing," that is its availability of compute power, software-as-a service, and Internet-all-the-way-down-to-the-sensors network will make a top-down system design approach extremely attractive.

3.4.2 Refining the Story

The story line relates to a surveillance system that is based on a number of identical (semi-) autonomous systems (Fig. 3.22). Each of them will make a map of the area for which it is responsible. The aerial map is the basis for the role the system will play in interaction to the persons in the area. One or more users will provide input, and systems will respond according to the role it has to play with that person.

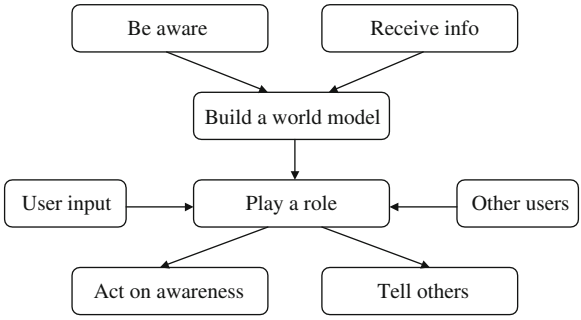

Fig. 3.22 Path diagram

The behavioral intention as expressed by the Path Diagram will be virtually implemented, i.e., independent of the geo-physical location, the place in the sensory network, and the way it is powered. Total independence seems far-fetched, but portability is achieved by building on a physical layer that can be easily configured for the different possibilities.

Consequently, the behavior will be based on an Application Programming Interface (API) that masks the physical reality from the behavioral needs. Next to portability, we need flexibility and reuse, i.e., a large variety created from largely a single body of software. This can be created on the middleware interface by defining a domain-dependent data structure.

We can identify now the existence of a subsumption hierarchy (see more details in Section 5.3.4), based on detection and diagnosis (Fig. 3.23). On every level, different sensors will be needed. In laying out the sensors over the field, we tend to minimize cable or hop distances. Therefore, an aerial organization seems to be called for. But some of the sensors may move across areas and will therefore have to be connected.

Gestures may be detected by a body suit, be part of a toy or be integrated in an area. This potentially makes this part of the network roaming and will be connected into the system through the middle layer, where the gestures are interpreted as part of activity.

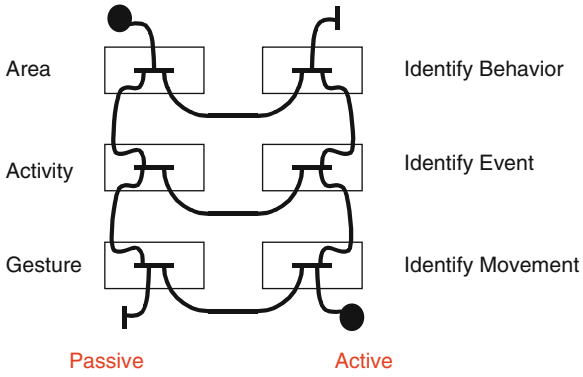

Fig. 3.23 A surveillance hierarchy

Comparing the Path Diagram with the surveillance hierarchy, we see that the hierarchy actually slices through the paths. By adding the subsuming interpretation, it has indicated that the processes are not all on the same level of abstraction and even have dominance on one another. This allows us to sketch how the actual system may be composed as a Wiring Diagram.

In the Wiring Diagram we will now identify the parts of the system and how they are connected bending the proposed hierarchy into suitable layers to provide the envisaged functionality. We may expect the contacts to the outer world (sensors and actuators) to be facilitated by sockets in a single Physical Layer, and not distributed over the functional hierarchy. This is not only because they will share the same data communication and powering networks but also because there will be a varying amount of intelligence in the peripheral devices. The sockets in the Physical Layer will then be needed to allow for arbitrary device to be connected while sharing internally a common output format.

In the next layer we need an activation part that will interpret the incoming data and produce the outgoing data. Data format are internally standardized to ensure that different compositions of the system will only have a limited effect. Likewise, new devices can be easily inserted, once adapted to the internal standards.

Lastly we need a layer to make decisions. Again we provide for a level of standardization, this time by means of a situational map. This allows that different strategies can be easily implemented as their only interface to the system is by means of the map. Different scenes will require different manipulations based on the data of the same map. Such scenes can therefore be applied to different parts of the map and to different persons on the same part of the map. Overall, this leads to the Wiring Diagram as shown in Fig. 3.24.

3.4.3 Moving to Scenes

The scenes in the scenario are then described in detail through examples of dialogs, say “shots.” Each scene corresponds with one proposed use case and should be enabled by a dedicated strategy process.

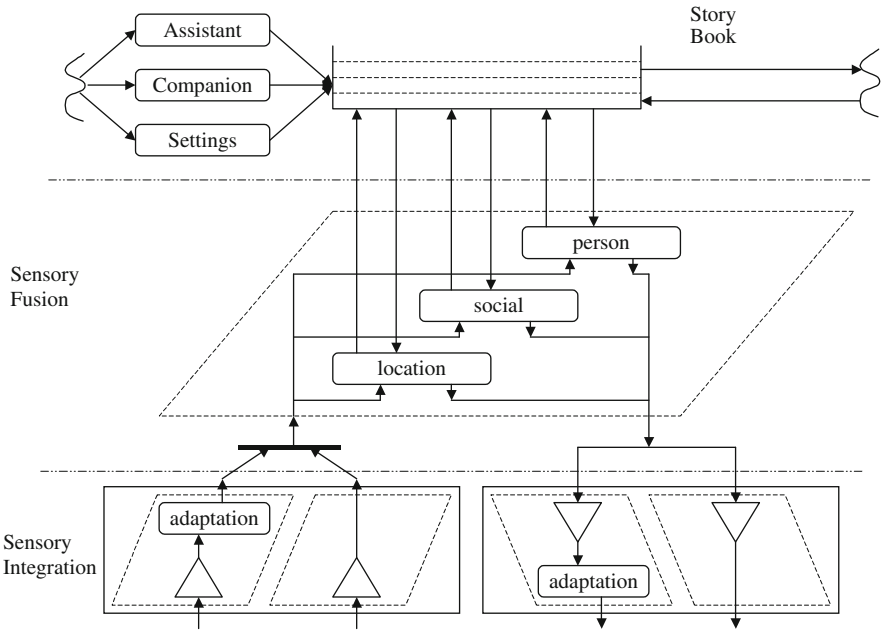

Fig. 3.24 Wiring diagram

The scene outlined here is a typical example of a social situation, where a direct conversation with the system is made and little additional system intelligence is required beyond the facility to converse. In this scene the kids come into the room. They come from school, with their backpacks. The cat feels their presence. He is in sleeping mode, so he does not move. The kids leave the backpacks on the sofa. One of them goes close to the cat and touches his ears. The cat wakes up and stares at him. He recognizes him as Michael.

iCat : "Hello Mike! How are you?"
Mike : "Fine, thanks. And you?"
iCat : "I'm fine too. I am bored, do you want to play a game with me?"
Mike : "Yes"
iCat : "How about TicTacToe? Do you want me to explain the rules?"
Mike : "yes, please"

The cat explains the rules and they start playing. The kid loses.

iCat : "You played really good. Do you want me to tell a joke?"
Mike : "Yes, thank you"

The cat tells a joke to cheer the kid up.

Apart from the dialogue details, we can already see a number of states appearing from which a sequence diagram and corresponding state machines can be derived (Fig. 3.25).

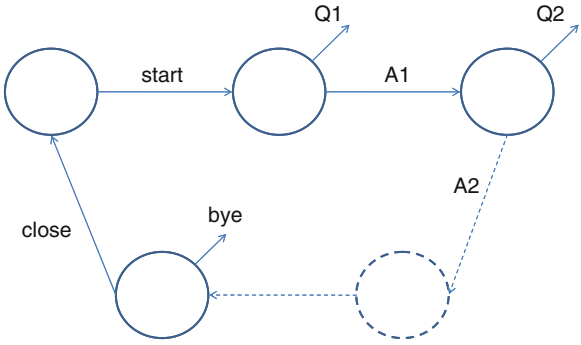

Fig. 3.25 Mike is talking to the iCAT

A rough analysis of the use cases contributes some technology issues: presence and touch sensors, camera, speakers, and microphone necessity; glance orientation, face recognition, speech synthesis, recognition, and understanding; visual recognition, game logics, organizer software, audio recording, etc.

3.5 Summary

This chapter confronts the worlds of microelectronic and system design. Embedded system design goes through a number of phases to turn a specification into a netlist and then back, continuously verifying consistency. The process is highly mechanized through complex computerized tools, but require expert handling.

Neural engineering helps where the specifications cannot be rigidly fixed. In general, neural networks provide an universal computation paradigm like an Excel sheet, where the links between the cells are not expressed but adapted. It enjoys popularity where measurement sets are available but have not been mined into a simple mathematical relation.

But our main issue with sensor networks is that the problem is larger than a single product. It is a collection of embedded products, which together and/or in parts react on an embedding environment. Therefore, a story board technique is discussed to get grip on the network needs of a problem. In the coming chapter we will continue with further connectivity issues.

References

1. Lederman L (1993) The god particle. Houghton Mifflin, Boston, MA
2. Böhm B (August 1986) A spiral model of software development and enhancement. ACM SIGSOFT Softw Eng Notes 11(4):14–24
3. Buyya R Introduction to grid computing: trends, challenges, technologies, applications. The Gridbus Project, The clouds computing and distributed systems (CLOUDS) Laboratory, The University of Melbourne, Australia. www.cs.mu.oz.au/678/grid-overview.ppt. Accessed 16 Oct 2010

4. Casimir HBG (1973) When does jam become marmalade. In: Mendoza E (ed) A random walk in science, an anthology compiled by Weber RL. Institute of Physics Publishing, London, pp 1–2
5. Randall L (2006) Warped passages. Penguin Books, London
6. Isermann R (1984) Process fault detection based on modeling and estimation methods – a survey. Automatica 20(4):347–404
7. van Veelen M (2007) Considerations on modeling for early detection of abnormalities in locally autonomous distributed systems. Ph. D. Thesis, Groningen University, Groningen, The Netherlands
8. Nguyen D, Widrow B (1989) The truck backer–upper: an example of self–learning in neural networks. Proc IJCNN Wash DC II:357–363
9. Jenkins RE, Yuhas BP (1993) A simplified neural network solution through problem decomposition: the case of the truck backer–upper. IEEE Trans Neural Netw 4(4):718–720
10. Geva S, Sitte J (1993) A cartpole experiment benchmark for trainable controllers. IEEE Control Syst 13(5):40–51
11. Haykin S (1994) Neural networks: a comprehensive foundation. Macmillan, New York, NY
12. Keegstra H, Jansen WJ, Nijhuis JAG, Spaanenburg L, Stevens JH, Udding JT (1996) Exploiting network redundancy for lowest–cost neural network realizations. In: Proceedings ICNN'96, Washington DC, pp 951–955
13. Hornik K, Stinchcombe M, White H (September 1989) Multilayer feed forward networks are universal approximators. Neural Netw 2(5):359–366
14. Narendra KS, Parthasarathy K (March 1990) Identification and control of dynamical systems using neural networks. IEEE Trans Neural Netw 1(1):4–27
15. Brooks RA (1986) A robust layered control system for a mobile robot. IEEE J Robot Autom 2:14–23
16. van der Klugt PGM (November 1997) Alarm handling at an integrated bridge. In: Proceedings 9th world congress of the association of institutes of navigation (IAIN), Amsterdam, The Netherlands
17. Shepanski JF, Macy SA (June 1987) Manual training techniques of autonomous systems based on artificial neural networks. In: Proceedings of the 1st international neural network conference (ICNN87), San Diego, CA, pp 697–704
18. Mecklenburg K et al (May 1992) Neural control of autonomous vehicles. In: Proceedings of the IEEE 42th vehicular technology conference (VTC1992), vol 1, Denver, CO, pp 303–306
19. Hess RA, Modjtahedzadeh A (August 1990) A control theoretic model of driving steering behavior. IEEE Control Syst 10(5):3–8
20. Franke U (May 1992) Real-time 3D road modelling for autonomous vehicle guidance. In: Johansen P, Olsen S (eds) Theory and applications of image analysis, Selected papers from 7th Scandinavian conference on image analysis, Aalborg, Denmark, 13–16 August 1991. World Scientific Publishing, Singapore, pp 277–284
21. Franke U, Fritz H, Mehring S (December 1991) Long distance driving with the Daimler-Benz autonomous vehicle VITA. In: Proceedings Prometheus workshop, Prometheus Office, Stuttgart Germany, pp 239–247
22. Neußer S, Nijhuis JAG, Spaanenburg L, Höfflinger B, Franke U, Fritz H (February 1993) Neurocontrol for lateral vehicle guidance. IEEE Micro 13(1):57–66
23. Rummelhart D, Hinton GE, Williams RJ (1986) Learning internal representations by error propagation. In: Rummelhart DE, Hinton GE, McClelland JL (eds) Parallel distributed processing. MIT Press, Cambridge, MA, pp 318–362
24. Troudet T et al (July 1991) Towards practical control design using neural computation. Proc IJCNN Seattle WA II:675–681
25. Schuermann B (2000) Applications and perspectives of artificial neural networks. VDI Berichte, VDI-Verlag, Dusseldorf, Germany, 1526:1–14
26. Jansen WJ, Diepenhorst M, Nijhuis JAG, Spaanenburg L (June 1997) Assembling engineering knowledge in a modular multilayer perceptron neural network. Digest ICNN'97, Houston TX, pp 232–237

27. Auda G, Kamel M (1999) Modular neural networks: a survey. Int J Neural Syst 9(2):129–151
28. ter Brugge MH, Nijhuis JAG, Spaanenburg L, Stevens JH (1999) CNN applications in toll driving. J VLSI Signal Process 23(2/3):465–477
29. Nguyen CT, (2003) Method and system for converting code to executable code using neural networks implemented in a very large scale integration (VLSI) integrated circuit. US Patent 6,578,020
30. Grunditz C, Walder M, Spaanenburg L (2004) Constructing a neural system for surface inspection. Proc IJCNN Budapest Hungary III:1881–1886
31. Spaanenburg L, (March 2007) Organic computing and emergent behavior, In: van Veelen M, van den Brink T (eds) Notes of mini workshop on dependable distributed sensing. Groningen, The Netherlands, pp 17–20
32. McLuhan M, Fiore Q (1967) The medium is the massage. Penguin Books, London
33. Venema RS, Bron J, Zijlstra RM, Nijhuis JAG, Spaanenburg L (1998) Using neural networks for waste-water purification. In: Haasis H.-D, Ranze KC (eds) Computer science for environmental protection ′98", Networked structures in information technology, the environment and business, Umwelt-Informatik Aktuell 18, No. I, Metropolis Verlag, Marburg, Germany, pp 317–330
34. Shlaer S, Mellor S (1992) Object lifecycles: modeling the world in states. Yourdon Press, Upper Saddle River, NJ
35. Moore A (6 August 2001) A unified modeling language primer. Electronic Engineering Times

Chapter 4
Sensor-Centric System Developments

Already the Phoenicians sailed the seas according to coastal marks, while in the early days of aviation navigation was based on roadmaps. Where no clear landmarks were available, as on full sea or in the desert, hemispherical marks like the sun or the moon were to be used. But such marks are not always in plain sight; under such circumstances, it was not wise to sail out. A simple storm may lead a ship astray and only when land is sighted can the location be calibrated again.

The electronic age allowed crossing the barriers posed by visual contact. Electromagnetic waves can pass through clouds; moreover, they can reach beyond the horizon. Networks of active radio beacons are constructed for sailing and flying purposes. By triangulation between three known beacons, the coordinates at the receiving side can be calculated. The system did not need to be precise, as it only served to fix the course for vehicles. On the shorter distances, simple line of sight or homing beacons at higher frequencies would provide for the necessary corrections for precise docking and landing.

Satellites bring digital technology to the craft of positioning. A network of geostationary satellites takes the role of the earthly beacons, while digital packets containing timing information replace the analog time-dependent (and therefore noise-sensitive) signals . Precision is raised to the range of earth meters. This opens entire new possibilities, aiming for location-awareness rather than direction-aware computing. The Global Positioning System (GPS) was originally introduced for military purposes, but non-military alternatives have also been under development.

GPS receivers have been rather bulky initially, but size and energy requirements have rapidly been reduced. An increasing number of embedded systems become equipped with GPS functions. An early example is in car retrieval: GPS devices inside cars allow the tracing of such vehicles wherever they have been moved by the owner, as well as other drivers. With international transport, the GPS have allowed to keep track of the whereabouts of the trucks. Such are clearly useful functions, but the key to the market success is not the high precision of the GPS system but rather the standardized architecture made applicable by mass-produced peripherals.

The societal consequences of location-aware computing can be impressive. Of old, location-aware functions are based on transponders. Examples are restraining orders for criminals (or civilians with criminal tendencies such as some known clans

L. Spaanenburg, H. Spaanenburg, *Cloud Connectivity and Embedded Sensory Systems*, DOI 10.1007/978-1-4419-7545-4_4,

of soccer fans). Such people have to report at regular intervals or carry a restraining device. Using GPS technology, any civilian can be restricted for specific activities. This could range from aid to elderly citizens and coaching of small children to "Big Brother is watching you" [1]. But, on the upside, it serves also to provide information and enjoyment. In Augmented Reality, knowing your location helps to plan your route, provide information on the tourist attractions around you, and explain the picture in front of you. In the Sonic Lighter [2], a map of the world shows the location of everybody who is using the App at that moment. The potential features can be even more interesting with the inclusion of an electronic compass [3] to give next to location also direction.

In order to support the development of extremely large sensor networks, we envision the development of a processing infrastructure for the collection, analysis, and visualization of large data sets from sensor network based on the cloud computing paradigm. This development will include many attributes available "as-a-Service" in the cloud computing environment, such as Software-as-a-Service, Security-as-a-Service, Safety-as-a-Service leading up to Infrastructure-as-a-Service, Analysis-as-a-Service, and Visualization-as-a-Service realizations.

In this chapter we will describe various sensor-based applications with increasingly larger cloud-based intelligence participation. In the following, we will first describe some applicable network technologies, after which we will describe several typical developments in extremely large intelligent sensor network applications.

4.1 Wireless Sensor Network Technologies

Over the past decades, continuous development in the semiconductor industry has achieved the miniaturization of integrated circuits. Various microelectronic sensors have been invented, which are tiny and powerful. At the same time, wireless communication technology has also made great progress. Together they have made the concept of Wireless Sensor Network (WSN) a reality. Wireless Sensor Networks have appeared in the past few years. Many applications have been proposed. They realize a vision of "Ambient Intelligence [4]" where many different devices will gather and process information from many different sources to both control physical processes and interact with human users. WSN realizes the interaction between people and the physical world.

One of the applications developed to utilize large-scale pervasive wireless sensor network is "Smart Dust." It was developed at the University of California at Berkeley and funded by the Defense Advanced Research Projects Agency. "Smart Dust" is designed to be a self-contained, millimeter-scale sensing and communication platform for massively distributed sensor networks [5]. As a military application, the concept of the project involves the use of thousands of tiny wireless sensors that can be spread over a large battlefield area, allowing enemy movements to be monitored.

Besides in military applications as mentioned above, WSN has a wide applicable area that may cover each aspect of human life and the ecological environment, such as disaster relief, environment control and biodiversity mapping, intelligent buildings, facility management, machine surveillance and preventive maintenance, precision agriculture, medicine and health care, logistics, and telematics [6].

Wireless Sensor Network consists of individual nodes that are able to interact with their environment by sensing or controlling physical parameters. The individual node contains at least some sensing, computation, and wireless communication functionalities. These nodes in the network have to collaborate to complete the task. As the applications are comprehensive and sophisticated, a single node usually does not fulfill the requirement. For instance, the "Zebranet" project that is carried out at Princeton University aims to set up an ad hoc WSN with the bandwidth and computational capabilities required to monitor the long-range migration, interspecies interactions, and nocturnal behavior of zebras in Africa [7].

4.1.1 Wireless Communication Protocols

There are several wireless communication protocols adopted by academia and industry to build the sensor nodes that are used for Wireless Sensor Network (WSN) applications. Research work on the MAC (Media Access Control) layer aims to find a more efficient network protocol. However, for different WSN applications there may be different optimal solutions. For example, a temperature sensor node requires very little communication bandwidth and is activated at a low duty cycle but it needs a long lifetime, while a wireless endoscopy capsule sensor node needs a higher bandwidth but the lifetime is only several hours.

Dependent on the locality dimension requirement for transmission of sensor collected data, and dependent on the required data rates, various communication protocols can be appropriate (Fig. 4.1), as listed in the following.

4.1.1.1 RFID

Radio Frequency IDentification (RFID) is the use of an object (typically referred to as an RFID tag) incorporated into a product, animal, or person for the purpose of identification and tracking using radio waves. It has two parts, a tag and a reader. Some tags can be read from several meters away and beyond the line of sight of the reader.

RFID is fundamentally based on wireless communication, making use of radio waves that form part of the electromagnetic spectrum. RFID systems use a wide frequency band from LF (low frequency) to UHF (ultra high frequency) and microwaves. Table 4.1 lists the typical RFID frequencies.[1]

[1] TMP275 Data Sheet, Texas Instrument

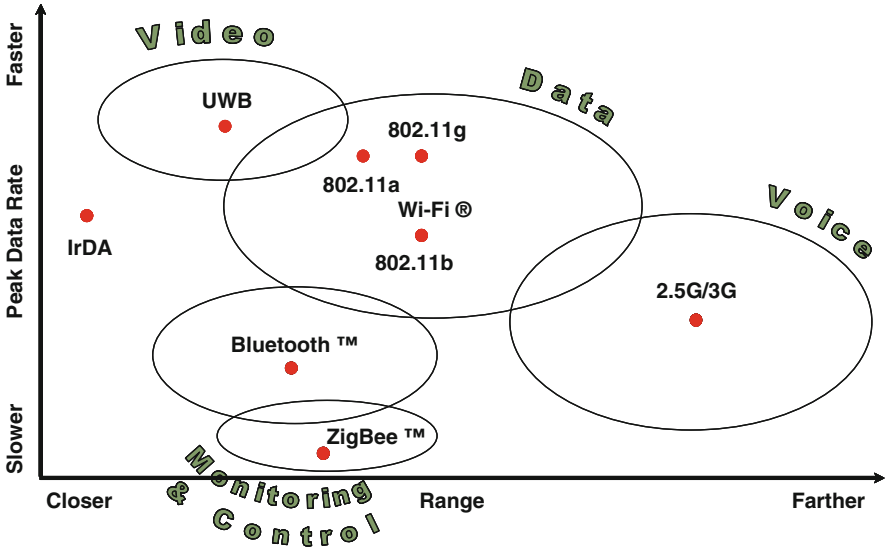

Fig. 4.1 Range and data rate tradeoffs in protocols (After Freescale Semiconductor)

Table 4.1 RFID operating frequencies

Band	LF	HF	UHF	Microwave
Frequency	30–300 kHz	3–30 MHz	300 MHz–3 GHz	2–30 GHz
Typical RFID frequencies	125–134 kHz	13.56 MHz	433 MHz, 865–956 MHz, 2.45 GHz	2.45 GHz
Approximate read range	Less than 0.5 m	up to 1.5 m	433 MHz = up to 100 m, 865–956 MHz = 0.5 to 5 m	up to 10 m

The choice of frequency affects several characteristics of an RFID system. In the lower frequency bands, the reading ranges of passive tags are short due to poor antenna gain. At higher frequencies, the reading range typically increases, especially where active tags are used. RFID systems are prone to interference from other radio systems. The LF band is particularly vulnerable, while the microwave band is the least susceptible. The performance of RFID systems will be adversely affected by water or wet surfaces. HF signals are better able to penetrate water than UHF and microwave signals due to their relatively long wavelengths. Metal is an electromagnetic reflector and radio signals cannot penetrate it. As a result, metal will not only obstruct communication but can also have adverse affects on the operation of RFID system. The HF band is affected more by metal than the LF band. RFID systems operating in the LF band have low data rates. Data rates increase with the frequency of operation. At LF and HF, inductive coupling and inductive antennas are used, which are usually loop-type antennas. At UHF and microwave frequencies, capacitive coupling is used and the antennas are of the dipole type [8].

There are three types of RFID tags.

- Active tags, which contain a battery, can transmit signals autonomously.
- Passive tags, which have no battery, require an external source to obtain power and provoke signal transmission via radio frequency.
- Semi-passive tags, which also contain a battery, still require an external source to provoke signal transmission.

Table 4.2 gives a brief classification of RFID tags. Passive tags need time to set up communication, which incurs delay because energy needs to be harvested from the reader. Passive tags can only operate in the presence of a reader. Semi-passive tags can operate without a reader, but communication is only available in the presence of the reader because the power for transmission is provided by the reader. Active tags are fully autonomous.

Table 4.2 Classification of tags

RFID Tag (W/R)	Active	Passive	Semi-passive
Battery	Yes	No	Yes
Availability of communication	Continuous	Within field of reader	Within field of reader
Communication start-up time	No	Yes	No

The passive RF transceiver is the unique feature of RFID communication. It recovers energy from carriers sent by the reader. By adopting the passive RF, real-time demand can be satisfied with no extra energy induced in the sensor nodes, solving the problem that tradeoff occurs between low-power and real-time wakeup in long-term standby sensor nodes in inventory and medical applications.

4.1.1.2 TinyOS

One of the first communication software application and the simplest, TinyOS, follows the main concept for most low-power measurement systems, the "Hurry up and Sleep." The objective is to sleep as often as possible to save power. TinyOS was developed at UC Berkeley. It is built out of components that provide a framework for concurrency and modularity. Its functionality consists of Commands, Events, and Tasks (Fig. 4.2).

Components store the state of the system and consist of interfaces and events. "Hooks" are provided for wiring the components together. Applications are additional components composed with the OS components and are event-driven. Typical for the communication application, a stack (Fig. 4.3) of components will be realized.

TinyOS represents a very small footprint – core OS is 400 bytes code + data. Libraries and components are written in nesC, an extension of C.

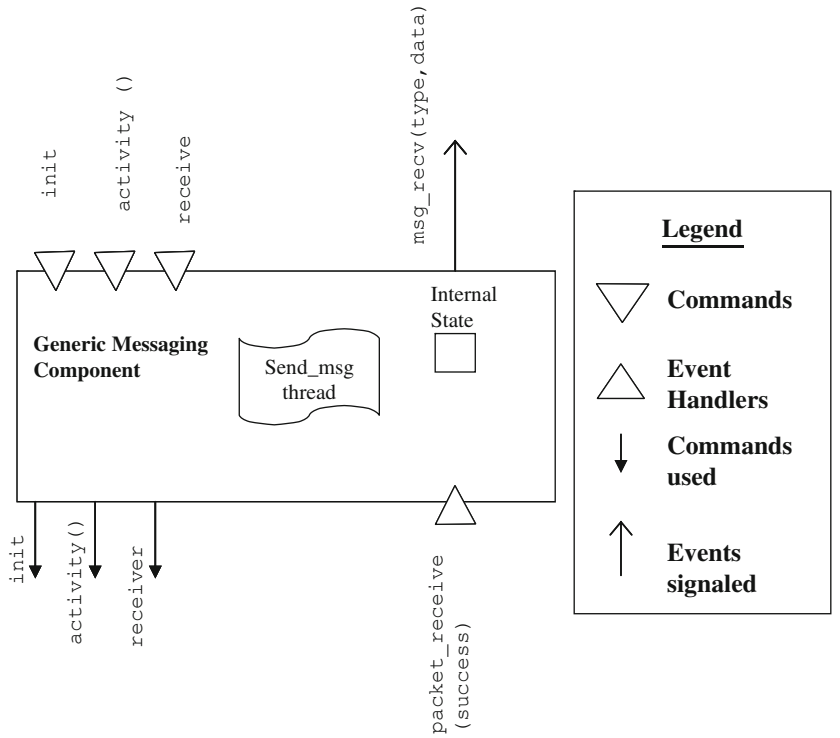

Fig. 4.2 TinyOS components [9]

4.1.1.3 ZigBee

ZigBee is a specification for a suite of high-level communication protocols using small, cost-effective, low-power digital radios based on the IEEE 802.15.4 standard for wireless personal area networks (WPAN), such as wireless headphones connecting with cell phones via short range. IEEE 802.15.4 defines the PHY (physical) and MAC layers; ZigBee defines the network, security, and application framework for an IEEE 802.15.4-based system. It works at 2.4 GHz frequency band. The transmission distance ranges from 10 to 100 m.[2]

ZigBee is intended to be simpler and less expensive than other WPANs, such as Bluetooth. It has efficient low-power connectivity and the ability to connect a large number of devices into a single network. ZigBee uniquely offers low-latency communication between devices without the need for network synchronization delays as required by Bluetooth. ZigBee targets at applications that require a low data rate, long battery life, and secure networking. It creates robust self-forming, self-healing wireless networks. The network connects sensors and controllers without

[2] http://www.meshnetics.com.

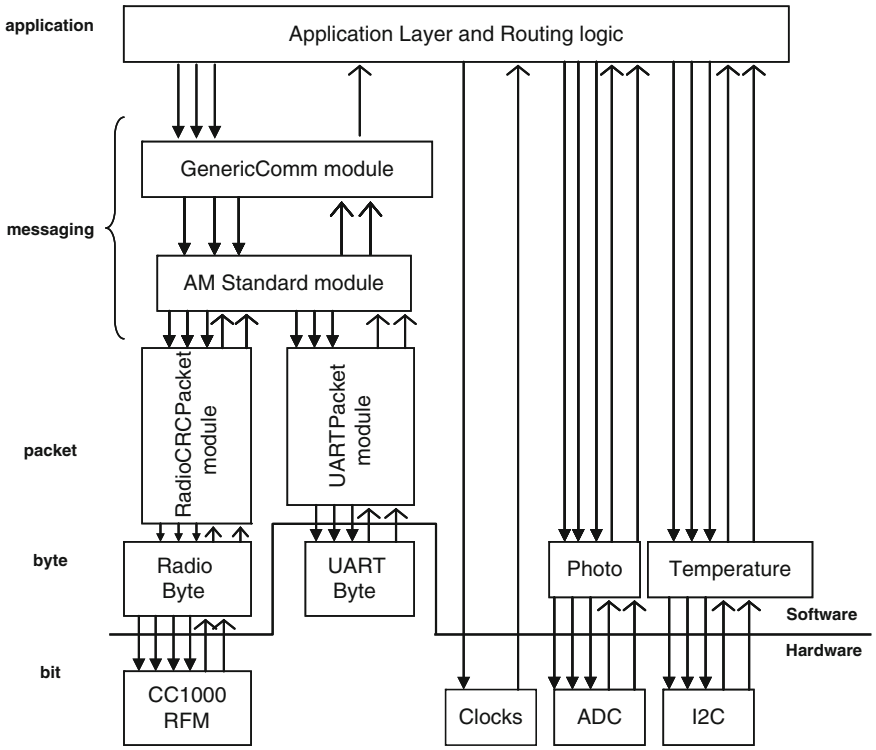

Fig. 4.3 Sample TinyOS application framework [9]

being restricted by distance or range limitations. The ZigBee protocol supports three network topologies, the mesh, the star, and the cluster tree. ZigBee networks let all participating devices communicate with one another, and act as repeaters transferring data between devices. Being mostly used in static networks with many, infrequently used devices, ZigBee allows for smaller packets over large network. The ZigBee architecture stack model (Fig. 4.4) at ∼32 kb represents an order of magnitude increase in complexity over the previously described TinyOS software environment.

4.1.1.4 Bluetooth Low Energy

Bluetooth Low Energy is an open radio technology for small devices. It addresses devices with low battery capacity and is easily integrated with traditional Bluetooth.[3] The transmission distance is 10 m with a frequency band at 2.4 GHz.

[3] http://www.bluetooth.com.

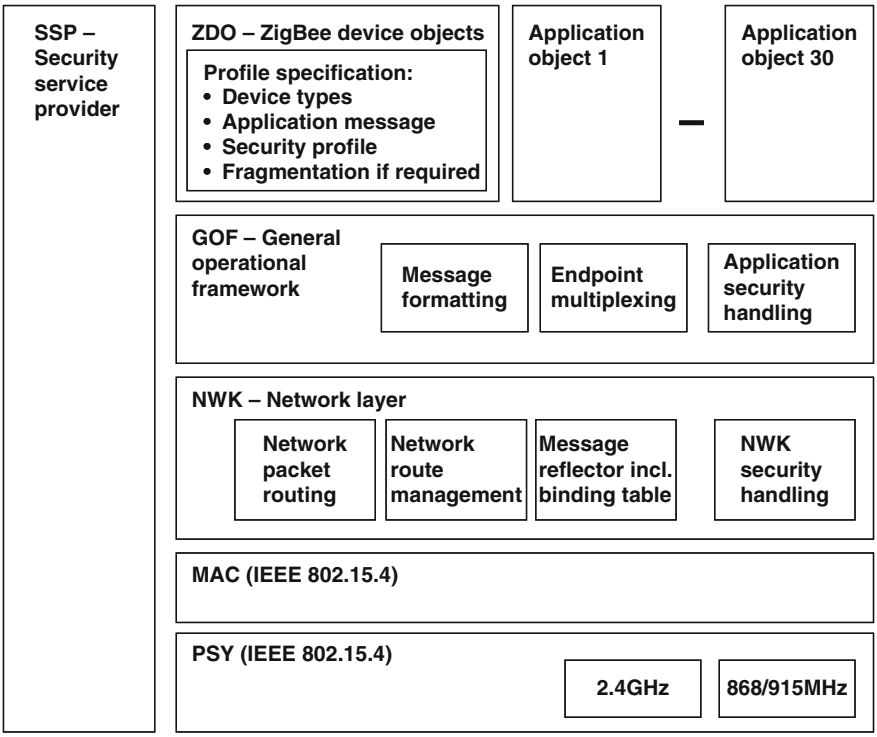

Fig. 4.4 ZigBee stack architecture [10]

The technology uses fewer frequencies (3 rather than 32) compared with standard Bluetooth, resulting in low-power consumption when connected. CSR (a global provider of personal wireless technology) has demonstrated that transferring data packets using Bluetooth Low Energy is 50 times faster than using standard Bluetooth, meaning that the devices consume as little as 1/50th of the power. In addition, in the established connection, the Bluetooth Low Energy devices used 1/10th the power required by standard Bluetooth.[4] End products based on this technology will have battery standby time measured in years, not just days or weeks.

Bluetooth (Fig. 4.5) generally handles larger packets over small ad hoc networks. Compared to ZigBee, it represents at 250 kb for the protocol stack, again an order of magnitude increase in complexity.

4.1.2 Power Management

Not all nodes in a distributed sensor network need to be in play at all times. Where possible a subset of nodes can be powered-up, other nodes can remain

[4] http://www.csr.com.

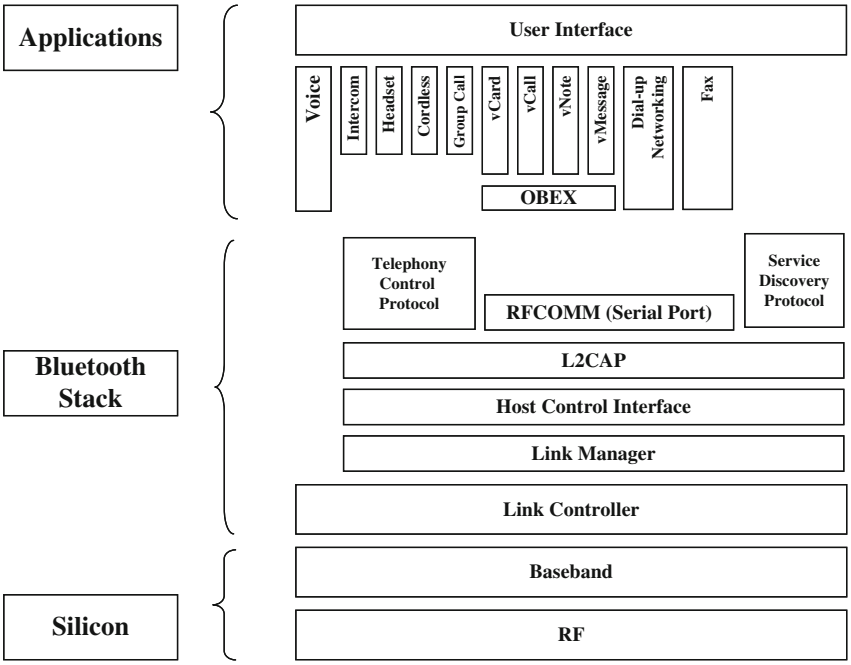

Fig. 4.5 Bluetooth[5]

powered-down. In addition, other approaches [11] to conserving energy could be applied. Power can be conserved at various levels:

- "PERT" [12] for power – System-level tradeoffs along critical computation paths and among general-purpose computer, digital signal processor, adaptive computing system, and ASIC implementation alternatives
- "Efficient" mathematics – Proper algorithm selection (for instance, choice of CORDIC [13] over multiply-add operations for FFT), incorporating Flynn's [14] notion of computational work
- "Optimizing" compiler – Optimization approaches specific for power, instead of for performance
- Device mapping – Space/power tradeoffs at the ACS and ASIC device level
- Cell/core power take management – Traditional power-down techniques as applied in hand-held equipment
- Design techniques – Circuit level design alternatives such as adiabatic wiring, dynamic voltage scaling (DVS), and asynchronous operation

Such measures mostly care of the energy consumption of a basic node. In a sensory network, such is a given and the focus will move to system architectural and

[5]The ZigBee Alliance, Inc.

implementation aspects. A typical example is in the passing of a single piece of sensory information within the network. One can discern the following options:

- Continuous – Continuous measurements do not need much local storage, but the fact that most of the transmitted data are likely to be about the same makes that a lot of energy is spilled.
- Episodic – Episodic measuring averages the values over a defined period before it is passed on. The data are stored locally, which eliminates the need for constant transmission and therefore saves energy. This is allowable for non-critical situations.
- Normal – Alarm measuring gives an a-period signal unless the data pass a defined threshold. In this sense, it is somewhere between continuous and episodic. The latter option remains needed as otherwise a nonfunctioning sensor might stay unnoticed and therefore a much needed alarm may be missed.

The good thing is, however, that these three principles can easily be handled by a single controller (Fig. 4.6).

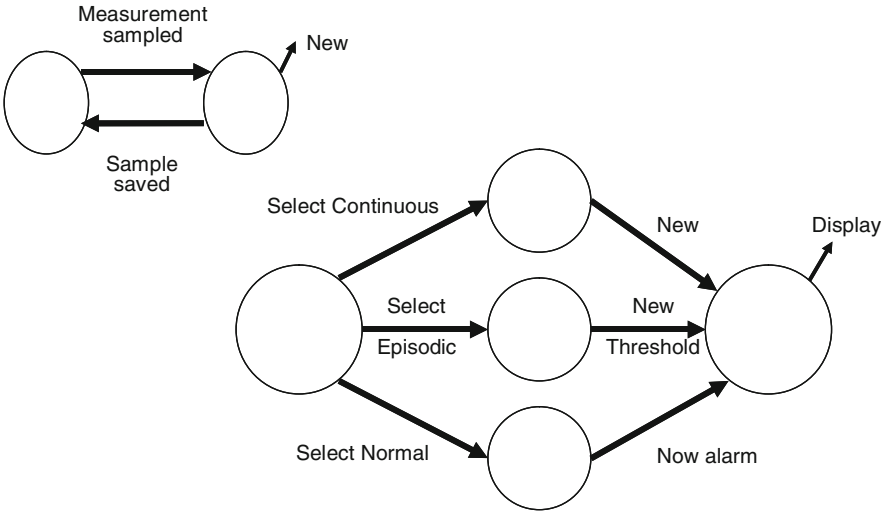

Fig. 4.6 A layered power management controller

Processing-versus-communication power consumption tradeoffs also need to be addressed at the network level. It assumes that transmission has set-up, run-time and close-down costs, also in energy consumption, and that they have characteristics different from computation. Consequently, energy reduction can be reached not only by choosing a suitable implementation for each but also by considering the combination. Sometimes it seems reasonable to have intelligent sensors, ones that do more calculation themselves and so reduce the need for communication.

4.1.3 Power Harvesting

Nonstationary and even stationary sensor nodes in a personal-area network will greatly benefit from the application of power harvesting technology and the concomitant power management techniques. Many forms of power harvesting techniques (Fig. 4.7) have been proposed and developed from piezo-electric [15], solar photovoltaic, thermo-electric, to micro-electric-mechanical systems (MEMS) devices, and electromagnetic generators. Figure 4.8 illustrates some aspects of on-body power generation possibilities.

Technology	Power Density [μW/cm^2]
Vibration - electromagnetic	4.0
Vibration - piezoelectric	500
Vibration - electrostatic	3.8
Thermoelectric (5°C difference)	60
Solar – direct sunlight	3700
Solar - indoor	3.2

Fig. 4.7 Example power densities of energy harvesting mechanisms [16]

4.2 Personal-Area Networks

Sensors for wellness assessment can be provided in clothing/body preferably with the application of power-harvesting technology. Integration of the data collection and analysis for manageable reporting will be crucial. Even more life-critical than the current practice in industrial automation event reporting, efficient false-alarm management needs to be provided in order to minimize nuisance reporting.

The wireless communication protocols discussed in the previous section are very popular for Body Sensor Network. They are low-power optimized protocols for battery-powered sensor nodes. The appropriate choice depends on the specific application, which differs by function, compatibility, and cost. Passive RFID technology is a good candidate for a sensor platform, since it avoids the significant power consumption.

4.2.1 Body Sensor Networks

While Wireless Sensor Networks (WSN) technology has evolved for a broadening range of applications, it does not specifically tackle the challenges associated with human body monitoring [18]. The human body consists of a complicated internal

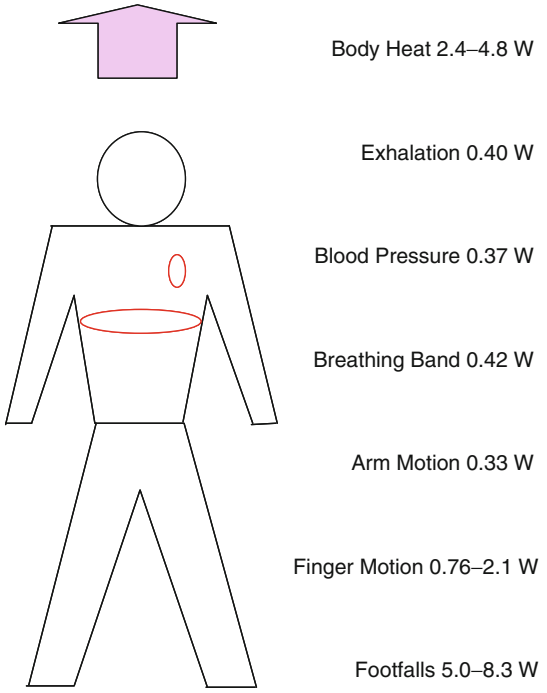

Fig. 4.8 Body-area watt-level available [17]

structure that responds to and interacts with its embodiment. Attaching these sensors on the skin, as well as implanting them into tissues, may achieve human body monitoring using a network of wireless sensors.

Compared with challenges faced by WSN, the human body requires a different type and frequency of monitoring. Therefore, it needs a specialized network, the Body Sensor Network (BSN), a term first coined by Prof. Guangzhong Yang of Imperial College in London.[6] Table 4.3 lists some of the differences between WSN and BSN.

In Yang's concept, a number of sensors is attached to the human body. Each sensor has a sensing part, a processor, a wireless transmitter, and a battery. A local processing unit controls wirelessly all the sensors. After capturing the data, the sensor node first carries out low-level processing, and then wirelessly transmits the information to the Local Processing Unit (LPU). In this way the data from all the sensors are collected by their respective LPUs, processed further, and then the information is seamless shared with home, working, and hospital environments.

[6] http://en.wikipedia.org/wiki/Body_sensor_network.

Table 4.3 Different challenges faced by WSN and BSN (partly taken from Yang) [18]

Challenges	WSN	BSN
Scale	As large as the environment being monitored (meters/kilometers)	As large as human body parts (millimeters/centimeters)
Node number	Great number of nodes required for accurate, wide area coverage	Fewer, more accurate sensors nodes required (limited by space)
Node function	Multiple sensors, each perform dedicated tasks	Single sensors, each perform multiple tasks
Node accuracy	Large node number compensates for accuracy and allows result validation	Limited node number with each required to be robust and accurate
Node size	Small size preferable, but not a major limitation in many cases	Pervasive monitoring and need for miniaturization
Variability	Much more likely to have a fixed or static structure	Biological variation and complexity means a more variable structure
Event detection	Early adverse event detection desirable; failure often reversible	Early adverse events detection vital; human tissue failure irreversible
Dynamics	Exposed to extremes in weather, noise, and asynchrony	Exposed to more predictable environment, but motion artifact is a challenge
Power supply	Accessible and likely to be changed more easily and frequently	Inaccessible and difficult to replace in implantable setting
Power demand	Likely to be greater as power is more easily supplied	Likely to be lower as energy is more difficult to supply
Wireless technology	Bluetooth, ZigBee, GPRS, wireless LAN, and RF already offer solutions	Low-power wireless required, with signal detection more challenging
Data transfer	Loss of data during wireless transfer is likely to be compensated by number of sensors used	Loss of data more significant, and may require additional measures to ensure Quality of Service

The Sensor Platform is defined by the selections made for sensors, processing module, and wireless transmitter/receiver module. Individual sensor nodes in the BSN are not easily wired to a power supply. From another point of view, a wired power supply could also limit the functional efficiency of the WSN, such as with implantable sensors. So generally, sensor nodes in the network have to rely on on-board batteries. Accordingly, the power efficiency of any proposed sensor platform is a very important figure-of-merit as a long operation time is usually desirable. Therefore, low-power technology is a main topic. The main consumers of energy on the sensor platform are the controller, the radio front ends, to some degree the memory, and, depending on the type, the sensors [6].

The BSN node is expected to be accurate and robust as the number of sensors attached to the body is preferably limited. Each node should perform multiple tasks. Therefore, the sensor platform should be suited to multiple sensors.

4.2.2 (Serious) Gaming

Serious games have been defined by Ben Sawyer of the Serious Games Initiative[7] as:

> any computerized game whose chief mission is not entertainment and all entertainment games which can be reapplied to a different mission other than entertainment.

Serious games have been proposed and developed in the areas of, for instance, medical rehabilitation ("rehabitainment"), preventive medicine, and educational training/simulation.

For its future, we envision the development of an integrated educational process based on the notion of Serious Games-in-the-Cloud, which is the application of Serious-Games-as-a-Service. This gaming orientation will serve as an introduction of concepts such as integrated sensors, cloud computing, industrial applications of serious games, educational notions of "Games for Learning."[8]

Serious games have been proposed for training sessions running on individual client processors. We envision that it is possible to up the (serious) "game" by the introduction of the additional introduction of distributed sensors, multi-participants, the Internet, and subsequently the Cloud [19].

4.2.3 Business- and Edu-tainment

Similarly, the division of labor between the cloud and the respective thin-client participants will become highly effective for socio-metric data collection and analysis purposes. The MIT Media Lab's socio-metrics Smart Badge [20] project (Fig. 4.9) illustrates a set of distributed sensors (on respective persons) within a physically confined space (an exhibition). The collector traverses on the outside of the confined network and observes (and reports) its information/awareness (business interest and meetings) content. Other examples of this topological configuration are the protection of buildings, schools, and responders on an emergency site.

More specifically,

> MIT researchers tracked people's social interactions at a conference gathering using a smart badge that incorporated an infrared sensor to gather data about face-to-face interactions, a wireless radio to collect data regarding proximity to other badges and send it to a central computer, an accelerometer to track motion of the participant, and a microphone to monitor speech patterns. At the event, the data from the infrared sensors was wirelessly transmitted to a computer that crunched the numbers, producing a real-time visualization of the event's social graph [20].

[7] http://www.seriousgames.org.

[8] Games for Learning Institute, http://g4li.org.

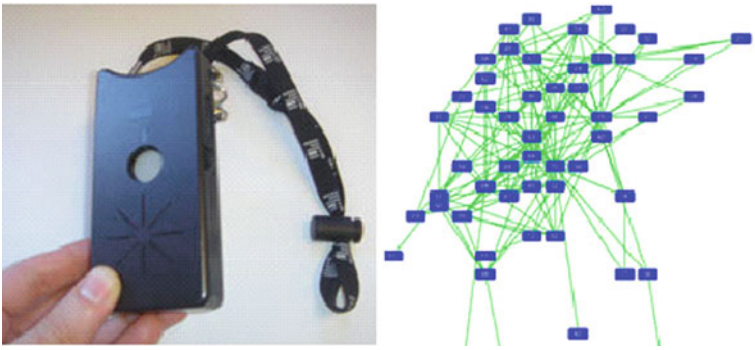

Fig. 4.9 MIT sociometrics

4.3 Monitoring and Observation

Situational awareness is more than just finding your location on the map. It has everything to do with observing the environment and being monitored by the environment for the good and safety of the individual.

4.3.1 Wear-Free Transmission

Wear-free transmission of signals can make a big impact on the reliable operation of machinery, such as in home, factories, automobiles, helicopters, and airplanes. Wear-free operation increases the overall reliability of the system, since less mechanical parts are involved. It will facilitate a dramatic reduction in assembly, testing, or generally in manufacturing costs.

WISE-COM (Wireless Interface for Sensors and Actuators), a proprietary [21] wireless communication standard, provides for reliable short-range interconnect of hundreds of devices in a factory environment. This will be useful especially in machine construction and operation, where cables move and tools are frequently changed. Wear-free transmission will mean that signal cables in production systems, which are prone to malfunction, will become obsolete.

Communication operates in the 2.4 GHz (ISM) band. It uses frequency hopping, with a total of 79 frequencies. One base station, or I/O module, supports 120 sensors or actuators in a range of 5 m. The protocol allocates each sensor or actuator a specific time slot and frequency for its transmission. This, combined with frequency hopping, avoids collisions.

The most unusual feature of the system is the way the individual wireless nodes receive their power to operate. Power is supplied by electromagnetic coupling, like a giant transformer without the iron core. The nodes are surrounded by primary loops in a (typically maximum) 6 m $\times$ 6 m $\times$3 m box shape, which produces a 120 kHz field. This induces current in small secondary coils in the sensors and actuators. The

amount of power induced this way ranges from 10 to 100 mW. Wireless power at a distance [22], in general, will solve a lot of problems in short-range transmission, ranging from not needing batteries to not needing cabling for power.

4.3.2 Care, Comfort, and Concern Monitoring

We have shifted to the related but more accessible domain of Home Automation to test concepts involved in the general collection of data in a stationary setting, such as are found in buildings. The house of the future will be Caring, Comforting, and Concerned (C3). Such functions need to be provided from a dynamically changing environment and reliably, as well as transparently, personalized for each individual in the house separately.

A house is initially meant to provide protection to the people sheltering inside for the dangers that lurk outside. But gradually it becomes more than only concerned. In the course of time, it has got filled with furniture and later also with audio and video devices to provide comfort. With the coming of personal healthcare devices, the notion of caring will be added. Patient and elderly monitoring are specific cases of C3.

The house is filled with objects. But few of such objects have relevance by themselves. Mostly, objects support a function. This function is shared with other objects, but not necessarily in the same way. A chair is to sit, but an office chair is also to give support during work while a couch is rather to relax. Nevertheless, one may get some comfort by sitting in an office chair if the couch is not available. On the other hand, the function is not confined to a fixed set of objects. One can sit on any object that provides a degree of support. This makes sitting not a mere concept in the compositional hierarchy, but an embedded function.

Another way to look at the situation is to notice that the relations between objects have a compositional nature by structure rather than by behavior. Office chairs are to be used together with an office desk. But office work is not identical to this. Where the structural relation (or relation by place in space) can be viewed as the syntax of ordering furniture, the behavioral relation provides meaning and can be seen as the semantics.

The home provides services to persons living there. Those services start at the door and are privileged to friends and relatives. Awareness about the people and their presence creates the basic home personality. The room needs to know about the persons and the occupancy of the other rooms to play the right kind of music at the right moment. Room temperature needs adjustment to age groups. Occasionally, access rights have to be enforced for parts of the house.

Typically, a gateway controls whether a room is entered or left. Sensors may be malfunctioning. This may be a simple lack in discriminating power, like: was the dog passing or a little child? The sensors may also be broken. But even when they function, such sensors provide no identification. We had identification at the door, but once we loose track such is gone.

Movement detection [23] and voice recognition can support the tracking, as it bridges the gaps between the gateway sensors. But eventually the person has to be approached more actively, i.e., addressed to get confirmation about his/her location and identity. Such actions are indicated in order in Fig. 4.10, but they actually take a combined effect.

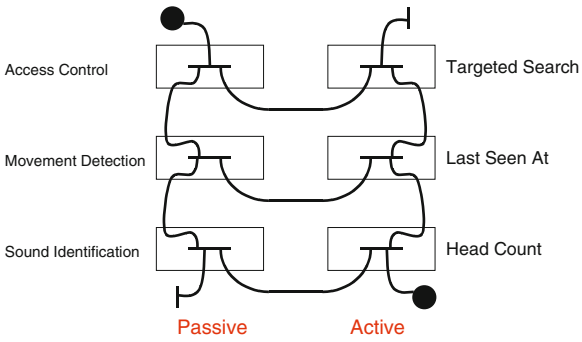

Fig. 4.10 Hierarchical layering of awareness

As illustrated, two neural networks are used that interact over a database. For instance, when a person's access to a room is noticed, it is administrated because in the end "what comes in has to go out." Also the position of persons is administrated and corrected every time he/she moves, because we need to know where to look when we are afraid that an accident has happened. And so on.

The basic house functions can be based on this constant awareness. For instance, music can be personalized to the person, his/her location, and historical preferences. But probably the most important consequence is accountability, which makes that, among others, imminent danger to the person's health can be noticed and brought to the attention of people that can help out.

Reliable collection and management of vital signs in transit, in emergency care, in hospital, and in assisted living at home facilitate general wellbeing. The continuous tracking of eating habits, (weight), blood pressure, glucose levels, and sleep times might result in a database from which interesting conclusions in regards to well-being can be analyzed and deduced. In addition to vital signs, data related to body position (such as fallen-and-I-cannot-get-up) and environmental conditions need to be provided.

4.3.3 Security Surveillance

Vision plays a major role in a number of safety issues, involving persons, objects, as well as environments. Persons move around under chaotic traffic conditions, being actively involved as driver or passively by walking around. Such chaotic conditions do not only involve traffic on the streets but also waiting rooms and shops. Here, it will pay to safeguard persons for accidents, potentially involving other persons.

Camera surveillance has long been applied to record situations as shopping without paying and robbing the bank. Nowadays, cameras are used for pre-active control as to detect behavioral patterns that indicate an incident to occur. In modern city centers, this enhances the effectiveness of police surveillances.

Also objects can provide a threat. Both in public buildings as in private areas, safety measures are nowadays necessary to preclude accidents that involve vehicles. Mostly there is just a simple message, spoken by a speech synthesizer that is either triggered by the time or by the situation. With the coming of cheap vision means, more and more the vehicle becomes autonomous, it is able to circumvent any (human or non-human) blockage. Nevertheless, we are still far away from the robot self-defense measures as described by Azimov [24].

Lastly, a security threat can result from unlawful conduct, such as the dumping of toxic waste. Cameras provide for a cheap means to guard remote and isolated locations, where a manned check-point would be too costly, and also to handle crowded situations where a guard would just hold up the passage. This shows that the benefit of computerized vision is not restricted to size, but covers also issues in energy and attention consumption.

Overall, we are concerned here with gate control: the enforcement of access limited to just a small number of persons. In normal life, a key gives access and any key bearer can therefore come in. If anybody else tries (or even manages) to get in, such might be noticed by other persons in the house or by innocent bystanders: there is a degree of social check. In most professional situations, this is not sufficient as either the single trespasser might not be visible in the crowd or the entry has severe consequences. Moreover, keys are easily duplicated or might get lost, while furthermore the point of entry might simply be forced. In the process of allowing a person access to a physical or virtual site, three levels of security can be discerned:

- Verification, where it must be certified that a valid access request is placed. Such a request will involve a sign (a password or a secret handshake).
- Identification, where the access request is made by a person, who bears an identity token (a document or a card) that needs to be checked on validity for him/her only.
- Authentication, where it must be ascertained that the person does not only have the right for access but is also the person who he/she claims to be.

Protection of assets, such as air fields, becomes more and more multi-sensor-oriented, as well as will involve more global intelligence. The ORganic Immense InfOrmation Networks (ORIION) at the University of Amsterdam and TNO ICT illustrates a set of distributed sensors with a collector traversing the sensor network collecting data from within its communication reach. It represents a topological configuration appropriate for the protection of water, levee, dike, and harbor infrastructures with a mobile information/awareness collection system.

The ORIION [25] project:

> represents research, development and field test of a practical network and information technology that make large scale protective infrastructures as dikes, energy related infrastructures as pipelines and dams, and early warning systems for forest fires and river

pollutions safer and cheaper. These networks have in common that they should be able to cover very large distances; to function in rural areas, and that large parts of the network can be destroyed during disasters.

4.3.4 Environmental Monitoring

The geographically distributed Low Frequency Array (LOFAR) as been developed by an international consortium is the world's first fully software radio telescope. In the early 1970s, the radar concept was innovated by emulating the moving mechanical parts through the phased control of a matrix of ferrite antenna elements. The next step is the emulation by the phased communication between partial antennae placed in a 400 by 400 km area.

LOFAR is constructed from about 100 nodes, distributed over a wide area. They are arranged in three long arms (with 28 nodes each) and three short ones (14 nodes each). Each node has 80 antennae that operate on a 2 Gbit/sec data frequency. They connect to the node for a first computation and compaction to create a synthetic aperture array, and comes on the network with 160 Gbit/sec. Using glass fiber technology and optical time multiplexing, this leads to a communication speed of 2.24 Tbit/sec for the shorter arms and 4.48 Tbit/sec for the long ones to reach the main computer in Dwingeloo, The Netherlands.

The LOFAR Central Processing Platform CPP is typified by a large cluster computer. This cluster is inhomogeneous in nodal hardware since the processing and data transport demands vary for the successive stages of the applications. Part of the application will benefit from specialized hardware available for particular processing tasks. For example, the correlation task in an imaging type application can be very efficiently executed on (programmable) logic arrays such as an FPGA. Therefore, specialized hardware is added to the microprocessor-based computers. This extra hardware is then available for the application programmers as a kind of co-processor.

Each of the 100 LOFAR stations (Fig. 4.11) has to process the incoming data from the 100 antennae. Each antenna is not much more than a specially inverted-V dipole above a ground plane, making them both nondirectional and broadband. The main task of a station is to synthesize a focused beam from the 100 data-streams coming from the dipoles. The shape of the beam is determined in the system in the so-called beam-former. One requirement is that the shape of the beam can be switched extremely fast, this for example to create a blind spot in the beam that follows a passing satellite. The total data-rate from all the receivers of the station is 1.6 Gbyte/sec. The result of the processing is a 64 Mbyte/sec data-stream that is send to the central processor. Finally, the system also has to be highly flexible, allowing for adjustments after the system has been installed.

Another extremely audacious example in support of our described Sensors-into-the-Cloud approach has been provided in the problem statement (by Harold Stone) for the recent 2009 Workshop on Interconnections within High Speed Digital Systems in Santa Fe.

It envisioned a future weather reporting and modification system comprising:

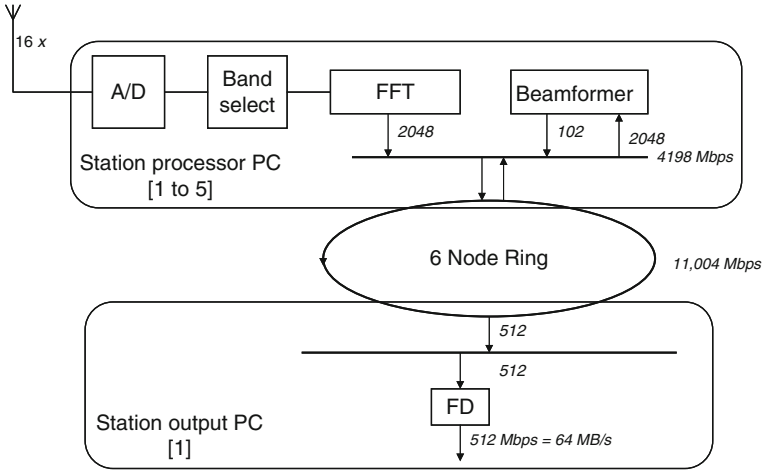

Fig. 4.11 Schematic view of a LOFAR node [72]

- A network of sensors on the surface of the continent that sample temperature, pressure, humidity, and wind velocity, and communicate these values with time of day and geographic location to a data center repository;
- Airborne sensors installed on commercial aircraft that collect temperature, pressure, humidity, wind velocity, time of day, altitude, and GPS coordinates;
- Cloud cover images from satellites;
- A data repository to hold the real-time information, and to collect historical data to enable simulation and analysis;
- A petaflop computer that can determine how and when to inject weather modification chemicals to attain specific results.

Clearly, these are very large data set applications.

4.4 Monitoring and Control

The personal environment can be adapted to personal needs. This environment can be a house, an office, a car, and therefore anything that needs to perform optimally to be enjoyable. A typical issue that has lately received much interest is the control of the environment to reduce energy consumption and therefore natural resources.

4.4.1 Smart Structures

Smart structures represent the use of sensors to collect data upon which decisions will be made that will impact the configuration and/or behavior of particular instrumented structures. The recently proposed US Department of Transportation (DOT) "Quick Highway Incident Detection and Incident Warning Systems" (Fig. 4.12)

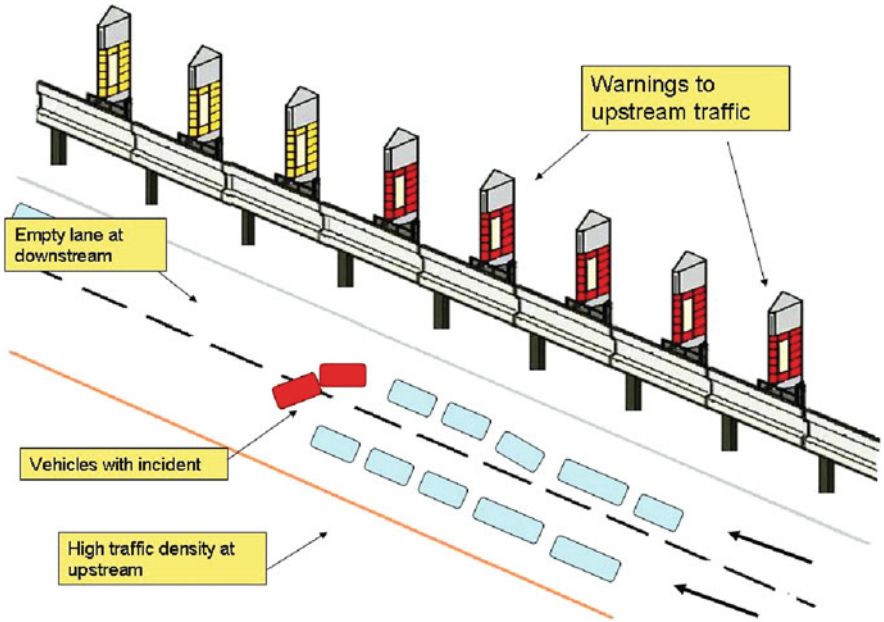

Fig. 4.12 Concept of operations, quick highway incident detection and incident warning system [26]

illustrates a set of distributed sensors with the collector traversing on the outside of the sensor network, observing only its boundary sensor nodes.

DOT states that:

> for this program, a new incident detection and warning system was proposed to address the issues of large spacing between detection stations along a highway. The basic concept was to use multiple low-cost detection stations with very small intervals. Also low-cost wireless communication devices and LED signs was to be co-installed with each detection stations. Since the stations are close to each other, it is feasible to detect and locate an incident very quickly, and also to use the LED signs to warn drivers for preventing secondary crashes. Low cost wireless short range communications can also be established through relaying from one station to another.

4.4.2 Traffic Control

Smart vehicles (Fig. 4.13) can be used on smart roads. This is not because they need special provisions in their embedding environment. For instance, the VITA vehicle mentioned in Section 3.2.3 is smart by itself. However, a number of collective functions need all swarm members to be equally smart, which is hard to enforce on a public road. Smart vehicles require large amount of instrumentation and information management in a largely nonstationary environment.

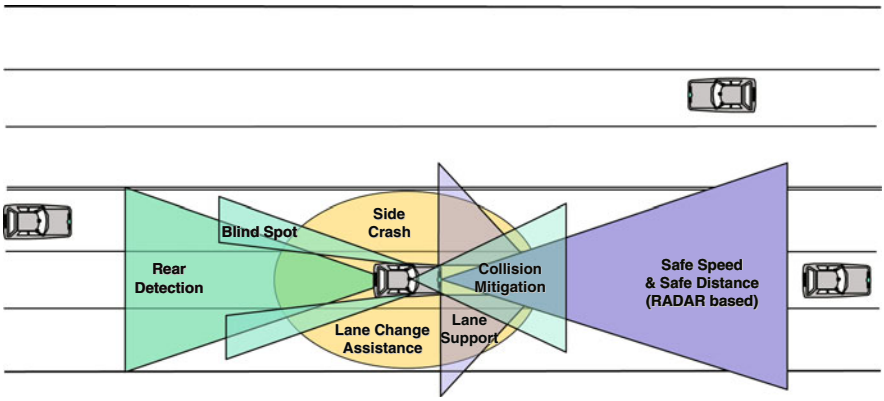

Fig. 4.13 Autonomous intelligent vehicle [27]

In the commercial world, companies shipping goods (especially valuable or perishable) face the problem of monitoring their goods (often on a smaller than ISO container level) in terms of handling, on time delivery, loss, and integrity. These actors often have transports that involve multiple modes of transport, as well as multiple carriers and potentially more than one container. These factors require a system that follows the cargo (boxes/pallets), instead of the unit carrying the cargo such as a truck, ship, or a particular container. In addition, we should consider city-wide traffic control based on widely distributed sensors and observations.

4.4.3 Smart Power Grid

The power grid especially with multiple power sources and drains needs to be smartly instrumented in order to be better optimized for cost and delivery. Accidents do happen. And while accidents are part of every day's life, it is important to create systems that can handle accidents without going berserk. This potential is based on two things: (**a**) the ability to detect something being wrong and (**b**) the potential to cure by either removing the origin of the problem or by working around it.

In infrastructure networks, systems cross vast distances, are highly nonlinear, and support strong interaction. To control such a system from a single authoritative node may not be practical as failures can spread unpredictably and instantaneous before coming to the attention of the control center.

The active management of conventional and unconventional power generation resources is also an example of the sensors-into-the-cloud opportunity. Large-scale distributed networks are not easy to control centrally. On 10 August 1996, the Western US power grid broke down and blacked out 11 US states and 2 Canadian provinces before being noticed, leading to a $1.5 billion damage. Apparently, central control only makes a system more vulnerable for the very problem it is designed to fix.

4.4.4 Industrial Automation

Industrial Automation is not an area with overly respect for science. It is about the fabrication costs of a product with defined quality, and few people care whether that is done according to the latest insights in control theory. Usually it is very late in adopting new technology.

Industrial Automation is also applied in an environment that is not necessarily spotless. Often it is oily and fat. Dust creeps in. You do not expect the cleaning lady to pass by here, and actually she rarely does. It is not because of this or at least not as a direct consequence, but it is rumored that much of the controls are in a bad condition. George Buckbee of ExpertTune claims that 30% of the control valves are having mechanical problems [28]. He continues that often such problems go unnoticed as the alarms are shut off to get rid of the flood of nuisance alarms (see Section 6.3.3).

A recent paper [29] on safety aspects of distributed sensor networks overviews the problems with model-based control in sensory networks, such as currently rolled out for the Electric Grid, and shows how networked intelligence can bridge between central and distributed control. It starts from the experience in industrial automation. This area gives more lessons on the potential threats in grid management.

Industrial automation is an area where special solutions to manufacturing processes have flourished for a long time. Only recently has the move been made toward Internet-connected Supervisory Control And Data Acquisition (SCADA) systems. A major reason for going in this direction is cost-cutting by using commodity solutions for partial problems. This has taken some time because the Internet is not real time. Even today real-time Ethernet [30] has a number of flavors and still a lot of work is needed to harmonize the different flavors into a single concept.

Unfortunately, a growing expertise in Internet technology is overshadowed by a large hacking experience already existing in that field. In a January 2008 press release, The Associated Press cites a source claiming that computer hackers have already managed to disrupt the electricity supply in multiple cities outside the United States [31]. The Homeland Security Department has attempted to stir the community with a video showing how hackers managed to blow up an enormous generator. Though the number of reported break-ins is small, it is clearly on the rise. Big control companies like Honeywell and Emerson aim to solve the vulnerability by the introduction of distributed control, but a good autonomous architecture is still not there.

4.5 Collective Intelligence

Every system displays behavior to the observer. For electronic systems, it is a design abstraction that once captured by a model hides the implementation. Large systems may easily involve an exploding number of states to built complex behavior and model synthesis becomes unwieldy. The problems become even more severe when

the electronic system is embedded in a reactive environment and behavioral model capture becomes a necessity. Occasionally, the model cannot comprise reality anymore and new behavior emerges. For such reasons, the creation of models from observations to detect, isolate, and diagnose faulty/correct behavior is indispensable [32].

The oldest interest concerns unintentional behavior in biological systems. Animals have an instinctive social behavior that involves eating, lactating, and reproducing. When a cow gets fertile, her activity pattern changes, which can be detected by a "stepcounter." The alternative visual monitoring has also been pursued using a point system based on a psychonomic Likert-type scale. Detection quality was low by the absence of suitable modeling, and the "stepcounter" is still the state-of-the-art. The newest interest is in intended behavior, especially ones with a criminal intent. It has been noted that robbers in public transport satisfy a few simple behavioral patterns: hang around the exits, frequently change seats, or close in on a potential victim. Where many public areas are equipped with Closed-Circuit TV (CCTV) systems, such situations are video-taped and can be used as evidence later on.

Increasing criminality and especially the threats of terrorist attacks have stimulated the interest in automated detection of abnormal behavior from CCTV streams. In a typical surveillance application, the number of cameras and the amount of captured video can be so large that a human operator cannot handle all the information from multiple input channels. Moreover, it is difficult for humans to watch video streams for a long time and remain able to analyze the data, detect all suspicious events, and respond to them in a timely fashion without missing anything. As a psychonomic approach will clearly not suffice, the patterns themselves must be modeled.

The key challenge is to extract relevant information from the scene, identify probably suspicious behavior and direct the attention of security personnel to the situation for further investigation. Unfortunately, it can be hard to automatically distinguish between normal and abnormal (or suspicious) events. In a place where everyone walks, running is an unusual behavior, while in places where everyone runs, walking is unusual. In our work abnormal events are defined as events that are not predictable due to time, condition, and location of their occurrence.

The popular approach in electronic control is based on the use of fault dictionaries [33]. Such assemblies of fault signatures are prepared for a limited set of assumed novelties under defined circumstances. However, the feature space is higher-dimensional for a typical surveillance system, and the observational variety is hardly limited. The cost of manual selection of algorithms and tweaking of algorithm parameters for each specific environment quickly becomes prohibitive for ubiquitous deployment of camera systems. Therefore, the key issue in automatic abnormal event detection is learning [33]. The surveillance system has to be generic and learn in a pre-defined manner any normal patterns of behavior for different types of objects detected in the scene, and build a model of normal behavior for the area under investigation. Abnormal events will then be detected when an observed event does not match the constructed model of normal behavior. Previous work in the area

of event detection varies widely due to different methods employed for learning, feature vector selection, and abstraction level of abnormality detection.

Motion is an important feature for abnormality detection, but motion alone may not be enough and other features may need to be used in combination with motion to achieve robust functionality. Different learning methods have been investigated in previous work by others. For example, Jodoin et al. build up an activity map during observation to effectively mask zones with high activity automatically from the event detection procedure, focusing on regions where activity would be unusual and possibly dangerous [34]. Zhang et al. use motion and color histogram features extracted from moving objects to train HMM (Hidden Markov Models) to build a reference model for usual events [35], while Owens et al. use Kohonen self-organizing neural networks to learn normal movement trajectories [36]. In Tehrani [37] FFNN (Feed-Forward Neural Networks) is used as the learning and modeling paradigm, and the applicability and performance of FFNN for motion-based abnormality detection is tested. In addition, the feasibility of implementing high-level real-time analysis of moving object behavior is studied.

Smart vision sensors for automated surveillance of public places can be based on Commercially-Off-The-Shelf (COTS) processors, but specialized image processors are required for massively crunching pixels into features while achieving real-time performance at a low-power budget. This poses additional restrictions on the implementation of the neural networks. Fully connected, monolithic neural networks have a limited capacity when implemented as a single digital ASIC, while they tend to give cache problems on COTS processors and usually do not agree with the fixed-point number representation popular for the cheaper embedded platforms. This suggests the use of heterogeneous platforms such as the smart WiCa (Wireless Camera) platform [38].

The WiCa platform is a high-performance low-power smart camera board developed by NXP Semiconductors (Fig. 4.14). The main processor on the WiCa board is the fully programmable and power-efficient Xetal IC3D chip [39], a 320-core SIMD (Single Instruction Multiple Data) chip with a peak performance of about 50 billion operations per second (50 GOPS) at its operating frequency of 80 MHz. This high performance is achieved through massively parallel operations, typically at the pixel level [40].

Fig. 4.14 NXP Semiconductors' WiCa wireless smart camera board

The WiCa board also contains one or two VGA color image sensors, an 8051 microprocessor for intermediate and higher level processing and control of global operations. A DPRAM (Dual Port RAM), is shared between the IC3D and 8051, and allows both processors to access the same data. Finally, WiCa uses an Aquis Grain ZigBee transceiver to communicate with a master computation unit or with other WiCa boards in case of a distributed camera network (Fig. 4.15).

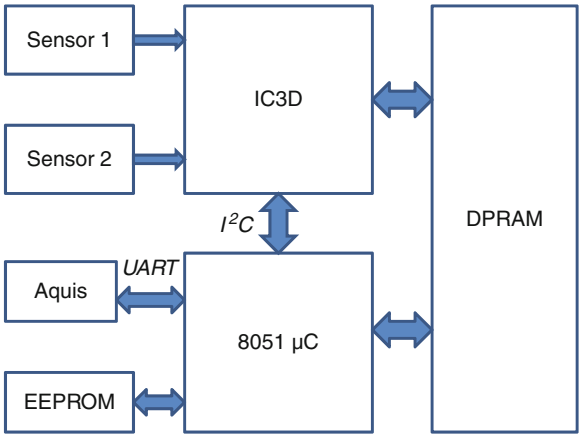

Fig. 4.15 WiCa block diagram

As monolithic neural networks are ill-famed for their learning problems, hierarchical structures appear advisable, leaving the question: how to build and distribute this hierarchy on the target platform.

Behavioral modeling has all the problems of time-series prediction. On one hand, series expansion is required to reach sufficient accuracy. This is similar to the series expansion used in mathematics, where a complex function is approximated through an iterative procedure [41]. On the other hand, it is also known that recursive computation allows for errors creeping in from continuous rounding. This is usually known as the finite-length effect and can make predictions for far into the future very unreliable. A typical example in artificial walking is discussed in Venema et al [42]. It is important to have a good grip on these phenomena as a neural network model can represent chaotic systems (e.g., the well-known Verhulst population model) [43]. Allowing for chaos effects in the model learning and/or execution will definitely reduce its potential for surveillance applications.

4.5.1 Collision Avoidance

In the early 1990s, the European automotive industry started their first collaborative research program, called Prometheus (see Section 3.2.3 for more detail). The next example comes from the PRO-CHIP project within that program. In a pre-decessing EU project, a large collection of data on a driving car was made. This collection is by

itself not usable as it describes mostly just driving along the road without changes in speed and/or direction. Data editing was necessary to eliminate catastrophic forgetting in adaptive networks, or prevent implicitly giving disproportionally high weight to simple behaviors in non-adaptive networks. For example, driving straight-ahead for a long time tends to destroy all the knowledge about taking a curve. This editing on the set can be a long and cumbersome process, as a lot of tests are necessary to check on the sensitivity of the results to the partial removal of training data.

A better approach is to define a closed set of typical driving situations, from which during learning a random selection with small random variations is made (Fig. 4.16). Different driver characteristics can be obtained by judiciously balancing the number of specific examples in the training set. Such behavior has been tested in a simulated environment with a variety of uncontrollable disturbances, such as side-wind while turning on a slippery road [44].

A neural network is a nonlinear system. Such a system may show chaotic behavior because of sensitivity for initial conditions. In the driver model this will lead to different behavior for every different starting position of the car. It is interesting to observe that this is true on the detailed level, but on the more abstract level of reasoning as introduced by the driving examples in Fig. 4.16, it has only occurred

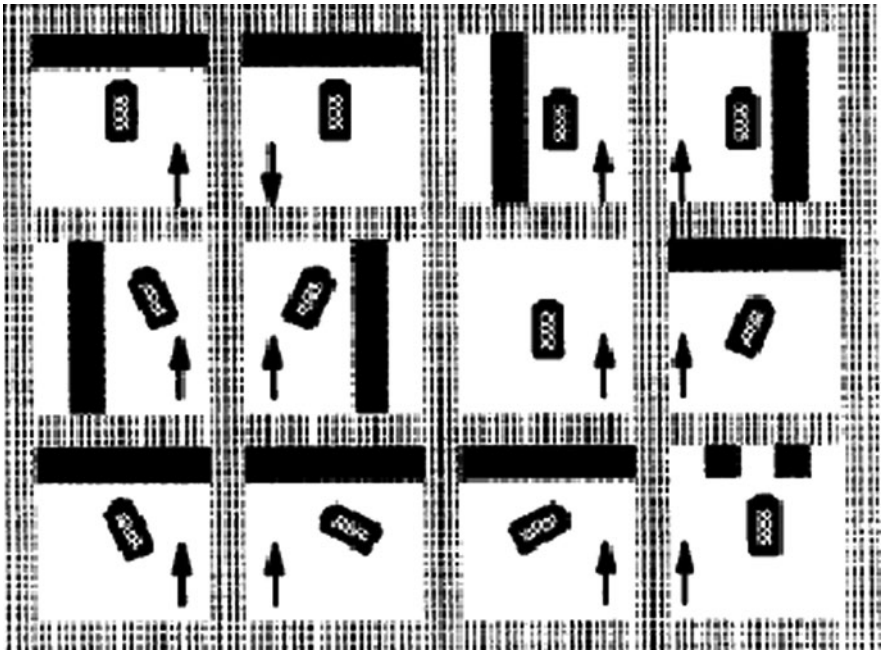

Fig. 4.16 Basic driving situations[9] [73]

[9](not shown is the 13th situation, where no blockade is near)

when the driver had not to take a curve or carefully avoid obstacles, such as the side of the road. Why is that?

The answer is that the curves and blockades put the driver in a straitjacket. There are far fewer ways to leave the curve unharmed than to enter it. In other words, the curves serve as points of reference for path convergence such that with every curve, accuracy gets largely restored. This becomes visible in car behavior when comparing a route with an occasional curve with a route with many curves.

For instance, after a U-turn all cars are almost at the same trajectory. The same reasoning applies to obstacle avoidance. This has two important consequences. First of all, it seems that trajectories are a meaningful signature of behavior. Secondly, it seems that behavior funnels acting as points of reference can restrict chaotic systems. In the experiments, both the simulation as the limited precision implementation has shown the same repeatable and deterministic trajectories. Figure 4.17 shows a typical testing ground. The neural control system has been realized on a Gate Array as a 14-20-2 feed-forward network [44].

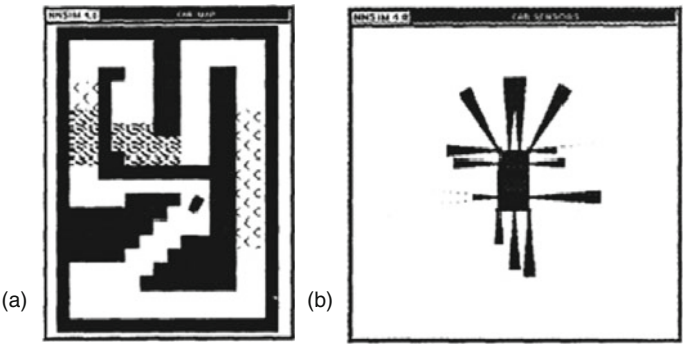

Fig. 4.17 (**a**) Screen showing a driving environment with wind (*arrows*) and ice (*shaded roads*) and (**b**) Screen showing the actual position, direction, and range of the thirteen distance sensors on the car

Avoiding collisions is one thing, driving properly another. This can be achieved through the three-tier hierarchy, discussed in Section 3.2.4. In the bottom layer it is trained to drive on the right side of the street. With a next set of examples, we introduce speed corrections at crossings (including the discrimination between streets and alleys). Finally in the top layer, street conditions like traffic signs, driving directions, and other traffic are dealt with. This achieves a model of a single car.

Continuous correct driving has been demonstrated at the 1991 PRO-CHIP meeting in Darmstadt. On one computer the street map of Stuttgart was shown to monitor the position of all cars involved. Then, for each car, the model was run on a separate computer where driving destination can be entered.

4.5.2 Tracking in a Billiard System

Based on the aforementioned experience we now simulate a so-called "Sinai billiard table" on an embedded computer vision system with limited dynamic range in its number formats (Fig. 4.18). The Sinai billiard system was introduced by Yakov G. Sinai and shows long-term unpredictability in a nonlinear dynamic system with unbounded accuracy [45]. In this system a (simulated) billiard ball is rolling on a square table in straight lines and changes its direction when it hits the walls, corners, or the circle in the middle of the table according to the usual laws (angle of incidence equals angle of reflection).

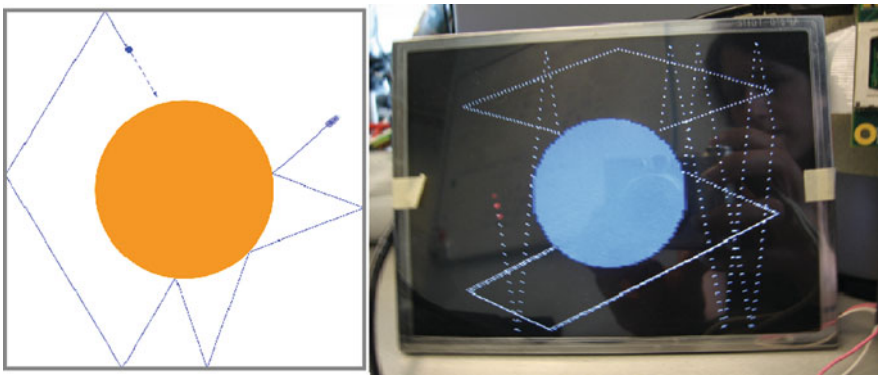

Fig. 4.18 Sinai billiard ball simulator

Having the position of the ball, its speed and direction, and the knowledge of its environment (the distance to the closest obstacle), the motion pattern can be learned in different situations separately as the ball faces different obstacles. The input flow vector to the selected neural network (NN) consists of the ball coordinates in two subsequent frames $f=\{x(t–2), x(t–1), y(t–2), y(t–1)\}$. In addition, coordinates of the ball in the next frame $T = \{x(t), y(t)\}$ are given to the networks as target outputs.

In the first case, a multilayer feed-forward neural network (FFNN) learns the linear movement of the ball (Fig. 4.19a), which can be expressed by the formulas $x[t] = 2 * x[t–1] – x[t–2]$ and $y[t] = 2 * y[t–1] – y[t–2]$.

A FFNN with a 4-2-2 topology is selected and trained off-line using supervised learning, with the training data set prepared using a MATLAB model of the Sinai billiard ball simulator. It is typical for a convolutional network that the local problem admits a simple construction (here two linear neurons would suffice), but a more generic set-up achieves an easier integration within the overall system. After training, the network has a good generalization results and acceptable error rate (Table 4.4).

For the next condition, a FFNN with similar topology is trained to learn the ball motion pattern after collision with the wall and corners (Fig. 4.19b, c).

$$m[t] = 2^{*}m[t-1] - m[t-2]; n[t] = n[t-2]$$

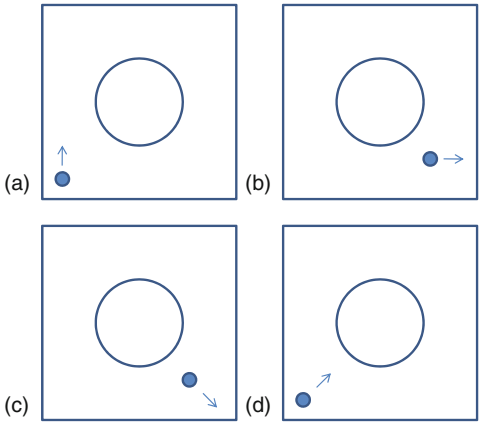

Fig. 4.19 Different motion patterns of the ball

Table 4.4 Training and testing results for linear movement pattern

Training		Test		
Set size	Error	Set size	Iter.	Error
200 data items	0.08%	200 data items	3000 epochs	0.09%

Table 4.5 Training and testing results for border collisions

Training		Test		
Set size	Error	Set size	Iter.	Error
200 data items	0.1%	200 data items	4000 epochs	0.09%

Here m=y and n=x when the ball collides with vertical walls and m=x and n=y when ball collide with horizontal walls. Error rate in both training phase and test phase is small and the networks learn the motion pattern successfully (Table 4.5).

Finally, the motion pattern of the ball after collision with the circle in the middle of the table is modeled by another FFNN, with 4-6-4-2 topology. Here the ball follows the following nonlinear pattern:

$$\text{Dist} = \text{sqrt}((\text{Xc} - x[t-1])^2 + (\text{Yc} - y[t-1])^2);$$

$$nx = (x[t-1] - \text{Xc})/\text{Dist}; ny = (\text{Yc} - y[t-1])/\text{Dist};$$

$$\text{d}p = -nx^* \text{d}x[t-1] + ny^* \text{d}y[t-1];$$

$$\text{d}x[t] = 2^* \text{d}p^* nx + \text{d}x[t-1];$$

$$\mathrm{d}y\,[t] \;= -2^{*}\mathrm{d}p^{*}ny + \mathrm{d}y\,[t-1]\,;$$

$$x\,[t] \;= x\,[t-1] \;+ \mathrm{d}x\,[t]\,; y\,[t] \;= y\,[t-1] \;+ \mathrm{d}y\,[t]\,;$$

All the above networks are trained and tested successfully with an error rate of around 1.4% (Table 4.6). This larger error is caused by training this network as a single structure instead of as an assembly of two networks, which it actually is. This confirms earlier observations that monolithic networks need to be avoided for reason of learning efficiency [46].

Table 4.6 Training and testing results for circle wall collisions

Training		Test		
Set size	Error	Set size	Iter.	Error
50 data items	0.16%	50 data items	11,000 epochs	1.4%

4.5.3 Trajectory Modeling

An imager-based surveillance system initially needs to be able to observe people and/or vehicles attempting to illegally cross physical borders. A network of imagers can improve the basic functions of unambiguous detection, identification, and tracking of intruders to the perimeter, with even greater range and lower false alarms.

In order to establish the trajectory of an object through space, it has to be identified and tracked through a sequence of images. There may be many objects needing evaluation. They may move independently or as a part of a collective, while some may finally prove to be irrelevant.

4.5.3.1 Moving Object Extraction

Typical event detection tasks in an automatic surveillance system include moving object detection, object tracking, trajectory analysis, and finally event detection and reporting (Fig. 4.20) [47]. In this section, the main building blocks of the system are presented and the proposed algorithms for each block are discussed. All the vision algorithms have been designed or modified with a massively parallel SIMD processor (e.g., IC3D on WiCa board) in mind.

Pixels in the image that are relatively constant over a specific time-span are labeled background (BG), and the other pixels are labeled foreground (FG) or moving object pixels. A simple method to find FG pixels is the average background model [48]. In this method the BG is modeled as the mean value of images in previous frames. FG can then be obtained from the difference between BG and the incoming frame as

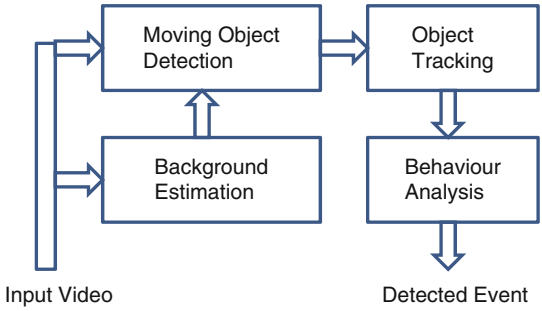

Fig. 4.20 Main framework in event detection task

$$BG\,(x, y, t) = a \times I(x, y, t-1) + (1-a) \times BG(x, y, t-1)$$

and

$$FG(x, y, t) = \begin{cases} 0 \ldots if\,|\,I\,(x, y, t) - BG(x, y, t)\,| & < Th \\ 1 \ldots if\,|I\,(x, y, t) - BG(x, y, t)\,| & \geq Th \end{cases}$$

where $I(x, y, t)$ represents the intensity of each pixel at location (x, y) and time t and α is the learning factor in the range of $0 < \alpha < 1$, typically set at 0.05.

By selecting a smaller α, the BG will adapt more slowly to changes in the view. This method performs well in separating FG from BG if the background changes only slowly. However, it leaves a trail for moving objects.

In order to have a fast method with low complexity and low memory requirements along with the ability to separate FG completely from BG, the average background model (with some changes) is employed. Using a relatively big α =0.5, the background adapts fast, leaving no significant trails, but some parts of the object might be interpreted as BG. To overcome this problem, the interior pixels of all objects are assigned to FG by a hole-filling method (Fig. 4.21). At the same time that the image is scanned from top bottom, and foreground pixels are generated in each line, the values of FG pixels are propagated in three directions, to the bottom line (Fig. 4.21b), from the rightmost FG pixel to the leftmost pixels in the same line (Fig. 4.21d) and from the leftmost FG pixel to the rightmost pixels (Fig. 4.21c). The intersection is then labeled as foreground (Fig. 4.21e).

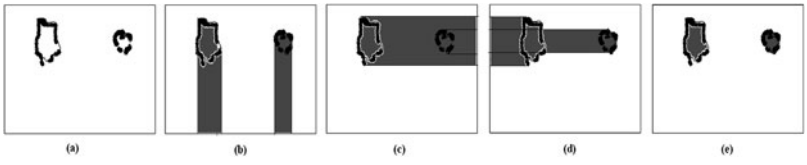

Fig. 4.21 Separating FG from BG pixels by hole filling

$$I'_{LR}(x,y) = 1 \; if \; I(x,y) = 1 \; or \; I(x-1,y) = 1 \; else \; 0$$

$$I'_{RL}(x,y) = 1 \; if \; I(x,y) = 1 \; or \; I\,(x-1,y) = 1 \; else \; 0$$

$$I'_{TB}(x,y) = 1 \; if \; I(x,y) = 1 \; or \; I(x,y-1) = 1 \; else \; 0$$

$$I''(x,y) = I'_{TB}(x,y) \cap I'_{RL}(x,y) \cap I'_{LR}(x,y)$$

After finding different moving object blobs in each frame, the next step is to extract some specific features, such as the bounding box for each blob. Such features are necessary for the tracking phase. A method is introduced that exploits the connectivity attribute of FG pixels and extracts borders for each isolated object. Labeling is done afterward by taking extracted borders into account. In this approach each frame is scanned only once from top to bottom. In each line, pixels belonging to FG are checked, and the ones which correspond to the leftmost, rightmost, uppermost, and lowermost location of each blob are extracted and saved in corresponding vector variable positions. The vector variables are named start row *(SR)*, end row *(ER)*, start column *(SC)*, and end column *(EC)*, respectively.

$$If \; I(x,y) = 1 \; and \; I(x-1,y) = 0 \Rightarrow SC(x) = 1$$

$$If \; I(x,y) = 1 \; and \; I(x+1,y) = 0 \Rightarrow EC(x) = 1$$

$$If \; I(x,y) = 1 \; and \; I(x,y-1) = 0 \Rightarrow SR(x) = 1$$

$$If \; I(x,y) = 1 \; and \; I(x,y-1) = 0 \Rightarrow ER(x) = 1$$

At the end of a frame scan, the SC(x) and EC(x) vectors are analyzed again to extract the horizontal borders Xmin and Xmax for each object, as well as the vertical borders Ymin and Ymax.

4.5.3.2 Trajectory Extraction

In Fig. 4.22 we illustrate this process to establish the moving object. In addition to the bounding box of objects, the Center of Gravity (COG) of each object is calculated for subsequent use in tracking and event detection as: $COG = \left(\frac{X_{\min}+X_{\max}}{2}, \frac{Y_{\min}+Y_{\max}}{2}\right)$.

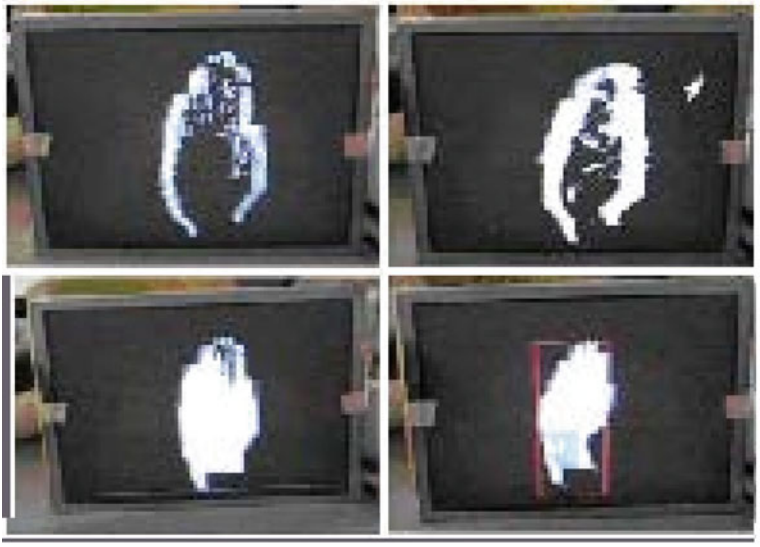

Fig. 4.22 Top shows Temporal Difference Method (*left*) and Background Updating Method with Learning Rate 0.5 (*right*). Bottom shows Filled Image with the Method described here (*left*), and the Final Result after finding the Blob Borders (*right*)

However, COG alone is not suitable for tracking and special event detection when the objects are humans moving relatively close to the camera. This is due to the non-uniform and erratic nature of human motion, caused mostly by movements of the arms alongside the body during walking or running. This results in periodic changes in the width of bounding box for the object. Therefore, another feature named Top of Head (TOH) is also extracted for each moving object. TOH represents the upper middle part of blobs.

$$TOH = \left(\frac{X_{\min} + X_{\max}}{2} \Big|_{y=Y_{\min}}, Y_{\min} \right)$$

Tracking can be defined as the problem of estimating the location of an object as it moves around in a scene and generates a trajectory. Since the regions in the image belonging to moving objects, the blobs, were obtained before, tracking now only needs to find the correspondences between blobs across frames. There are several methods for object tracking that can be used depending on the application, the type of moving object representation. and the type of features used. With the assumption of having a stationary camera and with blob displacements small compared to blob size, a simple correspondence of blobs appears sufficient. Tracking of objects is therefore done by matching blob features like COG or TOH in consecutive frames.

A metric for the closeness of feature points in consecutive frames is specified for use as a correspondence measure:

$$D_{ij} = \left[\left| \left(x_i - x_j\right) \right| + \left| \left(y_i - y_j\right) \right| \right].$$

If D_{ij} < threshold Th, then $O_i(t) \equiv O_j(t–1)$ is a match; otherwise $O_i(t) \neq O_j(t–1)$.

Tracking a moving object generates a sequence of successive locations of the object in the scene known as the trajectory of the object. Handling trajectories as spatio-temporal data streams is the essential part in event detection applications. As the first step, a vector that is going to be used as an input vector in event detection is specified and an index is assigned to each member of a trajectory based on its time stamp in the data sequence. A trajectory of an object i can be represented as $T_i = \{(x_1,y_1), (x_2,y_2), \ldots, (x_N,y_N)\}$, with N the number of frames.

Some scenes may include different physical pathways, which results in detection of different trajectories (Fig. 4.23). This makes learning of all the different paths difficult for a single classifier. Therefore, one may apply a preprocessing step before learning to classify trajectories into groups that each belongs to one path. In a simple method, trajectories can be classified in different groups based on their entry and exit points in the view. The entry point is the location where an object enters the view. Similarly, an exit point is the location where an object leaves the view [49]. In the work presented here, trajectories are grouped by their entry and exit point in the view, and a different classifier is used for each group.

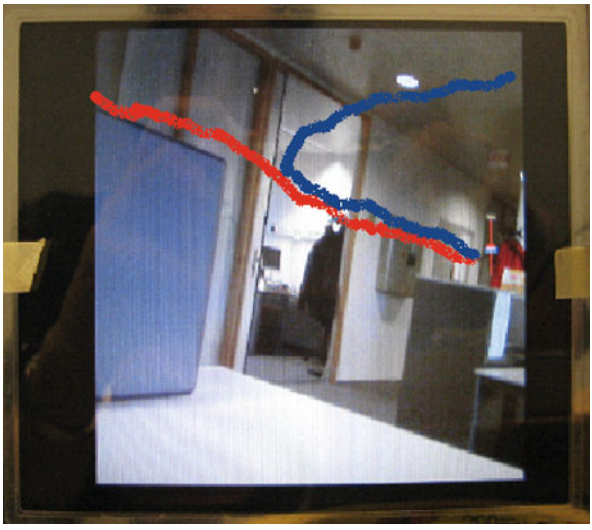

Fig. 4.23 Trajectories in a scene with more than one path

It is also sometimes necessary to find dominant trajectories from sets of observed trajectories in the scene that belong to one group, to support an efficient learning phase for building a model of trajectories (Fig. 4.24). Finally, in order to remove noisy variations and to minimize effects of time discretization, a temporal smoothing is employed to the elements of a trajectory. The smoothing function that we used is given by $S(s,t) = \mu \times x_t + (1 - \mu) \times x_{t-1}$ with $\mu \leq 1$ [50].

Fig. 4.24 A set of observed walking trajectories

4.5.3.3 Learning on the WiCa Platform

Post processing of the results obtained from trajectory point classifiers is also needed to conclude whether a trajectory is normal or not. Conditions examined in abnormality detection include $NL > \theta$, where NL represents the number of abnormal trajectory points and θ depends on the nature of the scene and application [51].

The extracted trajectory points of normal motion patterns of moving objects are saved in DPRAM of the WiCa board as a map of activity. The constructed map is then dumped from the DPRAM after a complete trajectory and analyzed using a MATLAB model, and the training data set for the Neural Net (NN) is prepared (Fig. 4.25). Next, the NN is trained off-line and the C code for the resulting network is automatically generated and then imported into the 8051 microcontroller.

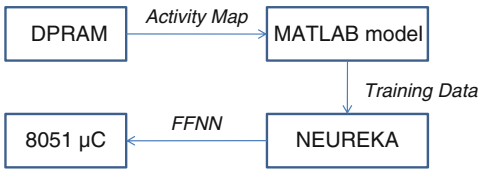

Fig. 4.25 Dataflow in learning phase

Feed-Forward Neural Networks (FFNN) are a form of NN where data are moving from input nodes to output nodes over multiple layers of neurons in one direction, crossing one layer at a time, and without any feedback [52].

In this approach FFNNs learn the motion pattern from the trajectories of moving objects. Inputs to an FFNN are the coordinates of the selected feature points (here

TOH) of moving objects in two subsequent frames. During the learning phase the target output to the network, which is the actual position (TOH) in the next incoming frame, is given and the weights of the network are adjusted. In the test phase, the network should be able to predict the approximate position of the feature points. Significant deviations from predicted results indicate unexpected changes in speed and direction of feature points, which can then be flagged as an abnormal event (Fig. 4.26).

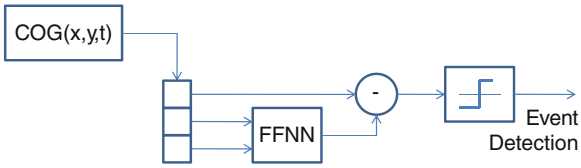

Fig. 4.26 Dataflow of testing phase

4.5.4 Cellular Neural Network Learning

Cellular Automata have always created a scientific attraction by displaying complex behavior on basis of a regular computing structure with simple rules. From the observation, that a Cellular Automata can both imitate biology and work as a Turing machine, the understanding has grown that they can also create autonomous behavior. Many Cellular Automata implementations have been made in software, but this approach will focus on a specific variation, the Cellular Neural Network (CNN), as it has a number of promising hardware realizations [53].

After the introduction of the Chua and Yang network [54], a large number of CNN models have appeared in literature. Like a cellular automaton, a CNN is made of a regularly spaced grid of processing units (cells) that only communicate directly with cells in the immediate neighborhood. Cellular Neural Networks [55, 56] are widely used with real-time image-processing applications. Such systems can be efficiently realized using macro-enriched field programmable gate-arrays. Early attempts in FPGA realization of CNN functionality have shown impressive potential [56, 57].

A DT-CNN (Discrete Time CNN), introduced by Harrer and Nossek [58], is a regular multidimensional grid of locally connected cells. Each cell c communicates directly with its *r-neighbors*, i.e., a set of cells within a certain distance r to c, where $r \geq 0$. E.g., if $r = 1$ we have 3×3 neighborhood and if $r = 2$ we have 5×5 neighborhood. Still a cell can communicate with other cells outside its neighborhood due to the network propagation effect.

The state of a cell c, denoted x^c, depends mainly on two factors: the time-independent input u^d to its neighbors d and the time-variant output y^d *(k)* of its neighborhood. The neighborhood always includes c itself. Equation 4.1 describes this dependency in a discrete time k, while Fig. 4.27 gives an illustration.

$$x^c(k) = \sum_{d \in N_r(c)} a_d^c y^d(k) + \sum_{d \in N_r(c)} b_d^c u^d + i^c \tag{4.1}$$

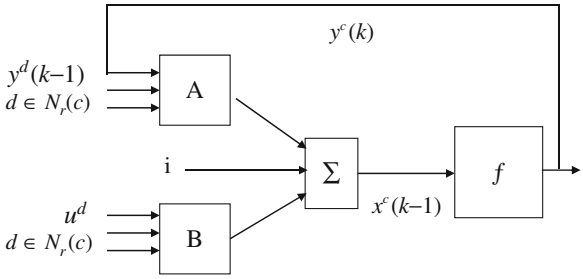

Fig. 4.27 Block diagram of a DT-CNN cell

Spatially invariant CNNs are specified by a control template A containing a_d^c, a feedback template B containing b_d^c, and the cell bias $i = i^c$. Node template, $T=<A, B, i>$, determines together with input image u and an initial output $y(0)$ completely the dynamic behavior of the DT-CNN. Convergence to a stable state will be achieved after a number of iterations n, i.e., at time $k = n$.

Although neighborhoods of any size are allowed in DT-CNN, it is almost impossible to realize large templates. Limited interconnectivity imposed by current VLSI technology demands that communication between cells is only local. Let us restrict ourselves to the use of 3 × 3 neighborhood, where templates A and B are 3 × 3 matrices of real-valued coefficients. Additionally, the input range of a DT-CNN cell is restricted to [–1, +1], due to the fact that grey-scale level is commonly used to the purpose of image processing. A value of –1 represents a white pixel and a value of +1 represents a black pixel. Other values represent grey levels in-between. Although floating-point representation of real-valued numbers is generally preferred, we employ fixed-point representation due to its friendliness in hardware realizations, especially for the use of multiplier primitives in FPGA. Furthermore, before any computations can be done, the image is assumed to be stored in an external storage space.

The operation of a DT-CNN on images covers many dimensions. The local operation is performed in a two-dimensional plane (width and length) and iterates in time. This data structure is a topographic map, serving a number of sensory applications though practice is predominantly confined to image processing. Due to the limited capacity of the CNN implementation, this has to be repeated over image slices and iterates over the surface to handle potential wave propagation. Finally, the operation is performed on sequences of images (Fig. 4.28). All this has to be facilitated on the two-dimensions in a FPGA. Consequently, the dominating architectural question is: how to reduce the dimensions from the functional requirements to the platform facilities?

4.5.5 Moving Target Detection

Motion detection is a central scheme in various areas of vision sensing, both in industrial as in consumer applications. The concept is based on the simple

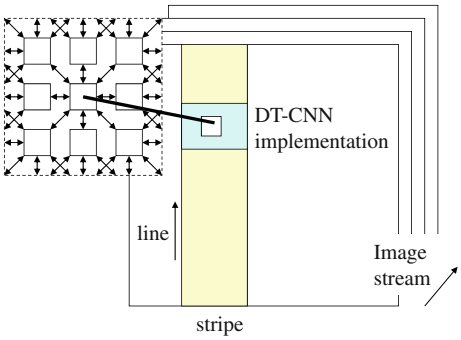

Fig. 4.28 Dimensionality of CNN image processing

observation that moving objects carry the most important features, in comparison to other details contained in background and still parts. Thus, detecting and coding the moving objects are essential for image understanding. The typical image-analysis algorithm consists of four main steps: (1) segmentation, (2) parameter (motion) estimation, (3) image synthesis, and (4) consistency observation [59].

The most important and computationally most complex one of these steps is the task of segmentation, which is accomplished by cutting a scene into different moving objects (regions). These objects have to be labeled and measured. Furthermore, consistency of object segmentation is an essential aspect to guarantee quality of the result. Therefore, maintaining a global uniform velocity is crucial to recognize the segmented objects [60].

The object-oriented image analysis scheme is widely accepted as an interesting and sophisticated approach for future video-coding systems with a very low bit-rate. By transferring the moving objects only, the transmission rate is greatly reduced. In Stoffels [60], the emphasis is on achieving object-oriented image compression for videoconferencing purposes on a CNN-UM hardware platform. The modeling of the motion can be eased from the understanding that labeled objects can only move within a Region of Interest: the remarkable features of a human face are the areas containing eye, nose, mouth, and ear. However, any deterioration of facial expressions decreases image quality drastically. Thus different "quality enhancing" steps are needed, which makes the segmentation algorithm rather complex.

In the work presented here, the problem is fundamentally different. A moving object is also changing location in its Region of Interest (RoI) and needs therefore also to be distinguished in every frame from a sequence of consecutive images. Once the object is segmented into a number of frames, the displacement of the object between two image frames must be extracted. Wrapping the moving object in an encapsulation box eases the computation of the displacement. The displacement of the box corresponds to the distance covered by the object in reality. The establishment of this correspondence is a problem by itself and often forces a need for calibration. When the whole object is observed in one of the frames, the difference between two consecutive frames shows two leftovers instead of one. Additional effort is then required to relate these two as belonging to a single object.

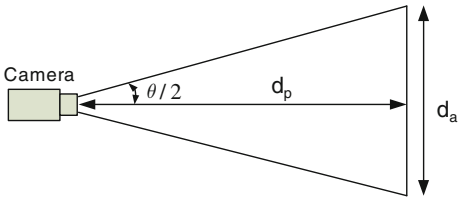

Fig. 4.29 Mapping of the image on the pixel map

The process of velocity measurement depends on a number of parameters, mainly coupled to the camera in use, such as image resolution, frame frequency f_f, and view angle θ. Another parameter of importance is the distance between the camera and the moving object, d_p. Given f_f in frames/seconds, d_p in meters and θ in degrees, the width of the captured scenery is $d_a = 2\,d_p\,\tan(\theta/2)$. Figure 4.29 illustrates a camera where the involved parameters are pointed.

An object with velocity v (m/s) will cover the distance d_a (meters) in $t = d_a/v$ (seconds). During this time, the camera takes $N = t.f_f = (d_a.f_f/v)$ frames. In other words, if all the frames are super-imposed, there will be N instances of the moving object on a single frame. If W, in pixels, denotes the width of frames delivered by the camera, the movement of the object corresponds to a displacement in pixels given by

$$n_p = W/N = (W \cdot v)/(d_a \cdot f_f).$$

The minimum velocity that can be detected corresponds to a single pixel displacement of the object. The nature of the applied template complicates the calculation of maximum speed when the object is close to the edge of the frame. In order to overcome this limitation, a 5% margin of the total frame-width is provided on both vertical edges. Obviously, the maximum displacement by means of pixels is correlated to the maximum object-velocity that can be detected.

Typical PAL camera specifications like horizontal view angle of 60°, 720 pixel wide frames, and frame-rate of 25 frames/s are utilized in Fig. 4.30, where the displacements of a 3-m long object are shown for different speeds. Obviously, the displacement depends on the distance d_p of the camera from the captured scenery in which the object moves. The size of the blob is dependent on d_p as well. Such dependencies can be resolved by nonlinear post-processing the blob sizes and displacement over a sequence of images 0 [61]. This effectively eliminates accuracy considerations from the here presented algorithm.

4.5.5.1 The Algorithm in Basic CNN Operations

The segmentation quality is significantly increased by pre-processing image-frames in order to remove disturbing noise. Noise filtering is achieved by iteratively applying the averaging template to the image (Fig. 4.31). The next step is to create a

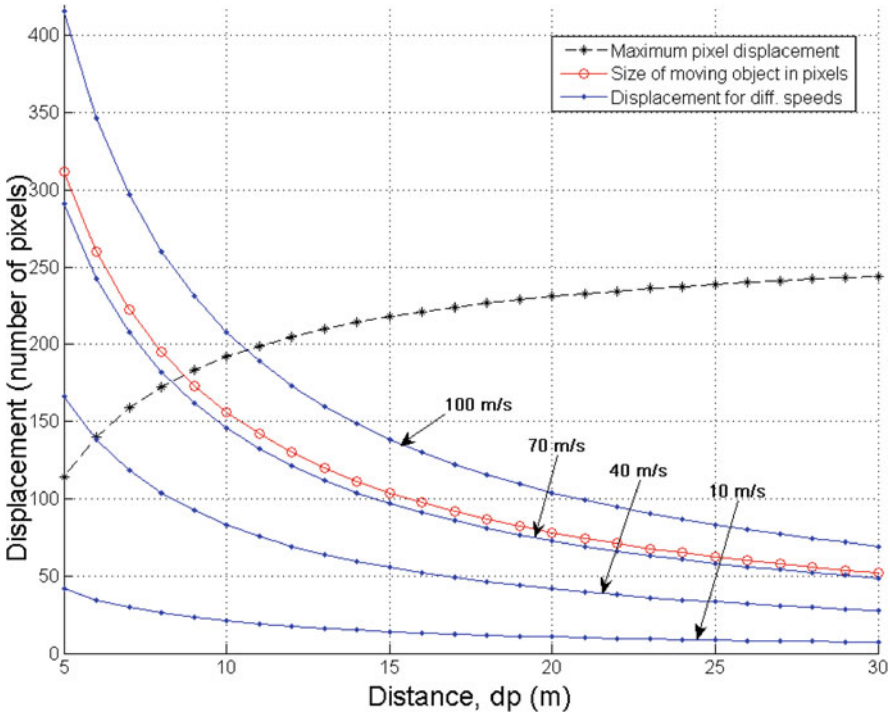

Fig. 4.30 Pixel displacement versus observation distance for several object velocities

mask around the Object of Interest (OoI), i.e., the moving object. In order to extract the RoI, all background information has to be removed. Calculation of the absolute value of the subtraction of two consecutive frames in the image sequence completes the task. In other words, if f_1 and f_2 are first and second frames, respectively, the result is given by $|f_1 - f_2|$. The resulting output map has its darkest pixels where both frames differ, while background information is covered in grey.

Although the output map includes all necessary contour pixels for segmentation, further contour enhancement is needed to achieve sufficiently exact segmentation. Contour lines vary in thickness due to the sharpness of object edges. An appropriate choice of the gradient threshold reduces the effect of sharp/smooth edges, but we follow the proposition made in Stoffels [60], and apply the approach of *skeletonization* (Fig. 4.31). Applying this powerful line-thinning algorithm iteratively reduces lines of arbitrary and varying thickness to their center pixels. Quality of the intermediate result is, however, still reduced due to the presence of some isolated pixels that might even inhibit the creation of the encapsulation box around the moving object. These pixels are easily removed by applying the template of Isolated Pixel Removal (Fig. 4.31). The resulting output map is now free of disturbing information, and the task of segmentation is accomplished by thresholding the value of all pixels using a hard-limiter. A binary mask is then created, and combining the corners of that mask

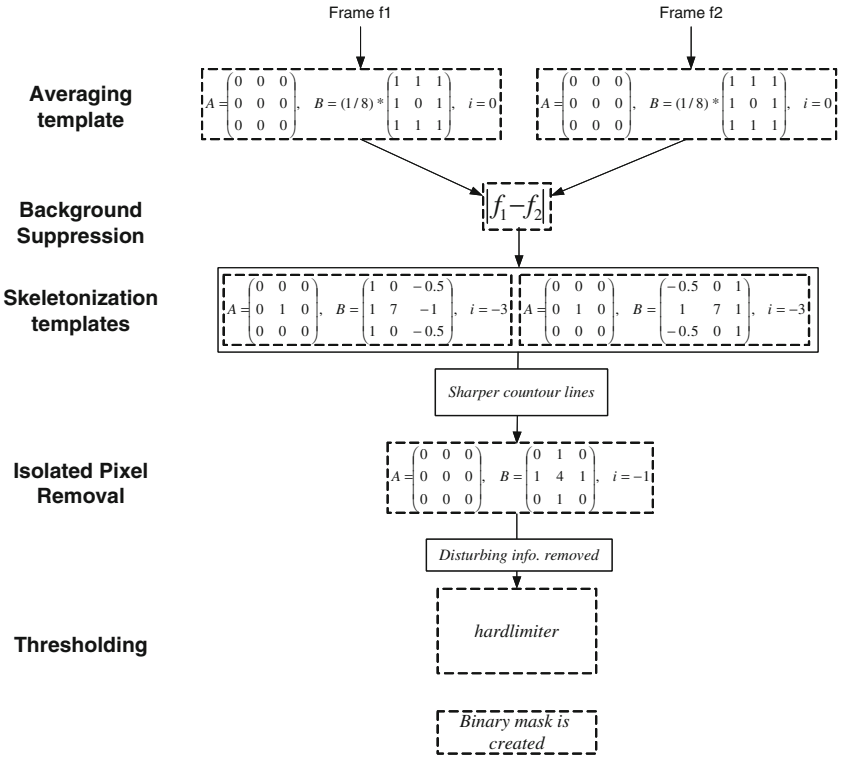

Fig. 4.31 Flow diagram for the template application

easily creates a black box covering the object of interest. As the differences between the frames are not necessarily co-located, they must be linked in order to establish the OoI.

In the general case, any blob that appears in both frames must be checked. The problem is eased when the direction of movement can be predicted from the past or by hypothesis. For instance, a train will move along a track and its direction will be clear from the outset. As the early movement will show co-located blobs (Fig. 4.32a,

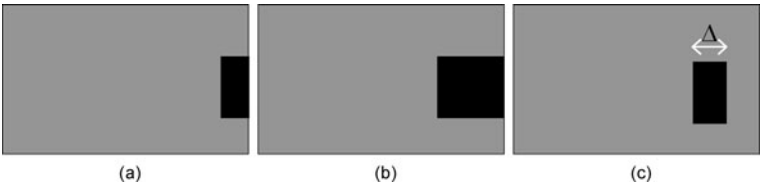

Fig. 4.32 Measuring the displacement of an object moving from right to left in the scenery. Displacement (shown in (**c**)) of the moving object is the difference BETWEEN the black boxes in (**a**) and (**b**)

b), the direction can be predicted when the speed rises and object splitting occurs. Verification of the hypothesis can be achieved by enlarging the blob in the frames in the direction of the movement, followed by an AND operation. This supplies us already with a box at length of the displacement.

When the blob has not split, we have to repeat the same segmentation procedure with the following pair of frames, i.e., f_3 and f_4, creating another black box. Two different facts are extracted by comparing the position of the boxes: motion direction and, most important, displacement of the moving object between frames f_2 and f_4. As time between consecutive frames is known, we only need metric information about one of the details in the scenery (preferably the moving object itself) to determine the velocity of the moving object. In Fig. 4.32c, the displacement of the moving object is illustrated with the difference Δ of the two black boxes.

4.5.5.2 Verification and Test

MATCNN[10] is a flexible and easy-to-use test environment for single-layer CNNs developed at the Analogic and Neural Computing Laboratory, MTA-SzTAKI, in Budapest, Hungary. Although basically built to simulate analog CNN implementations, it provides a very good toolbox (for MATLAB) to verify the approach. The toolbox is equipped with a library of many different 3×3 templates.

In the following, all steps in the flow diagram in Fig. 4.31 are applied to a number of frames captured from a video sequence of a locomotive of type SJR6. First of all, the averaging template is iteratively applied on the first two frames of the video sequence. Figure 4.33 shows the results after 25 iterations with a time step of 0.019τ.

The background is then faded in order to outline the moving object, which is achieved by calculating $|f_1 - f_2|$. As seen in Fig. 4.34a, the background is replaced with grey pixels.

Fig. 4.33 First two frames (f_1 and f_2) of the video sequence after applying the averaging template for a number of iterations

[10] http://lab.analogic.sztaki.hu/Candy/

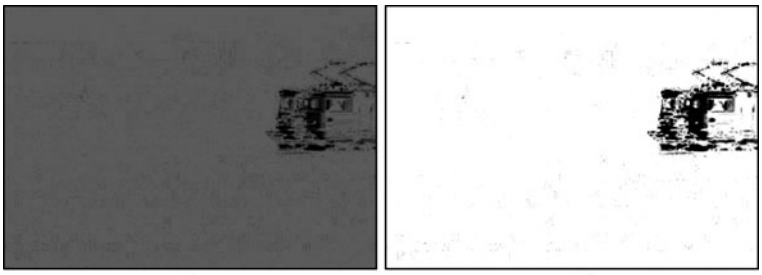

Fig. 4.34 (**a**) Resulting image of the absolute value of the difference of two consecutive frames. Darkest pixels are observed where the two frames differ as most. (**b**) Intermediate result after skeletonization, where the isolated pixels can easily be noticed

The algorithm of skeletonization performs iteratively in eight subsequent steps, where each step peels one layer of pixels in a certain direction. After one iteration is accomplished, the pattern is one pixel thinner in all directions. The algorithm stops when no difference between the input and the output is obtained. Assuming the object moves horizontally only, applying the templates of skeletonization for two directions only, i.e., west and east (Fig. 4.31), is sufficient. The templates are applied iteratively for 25 iterations each, with time step of 0.019τ. The result is shown in Fig. 4.34 b, where isolated pixels are easily noticed. Isolated pixel removal is applied iteratively for 25 iterations with time step 0.04τ to remove these pixels. Figure 4.35a depicts the resulting image.

Segmentation of the moving object is accomplished by creating a binary mask using the hard-limiter function. Figure 4.35b depicts the final result with the binary mask. Finally, the object of interest is covered with a black box, as shown in Fig. 4.32b.

All steps of velocity estimation from averaging to skeletonization through isolated pixel removal have been implemented on a parallel FPGA implementation of the CNN algorithm, called Cabellero [62]. Subsequent steps of the algorithm, i.e.,

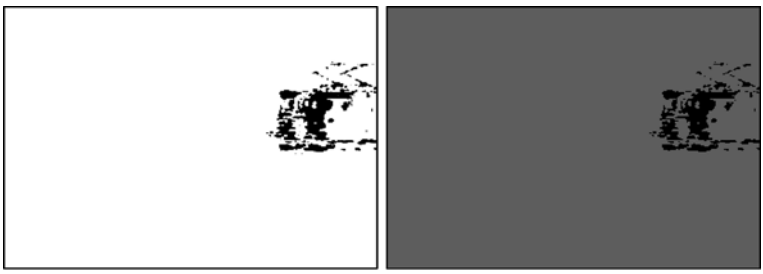

Fig. 4.35 Applying the template of IPR removes all isolated pixels (**a**). Procedure of segmentation is completed once the binary mask is created (**b**)

creation of both the binary mask and the black box covering the moving object, are performed on a PC by, preferably, using MATLAB toolbox.

The task of post-processing is completed by calculating the value of Δ (see Fig. 4.32c) and, thus estimating the velocity of the moving object. This design maintains the regularity of a CNN and paves the way for future modification, e.g., usage of two-layered CNN [63] needed for implementation of steps 2, 3, and 4 in image analysis algorithm mentioned before. Hence, Caballero is chosen to serve as a model for the implementation of our algorithm in hardware.

The design performs on the same video sequence used for verification. The intermediate result obtained after skeletonization (Fig. 4.36) differs, however, from the expected result noticed in Fig. 4.34b. This quality reduction is due to the use of squashing function. A Caballero cell applies a tanh function for scaling the 21-bits wide calculation result down to 8-bits output value, which leads to a reduced precision. One way to overcome this limitation is by using the template of averaging (Fig. 4.31) again between the operation of skeletonization and IPR. Figure 4.36 shows the intermediate result after each step.

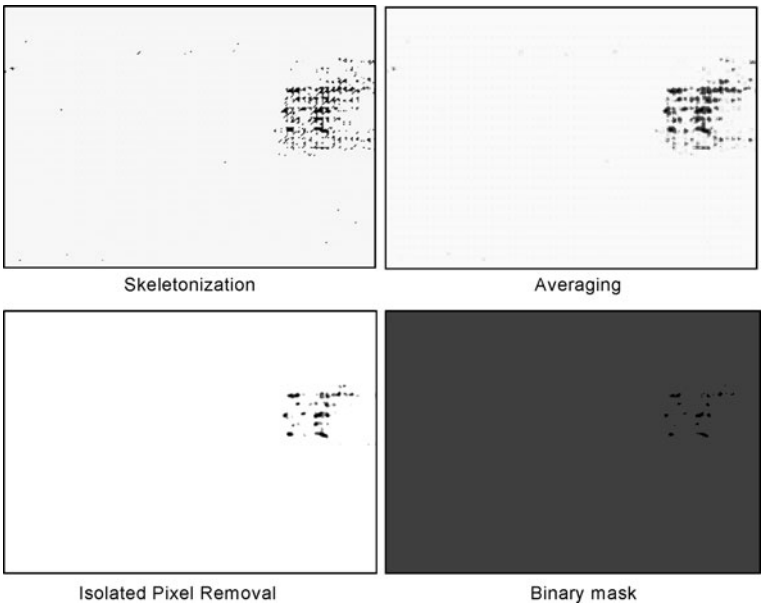

Fig. 4.36 The intermediate results of all steps as obtained from the post place and route simulation

4.5.5.3 Experiment Evaluation

A careful look at Fig. 4.30 tells that displacements captured with parts of the blob (object) outside the scenery (Fig. 4.33) guarantees obtaining one solid box after the step of background suppression. On the other hand, separation between blobs occurs

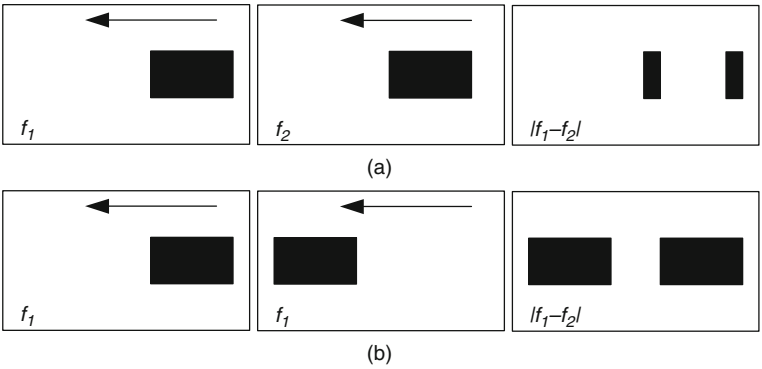

Fig. 4.37 Separation between blobs due to different speeds: "Slow" object in (**a**) and a "Fast" one in (**b**). The arrows indicate the direction of the movement

for all displacement values above maximum blob size. Furthermore, two leftovers are observed when one of the frames contains the whole object and the displacement is above the maximum size of the blob (Fig. 4.37). The size of the obtained blobs is highly dependent on the speed of the object and the frame frequency f_f. In the extreme case, one of the ends of the object is seen in one frame while the other end only appears in the successive frame. This case is discarded, as no measurements are possible. The alternative is to operate so fast that co-location still exists between subsequent images.

Taking the difference between frames will not suffice, when there is separation between blobs. The separation between the blobs has to be bridged. This case can be taken care of in the algorithm by including more steps.

Alternatively, tuning the entries of the templates compensates for the noise introduced due to internal truncation of data. Recently, it has been shown that the internal 21-bits value in Caballero can be scaled down to 7 bits without loss of accuracy [64]. Unfortunately, this approach has not been tested here.

The templates for thickening only the right-hand side and left-hand side are applied on the first and second image frames, respectively. A logical AND template is applied to the resulting image frames to get an intermediate image frame. Now the operation is performed on the other sides, i.e., the templates for thickening left-hand side and right-hand side are applied on the first and second frames, respectively. The resulting image frames are again logical AND-ed to get another intermediate image frame. These two intermediate frames are logical OR-ed. Hence, the image frame derived from this operation will be free of pixel separation between the two blobs. The value of Δ can be calculated by taking the difference between the separation-free frame and the first frame. The steps are illustrated in Fig. 4.38.

All in all we have shown how velocity estimation can be handled within a single CNN implementation. The final prototype can isolate and measure a moving object at a speed of 250 frames per second, which is sufficient for the usual applications as in law enforcement.

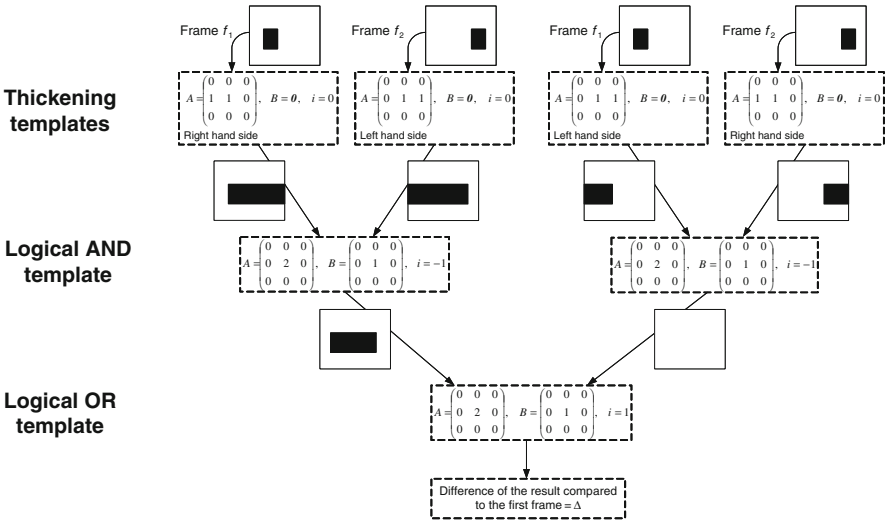

Fig. 4.38 Extended algorithm for handling fast-moving objects. the direction of movement is from *right* to *left*

4.6 Multi-sensor Intelligence

Neural networks tend to become hard to train, when the structures grow in complexity. Modular neural networks seem the answer, but they are vulnerable to unlearning effects. We need to stress knowledge integrity as the goal of training, facilitated by time-ordered scheduling. This allows neural modules to behave as software objects and therefore enables knowledge capture by construction [65]. Where measurement data are hard to come by or costly to gather, such a fusion of knowledge from different sources is the efficient way to create a quality neural model of the non-modeled process.

4.6.1 Intelligence Gathering

Surveillance Applications are also known as "Electronic Eyes and Ears" [66]. There are many variations ranging from the Wall between USA and Mexico – and similar closed environments – to a Home for Elderly Citizens. Such large-area sensory applications are generally showing a form of Electronic Awareness, for instance:

- Location awareness, where the device is informed about its spatial coordinates.
- Social awareness, where the device is informed about its role in an interaction between many devices.
- Personal awareness, where the device is informed about its human-like personality.

Also mixed levels of awareness can occur. For instance, situational awareness can be used where a device plays social roles depending on where it is. Clearly, awareness is a set of hierarchically layered notions. But the question remains: how? We will find the answer in the subsumption hierarchy.

The main purpose of the suggested hierarchy is to quantify the situation. This is more than simply maintaining access rights or handling privileges. When children are playing, it needs to be monitored whether they are still playing and whether their interaction will not bring danger to the individual's health. But on background it must still be known where they are. In other words, the three following ingredients are necessary:

- Passage restriction (a person has only allowance for a certain area) and person authorization (the person is who he claims to be)
- Activity restriction (an activity involves certain persons) and person verification (the person can in principle play his part)
- Gesture restriction (a gesture involves a person's typical behavior) and person identification.

A gesture reflects emotions, like joy or fear. When a subject displays fear, special attention is required even though the cause is not known yet. This is a sound biological principle, where fear causes the autonomous action of the thalamus to flee while temporarily bypassing the cortex, in which the diagnosis of the threat is made.[11] Where the activity may bring the subject in a dangerous situation, the reason for fear can almost be guessed. It is only a guess, but there is every reason to make that guess. Still, it could be wrong and the guess needs to be verified. Such activities are likely to depend on a specific environment. Being in such an environment makes it even more likely that the threat has occurred.

4.6.2 Multi-sensor Fusion

A borderwatch system initially needs to be able to observe people and/or vehicles attempting to illegally cross a physical border. We envision the detection systems to be placed at strategic locations along the border of interest. The detection system as minimum should consist of a low-cost commercial radar subsystem, a multi-spectral infrared subsystem, and a panoramic view visual subsystem. We believe that an integrated multi-sensor learning solution will result in better performance than any of the individual sensors. The sensor locations should be solar cell-powered. The individual locations should be wirelessly interconnected in a fault-tolerant (chordal-bypass) pattern. It should be possible to build man-portable and mobile versions of the fixed-location detection station.

[11] NN, The amydala and its allies, http://thebrain.mcgill.ca/flash/i/i_04/i_04_cr/i_04_cr_peu/i_04_cr_peu.html

Previous studies [67, 68] on detection of pedestrians and vehicles performed at MIT's Center for Biological and Computational Learning have indicated that (for a single sensor!) already with small sets of (ten) positive examples, a reasonable false-positive rate can be attained. Similarly, a recent (single sensor!) study [69] at McGill University's Center for Intelligent Machines shows that an FPGA-based system can detect people accurately at a rate of about 2.5 frames per second. A study [70] at Carnegie Mellon University has performed people detection and tracking for a panoramic video mosaic.

We could apply a Field-Programmable Gate Array (FPGA) board in the detection station for initial people and/or vehicle Region-of-Interest (ROI) detection. The FPGA board would implement Cellular Neural Networks for detection. An unusually large (more than 4 GBytes of SDRAM) amount of memory on the board would facilitate the highly efficient implementation of CNNs [71]. In Section 4.5.5, we described the use at Lund University in Sweden of CNN (implemented as a Network-on-Chip) on an FPGA board (Cabellero) to isolate a moving object and measure its velocity at a 250 frames per second rate [62].

A unique aspect of this approach is that CNNs will be used in all the stages of analysis, be it spectral, temporal, or spatial. In addition, the CNNs will introduce a level of learning for target detection and recognition.

4.7 Summary

The motor of sensory networks is the wireless communication. Where the energy costs of the transmission are dominating the design of wireless networks, there is a continuous improvement in energy use, storage, and harvesting to provide the functional freedom that we want. We have listed some popular standards without going into much detail as steadily such characteristic values will change.

Based on wireless communication, some product-like sensory networks have evolved. This first generation is typified by the existence of a common goal. In the past, one would find a single product offering; here we see a collection of devices that provide parts of a desired functionality. For instance, in the Body-Area Network, some selected sensors are brought together on a piece of textile to serve a single, collective purpose. In this sense they are virtually wired.

The second generation, as is currently already being prototyped in the laboratories, will cover a more general functionality and will also be based on sensory resources rather than sensors. A typical functionality has to do with the Caring, Comforting, and Concerned house. The architecture as such is not bound by a piece of control software, housed in one of the network parts. Instead, it will find sensors that can support the intended functionality and apply them for that purpose. This already brings redundancy into the equation.

A typical example is behavioral modeling based on the movements of a person in the house. One way to extract such movements is from the mobile phone that is carried by the person. Using the internal compass, acceleration, and GPS resources,

it can independently determine its trajectory and communicate that to the outside world. The issue is of course that the person may not carry his/her phone or carry the phone of somebody else. This is similar to the discussion in the automotive industry in the early nineties: do we need traffic signs that actively communicate with cars, or cars that can read the traffic signs. Similarly, human trajectories can be broadcasted by the person or "seen" by the room.

Clearly, the second generation of sensory systems will be based on multiple sensors, where functional overlap between the sensors will help to create the intelligence required by the network to not only perform the intension but also give it a fair amount of dependability and therefore safety.

References

1. Orwell G (1950) 1984, Signet classics, July 1950
2. Perry TS (September 2009) Ge Wang: the iPhone's music man. IEEE Spectr 46(9):24
3. Jones WD (February 2010) A compass in every smartphone. IEEE Spectr 47(2):12–13
4. Aarts E, Marzano S (eds) (February 2003) The new everyday: views on ambient intelligence. Uitgeverij 010 Publishers, Rotterdam The Netherlands
5. Kahn JM, Katz RH, Pister KSJ (September 2000) Emerging challenges: mobile networking for "smart dust". J Commun Netw 2(3):188–196
6. Karl H, Willig A (2005) Protocols and architectures for wireless sensor networks. Wiley, San Francisco CA
7. Sung M, Pentland A (July 2005) Minimally-invasive physiological sensing for human aware interfaces. In: Proceedings of the 3rd international conference on universal access in human-computer interaction, Las Vegas, NV
8. Xu Y, Heidemann J, Estrin D (July 2001) Geography-informed energy conservation for ad hoc routing. In: Proceedings of the 7th annual international conference on mobile computing and networking (Mobicom2001), Rome, Italy, pp 70–84
9. Rajan S (2005) Dynamically controllable applications for wireless sensor networks. Master's thesis, Mississippi State University, Mississippi State MS
10. Dvorak J (September 2005) IEEE 802.15.4 and ZigBee overview. Tutorial, Motorola
11. Ranganathan P (April 2010) Recipe for efficiency: principles of power-aware computing. Commun ACM 53(4):60–67
12. Moder JJ, Phillips CR (1970) Project management with CPM and PERT, 2nd edn. Van Nostrand Reinhold Company, New York NY
13. Volder JE (September 1959) The CORDIC trigonometric computing technique. IRE Trans Electron Comput EC-8:330–334
14. Flynn MJ, Hoevel LW, (July 1984) Measures of ideal execution architectures. IBM J Res Dev 28(4):356–369
15. Chait AL Solving The Last Milli-Mile problem in vehicle safety; the EoPlex approach to powering wireless tire pressure sensors. White paper of EoPlex Technologies Inc. Available at www.eoplex.com
16. Calhoun BH, Daly DC, Verma N, Finchelstein DF, Wentzloff DD, Wang A, Cho S-H, Chandrakasan AP (June 2005) Design considerations for ultra-low energy wireless microsensor nodes. IEEE Trans Comput 54(6):727–740
17. Starner T (1996) Human-powered wearable computing. IBM Syst J 35(3–4):618–629
18. Yang GZ (ed) (2006) Body sensor networks. Springer, London
19. Ross PE (March 2009) Cloud computing's killer app: gaming. IEEE Spectr 46(3):14

20. Greene K (January 2008) Smart badges track human behavior. MIT Technol Rev (online) 30 Jan 2008
21. ABB Automation Technologies (January 2009) Festo adopts ABB's wireless standard. Control Engineering Europe (online), 13 Jan 2009
22. Schneider D (May 2010) Electrons unplugged. IEEE Spectr 47(5):34–39
23. Nam M-Y, Al-Sabbagh MZ, Kim J-E, Yoon M-K, Lee C-G, Ha EY (June 2008) A real-time ubiquitous system for assisted living: combined scheduling of sensing and communication for real-time tracking. IEEE Trans Comput 57(6):795–808
24. Azimov I (1950) I, Robot. Gnome Press, London
25. Wings S, ORIION {Organic Immense Information Networks}", Base^{N} project, cordis.europe.eu/fp7/ict/e-infrastructure/networking. Accessed 16 Oct 2010
26. Department of Transportation and Federal Highway Administration, Quick highway incident detection and incident warning system. DTRT57-07-R-SBIR Solicitation, 15 Feb 2007
27. Brignolo R (May 2006) The co-operative road safety SAFESPOT integrated project, Centro Ricerche Fiat. In: APSN network and APROSYS integrated project 6th annual conference, Vienna, Austria, 12 May 2006
28. Greenfield D (March 2010) When is PID not the answer? Control Eng 54–57
29. Spaanenburg L (December 2010) Ensuring safety in distributed networks. In: IEEE conference on decision and control (CDC2010), Atlanta, GA
30. Kirrmann H (2005) Industrial Ethernet: IEC, go back to the negotiation table. Control Engineering Europe, April/May 2005, pp 25–28
31. Robinson D (2008) A honeypot for malware. Control Engineering Europe, April 2008, pp 10–11
32. Isermann R (1984) Process fault detection based on modeling and estimation methods – a survey. Automatica 20(4):347–404
33. van Veelen M (2007) Considerations on modelling for early detection of abnormalities in locally autonomous distributed systems. Ph.D. Thesis, University of Groningen, Groningen, The Netherlands
34. Jodoin P, Konrad J, Saligrama V (2008) Modeling background activity for behavior subtraction, Second international conference on distributed smart cameras (ICDSC-2008). Stanford University, California, September 7–11, 2008, pp 1–10
35. Zhang D, Gatica-Perez D, Bengio S, McCowan I (June 2005) Semi-supervised adapted HMMs for unusual event detection. In: IEEE conference on computer vision and pattern recognition (CVPR2005), vol I, San Diego, CA, pp 611–618
36. Owens J, Hunter A (July 2000) Application of the self-organization map to trajectory classification. In: Proceedings of the 3rd IEEE international workshop on visual surveillance (VS2000), Dublin, Ireland, pp 77–83
37. Tehrani, MA, Kleihorst R, Meijer PBL, Spaanenburg L (2009) Abnormal motion detection in a real-time smart camera system. In: Proceedings third ACM/IEEE international conference on distributed smart cameras (ICDSC 2009), Como, Italy, 30 August – 2 September, 2009
38. Kleihorst R, Schueler B, Danilin A, Heijligers M (October 2006) Smart camera mote with high performance vision system. In: ACM workshop on distributed smart cameras (DSC2006), Boulder, CO
39. Abbo A, Kleihorst R, Choudhary V, Sevat L, Wielage P, Mouy S, Heijligers M (2007) "Xetal-II: A 107 GOPS, 600 mW massively-parallel processor for video scene analysis. In ISSCC2007 Digest of technical papers, San Francisco, CA, pp 270–271, 602
40. Wu C, Aghajan H, Kleihorst R (September 2007) Mapping vision algorithms on SIMD architecture smart cameras. In: First ACM/IEEE international conference on distributed smart cameras (ICDSC07), Vienna, Austria, pp 27–34
41. Muller J-M (2006) Elementary functions, 2nd edn. Birkhäuser, Berlin
42. Venema RS, Ypma A, Nijhuis JAG, Spaanenburg L, (1996) On the use of neural networks in time series prediction with an application to artificial human walking. In: Proceedings world congress on neural networks WCNN'96, San Diego, CA

43. Peitgen H-O, Jürgens H, Seupen D (1992) Fractals for the classroom. Springer, New York, NY
44. Nijhuis J, Hofflinger B, Neusser S, Siggelkow A, Spaanenburg L (July 1991) A VLSI implementation of a neural car collision avoidance controller. Proc IJCNN Seattle WA 1:493–499
45. Sinai YG (1970) Dynamical systems with elastic reflections. Russ Math Surv 25: 137–191
46. Spaanenburg L (April 2001) Unlearning in feed-forward multi-nets. In: Proceedings ICANNGA'01, Prague, Czech Republic, pp 106–109
47. Hu W, Tan T, Wang L, Maybank S (August 2004) A survey on visual surveillance of object motion and behaviors. IEEE Trans Syst, Man, Cybern C Appl Rev 34(3):334–352
48. Heikkila J, Silven O (June 1999) A real-time system for monitoring of cyclists and pedestrians. In: Second IEEE workshop on visual surveillance, Fort Collins, CO, pp 74–81
49. Morris BT, Trivedi MM (August 2008) A survey of vision-based trajectory learning and analysis for surveillance. IEEE Trans Circuits Syst Video Technol 18(8):1114–1127
50. Dahmane M, Meunier J (May 2005) Real-time video surveillance with self-organizing maps. In: Second Canadian conference on computer and robot vision (CRV2005), Victoria, BC, Canada, pp 136–143
51. Lee KK, Yu M, Xu Y (October 2003) Modeling of human walking trajectories for surveillance. In: Proceedings of IEEE/RSJ international conference on intelligent robots and systems (IROS2003), vol 2, Las Vegas, NV, pp 1554–1559
52. Haykin S (1999) Neural networks: a comprehensive foundation, 2nd edn. Prentice Hall, New Jersey
53. Spaanenburg L, Malki S, (2005) Artificial life goes In-Silico", CIMSA 2005 – IEEE international conference on computational intelligence for measurement systems and applications. Giardini Naxos – Taormina, Sicily, Italy, 20–22 July 2005, pp 267–272
54. Chua LO, Yang L, (October 1988) Cellular neural networks: theory. IEEE Trans Circuits Syst 35:1257–1272 and 1273–1290
55. ter Brugge MH, (2004) Morphological design of discrete-time cellular neural networks, Ph. D. Thesis, Rijksuniversiteit Groningen, Groningen, The Netherlands
56. Nagy Z, Szolgay P (2002) Configurable multi-layer CNN-UM emulator on FPGA. In: Tetzlaff R (ed) Proceedings of the 7th IEEE workshop on CNNs and their applications. World Scientific, Singapore, pp 164–171
57. Uchimoto D, Tanji Y, Tanaka M (1999) Image processing system by discrete time cellular neural network. In: Proceedings of 1999 international symposium on nonlinear theory and its application (NOLTA1999), vol 1, pp 435–438
58. Harrer H, Nossek JA (September/October 1992) Discrete-time cellular neural networks. Int J Circuit Theory and Appl 20(5):453–467
59. Grassi G, Grieco LA (2002) Object-oriented image analysis via analogic CNN algorithms-Part I: motion estimation. In: Proceedings of 7th IEEE international workshop on CNNs and their applications, Frankfurt/M, Germany, pp 172–179
60. Stoffels A, Roska T, Chua LO (1996) An object-oriented approach to video coding via the CNN universal machine. In: 4th IEEE international workshop on CNNs and their applications, Seville, Spain, pp 13–18
61. Spaanenburg L et al (2000) Acquisition of information about a physical structure, Patent WO 000 4500, 27 January 2000
62. Malki S, Deepak G, Mohanna V, Ringhofer M, Spaanenburg L (2006) Velocity measurement by a vision sensor, CIMSA 2006 – IEEE international conference on computational intelligence for measurement systems and applications, La Coruna – Spain, 12–14 July 2006, pp 135–140
63. Arena P et al (1996) Complexity in two-layer CNN, 4th IEEE international workshop on CNNs and their applications, Seville, Spain, pp 127–132

64. Chen D, Zhou B (2005) Digital emulation of analogue CNN systems. Master's Thesis, Lund University, Lund, Sweden
65. ten Berg AJWM, Spaanenburg L (September 2001) On the compositionality of neural networks. In: Proceedings ECCTD, vol. III, Helsinki, Finland, pp 405–408
66. Tanenbaum AS, Gamage C, Crispo B (August 2006) Taking sensor networks from the lab to the jungle. IEEE Comput. 39(8):98–100
67. Papageorgiou C, Poggio T (1999) Trainable pedestrian detection. In: Proceedings of the 1999 international conference on image processing (ICIP ′99), Kobe, Japan, October 24–28, 1999
68. Papageorgiou C, Poggio T (1999) A trainable object detection system: car detection in static images. MIT Center for biological and computational learning paper No. 180, October 1999
69. Nair V, Laprise P, Clark JJ (2005) An FPGA-based people detection system. EURASIP J Appl Signal Process 7:1047–1061
70. Patil R, Rybski PE, Kanade T, Veloso MM (2004) People detection and tracking in high resolution panoramic video mosaic. In: Proceedings of 2004 IEEE/RSJ international conference on intelligent robots and systems, Sendai, Japan, September 28 – October 2, 2004
71. Spaanenburg H, Thompson J, Abraham V, Spaanenburg L, Fang W (2006) Need for large local FPGA-accessible memories in the integration of bio-inspired applications into embedded systems, 2006 international symposium on circuits and systems, Kos, Greece, May 2006
72. Achterop S, de Vos M, v. d. Schaaf K, Spaanenburg L (November 2001) Architectural requirements for the LOFAR generic node. In: Proceedings of the ProRISC'01, Veldhoven, The Netherlands, pp 234–239
73. Nijhuis JAG (January 1993) An engineering approach to neural system design. PhD thesis, Nijmegen University, Nijmegen, The Netherlands

Part III
Everything-in-the-Cloud

According to the Law of the "Conservation of Trouble," technology innovation results in benefits, but these benefits come at a price. For instance, development of multi-chip packaging results in devices with smaller size and increased compute density, but at the price of additional observability needs for diagnosis and test. In our case of cloud computing, increased capabilities will come at the price of increased needs for security and safety.

Technology needs to be applied to access working status of network elements, including self-test, tampering, adversity spoofing, and maliciousness all the way to "Byzantine Generals" network analysis. It is extremely important to access the integrity of the collected data, in addition to collecting the data itself. This increased level of safety and security technology is a missing element in many currently applied intelligent sensory networks.

The third part of the book addresses concerns for security and safety that will need to be supported in order to result in successful cloud-centric sensory networks.

Highlights

Everything-in-the-Cloud

The emphasis in cloud computing is to put as much as possible processing capabilities in the cloud, in order to keep the thin-client sensor nodes as simple as reasonable. This will become more and more significant as the intelligence gathering sensor networks become larger and larger, e.g., for weather and patient monitoring.

Enforced Security

Additional levels of computing and communication security will be the price to be paid for the availability of increased cloud-based processing capabilities. Security provisions need to be present to safeguard processing as well as communication resources. Various forms of redundancies could be provided in the cloud and the interconnection media.

Enforced Safety

Additional levels of system safety will be the price to be paid for the availability of increased cloud-based processing capabilities. Authentication of sensor network intelligence sources, including the detection of faulty and malfeasance behavior, will be required, in order to build and increase "trust" in integrated sensor-cloud systems.

Noise Society Impact

Trust in sensor intelligence data will need to be challenged and established, as a result of the current anonymity of the sensors as viewed by the Internet/cloud. A form of non-cooperative authentication or signature might need to be developed for sensor intelligence gathering network nodes (not necessarily for the Internet access points themselves).

Chapter 5
Security Considerations

In this and the following chapter we will review dependability requirements for cloud-centric sensor systems. Security[1] and safety represent the higher-level price to pay for improved computational capabilities and intelligence gathering. Security, as reviewed in this chapter, represents those system aspects that will allow a system of sensor nodes, without internal or external malicious intent, to operate with potentially faulty sensor nodes, computational resources, and/or communication paths. Security will be focused on making the system work and will be based mainly on static aspects of system well-being.

Safety, as reviewed in the subsequent chapter, concerns itself with the detection of (and survival over) internal or external malicious intent. Safety will focus on reacting to unexpected events and will therefore be based on dynamic aspects of system well-being. The described increased capabilities of security and safety technology are the missing elements in many currently applied intelligent sensor networks.

Foremost, when data arrive from a sensor into the cloud, we need to know that the data came from a working sensor node, that it has not been corrupted in transmission, and that the data were not maliciously created or introduced. Especially when life-critical decisions are based on the data, the emphasis on security and safety will be critical.

5.1 Reliability

In the course of time the major cost factor for an IC has regularly shifted. In the beginning it was hard enough to get a few devices fabricated but, when the yield started to rise, the design became the next hurdle. With increasing product

[1]Definition of Security (Merriam-Webster Online Dictionary): 1: the quality or state of being secure: as a: freedom from danger: safety b: freedom from fear or anxiety c: freedom from the prospect of being laid off <job security>, 2 a: something given, deposited, or pledged to make certain the fulfillment of an obligation b: surety, 3: an instrument of investment in the form of a document (as a stock certificate or bond) providing evidence of its ownership, 4 a: something that secures: protection b (1): measures taken to guard against espionage or sabotage, crime, attack, or escape (2): an organization or department whose task is security

L. Spaanenburg, H. Spaanenburg, *Cloud Connectivity and Embedded Sensory Systems*, DOI 10.1007/978-1-4419-7545-4_5,

complexity, additional effort is required to validate the fabricated device with the least effort affordable for a small price. And currently the worry is reaching even to outside the product and includes such lifetime effects as maintenance. Clearly test, debug, and validation have become essentially parts of a modern design philosophy.

What we describe in this section is aimed at three different phases during the life of a product. First of all, even formal verification needs a certified part, the *golden product*. If faults are left behind, they are probably hard to find anyway; once such a fault becomes noticeable, it will be even harder to find when the part is already integrated in a larger environment. But for an ASIC there may be no need to test beyond the confinement of the envisaged application. By and large we want to find out whether the device has some fault. For production purposes, such a test is sufficient. Diagnosis will only be performed for off-line quality assurance to support the ongoing effort of fabrication improvement by device characterization (Failure Mode Analysis or FMA).

5.1.1 Basic Concepts

There are many reasons why things can go wrong, and it is not necessarily a problem when things go wrong. The issue of quality is only at stake when the fault effect is seriously harmful or costly. There are so many levels of concern that the English language distinguishes between the failure, the fault, and the error. To state this in a single sentence:

> an error is a discrepancy between observation and specification that manifests as fault and leads to a failure.

Let us try for a little bit of conceptual understanding, as faults may be suppressed due to error redundancy. Even when faults are present, a system failure is not guaranteed because fault tolerance can keep the system alive. Let us take the example of a computer. An error is present when a transistor is designed, but not fabricated. When this is only one of the many transistors to pull-down a floating line, the function will only be performed slower and we do not have to notice the difference. When the transistor is the unique device to pull-down a line inside a ROM to provide constants to a multiplier, we will still not encounter this fault when the multiplier is not exercised within this interval. But when it does, the application may be seriously failing to launch a rocket.

In other words, the error lies at the heart of the problem. Without an error, there will never be a fault, let alone a failure. So we like to avoid errors as much as possible. One reason for errors to occur is because of the complexity of the problem we like to solve or because of the lack of intellectual challenge in coming up with the solution. So, instead of running the chance of getting bored, we may as well automate the process. If the solution is not that clear-cut, then version control to account for our diverse attempts is the least we can do.

The alternative is to apply formal proof techniques. This implies that there is an abstract rendering to which we can compare. There is gradual progress in this direction, but by and large it is still no competition for pure paying attention. Most

of the errors are of a horrifying stupidity and can be found by visual inspection. The key to error avoidance is simply to have the quality documentation that allows innocent bystanders to transparently locate what the designer has left behind.[2]

Errors that are still present after inspection should be easy to detect by non-visual means. In modern technology, we often rely on modeling and simulation. As always there is a balance at stake between costs and benefits. Simulation is a very computation-intensive affair for pinpointing errors at the smallest level of detail. Hence, it is a good thing to simulate only on parts where the existence of faults has been shown by *profiling*. This technique serves also to compact the test error set to a good compromise between size and coverage. In general, we want the existence of errors to be established as soon as possible, even if the granularity leaves much to be desired. As the design is a collection of individually designed objects, immature parts will often show by clustered errors within an object that calls for separate attention.

So far we have ignored the problem that errors are by nature unknown. If we do not want to test exhaustively, the question rises: how much testing is enough? If the fault effects are not really catastrophic, one might stop testing when the sensible benchmarks will run, or when it is unlikely that major errors will shown up within the guarantee period.

A popular way to characterize the maturity of a design is *fault seeding*. A number of faults are inserted and the tests are run. From the number of faults that got inserted and detected versus the number of overall detected faults, an estimate can be made of the actually remaining ones in the current design. This is the statistical counterpart of the approach that we have advertised implicitly in this text: *fault probing*. Keeping such test sets closely administrated with the various design versions supports the benchmarking approach. This is also the background of the naming convention for waveform and test-bench files enforced in the Xilinx design environment.

We have hardly made reference to hardware or software in the former discussion. The reason is not that hardware and software testing is identical. There are some differences of a fundamental nature, but not so much as often stated in literature. There is a difference in maturity in the different aspects of testing, but with the coming integration of hardware and software into a single design attempt, the differences will disappear.

The essential problem is in the abstraction scope. In software, we will find primarily design faults coming in clusters. In hardware, the breakdown of individual components at an experimentally estimated rate lies at the source of all problems. Design problems are usually eliminated quite soon, but the ones that are left will combine with fabrication faults but otherwise not cluster.

5.1.2 Probing Cases

Testing and verification are the key to robust and reliable product development. As shown in Fig. 5.1, one can never test enough. However, from a business development

[2]Data Logic, 1989.

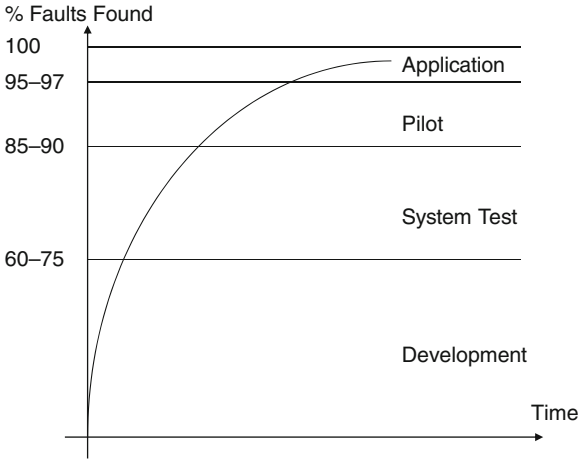

Fig. 5.1 Product development life-cycle

point of view, testing and verification are mere overhead and a product should be released when the fault count is acceptably low. The price and therefore value of "acceptable" is rather fuzzy and once sees, for instance, in the occasional automotive application that the amount of testing and verification is gradually decreased till suddenly the product gets into reliability problems.

Augustine [1] even went so far as to postulate that the price of a product can be estimated from the amount of testing that has taken place during its development.

The basis of testing and verification is the execution of use cases to the product that will reveal failures if present. Such use cases are normally given as test pattern sets. Such sets may differ on basis of the specific goal one wants to achieve. In general, we find benchmarks, extreme marks, and stress marks.

- Benchmarks are collections of typical pattern sets that are primarily meant to compare different hardware platforms. Originally collected for giving insight into computer architectural parameters and consequently misused for marketing considerations, they have gradually moved into the application area. Different applications have different needs, and a benchmark can therefore be used to find whether a proposed hardware platform suits the desires of the client company. To keep the comparison as fair as humanly possible, the benchmark sets are maintained and monitored by non-profit organizations. For embedded systems the organization is the EMBC: the Embedded Microprocessor Benchmark Consortium.
- Worst-case measurement originates from the design centering of analog circuits. The analog components have a min/max parameter spread around the typical value. Design centering is used to design the circuits such that the components have parameter values that in their min/max fluctuations will never bring the circuit outside its specification. In the general case, designing a system for worst-case brings a lot of overhead as the worst-case will almost never happen.

Furthermore, the worst-case is usually defined on the corners of the hypercube that encloses all the possible system values. These corners points may even not be existent in real operation.

- Stressmarks [2] set a compromise between the former two. Such pattern sets are benchmarks that bring extreme conditions of the system performance to the fore. The underlying assumption is that a system which behaves well according to the stressmarks will have a reasonable base performance. A typical stress mark is implied in the quote:

> Two CoreMark scores for the TI Stellaris were submitted recently. It is interesting to note that while the only difference between the submissions is the frequency, the CoreMark/MHz has changed (1.9 at 50 MHz vs. 1.6 at 80 MHz; a 16% drop). Since the device does not have cache, the CPU frequency to memory frequency ratio may come into effect, and indeed we find that the flash used on the device can only scale 1:1 with the CPU frequency up to 50 MHz. Once frequency goes above 50 MHz, the memory frequency scales 1:2 with the CPU[3]

In general, the patterns for test and verification will be applied in a layered way, where first the core functionality is checked and subsequently tested functions are in turn used to exercise remaining ones.

5.1.3 Software Metrics

Reliability can be measured in customer changes requests (usually per thousand or kilo lines of codes or kloc), faults per (usually kilo) line of code (typically 0.1 for a mature product) of failure arrival rate. There are in general three ways to improve the reliability: by quality assurance, by risk assessment, and by good design practice.

Quality assurance is usually done by a reliability growth model that typically assumes independent faults with a uniform distribution on a problem with high granularity. Its effectiveness is based on the assumption that "it is possible in most cases to obtain accurate measures and to have reasonable confidence in a particular situation so long as the required reliability levels are relatively moderate [3]." Herein "moderate" usually means less than 25%.

For fault probing according to Nelson [4], a computer program p is a specification of the function F on the set E of all input data. The operational profile of E is the set of p_i. Then the chance $R(t)$ that a run i of p will produce an acceptable output at time t is formulated as $R(t) = e^{-\int_o^t h(u).du}$ where $h(t_i) = -\ln(1-p_i)/\Delta t_i$ with Δt_i as the execution time of run i.

Risk assessment looks at the time between failures where such times between failures are assumed to be independent, while having an equal probability of fault exposure, independence of fault occurrence and perfect fault removal. In the Goel-Okumoto model, a non-homogeneous Poison process $P(N(t) = n) =$

[3] EMBC blog, 8 March 2010.

$H^n(t).e^{-H(t)}/n!$ is assumed for n failures as rate function $N(t)$ and $H(t)$ remaining faults at time t. This implies further a failure intensity function $d(t) = \{d(H(t))/dt\} = a.b.e^{-bt}$, with $H(t) = a\left(1 - e^{-bt}\right)$.

For the application of such formulae, one should take a fundamental difference between hardware and software into consideration. In hardware, individual components fail at an experimentally estimated rate. Design faults are a minority of all faults, and units fail independently of one another. In contrast, design faults can be assumed the only ones present in software. Moreover, they are seldom independent of one another.

5.2 Trustability

The purpose of security will be to increase trust[4]-worthiness of the data, the computing, the communication, and the system. A computer is secure if you can depend on it and its software to behave as you expect. This concept is often called trust: you trust the system to preserve and protect your data [5]. Trust can be established at the circuit level, the compute level, and the system level.

At the highest level the various security and trustability aspects will need to be integrated into a summary conclusion of total trustworthiness. Similar to the proper scheduling of dynamic memory refresh and also self-test, trust building and evaluation will need to be scheduled appropriately during the system application timeline.

5.2.1 Trusted Circuits

At the lowest level, we need to assure that components involved in the sensors are devoid of computational elements with undesired side effects in execution as a result of the presence of extraneous circuitry (Fig. 5.2).

Existing techniques to create testability, maintainability, and security are based on majority voting, multiple versions, and encryption. For embedded systems in submicron technology, these come together in the following ways.

- Critical computations can be protected by Arithmetic-Modular AN-codes, etc.
- Majority voting can be largely applied on the gate level to mask away soft errors. Multiple versions have become popular for test enhancement, such as BILBO,

[4]Definition of Trust (Merriam-Webster Online Dictionary): 1 a: to place confidence: depend <trust in God> <trust to luck> b: to be confident: hope, 2: to sell or deliver on credit transitive verb, 1 a: to commit or place in one's care or keeping: entrust b: to permit to stay or go or to do something without fear or misgiving, 2 a: to rely on the truthfulness or accuracy of: believe <trust a rumor> b: to place confidence in: rely on <a friend you can trust> c: to hope or expect confidently <trusts that the problem will be resolved soon>, 3: to extend credit to

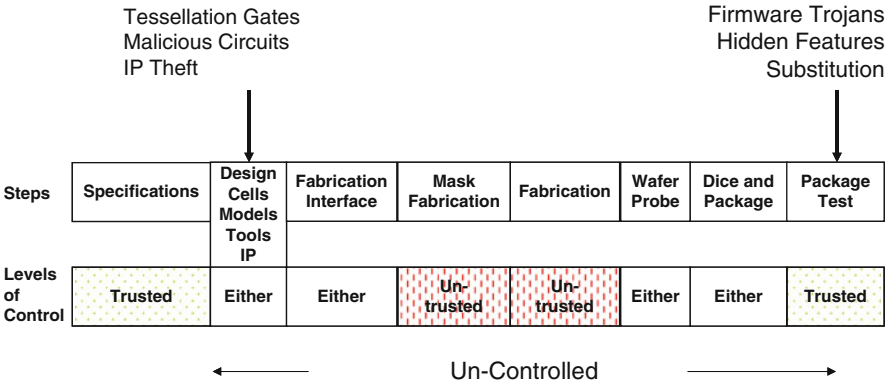

Fig. 5.2 Controlled and uncontrolled boundaries of the chip development process[5]

and involve some degree of majority voting as well. On a system level this can also be used to create a degree of reliability.

- Encryption is being used on the hardware level to avoid tampering, or to make the external communication more secure. The combination of multiple versions and encryption can aid to raise the level of trust within a system.

We envision the feasibility of providing protection along the UML to FPGA code generation chain. This approach may result in a complete design description (middleware) of a "trustable virtual machine," and a potential analysis of the degree of trust it would bring to FPGA users. Methods of ensuring trusted functions for field-programmable gate array devices from third parties are definitely of interest, given the proliferation of FPGAs in many deployed systems.

At the start of a design (see also Chapter 3), the high-level complex system architecture is implemented in UML. Most contractors and designers generally use Rational's Rose UML tools for system definition. UML designs generally do currently not seamlessly feed into detailed design. However, in a SystemC-based system design concept, we would allow for UML-to-SystemC front-ends such as have been described by STMicroelectronics [6] and the National University of Singapore [7].

Many tools exist for the actual synthesis of FPGA designs. Choices are among Xilinx FEXP, FPGA Express/FPGA Compiler II (Synopsys), LeonardoSpectrum/Exemplar (Mentor Graphics), or Synplify (Synplicity). For instance, Synplicity's Synplify product takes Verilog and VHDL Hardware Description Languages (HDL) as input and outputs, an optimized netlist in the most

[5]DARPA Microsystems Technology Office (MTO), TRUST in Integrated Circuits Program, Briefing to Industry, Brian Sharkey, i_SWCorp, 26 March 2007.

popular FPGA vendor formats. Celoxica's Agility compiler supports SystemC synthesis with a flow similar to that of their Handel-C (C-subset) tools. The Agility Compiler takes in SystemC and outputs IEEE compliant VHDL and Verilog RTL, as well as optimized EDIF netlists for FPGA.

The envisioned approach starts from a UML description, where the design development can be supported by automatically introducing trust in terms of multiple versions and encryption. Many architectural alternatives exist, which have to be evaluated before C++-code is produced from the UML source. In this sense the approach adds on current endeavors in UML to SystemC, and therefore C++, conversion.

As developments in FPGA device design advance, new generation of FPGAs increasingly are incorporating complex logic, such as whole microprocessor cores in their design. This trend along with continual improvements in both the performance capacity of FPGA hardware and the internal memory capacity of these devices has led to new opportunities for very complex FPGA applications to be developed. This, in turn, has increased the need for tools to support the "trustable" development of these complex and performance-sensitive applications. Of course, the same approach would be applicable for ASICs (Application-Specific Integrated Circuits).

As shown in the evaluation tree in Fig. 5.3, only performance optimization at higher levels of abstraction will payoff in the long run. Generally, more degrees of (design) freedom are present at those levels. It is a misconception to expect to be able to use FPGA personalization bit-level code for update/upgrade. Too many technology-specific design decisions have been made to get to that particular synthesized code pattern. A similar (reverse) argument can be made relative to trustability. It is our hypothesis that at higher levels of abstraction it will be easier to detect malicious code than at lower, closer to FPGA bit patterns, levels. Fortunately, at higher levels in SystemC many opportunities exist for simulation, verification, and performance evaluation.

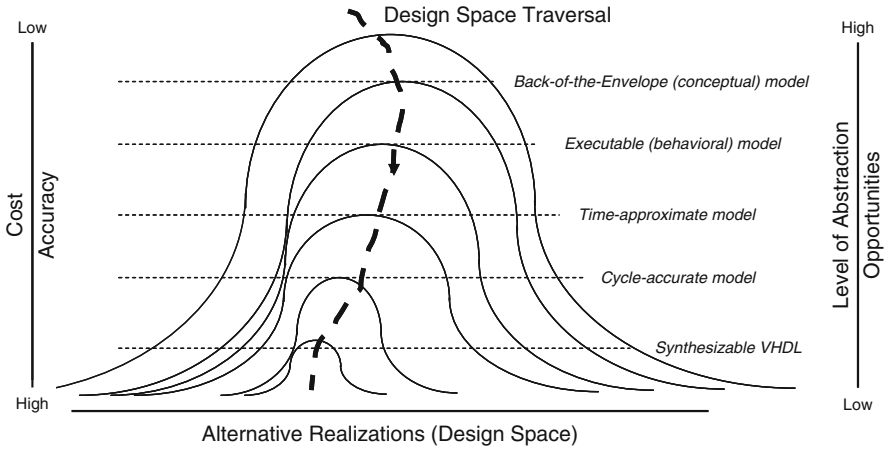

Fig. 5.3 The design pyramid [8]

Of course, after the proper linkages, a fully integrated turn-key UML to FPGA code development chain would be the best for overall trustability. The user/programmer would then have no impact on intermediate sub-tool interfaces. The key insight is that SystemC provides a very attractive and expressive language for complex system execution, as well as for its historical use for systems simulation and verification. Through the tightly integrated coupling of system simulation with system execution, it dramatically reduces design cycles costs and overall system quality dramatically improves. SystemC allows for seamless, single language system designs from a top-level down to lower-level hardware/software co-designs and ultimately "trusted" implementations.

The same arguments regarding trust as applied to the ASIC/FPGA design flow apply also to the application of replacement parts, in addition to the protection against use of stolen, counterfeited, or mislabeled parts.

5.2.2 Trusting-in-the-Edge

In principle what we need to trust is the edge of the network where most of the sensors are located. We need to protect the edge nodes against a hostile environment. At times the edge might include partially damaged devices with some remaining but faulty life. We need to be able to interrogate the edge nodes to see whether they are still trustworthy, and of course to what extent. The science of deployed sensor network security, or how to protect the nodes from any or all physical attacks, will have to experience a renaissance.

Since the notion of cloud-centric sensor networks is all about agile response to changing application scenarios, we will see the proliferation of software implementing even security-critical functions. This software will, however, reside within a sophisticated hardware/software architecture generally capable of protecting it from malicious compromise, and if not, with the ability to self-test and report on its own integrity.

The edges may be in parts of the network owned by third parties. This is the typical situation in large distributed networks and precludes that everything can be tested and verified by oneself. This brings trust back in one of two ways: edge intelligence and party authentication.

An old example of edge intelligence is in the Loop-Back structures used to separate a digital circuit part from the analog world. In a first layer, the analog I/O-port of a digital processor is short-circuited at the edge: the digital Loop-Back. This allows the digital processor to test its functionality including the port. In a second layer, the analog port is short-circuited instead: the analog Loop-Back. This allows the digital processor to test its functionality including the analog peripheral, but using a known, programmed analog value.

The Loop-Back theme has returned in the JTAG Boundary Test. On the board level there was a need to replace the *bed-of-nails test* technology, while on the chip level the concern for long synchronized data transfer was growing. The idea was launched to bound a function block by registers into a single clocking regime. This

can support a global asynchronous transfer between the blocks. In 1984, a redesign of the Intel 8045 microcontroller has shown this *Globally Asynchronous Locally Synchronous (GALS) concept* [9] to be very effective in reducing design complexity and optimizing performance (Fig. 5.4) [10]. With the registers in place on the block boundary, the same concept can also be used for testing purposes. And the concept is clearly no limited to the boundaries of functional blocks, but can easily be stretched to larger conglomerates and this is how it connects to the JTAG Boundary Scan.

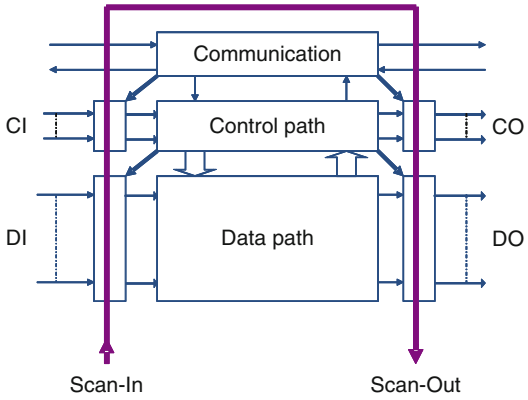

Fig. 5.4 Principle of boundary scan

The alternative (or rather the necessary complement) is trust-by-authentication. Where the other system parts have different owners, we simply have to trust that they are competent enough to do their part of the test and verification. These notions represents new capabilities currently not provided in fielded sensor network, capabilities without which realistic cloud-centric sensor networks would not be viable, even would be dangerous to apply.

5.2.3 Secure Mobility

Mobility adds an altogether new dimension to sensor networks. It used to be acceptable for mobile, that is tactical, systems to be less assured than stationary, that is strategic ones, because the perishable information that mobile network systems carried had a shorter intelligence lifetime. This is no longer the case, automobiles drive, patients walk, etc., still with access to the network.

In this new paradigm, the mobile edge operators will need to access the very same cloud facilities as the stationary networks, via their equally secure mobile access methods. Secure mobility adds, of course, the continuous need to know, or to be able to reliably predict, the exact location of the sensor network elements. This will include the testing for tactical realities such as temporary (wireless loss in tunnels, etc.) or permanent loss of equipment. Authentication becomes an extremely critical element in mobile sensor networks.

5.3 Resiliency

Resiliency,[6] the survival over malfunction, will be addressed in the following sections in available compute resources, in communication bandwidth, and in the functional operations of the sensor data collection nodes in the network.

5.3.1 Fault Tolerance

There are a number of ways to improve the fault tolerance of a sensory system. In essence, it all comes down to decision-making based on redundancy as made available through the system structure. Redundancy means that there are more available resources than absolutely necessary. One may therefore be inclined to see redundancy as overhead, as without redundancy the system will still run but cheaper and more efficient. But that is only true in the ideal case. In the non-ideal case, the system is vulnerable to faults.

The question then rises how much that costs and what the benefits are. If the fault is threatening human life, no cost is too high; if the fault is just a nuisance, any cost may be too high. Apparently there is a spectrum of cost/benefit balances, and this will be reflected in the diversity of architectural measures.

5.3.1.1 Majority Voting

Where the single sensor is too vulnerable to be acceptable for robust applications, the first-hand solution is to have more of them. Assuming the fault to be destructive, multiple copies of the same (Fig. 5.5) allows deciding which sensor is affected and which ones are not. Two sensors allow already detecting that one is wrong, three or

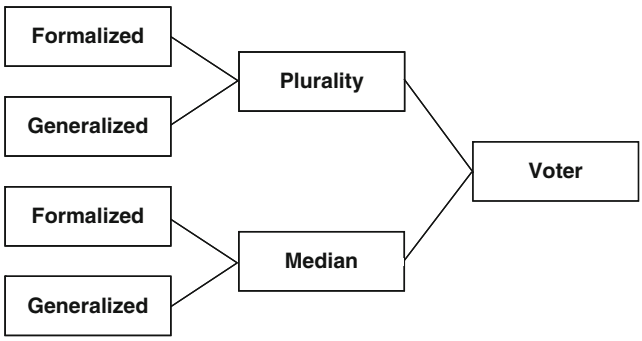

Fig. 5.5 *N*-Modular redundancy

[6]Definition of resilience (Merriam-Webster Online Dictionary): 1: the capability of a strained body to recover its size and shape after deformation caused especially by compressive stress, 2: an ability to recover from or adjust easily to misfortune or change

more facilitate determining which one. This is only true where only one sensor is faulty.

One can learn a lesson from the Ariane 5 satellite launch, where two faulty computers decided that the third (and actually correct) one had to be disabled. Overall, it can be seen that for n simultaneously faulty sensors, we need at least $2n+1$ devices.

Unfortunately, the case of a single destructive fault is rare. It does exist, but it is an exception rather than a rule. One reason is that a fault may influence different sensors in different ways. There may be an underlying process from which the fault emanates that is shared by a number of sensors. The other reason is that having multiple expensive sensors is too costly. So, in order to improve robustness while not increasing cost, one may allow a lower quality of sensors.

This may seem counter-intuitive. How can accepting a lower quality improve the result? The reason is that we step away from simple voting and move on to an averaging scheme. Let us illustrate this with a simple arithmetical example. Of n–1 sensors give a value "1," while a single sensor provides a value "a," majority voting will result in the value "1" while averaging gives $(n–1+a)/n = 1– (1–a)/n$, which for a sufficiently high n is roughly the same. This shows that for a large number of sensors the individual quality becomes less important.

Clearly the number of sensors cannot be raised arbitrarily. But here statistics comes to the rescue. By proper characterization over time, we can determine the likelihood of the sensory value combination. Unfortunately, this does not remove the dependency on lifetime effects, which may cause all sensors to gradually deteriorate in perfect harmony. There is no easy way to get out of this dilemma and other schemes are still needed.

5.3.1.2 Multiple Versions

A typical area where majority voting has never been possible is in the detection of (software) system design flaws. Putting a single program together is hard enough, writing the same program twice while not making the same design mistakes is not very likely. The solution exploited in software engineering is therefore to have different descriptions of the same intended function. In terms of sensors, this entails the use of different technologies. For instance, temperature can be measured from the expansion of a liquid as well as from the color of the heated material. The chance that two technologies share an underlying process is remote, while it seems utterly unlikely that such a process influences both measurements in a similar manner.

It frequently occurs that the same reading can be obtained from more than one sensor, even if they are not the same. For instance, a temperature can be measured from a thermometer and through a camera. In sensory networks the existence of multiple versions comes in through the backdoor. No sensor is ideal and the reading is usually a combination of several physical effects. This creates a set of overlapping views where each sensor has a primary measurement, but in combination with a number of secondary ones. Unfortunately, this situation is often masked away by filtering in measurement instruments. This builds an argument for using the raw sensors.

The diversity in the sensors can have more causes. An obvious one is from the location of the sensor. When, for instance, a number of distance sensors are distributed over the perimeter of a robot, they will have an angular view. Having overlapping angles allows bridging a malfunctioning one. Is this really multiple version or just a local form of majority voting? We see it as the former for two reasons. Firstly, no two sensors are ever the same. Secondly, angles differ, while measuring the object, because the observation changes. Consequently, the objects are nonlinearly deformed and the amount of occlusion will change. An example is in measuring from video fragments, where additional accuracy can be reached by taking details from subsequent frames into account.

When sensors are to be distributed, they can also be bound to diverse carriers. This virtualizes the sensor concept to the cooperation of a number of contributing parts. Seemingly they are disjoint, but a judicious flow of information between them will finally create the desired functionality. This is the ultimate multiple-version situation, where (a) the overall sensor function is not physically packaged in a single container, and (b) even the parts of the sensor function can be redundant. This creates a sensor structure that adheres to the concept of modularity.

5.3.1.3 Mutual Trust

The redundancy concept implies a global decision-making. But having a central control can only be part of a solution. The reason is that, though data usually flow from input to output, faults may take arbitrary ways. Furthermore, while data propagate as designed, faults have their own timing. Such non-trustable behavior makes central control hard to maintain. Horror stories about mishaps on the Electric Grid illustrate this with very expensive examples (see Section 6.3.2).

Reliability issues on the virtual sensor are remarkably similar to those on a network. Hence we can discuss similar solutions. But because a sensor is not a network node, the option of making each part self-testing has no additional virtues. The second option is to have the sensor parts test one another.

5.3.2 Fault-Tolerance in Computing

Fault-tolerance[7] can be provided by introducing various forms of redundancy in the compute and/or communication architecture:

- Software Fault-Tolerance
  - Repeated execution of application sequentially or in lockstep, followed by a voting procedure and/or a possible repeat operation. This is a pipelined architecture version of providing software fault-tolerance.

[7]Definition of fault-tolerance (Merriam-Webster Online Dictionary): relating to or being a computer or program with a self-contained backup system that allows continued operation when major components fail

- Self-Test
  - Testing locally of a processing node either by preloaded functional code or by use of circuitry (BILBO) in the boundary-scan mechanism. The test can be performed and directed at the module level.
- Roving Nodes
  - Redundant nodes can be taken out of operation for self-test, followed (if still functionally capable) by their re-introduction into the processing stream and the selection of subsequent roving nodes.
- Chordal Bypass
  - Ring or macro-pipeline connected processing elements can be provided with an additional connective that bypasses subsequent node(s). The bypass length needs to be selected, as well as any bidirectedness.
- Triple Redundancy
  - By performing three instances of the same computation in parallel, followed by a voting procedure, we can handle failure of one out of three processing elements
- Degraded Mode
  - It is still possible to perform the proper functionality after redundancies have been consumed, though not at the proper performance level. A schedule can be established for various levels of degradation.

5.3.3 Secure Communications

A high level of accuracy has to be maintained in network communication with and without redundancy in the connectivity:

- Redundancy-in-Space
  - Redundancy can be provided in the parallelism of computing and networking. Results can then be voted on for accuracy (Fig. 5.6a).
- Redundancy-in-Time
  - Redundancy can also be provided by repeated computations (in a pipeline) over time, followed by comparison of the subsequent results (Fig. 5.6b).
- Redundancy-in-Information
  - Of interest in sensor networks plugged into the cloud are “tiny” versions of encryption and decryption algorithms. Power and interconnect bandwidth limitations will make this obvious.

Information redundancy has been widely applied in digital communication techniques. In the beginning we see duplication, for instance, when checking sent data versus received data. Then, in an effort to speed up communication by not having

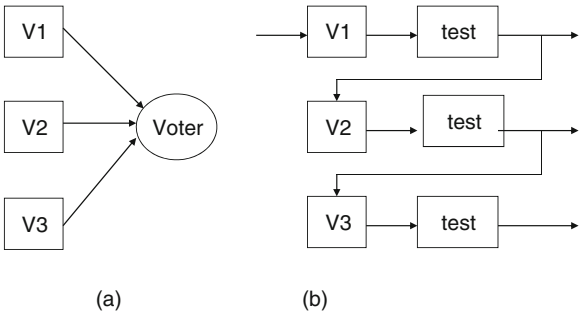

Fig. 5.6 Redundancy in (**a**) Space and (**b**) Time

to wait for a response, we have arrived at parity coding, check sum coding, cyclic codes, and arithmetic codes. They are still widely in use and have even spread to neighboring areas such as video stream compaction. Our interest here is mainly en- and decoding for security purposes.

Cryptography is rapidly diffusing into society. With the explosive growth of the web-based appliances market, one can no longer afford to go unprotected. Cryptography is needed and it must often be hardware-supported to handle product tampering at any time. Trusted hardware will have encrypted data- and instruction flows as a first line of defense [11]. However, encryption still counts as overhead and must not take too much of the design and implementation budget.

Traditionally, enciphering has been based on blocks, fixed-sized chunks of data. A block cipher is a type of symmetric-key encryption algorithm that transforms a fixed-length block of plaintext (unencrypted text) data into a block of ciphertext (encrypted text) data of the same length. Famous block ciphers, such DES and AES, need to iterate a round function many times to keep the code from breaking by sheer computational power [12]. Unfortunately, the hardware implementation of block ciphers is quite costly. The best block cipher for hardware is probably AES [13].

Another class, stream ciphers, is especially interesting for embedded products. A stream cipher is a type of symmetric encryption algorithm that operates on smaller units of plaintext, usually bits, in a time-varying manner. Well-known examples as A5/1 (used in GSM) and E0 (used in Bluetooth) are not acclaimed for their safety [14], but are definitely low in overhead as embedded program. The stream cipher Snow is developed at Lund University [15–17]. The current version Snow 2.0 has improved security and is one of the two dedicated stream cipher designs that have been applied to ISO/IEC 18033-4 [18].

5.3.4 Redundancies in Large-Scale Sensor Networks

Redundancies can be created at various levels in multi-sensor networks. Specific benefits can be attained for homogeneous and heterogeneous sensors arrays. In

addition, increased performance can be gained from spectral, temporal, and spatial analysis, as well as from the application of learning techniques.

5.3.4.1 Homogeneous Multi-sensor Arrays

Several examples of sensors arrays have been built and evaluated. It has been noted that for most of them not all sensors in the array are needed as redundancy exists. Reasonable performance can quite often be attained from a subset of devices.

- Multiple Microphones
  - MIT LOUD 1020-node Microphone Array [19] used for multiple speakers' speech recognition and enhancement. A reasonable performance can be obtained already from 60 microphones.
- Multiple Aperture Imaging
  - Stanford University's Multi-Aperture Image Sensor [20] is able to capture depth information, provide close proximity imaging, achieve color separation and increased tolerance to defective pixels in a single sensor chip.
- Multiple Cameras
  - Large Camera Arrays [21] can be applied for integrated accuracy at various image distances from the cameras. The general application is one called Computational Photography.
- Multiple Wireless Antenna and Receiver Nodes
  - OrderOne Networks [22], a mesh network with 720 transmit/receive nodes in the Wireless Information Network Laboratory at Rutgers University.

5.3.4.2 Heterogeneous Multi-sensor Fusion

An integrated multi-sensor detection system, based on for instance a radar subsystem, a multi-spectral infrared subsystem, and a panoramic view visual subsystem, will result in better performance than by any of the individual sensors. With image co-registration, subsequent imagery can be integrated for improved signal-to-noise performance.

5.3.4.3 Heterogeneous Multi-level Subsumption

For sensor networks some level of cross-functional redundancy is not only needed but also demanded. That notion is based on the subsumption[8] principle of Rodney Brooks [23]. When he presented the construction of intelligent robots, it was noted

[8]Definition of Subsuming (Merriam-Webster Online Dictionary): to include or place within something larger or more comprehensive: encompass as a subordinate or component element <red, green, and yellow are subsumed under the term "color">

that there is a functional hierarchy involved so that if one sensor fails, other sensors can at least help out to achieve the desired goal. For instance, vision can be used to measure distance but if the eye fails that the ear can provide some help by interpreting echoes, and so on.

The subsumption of sensory handling will make a number of assumptions on how the recipe requirements will be entered. Using speech, natural language processing is an option, but not a necessity. Recipes have a rigid structure, and often a few simple and well-chosen words are sufficient to make an acceptable selection. At the same time, the person can be identified to check on medical restrictions or to enable a personal delivery. When the voice fails, such words can still be typed and SMS-ed, but authentication is limited. It is not necessary that the owner of the telephone expresses the meal wish. It is also possible to fill a form manually. In that case authentication is not really hard, but rather cumbersome and therefore not really providing an inviting offer.

Subsumption can indicate not only how a particular system function can be supported by various processes but also how this function can be accommodated in various system components. This provides the glue between the Path and the Wiring diagram when strict orthogonality is destroyed by redundancy. Or, to put it more positively, subsumption must be made explicit to design implicit fault tolerance on basis of redundancy.

We can identify now the existence of a subsumption hierarchy, based on detection and diagnosis. On every level, different sensors will be needed. In laying out the sensors over the field, we tend to minimize cable or hop distances. Therefore, an aerial organization seems to be called for. But some of the sensors may move across areas and will therefore have to be connected.

5.3.4.4 What Comes Next in Secure Sensor Networks?

When you are bothered by a dinosaur, killing it may ease your problem. When you are bothered by a swarm of flies, killing one fly is not enough. When the computer was a single frame in a single room, security was a simple matter of keeping the door shut. With the Internet connecting everyone and everything all over the world, security has left through the chimney.

Large creatures are more vulnerable than small ones. And the same is true for computer networks. Even when security is high on the communication links, the fact that there are many points to tap already decreases the safety. The problem is that there is no safety by composition. Small networks can be formally proven to be safe; but when larger ones are assembled from smaller, then they do not automatically inherit their safety. This can be attributed to the fact that together they cover new areas in the problem space. In a simple example, suppose we have a function $f(x,y)$ and $g(y,z)$, then the composition of f and g lies in the (x,y,z) field, which covers much more than each of the parts. This problem seems to be eliminated by assuming that f and g are independent. Unfortunately, the composition brings by nature dependency into play.

This is especially true when faults can occur. The usual assumption in studying faults is that their occurrence will not increase the dimensionality of the system, for instance, by introducing new states (i.e., historical effects or memory). This is a severe limitation. Furthermore, a fault may change the intended independency of the parts. So, any system that is designed from the ideal (i.e., the intended behavior) is likely to display new behavior when mishaps happen. This has two consequences. Firstly, abnormal behavior has to be expected as it can in principle not be avoided (Murphy's Laws are no incident but lessons from the past). Secondly, abnormal behavior must be taken into account from the start of the design.

We have suggested that redundancy brings more safety. It should be noted that we did this in connection with the use of the subsumption hierarchy and not for the general case. The fact that redundancy brings stability has generally been accepted. A typical example is in cellular structures where the strong local interconnect brings all the advantages of feedback structures into play [24]. When errors creep in, the intended function remains (Fig. 5.7).

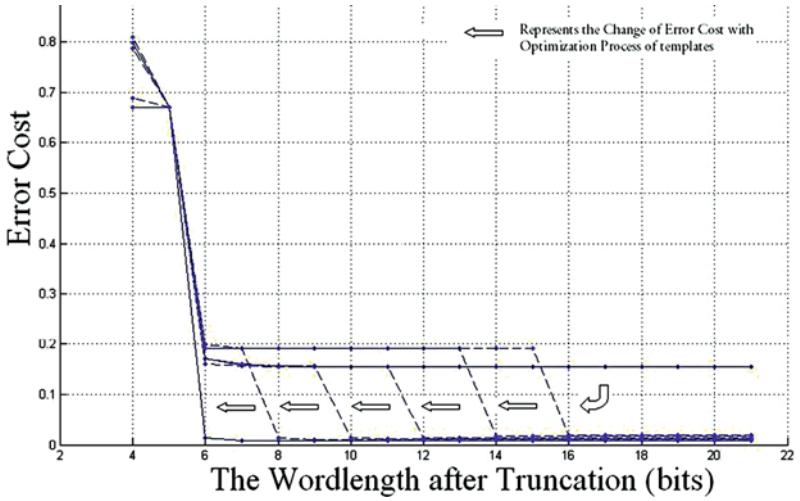

Fig. 5.7 CNN Stability in the presence of numerical errors

This has worked nicely because the structure is fully symmetrical. This makes a Cellular Neural Network (CNN) a good platform to look at such phenomena. When symmetry is not the case, the effect of redundancy becomes less clear. Furthermore, the effect is only illustrated for the static case and may be less clear for dynamic patterns (Fig. 5.8). CNNs are typically second-order differential equation sets and therefore they can be brought to handle dynamic patterns.

Actually they are numerically hard to achieve [25]. The coefficients have to be chosen very precisely, as otherwise the patterns will die out or even do not appear. This is similar to what has been observed by Buldyrev [26]. Most of the faults die out

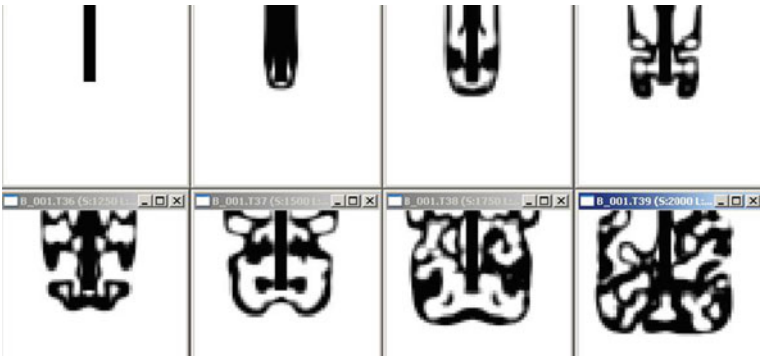

Fig. 5.8 Autonomous pattern generation in a CNN

almost immediately. However, under special circumstance, even the slightest fault will escalate and bring a deluge of errors spreading fast through the entire system. This exemplifies once again the need for compartimentation (to be addressed in Section 5.4.1).

5.4 Authentication

Authentication[9] is clearly the highest level of security, but also poses the most severe technical problems. Biometric measures have been studied to ensure the identity of a person. It is assumed that biological diversification is large enough that each finger, iris or voice has a different imprint.

Unfortunately, no classification system has been developed yet that works acceptably perfect from a single imprint source. This is established by the Receiver Operation Curve (ROC), where the balance between acceptance and reject defines the native shortcomings from the single classifier. Multiple classifiers, using more than one imprint source simultaneously, will lead to substantial improvements.

Even larger problems might exist in the distinction of individually (on paper) identical sensor subsystems. Electrical or other computational techniques might be involved in the distinction process.

[9]Definition of Authenticate (Merriam-Webster Online Dictionary): 1 obsolete: authoritative, 2 a: worthy of acceptance or belief as conforming to or based on fact <paints an authentic picture of our society> b: conforming to an original so as to reproduce essential features <an authentic reproduction of a colonial farmhouse> c: made or done the same way as an original <authentic Mexican fare>, 3: not false or imitation: real, actual <based on authentic documents> <an authentic cockney accent>, 4 a of a church mode: ranging upward from the keynote – compare plagal 1 b of a cadence: progressing from the dominant chord to the tonic – compare plagal 2, 5: true to one's own personality, spirit, or character

5.4.1 Sensor Accreditation

On the Internet there is complete anonymity between the different users. However, the anonymity creates also all kind of problems such as spam, phishing. It creates a high level of information distrust, a kind of Internet societal noise. Since there is a distinct difference between people and sensors, there might not be the same need for anonymity for sensors as there currently is for people. The current upgrade from IPv4 to IPv6 will provide the opportunity to have distinct identifiers for a relative large amount of sensors.

In true sensor-into-the-cloud systems, there might be need to remove anonymity on the sensor site. It will be necessary to know the identity of the individual sensors, their assigned functionality, as well as their respective locations. If not explicitly provided with the identity of the sensor, we will need to develop methods for implicit identification and accreditation.

One way to achieve this is by compartimentation, such as for instance in social networks. In such networks, the idea is to provide trust to parties with whom a social trust exists, like for instance family members. Extending such social relationships to sensors as being part of a family can be done on top of an Internet addressing scheme.

Most often, off-line network problems have a very basic cause. For instance, the quality of the electricity supply leaves nowadays much to be desired. Regularly, the supply is interrupted for a short time. If this time remains below 10 ms, the human operator will not even notice it, but the machines get affected at least to the degree of a noticeable reduction in lifetime. If the supply elapse takes longer, the electronic equipment gets deregulated.

A much-advocated solution is the insertion of an UPS to bridge such gaps. This has caused a consequential problem in an Internet router, where the electronic equipment kept functional but the ventilator stopped, causing overheating of the system followed by a melt down. Future local supply systems can remedy this fault category, but still leaves other issues such as simultaneous switching untouched.

5.4.2 Sympathetic Testing for Trustability Assessment

In order to assess functional performance at the sensor nodes, self-test routines will have to be performed. The test will be performed under control from the cloud. The end result from the test needs to result in an as unique as possible signature, a digital fingerprint.

Suppose one of the cloud's clients is trusted. Then still the question remains whether the client is who it claims to be. In contrast to authentication of human beings, the node has no name and no identity; but even if that would be the case, then identity theft is still an issue. So the question is to establish the identity in a way that cannot be defrauded.

For human beings we have the alternative of biometric authentication, where we rely on the uniqueness of the way in which nature creates living creatures. We need clearly something similar here.

In Section 5.1.2 we have discussed the stressmarks of a digital system. There are patterns by which critical parameters of the implementation can be found. We quoted there a test whereby a break-point in the relation between system response time and basic clock speed could be established. This principle is sound, but still too little discriminatory to serve as a reliable digital signature.

Let us take a look at two classical examples of discriminatory computer architecture tests, both taken from the Hennessey and Patterson book [27]. The first example is on different system speed for different arrangements of the memory hierarchy.

A typical benchmark for a hierarchical memory system is shown in Fig. 5.9 [28]. The lower curve relates to the situation where the L1 cache covers all. Its effect seems to stop at 16 k. For the higher curve, relating to the L2 cache, the end seems to come around 256 k. It seems that block replacement kicks in for blocks at 8 bytes. If we make a program that has the maximal problems with the cache contents (something you would not do normally), we get clearly discriminated for the miss penalty values. Loosely measured, this comes to 15 ns for the L1 and 55 for the L2. Clearly, the trick is to use "an unfair benchmark."

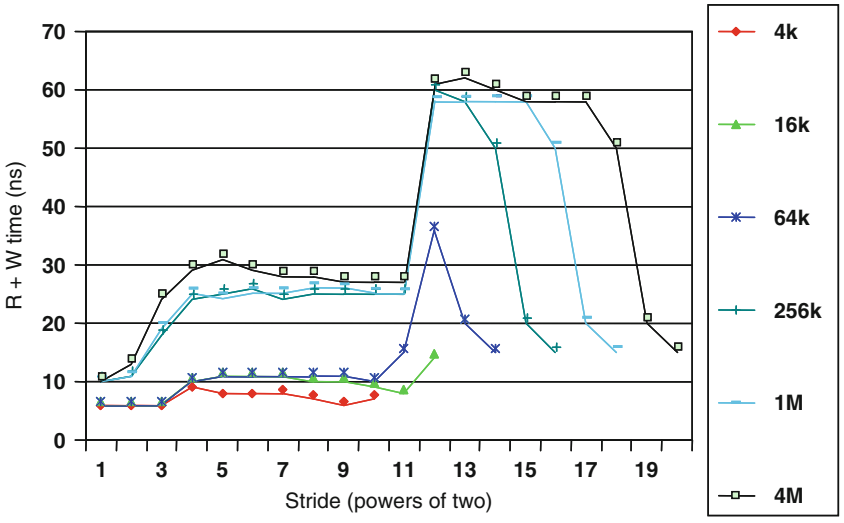

Fig. 5.9 A Sunblade 1000 is Benchmarked (which has separate L1 instruction and data caches as well as a unified L2 cache)

In the next example [28], there are two machines with the same processor and main memory, but different cache organizations. Assume that both processors run at 2 GHz, have a CPI of 1, and have a cache (read) miss time of 100 ns. Further,

assume that writing a 32-bit word in main memory requires 100 ns (for the write-through cache). The caches are unified – they contain both instructions and data – and each cache has a total capacity of 64 kB, not including tags and status bits.

The cache on system A is two-way set associative and has 32-byte blocks. It is write-through and does not allocate a block on a write miss. The cache on system B is direct mapped and has 32-byte blocks. It is write-back and allocates a block on a write miss.

First we will now look for a program that makes system A run as fast as possible relative to system B's speed. For this we take a look at the MIPS code in Fig. 5.10. We make two assumptions in this code: First, the value of *r0* is zero; second, locations *f00* and *bar* both map onto the same set in both caches. For example, *f00* and *bar* could be 0×00000000 and 0×80000000 (these addresses are in hexadecimal), respectively, since both addresses reside in the set zero of either cache.

Fig. 5.10 MIPS code that performs better on Cache A

```
F00:    beqz    r0, bar     : branch iff r0 == 0
                .
                .
                .
bar:    beqz    r0, f00     ; branch iff r0 == 0
```

On Cache A, this code only causes two compulsory misses to load the two instructions into the cache. After that, all accesses generated by the code hit the cache. For Cache B, all the accesses miss the cache because a direct-mapped cache can only store one block in each set, yet the program has two active blocks that map to the same set. The cache will "thrash" because when it generates an access to *f00*, the block containing *bar* is resident in the cache, and when it generates an access to *bar*, the block containing *f00* is resident in the cache. With all access hits, Cache A allow the processor to maintain CPI = 1. Cache B misses each access at cost of 100 ns, or 200 ns clock cycles. This Cache B allows its processor to achieve CPI = 200. Cache A offers a speed-up of 200 over Cache B.

Now we look at a program that makes system B run as fast as possible relative to system A's speed. For this we take a look at the MIPS code in Fig. 5.11. We make two assumptions: first, locations *baz* and *qux* and the location pointed to by *o(r1)* map to different sets within the caches and are all initially resident; second, *r0* is zero (as it always is for MIPS).

```
Baz:    sw      0(r1), r0   ; store r0 to memory
Qux:    beqz    r0, baz     ;branch iff r0 == 0
```

Fig. 5.11 MIPS code that performs better on Cache B

This code illustrates the main thrust of a program that makes a system with Cache B outperform a system with Cache A, that is, one that repeatedly writes to a location that is resident in both caches. Each time the *sw* executes on Cache A, the data stored are written to memory because Cache A is write through. For Cache B, the

sw always finds the appropriate block in the cache (we assume the data at location *0(r1)* are resident in the cache) and updates only the cache block, as the cache is write back; the block is not written to main memory until it is replaced.

In the steady state, Cache B hits on every write, and so maintains CPI = 1. Cache A writes to memory on each store, consuming an extra 100 ns each time. Cache B allows the processor to complete the execution of one iteration in two clocks. With Cache A the processor needs 202 clocks per iteration. Cache B offers a speedup of 1012 over Cache A.

5.4.3 Remote Entrusting

The European[10] research project, RE-TRUST, which stands for Remote EnTrusting by RUn-time Software authentication, provides a novel methodology for both software-only and hardware-assisted remote entrusting (RE).

Whereas hardware-assisted entrusting requires a special chip either on the computer's motherboard or inserted into a USB drive, RE-TRUST uses logic components on an untrusted machine to enable a remote entrusting component to authenticate – via the network – the untrusted machine's operation during runtime. This means it ensures that the software is running properly and that the code integrity is maintained, thus almost completely guaranteeing security.

5.5 Security-as-a-Service

Oberheide et al. [29] have described an innovative technique for virtualized in-the-cloud security services for mobile devices, a Security-as-a Service approach for sensor-based thin-clients. Antivirus software on your personal computer could become a thing of the past thanks to this new "cloud computing" approach to malicious software detection developed at the University of Michigan. Key to their approach is a new model for malware detection on end hosts based on providing antivirus as an in-cloud network service.

Instead of running complex analysis software on every end host, they have suggested that each end-client runs a lightweight process to detect new files, send them to a network service for analysis, and then permit access or quarantine them based on a report returned by the network service (Fig. 5.12). Multiple, heterogeneous detection engines should identify in parallel the presence of malicious and unwanted software. Similar to the idea of N-version programming, they proposed the notion of N-version protection and suggest that malware detection systems should leverage the detection capabilities of multiple, heterogeneous detection engines to more effectively determine malicious and unwanted files.

[10] http://cordis.europa.eu/ictresults.

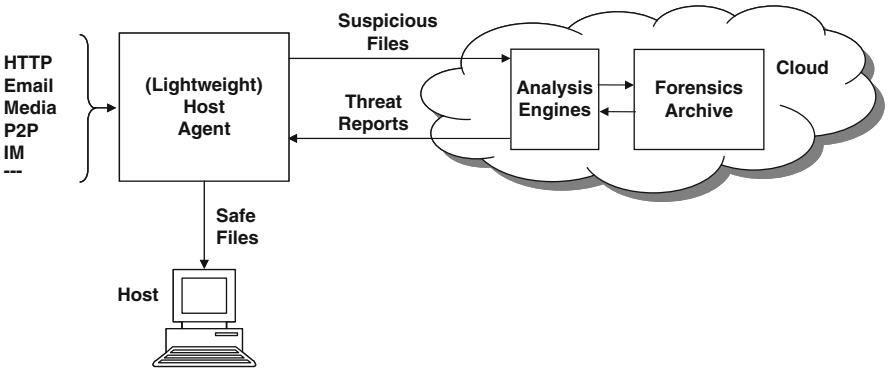

Fig. 5.12 Architectural approach for in-cloud file analysis service

The University of Michigan's CloudAV [30] is accessible to any computer or mobile device on the network that runs a simple software agent. Each time a computer or device receives a new document or program, that item is automatically detected and sent to the antivirus cloud for analysis. The CloudAV system uses 12 different detectors that act together to tell the inquiring computer whether the item is safe to open. To develop this novel approach, the researchers have evaluated 12 traditional antivirus software programs against 7,220 malware samples, including viruses, collected over a year. The vendors tested were Avast, AVG, BitDefender, ClamAV, CWSandbox, F-Prot, F-Secure, Kaspersky, McAfee, Norman Sandbox, Symantec, and Trend Micro. They have found that CloudAV provides 35% better detection coverage against recent threats compared to a single antivirus engine and a 98% detection rate across the full data set. Also of interest is that not all detectors need to be applied to receive a reasonable performance. Moving from one to two detectors results in a large return, from two to three in a lesser and so on (Fig. 5.13).

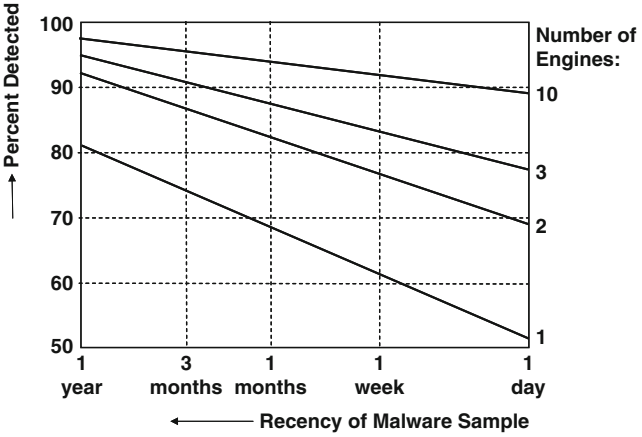

Fig. 5.13 The continuous coverage over time when a given number of engines are used in parallel [30]

Of course, this performance may be dependent on the order of selection and the quality of the respective detectors.

A similar cloud-based approach, in this case for the detection and remediation of software-based failures, has been advocated in the Dependable Systems laboratory at the (Ecole Polytechnique Federale de Lausanne) EPFL's Dimmunix system. In it, the notion of deadlock immunity is introduced – a property by which programs, once afflicted by a given deadlock, develop resistance against future occurrences of similar deadlocks.

Dimmunix [31] develops deadlock immunity with no assistance from programmers or users. The first time a deadlock pattern manifests, Dimmunix (in the cloud) automatically captures its signature and subsequently avoids entering the same pattern. Signatures can be proactively distributed to immunize other users who have not yet encountered that deadlock. Dimmunix can be used by customers to defend against deadlocks while waiting for a vendor patch, and by software vendors as a safety net.

5.6 Summary

The explosion of the embedded world leads to a major security issue. Where each sensory system is invisibly serving society, reaching out to the edge of the electronic reality, any error has major effects before it becomes noticeable. We see this in the history for Electricity Generation, where ICT has helped to decrease the amount of service problems, but on the other hand made the remaining ones bigger and bigger.

In ship building one is well aware of the Titanic disaster. Any hole in the hull may cause the ship to sink in a couple of minutes. In the military practice this has given rise to compartimentation, such that the ship contains a large number of compartments that can each be sealed off in case of an emergency. With the increasing size of sensory systems, it seems reasonable to take similar measures.

Unfortunately, the system's architecture is not fixed by design. Therefore, we need another principle than designed composition to define compartments. This chapter proposed to take ownership as the ruling principle. If the sensory system is composed from authenticated parts, "incoming water" is less likely to occur and therefore security is raised.

However, one has to be aware of the fact that security will be hard to push to 100%. There will be always somebody that leaves the door between compartments wide open. Therefore, next to security, we also have to be attentive to safety.

References

1. Augustine NR (1987) Augustine's law. Penguin Books, New York, NY
2. Kumar S, Pires L, Ponnuswamy S, Nanavati C, Golusky J, Vojta M, Wadi S, Pandalai D, Spaanenburg H (February 2000) A benchmark suite for evaluating configurable computing systems -status, reflections, and future direction, Eight ACM International Symposium on Field-Programmable Gate Arrays (FPGA2000), Monterey, CA

3. Böhm B (August 1986) A spiral model of software development and enhancement. ACM SIGSOFT Softw Eng Notes 11(4):14–24
4. Nelson VP (July 1990) Fault-tolerant computing: fundamental concepts. IEEE Comput 23(7):19–25
5. Garfinkel S, Spafford G (1997) Web security and commerce. O'Reilly Nutshell, O'Reilly Media, Sebastopol, CA
6. Vanderperren Y, Pauwels M, Dehaene W, Berna A, Ozdemir F (2003) A SystemC based system on chip modeling and design methodology. In: Muller W, Rosenstiel W, Ruf J (eds) SystemC: methodologies and applications. Kluwer, Norwell, MA
7. Tan WH, Thiagarajan PS, Wong WF, Zhu Y, Pilakkat SK (June 2004) Synthesizable SystemC code from UML-models. In: International workshop on UML for SoC-design (USOC2004), San Diego, CA
8. Lieverse P, van der Wolf P, Deprettere E, Vissers K (2001) A methodology for architecture exploration of heterogeneous signal processing systems. J VLSI Signal Process Signal, Image Video Technol 29(3):197–207
9. Spaanenburg L, Duin PB, Woudsma R, van der Poel AA (7 April 1987) Very large scale integrated circuit subdivided into isochronous regions, method for the machine-aided design of such a circuit, and method for the machine-aided testing of such a circuit. US Patent 4656592
10. Spaanenburg L, Duin PB, van der Poel AA, Woudsma R (March 1984) One-chip microcomputer design based on isochronity and selftesting, Digest EDA'84, Warwick, England, pp 161–165
11. Smith S (2003) Fairy dust, secrets, and the real world. IEEE Secur Priv 1(1):89–93
12. Daemen J, Rijmen V (2002) The design of Rijndael. Springer, Heidelberg
13. Su C-P, Lin T-F, Huang C-T, Wu C-W (December 2003) A high-throughput low-cost AES processor. IEEE Commun Mag 41(12):86–91
14. Ekdahl P, Johansson T (2000) Some results on correlations in the bluetooth stream cipher, Proceedings of the 10th joint conference on communications and coding, Obertauern, Austria, pp 16
15. Ekdahl P (2003) On LFSR based stream ciphers analysis and design, Ph.D. Thesis, Department of Information Technology, Lund University, Lund, Sweden
16. Fang WH, Johansson T, Spaanenburg L (August 2005) Snow 2.0 IP Core for Trusted Hardware, 15th international conference on field programmable logic and applications (FPL 2005), Tampere, Finland
17. Fang W (2005) A hardware implementation for stream encryption snow 2.0, M. Sc. Thesis, Department of Information Technology, Lund University, Lund, Sweden
18. Mitchell CJ, Dent AW (2004) International standards for stream ciphers: a progress report, stream ciphers workshop, Brugge, Belgium, pp 121–128
19. Weinstein E, Steele K, Agarwal A, Glass J (April 2004) LOUD: A 1020-node modular microphone array and beamformer for intelligent computing spaces. MIT, Cambridge, MA, MIT/LCS Technical Memo MITLCS-TM-642
20. Fife K, El Gamal A, Wong H-SP (February 2008) A 3 M pixel multi-aperture image sensor with 0.7 μm Pixels in 0.11 μm CMOS. In: IEEE international solid-state circuits conference (ISSCC2008) digest of technical papers, San Francisco, CA, pp 48–49
21. Wilburn et al (July 2005) High performance imaging using large camera arrays. ACM Trans Graph 24(3):765–776
22. Boland R (July 2007) No node left behind. AFCEA Signal Mag 61(11):27–32
23. Brooks RA (March 1986) A robust layered control system for a mobile robot. IEEE J Robot Autom 2(1):14–23
24. Fang W, Wang C, Spaanenburg L (August 2006) In search for a robust digital CNN system. In: Proceedings 10th IEEE workshop on CNNA and their applications, Istanbul, Turkey, pp 328–333
25. Spaanenburg L, Malki S (July 2005) "Artificial life goes 'In-Silico'". In: CIMSA 2005 – IEEE international conference on computational intelligence for measurement systems and applications, Giardini Naxos – Taormina, Sicily, Italy, 20–22, pp 267–272

26. Buldyrev SV, Parshani R, Paul G, Stanley HE, Havlin S (April 2010) Catastrophic cascade of failures in interdependent networks. Nature 464(7291):1025–1028
27. Hennessy JL, Patterson DA (2003) Computer architecture, a quantitative approach, 3rd edn. Morgan Kaufman, San Francisco, CA
28. Hennessy JL, Patterson DA (1996) Computer architecture, a quantitative approach, 2nd edn. Morgan Kaufman, San Francisco, CA, Exercises 5.1 and 5.2
29. Oberheide J, Veeraraghavan K, Cooke E, Flinn J, Jahanian F (June 2008) Virtualized in-cloud security services for mobile devices. In: Proceedings of the 1st workshop on virtualization in mobile computing (MobiVirt'08), Breckenridge, CO, pp 31–35
30. Oberheide J, Cooke E, Jahanian F (July 2008) CloudAV: N-Version antivirus in the network cloud. In: Proceedings of the 17th USENIX security symposium, San Jose, CA
31. Jula H, Tralamazza D, Zamfir C, Candea G (December 2008) Deadlock immunity: enabling systems to defend against deadlocks. In: Proceedings 8th USENIX symposium on operating systems design and implementation (OSDI), San Diego, CA

Chapter 6
Safety Considerations

Safety[1] concerns itself with the detection of (and survival over) internal or external malicious intent. The described level of safety technology is a missing element in many currently applied intelligent sensor networks [1, 2]. Technology is included to access working status of network elements, including self-test, tampering, adversity spoofing, and maliciousness all the way to "Byzantine Generals" network analysis.

In addition, anytime we find anomalous behavior in a sensor network, we need to manage that information. Quite often one anomaly leads into an avalanche of subsequent alarms.

Another safety aspect is that of an ad hoc network. If the injector decides that the network has a particular topology like a ring and forces all sensors to communicate in this fashion as if the chosen topology was physically in place, an outside force might find it hard to know how to talk to the system since, when not calling it topologically right, you instantly give yourself away as not being part of the system. And the topology of the system can change arbitrarily in the eyes of an outsider.

6.1 Adversity Action Detection

Not only does the integrity of the initial network need to be established but also the soundness of any network growth, as well as the detection of any additional tampering or pretending. In principle, what we are after is to be able to completely "trust the edge," where the data delivering sensor nodes are located.

[1] Definition of Safety (Merriam-Webster Online Dictionary): 1: the condition of being safe from undergoing or causing hurt, injury, or loss, 2: a device (as on a weapon or a machine) designed to prevent inadvertent or hazardous operation, 3 a (1): a situation in football in which a member of the offensive team is tackled behind its own goal line that counts two points for the defensive team – compare touchback (2): a member of a defensive backfield in football who occupies the deepest position in order to receive a kick, defend against a forward pass, or stop a ballcarrier b: a billiard shot made with no attempt to score or so as to leave the balls in an unfavorable position for the opponent c: base hit.

L. Spaanenburg, H. Spaanenburg, *Cloud Connectivity and Embedded Sensory Systems*, DOI 10.1007/978-1-4419-7545-4_6,

Recent DARPA[2] information survivability research (Fig. 6.1) focuses on technology that will guarantee that critical information systems continue to function adequately in the face of attack, even when the precise type of attack has not been anticipated.

The High Confidence Systems (HCS) research and development project (as listed in the CICRD BlueBook 1999) focuses on the critical technologies necessary to achieve high levels of availability, reliability, security, protection, and restorability of information services. Systems that employ these technologies will be resistant to component failure and malicious manipulation, and will respond to damage or perceived threat by adaptation or reconfiguration.

Applications requiring HCS technologies include national security, law enforcement, life- and safety-critical systems, personal privacy, and the protection of critical elements of the National Information Infrastructure. Systems for power generation and distribution, banking, telecommunications, medical implants, automated surgical assistants, and transportation also need reliable computing and telecommunication technologies.

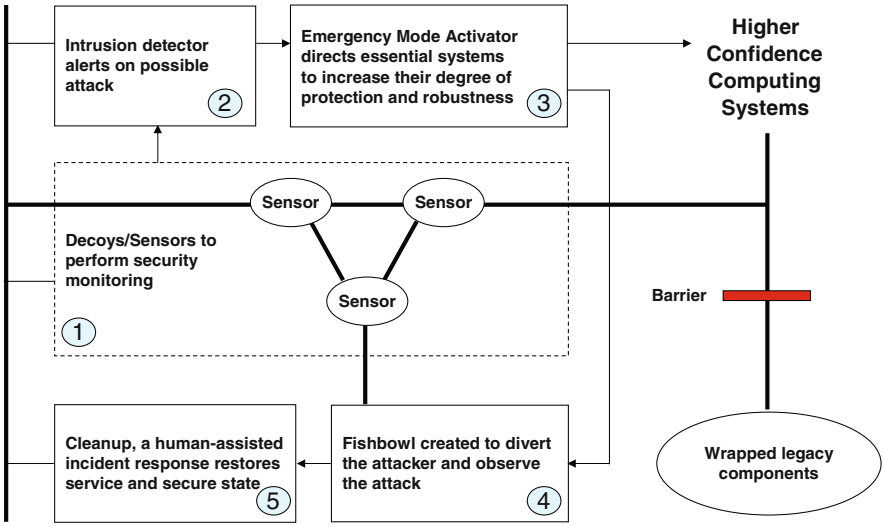

Fig. 6.1 DARPA information survivability research overview[3]

[2]Defense Advanced Research Project Agency.

[3]CICRD BlueBook 1999, High Confidence Systems, Networked Computing for the twenty-first Century.

6.1.1 Jamming

Sensor communications can be jammed during their (wireless) transmissions. Redundancy of communication paths can improve on the probability of successful transmissions. MIMO (Multiple Antennas, Multiple Receivers), frequency hopping, and diversity techniques could also be applied to improve the probability of successful transmission/reception.

In the case of jamming no new (unreliable) data will arrive into the cloud; however, the quality of the data could be an issue.

6.1.2 Tampering

The sensors can be tampered with: their data and program memory can be exchanged, their firmware PROMs can be exchanged, etc. Authentication techniques based on physical properties of the sensor/processing node can be applied to detect tampering. Tampering detection technology based on visual inspection will of course not be applicable.

Data from tampered sensor nodes will need to be removed as a result of analysis to be performed in the cloud. This will especially be needed from life-critical decision-making based on the sensor data.

6.1.3 Invading

New faux (compromised) sensors can be introduced into the network, faking "valuable" information. The new nodes can be in new locations, which may be detected from known location (GPS) data. They can also replace removed nodes, which may be found out by authentication.

Similarly, in this case data from the invading sensor nodes will need to be removed as a result of analysis to be performed in the cloud. Also in this case this will especially be needed from life-critical decision-making based on the sensor data.

6.1.4 Antipathetic Testing for Malfeasance Assessment

In order to determine malicious intent and behavior at the sensor nodes, it may be necessary to run specific self-test-like routines in the nodes. The tests results can be compared against those of a known trustable node. From the cloud's point-of-view, we are looking for out-of-the-ordinary behavior, our focus is not on plain functionality.

In addition, we would look for physical properties that will identify the particular node. A node could, for instance, be authenticated by "fingerprinting" properties of its memory [3]. Next to authentication, we need to know that no other party is directing the operations of the network node. In other words, we need to know that no hackers are manipulating the sensory network [4].

Ultimately, a more pro-active approach would be to have the network nodes contain some form of self-verification. A recent Columbia University proposal [5] describes a processor that contains two engines hardwired to monitor chip communications for anomalies. One of them, TrustNet, alerts when more or fewer instructions are executed than expected. The second, DataWatch, looks for signs that the processor has been maliciously modified.

6.2 Byzantine Generals

The "Byzantine Generals" problem concerns itself with situations in which information is received from more or less reliable sources. The question is whether or not, out of the "noisy" responses, one can still deduct the correct information or decision.

In their seminal paper on this subject, Lamport, Shostak, and Pease [6] have shown how reliable computer systems must handle malfunctioning components that give conflicting information to different parts of the system. They did this by describing a situation concerning a group of generals of the Byzantine army camped with their troops around an enemy city. The battle plan for the attack must be agreed upon by the generals. The generals are at different locations around the embattled city. Communication among the generals takes only place by messenger. However, one or more of the generals may be traitors who will try to confuse the others. The problem is to find an algorithm to ensure that the loyal generals will reach an agreement.

In their paper, Lamport, Shostak, and Pease showed that, using only oral messages, this problem is solvable if and only if more than two-thirds of the generals are loyal; so a single traitor can confound two loyal generals. With unforgeable written messages, a situation in which the messages can be passed on without alteration, the problem is solvable for any number of generals and possible traitors.

6.2.1 Faults

A fault is like a transmission from a "false" general. For the case of the sensory network this can be a node that failed, is tampered with, or has been invasively added to the network. In a homogeneous network the Byzantine generals' problem solution will help decide in the case of conflicting data.

6.2.2 Signed Messages

The use of signed messages can improve the reliability of transmissions. Similarly for the case of the sensory network, any form of attribution or stamping will simplify the decision process.

6.2.3 Missing Communications

A missing communication is neither a false nor a true statement. In the case of the sensory network this represents not operational communication paths. Redundancy of transmission will be advantageous in this case.

6.3 Emergent Behavior Detection

It is extremely important to access the integrity [7] of the collected data, in addition to collecting the data itself. In addition, we need to manage the occurrence of detections from the sensors, the anomalies, and their respective selection criteria. In any detection and alarm management system, there also is an element of feedback (threshold level) in fine-tuning the number of detections that can be handled in a reasonable time period.

6.3.1 Information Reporting and Management

Abundant data, easily accessible from the sensor networks, lead to situations where the true data consist side by side with "rumors, lies and errors" [8]. Before reporting on findings, a careful weeding out of the true data (the truth) needs to take place.

Corrective models for set-point predictions of an industrial processing system show time-varying global disturbances (not observable by the mathematical models) that are present and must be suppressed. Without intent, emergent behavior appears in such distributed systems due to non-deterministic computing and is spread indirectly by feedback control algorithms. Consequently, a gap between the desired behavior as physically understood and the actual behavior comes to the surface.

In this section we assume that the measurement data are already cleaned from false indications. Only actual faults exist and will eventually become observable. We see these as the result of parameter variations over dimensions not included in the model. Such extra dimensions cover a number of related models, of which the fault models are discrete samples. This brings the need for two essentially antagonistic views on the same reality (Fig. 6.2):

(a) the *process model* reflecting the proper operation of the network, and
(b) the *detection model* reflecting the abnormalities.

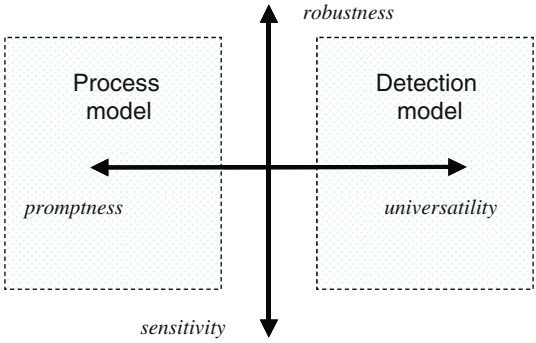

Fig. 6.2 The parameter space for novelty detection and isolation (NDI)

The former relies on the extraction of physical coefficients from model parameters [9]. The aim is to find the Degrees of Freedom (DoF) in the model that describe the process in full complexity while minimizing the least square error (*risk minimization*). Domain experts usually add as constraint that the model can be humanly interpreted. The relevance of a known disturbance can be built-in by ensuring that the error on desired behavior is a measure of the "faultiness" of the observed behavior.

To handle unknown faults, the generic sensitivity (universality) needs to be improved by increasing the DoFs in the model. However, increasing the DoFs conflicts with risk minimization. Simplifications may be called for to reach the desired separation of concerns. The justification for simplifications comes from properties on the independence of local processes, allowing for modular models and linear behavior. Indeed, if a system composed of multiple processes is in a stable equilibrium in its state space, linearization is allowed and will break the dependence between subprocesses. Unfortunately, these properties are insufficient as soon as the system drifts from the desired stable equilibrium (Fig. 6.3).

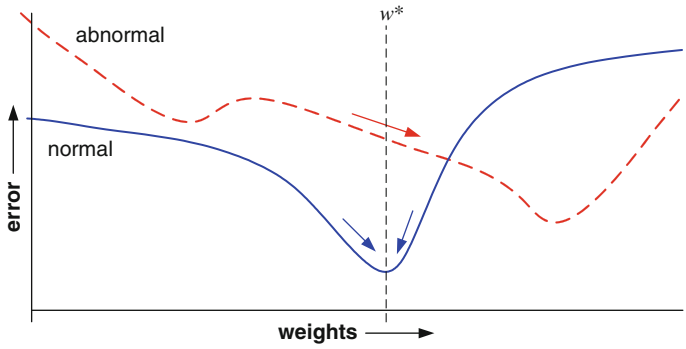

Fig. 6.3 Set-point becomes non-optimal when abnormality occurs

The alternative proposed here is to break the dimensional dependence between model complexity and model risk to improve both overall sensitivity as well as confidence in the relevance of detected faults. Models used for detection must therefore extend beyond the desired process behavior, intrinsically augmenting the model by extending the capabilities of the model beyond the DoF needs for describing the normal systems behavior. The underlying philosophy is the multi-version technique (see also Chapter 5), as popularized with great success in other technological fields. We argue that splitting the demands over two models seriously reduces (if not eliminates) the bias-variance problem, which in the FDI era was an undividable part of abnormality detection. Such models need to match two opposing worlds by creating two different but dimensionally overlapping views on the same reality.

Initial experiments in early detection of abnormal emergent behavior have been performed on several levels of the control hierarchy (Fig. 6.4). On the tactical level, van Veelen has done an extensive analysis of hot-strip mill data, while on the strategic level he has looked at a telephony network. Later, the applicability in large-area telescopes has been treated [10]. Such experiments show the flexibility of neural engineering, but are limited to detection as reaction time is of the essence.

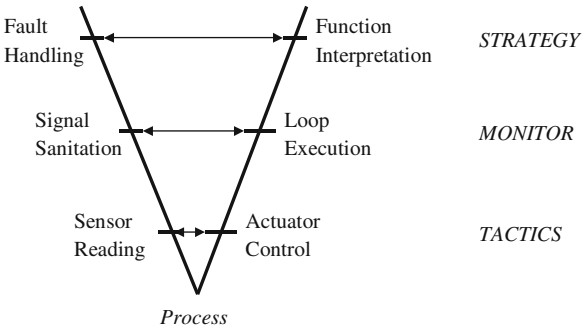

Fig. 6.4 V-chart for separated process and detection modeling, illustrating the dedicated but hierarchically layered focus over all control level

The learning process creates a sense of history, but taking care of a nonlinear dependence and without the need to reduce the number of observed parameters. This is of interest where the process interacts with its environment. There will be different structures in different views. Furthermore, the neural network evades the need to perform an m-dimensional mapping and matching. Instead, we have a simple mechanism that can be easily added to an existing software object as a guard mechanism.

We have to distinguish between two types of guards: off-line and in-line. The in-line guards are directly related to an existing software object. The off-line guards are required to handle faults that creep into the system from a nonfunctional part of the environment. This is depicted in Fig. 6.4 as V-chart for the Power Grid control case [11]. This picture has a striking resemblance to model separation shown in Fig. 6.2.

6.3.2 Anomaly Detection

Anomaly in a dynamical system [12] is defined as a deviation from its nominal behavior, and can be associated with parametric or nonparametric changes that may gradually evolve in the system.

Early detection of anomalies in complex dynamical systems is essential not only for prevention of cascading catastrophic failures but also for enhancement of performance and availability. For anomaly detection, it may be necessary to rely on time-series data generated from sensors and other sources of information, because accurate and computationally tractable modeling of complex system dynamics is often infeasible solely based on the fundamental principles of physics.

It is interesting to note here how anomalous behavior is often ignored as just being the result of incomplete analysis. Leon Chua has investigated the use of higher-order primitives in the modeling of dynamic nonlinear systems. From network theory, the use of the resistor, inductor, and capacitor is known, but Chua spotted the need for a fourth primitive: the Memristor [13]. This observation went largely unnoticed until HP got reminded, when trying to properly model a piece of nano-technology, which confusingly became also named Memristor. Initially, they did not manage to complete the model as apparently many others suffered in the past; but using Chua's technique their problems were over [14].

Emergent behavior [15] has become a major threat to modern automation. It was not so visible in the early days of automation, when a single machine was modeled and controlled. The model was developed to be robust enough for providing reliable results despite noisy, irreproducible, and incomplete measurement data. Faults are modeled separately, and Fault Diagnosis and Isolation (FDI) is based on classifying machine behavior [16] (Fig. 6.5) as normal or having a known and characterized fault [17, 18].

Classical abnormality detection starts from the single fault assumption. Faults are observed at the output, but can hardly be distinguished from other faults on the same path to the system output. When a fault is observed and leads to an operator alarm, the remedy is not always clear. The single fault may actually lead to a flood of alarms messages that annoys the operator, but does not always create disasters.

Fault-observability should therefore address not only the **A**bility to see a fault but also the **A**menability to rank its urgency and the **A**gility to react accordingly. Without such AAA measures, alarms tend to be simply ignored in practice, and it is necessary to regularly perform a health check to bring the alarm flood back in proportion.

But the realm of automation has grown into networks. Abnormalities do not only give errors but can also travel over the network to reach other machines. Most of these networks grow without an overall architectural vision, but rather by means of a local preferential attachment. This lack of predetermined structure does not mean that there is no structure at all. On the contrary, it has been noted that the seemingly chaotic self-organization leads to a clear structure with distinct properties, though different from designed networks [19], and display therefore emergent behavior.

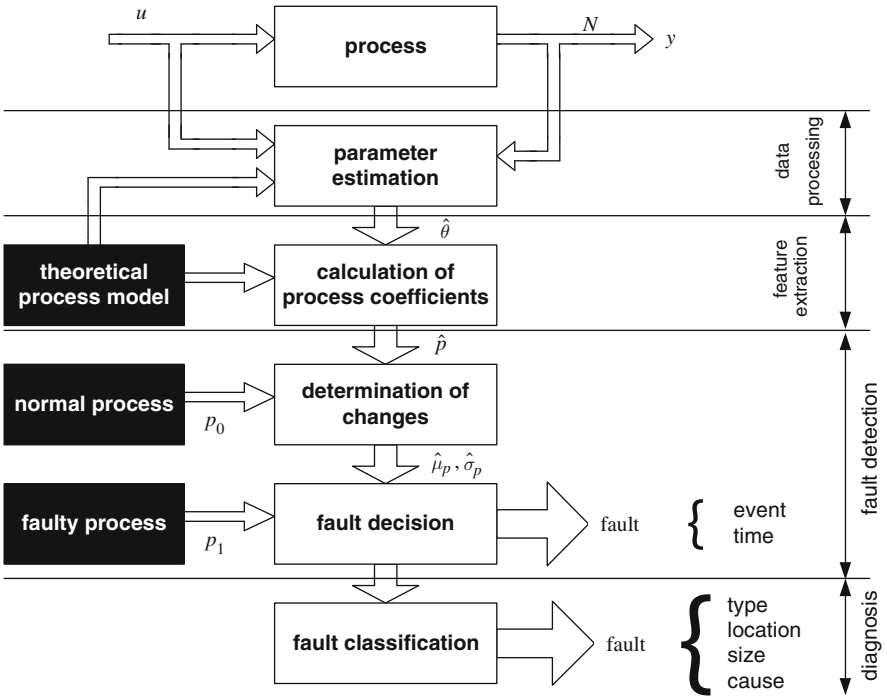

Fig. 6.5 Classical FDI development phases

A recently reported study [20, 21] has noted that networks that would be fully resilient on their own could become fragile and prone to catastrophic failure when connected to other similarly well-protected networks, a finding with troubling implications for tightly linked modern infrastructures [22].

The global structure of networks with inherent communication delays has also become notorious for abnormal behavior. For instance, default distribution of a programming error as part of the maintenance procedure caused in 1992 the New Jersey blackout. Only a couple of years later, the Allston-Keeler (August 10, 1996) and the Galaxy-IV (May 19, 1998) disasters gave rise to a concerted research activity on Self-Healing Networks [11]. A series of three disasters on the Electricity Grid in autumn 2003 (in respectively America, Sweden, and Italy) suggests that little progress has been made. And these are only the tip of the iceberg [23].

Electronic networks (and especially wireless ones) are not exempt from emergent behavior. Even when the design has been perfect, ageing and wear can develop in unknown ways. Moreover, embedded systems are becoming more of the reactive nature that makes abnormal behavior likely to emerge. Where autonomous nodes work together, they tend to pass not only the good but also the bad news. Consequently, special measures are required to make them "immune" for sick neighbors [24]. Of course, the degree of immunity (or self-healing) must be dependent

on the fatality of the sickness. The critical point is clearly how to differentiate between the need to globally adapt and the demand to locally block a fault effect from spreading.

In addition, redundancy allows overruling a malfunctioning part to decrease the fault sensitivity. It is usually eliminated to bring down initially product cost. However, an optimal design should exploit redundancy to the fullest, because it reduces maintenance costs over the lifetime of the system [25]. Most effective is partially redundancy, though this will inevitably raise the need to contain emergent behavior.

6.3.3 Nuisance Alarms

In industries the production is supervised by control system that monitors analog and digital signals from the plant and checks on limits for the values that are allowed for a signal – alarm limits. Whenever a signal exceeds its limits, the alarm system activates.

An alarm should only be activated when the process does not behave as expected, that is before anything critical has happened, but only when a critical event is about to happen. The alarm has to be unmistakable and there should be no more than one alarm. A well-designed control system with well-tuned alarm limits gives the operators the possibility to efficiently monitor and control the process from a control room. It is however impossible (in practice) to set the alarm limits without introducing false alarms [26]. But also for other reasons like outliers, disturbances, novelties, or abnormalities [10], the attention of the operator may be attracted without any obvious reason. This is not only a nuisance, but also potentially dangerous as it may distract the attention of the operator from real problems and slow down his reaction. An even more dangerous situation may occur if the operators believe that an alarm will precede every fault situation, but too widely set limits has made the alarm "silent." It is a common problem that one single fault may cause many others. Alarms are then triggered by the faults causing an alarm flood, but it cannot be trusted for its time order. The primary alarm may not come first; in fact, it may not come at all because its communication is blocked or because it is silent. This makes it even harder for the operator to come up with the right corrective procedure. Model-based reasoning then seems a viable remedy [27]. But except from the fact that the model has to be accurate, it is also depending on the correctness of the control system and the alarm limits.

Ahnlund and Bergquist found that alarm systems [28] are badly designed. Some alarms are unnecessary for operating the plant, while some important alarms are missing. Or the alarm system is unable to handle different operating states. For example, an alarm system works well in normal operation, but fails in case of a major accident. The alarm system presents too much (inadequate) information. The operator is flooded and thus unable to separate important alarms from minor events. The alarm system is badly tuned. Alarm triggers due to too tight limits, controller-caused oscillations, unfiltered noise disturbances, outliers, transients, etc.

The use of set-points is popular to bring control applications in a desired state. Unfortunately, in a typical production line all the individual machines receive a locally optimized setting. Even when that is not the case upon system installation, it may easily creep in through a later parts replacement. Misalignment of the set-points will then cause an "alarm flood," the notorious music in many fabrication halls that signals potential hazards but more irritates than warns people.

As system parts may vary, each set-point needs to be adapted to the actual circumstances. To reduce such sensitivity, one usually allows set-points to operate in intervals (Fig. 6.6). Alarms will then be triggered when the operation steps outside the allowed interval. This eases the control procedure, but does not seem to reduce the alarm flood. On the other hand, the alarm directionality helps to logically deduce better settings from the alarm logs.

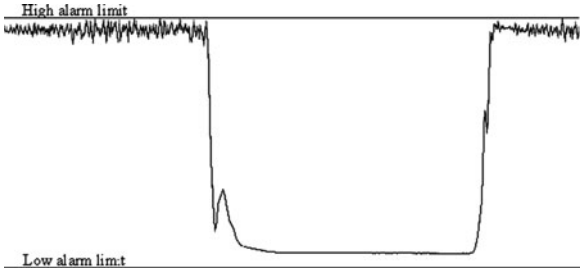

Fig. 6.6 The high alarm limit is set too tight and is therefore sensitive to noise. The low alarm limit on the other hand is set too wide, and does not activate even though a plant trip occurs

General methods for reduction of nuisance alarms, i.e., alarm cleanup were first addressed as "alarm sanitation" [29]. Very little is written about the concept except in J. Ahnlund and T. Bergquist's 2004 Lic. Thesis at Lund University [30]. The problem with nuisance alarms has however burdened the industry for decades, and some reduction techniques have emerged from engineers dealing with these problems. The Engineering Equipment and Materials Users Association (EEMUA) have presented a comprehensive guide for design, management and procurement of alarms systems [31]. Less technical detailed guidance can be found in several publications, for instance, by Veland [32] et al.

A conventional alarm cleanup is rather uncommon, since it is carried out by hand and therefore both time-consuming and expensive. J. Ahnlund and T. Bergquist's alarm cleanup SILENT CONTROL^{TM} uses a general computerized tool and is a first step to an enhanced overall alarm situation. They take advantage of the control systems built-in functions, as shown in Fig. 6.7. These functions are only applied on the alarm system and do not interfere with the operating of the process. It means to keep the methods generally applicable on any process and as user-friendly as possible. This requires built-in intelligence that combines new research results and known algorithms in various technical and mathematical areas.

SILENT CONTROL^{TM} is not integrated into a control system. Instead, it has been applied as an off-line approach that examines the process in the past and then tries to predict the behavior in the future. It is based on the assumption that the process signals characteristics are stable in normal operation. In most (if not all)

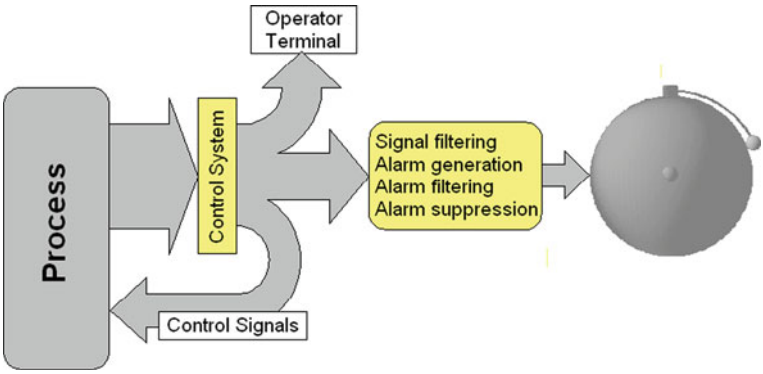

Fig. 6.7 The Control System receives Signals from the plant, where some signals are used for the controlling of the plant or displayed at the operator terminals. A subset of the signals is used for the alarm system. Whenever a signal exceeds its alarm limit, the alarm system activates. The applied signal processing methods consist of signal filtering, alarm filtering, and alarm suppression, and do not interfere with the system's ability to control the process

industrial processes that is true. All improvements suggested from the tool have to be evaluated and accepted by operational personnel with process understanding before they are implemented in the control system. Over the years, the toolbox has successfully provided maintenance services to the energy production industry in Northern Europe.

6.3.4 Redundancy by Intelligent Agents

In the Complex Interactive Networks/Systems Initiative (CIN/SI) the remedy is sought in Self-Healing Networks: an agent-intensive concept where agents sense abnormalities to reconfigure the system by local measures, essentially breaking the occurrence of cascading faults. On 10 August 1996 a fault in Oregon led to an excess power surge that started a chain reaction across the continent. Though the problem was already a major one, cascading gave it an unwanted extra dimension.

The situation becomes even more disastrous when taking the current tendency toward heterogeneous networks into account. Signaling along the power grid uses the ICT network including satellite communication for large distances. Furthermore, the road infrastructure is involved with all its usual congestions. Overall, the problem takes on a dimensionality that defies the conventional mathematical analysis as the basis for control. With an eye on the catastrophic accidents that have already passed, there is a clear need for alternatives.

Self-Healing Networks are based on autonomous, intelligent control, as there is no succinct, closed-form specification of unknown faults. They are based on a multi-agent architecture consisting of a reaction layer for immediate response, a coordination layer to discriminate between urgencies of request, and a deliberation layer to validate the feasibility of current plans and commands. Agents can be cognitive (i.e., rational or even intentional) or reactive with a hardwired stimulus/response function.

The CIN/SI work began in spring 1999. The 5-year, $30 million effort combines 28 universities under the Government-Industry Collaborative University Research program. But even before that time, the name has been used. In 1993, AT&T announced the self-healing wireless network. In 1998, SUN acquired RedCape to use the Policy Framework for self-healing software. Also in 1998, HP released a self-healing version of its OpenView Network Node Manager. And in January 2001, Concord Communications announced self-healing technologies for the home.

But most of the current effort is spent on ATM networks. Here, self-healing is considered and implemented already on the basic network layers. SONET and SDH provide such properties on the physical layer. Similar properties on layer 2 and 3 of the ISO/OSI model allow for reconfiguration. But these measures handle only part of the problem: to detect, diagnose, and isolate the unknown. And so, still more work is required to come to the much required level of security: the guaranteed availability of resources during unknown operational conditions.

This makes Intelligent Web-based applications the preferred testing ground for Self-Healing networks. Reconfiguration comes as a natural opportunity; complexity is scaled on the network level; and the coordination is loose enough to provide for self-test, diagnosis, and repair in a way that does not require each house owner to become a fully trained technician.

The step from simple neural networks to communicating intelligent agents or hierarchical neural nets is small. Such agents are smaller trees that may mutually trigger in a distributed network. Whether by agents or by experts, the goal remains an efficient reasoning and the decisions will be based on data from the "work floor." Dedicated tools in a variety of technologies are at work, because the steady stream of data will flood the experts (just like humans [33]) at this low level of detail. From Fig. 6.8 we find that our subject is an important ingredient in the quest for the self-healing network [11].

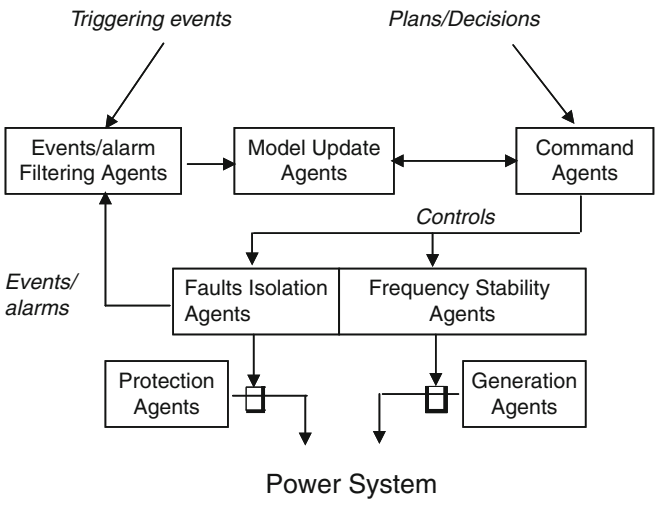

Fig. 6.8 Reactive and coordination layer in self-healing multi-agent smart grid control

Hierarchical neural networks can be applied to provide structure to a distributed sensory network such as a house or a production line. We envision that the layered control should not be based on merely "divide-and-conquer," but on a holistic view that goes down in abstraction while moving down the hierarchy. As the upper layers cover the same control problem as the lower ones, they contain an amount of redundancy that naturally provides for a sanity check. As a consequence, alarm sanitation becomes a given, and does not require special measures. As fault handling requires different models than control itself, mixed architectures as depicted in Fig. 6.8 should be avoided. Instead, we find that every level contains separate parts for control and detection. In the SILENT CONTROL^{TM} package, a number of signals have been determined, of which the detection can indicate the need for fault detection. Figure 6.9 gives an overview. Having these signal detectors in-line allows for classifying alarms according to their potential impact. This then gives sufficient information in the hierarchically distributed neural net to either (a) handle the alarm locally while logging the event centrally at a suitable time or (b) pass the alarm directly to the operator.

6.4 Identity Assurance

One has always sought a trusted person to pass sensitive information along. Because security is better than trust, the information was made unreadable (or even invisible) to outsiders. This alleviates the requirements on the carrier, but does not remove them altogether. If not all involved persons can be trusted or their identity can be assumed by a third party, the transfer of information may be tampered with. Even seemingly simple situations, as in the Byzantine Generals problem, can be problematic to prove secure.

In modern communication technology, the role of the persons is taken over by key codes. Like in the case of a bank vault, the information can only be accessed when all the keys are correct. The fraud sensitivity then moves from information theft to identity theft. Once the key is stolen, a third party can pose as the person that is identified by the key. Daily news is full of stories about fraud based on PIN codes and smart cards. This is like putting the horse behind the wagon, and biometrics has been proposed as the obvious improvement as it relates directly to the person instead of via some randomly assigned number.

6.4.1 Biometric Authentication

The derived key value should be too big to guess, while the construction is too random to allow improved guessing by structural observations. This is the case with living material that grows and changes through underlying, but not fully understood, processes. The same problem makes weather so hard to predict despite the use of supercomputers. The observation is on a process that is the outcome of a number

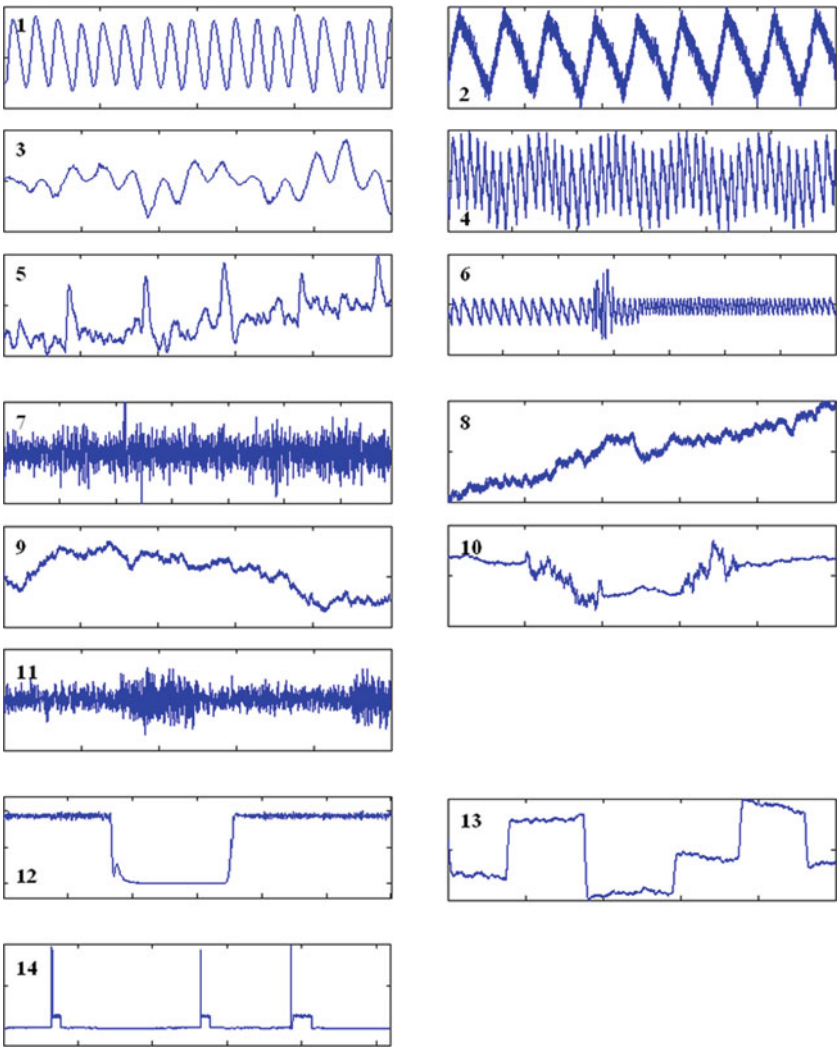

Fig. 6.9 Examples of SILENT CONTROL^{TM} Signal Classes. Signals 1–6 are periodic, signals 7–11 are slowly varying, 12 and 13 are multiple steady-state signals and signal 14 contains outliers

of non-observable processes. This makes for a large variety. If moreover the observation can be confined to a living process, defrauding becomes extremely hard to achieve, if ever.

For these reasons, biometrics technologies promise to provide highly secure authentication systems. They are usually classified according to the ease-of-use, cost, accuracy, and perceived intrusiveness. The widest known biometric disciplines are fingerprint, facial, and vein recognition. But there are more. Table 6.1 lists the principle methods and a number of body parts on which they have been applied.

Table 6.1 Biometric Technologies (Adapted from RAND Report, June 2000)

	Object	Robustness	Accuracy	Intrusion
Geometry	Hand	Moderate	Low	Touching
	Finger	Moderate	Low	Touching
Structure	Finger Tip	Moderate	High	Touching
	Iris	High	High	Near
	Face	Moderate	Moderate	Near
Vascular	Palm	High	High	Casual
	Finger	High	High	Casual
	Retina	High	High	close
Dynamics	Speech	Moderate	Low	Remote
	Writing	Low	Moderate	touching
	Key-stroke	Low	Low	Casual

Early attempts to bring something practical to society have been on measuring the outline of a hand (in operation at Ben Gurion airport since 2003) or a skull. From there on, the move has been toward more complex features, or to a higher dimensionality in time or space. In space, we moved from outline to 2-D textures and subsurface vascular patterns. In time, detection schemes based on speech, writing, typing, and walking have been developed.

Most of these methods are susceptible to fraud as they lack the link to Nature. This applies clearly to still observations where the image generation does not rely on living organisms (geometry and texture). For instance, fingerprint provides a good balance between the four characteristics listed in the previous paragraph, making it the most common technology nowadays. But such systems are demonstrated to be very sensitive to fraud by reproduction, while technical considerations such as sensor quality and temperature decrease the usability. One may conjecture that this argument also applies to dynamic observations. For instance, the dynamics of speech are hard to copy, but not impossible for single words or short sentences. On the other hand, vein patterns of either a palm/finger of the hand or the iris of an eye offer stable, unique, and repeatable biometrics features.

The oldest vascular systems are based on eye veins. They come in two categories. Firstly, retina-scanning uses the capillary patterns in the retina that absorbs light and therefore can be made visible by proper illumination. On the inside of the eye cavity, the incoming light falls on a sandwich of three layers, consisting of the capillaries, the blood vessels, and the supporting fibers. The scanner should be close to the lens to see the retina as the lens opening is small, and perfect alignment without eye movement is necessary to have a reproducible measure. The use for authentication was already suggested in the 1930s [34], but it took until 1984 to become commercially exploited by EyeDentify. The technique is intrusive and possibly damaging for the eye. Furthermore, the stability of the pattern is not guaranteed as all kind of diseases may change it.

The competitor is the Iris scanner. It uses the texture of the iris, the anatomical structure covering the pupil of the eye for controlling the diameter and size of the

opening and therefore the amount of light. In its emphasis on structural features, iris scanning is similar to finger printing. The iris patterns can be observed from a somewhat large distance and the technology is therefore less intrusive. This has made for a clear competition with retinal scanning, starting from the introduction of the first IrisAccess system in 1994, originally developed by John Daugman of Cambridge University [35]. Today, iris scanners are still based on the same algorithmic concepts. Unfortunately, such a system can be easily fooled for glasses and lenses worn by the user.

Hand vein patterns have been considered for identification purposes already in 1987 by Joe Rice in the United Kingdom, but his employer saw no need for further development [36]. An early device using the palm of the hand has been introduced to the market in 1997 by Advanced Biometrics. Later on, other places to find the veins (the back of the hand and the ventral/dorsal side of a finger) have been exploited. The main three players in the hand vein recognition markets are similarly divided. All three are located in Japan and initially developed the devices by the local market.

One may wonder: if vascular methods are so advantageous, what kept it so long? The answer is cost and therefore technology. It extracts vein locations from light absorption patterns due to oxygen-depleted hemoglobin observable in the near infrared spectrum. It is only on the breakthrough of LED technology that this has become a practical alternative. The principle mechanism guarantees the elimination of an easy fraud by copying, while the structure is experimentally proven to be unique in a test of 40,000 genuine versus 50,000,000 imposter images [37].

The first generation of hand vein devices has targeted the demonstration of principle soundness. The vein images are obtained using two near-infrared LED arrays, a CCD camera, and a video capture card. An early extraction algorithm [38] uses fixed-point numbers to limit the processing time, and patterns are recognized with 99.45% success. Images are cleaned and compared within 100 ms per person. Nowadays, devices show persistent results with False Reject Ratio FRR < 0.01% and False Accept Ratio FAR < 0.00002% [39].

The detection of veins in the palm of the hand has been improved by Fujitsu to support contactless applications [37]. Similar is the use of the back of the hand [40]. The primary advantage of this PalmSecure device is that it allows “no-touch” authentication, which is important for access control in public areas. This keeps the scanner clean while the user needs no special preparation (like taking the gloves off). Furthermore, environmental light plays no role in the reading. Hitachi uses the finger veins in the VeinID that has been successfully employed by major financial institutes in Japan [41]. The Sony Mofiria system looks from the side to the finger, having typical mobile applications in mind.

The early hand vein recognition devices have been successfully applied to control public access, for instance, airports and banking terminals. Therefore, the size, price, and user-friendliness were of moderate importance. In a sense they were computer peripherals. The newer devices stress the need for price and size reduction to integrate them into consumer systems, such as mobile telephones and laptops.

As their main purpose is to check on the authorization of a specific user, they serve as electronic keys. Such products have been demonstrated at the CEBIT in 2009.

A typical issue with biometric devices is the need to enroll. The lower the robustness of the sensor, the more samples have to be taken. Furthermore, the complexity of the algorithm takes its toll on the actual authentication time. For an iris scan the enrollment takes 45 s, and another 15 s for every authentication event. A vein scan requires more time as it has to handle a volume rather than a surface. In public places such as stadiums the need is for casual, real-time authentication of large amount of people. In order to avoid excessive waiting queues, the execution time per person must be further brought down by dedicated hardware (Fig. 6.10), like the CNN-based Biometric Authentication System (CBAS) universal biometric engine [42].

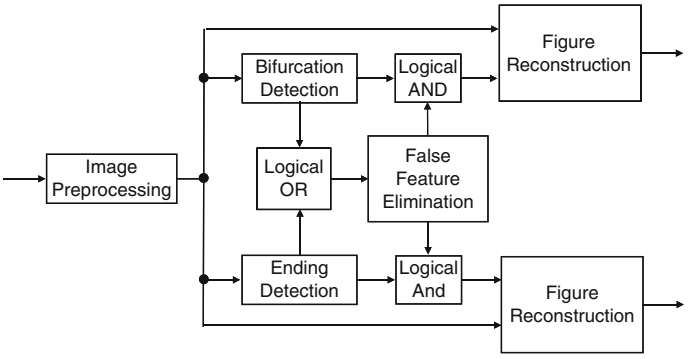

Fig. 6.10 CBAS algorithmic structure

6.4.2 BASE for Multi-modal Sensing

Homeland Security targets the insurance of safety in public places,[4] which is largely attempted by access control, for instance, at stadium gates or country borders. It addresses both the authorization of people and the detection of potentially dangerous material. In a previous section we have discussed that vascular detection principles are very effective to identify people, even when wearing a burka and having grown different facial hair. But Homeland Security requires real-time performance for such a technology to be practical (Fig. 6.11).

There are two distinct phases in the application of biometric measures. During the enrollment, the person is measured for the feature. Distance and orientation of the measured body part are parameters that influence the metric. Therefore, application should be repeated several times to either get a correct measurement according to some measure of goodness or to reach a variety that covers all practical situations.

[4]Biometric Authentication Systems Enterprise.

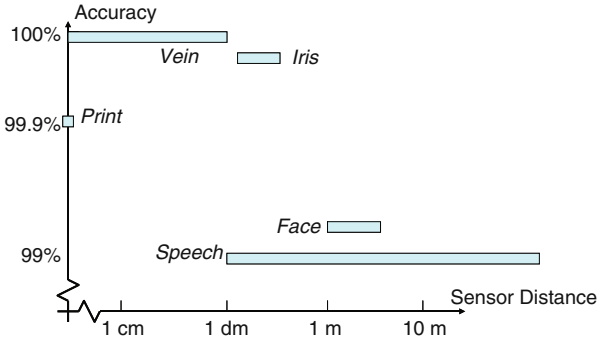

Fig. 6.11 Tradeoff between biometric methods

The goal is then to achieve a data set that allows a high-probability identification upon application. A typical example is in iris recognition, where several measurements at different distances are needed to ensure a high identification rate, though in theory the recognition could be 100% correct.

To eliminate such practical problems, it has been suggested to use multiple metrics. A multi-modal classifier uses the effect that different classifiers have different weakness for operating on the same input. Simple voting between multiple classifiers will then bring out the best result according to the democratic majority. For biometric authentication, the multi-modal principle seems at first little applicable. Suppose a sensor operates on hand shape and hand print. Both operate on different properties of the hand and can therefore compensate each other's weaknesses. Unfortunately, the sensor can be easily misled by artificial input.

As a second example, when mixing pin card with hand vein technology, the problem is reduced to the mere identification whether the person with the access rights according to the pin card is actually the person he/she claims to be. Consequently, only a limited amount of samples have to be checked and no extensive database search is needed. The weak point of the system is the pin card. Once the code on the pin card is hacked, it becomes possible to transfer the access rights to the person that offers his hand. In other words, authentication based on the hand veins is still required and the addition of the pin card gives no additional security.

The variety in existing vascular systems is largely based on proposals to limit the freedom in applying the body part. In this respect, placing the hand flat on the sensor is better than being contact-free. Sony has suggested a fixture for fingers such that the rotational freedom is eliminated. All such suggestions for a dedicated sensing fixture loose out on the need for regular cleaning while still no 100% repeatability is achieved. No fixture that fits all can be achieved considering the difference in size of hands and figures between persons.

An alternative is to algorithmically transform the vein image by post-processing into a known shape. Mathematically, this can ensure that both the enrolled and the called image are taken under comparable conditions. This is by itself not surprising. The same principle has been used to compensate the shortcomings of conventional motor mechanics. Electromechanics has made the car vastly more efficient, an early

example of an embedded system. Currently, such car electronics represent more than 30% of the component costs. Such replacements, additions, and innovations cause a steady doubling of digital system designs per 5 years.

Compensation of diaphragm and rotation are by themselves two-dimensional problems. But even then a measure of quality is needed. For instance, it has been suggested to check on the location of the knuckles to see whether the hand has been observed closely enough [43]. The knuckles can be used to give orientation to the observed image.

Such considerations are of lesser importance in the health industry. There the relevance of authentication is limited to fraud detection, such as in billing and in access to an Electronic Health Dossier. Nevertheless, the measurement principle was already in use for other, more medical purposes. A typical example is in direct aid of burn wounds, where it is important to find how much skin has been burnt away. Instead of the conventional laser scan, it is also possible to use reflected light. The blood stream causes a frequency shift based on the Doppler effects, which can be measured by correlating the intensity of the original 1 kHz and the reflected light.

When the veins are close to the surface, such effects are easily measured. Popular locations are ear and tongue, but also fingertip and toe are used. As the blood stream carries the heart beat, measuring light intensity shift is also possible to detect cardiovascular diseases. Using light of different wavelengths, the oxygen saturation in the blood and the mechanical state of the blood vessels can be established. Such techniques are better known in medicine as "pulsoxymetry."

A recent discussion is on the use of mere infra-red absorption spectroscopy for diverse molecules such as alcohol. Again light of different wavelengths is used, because the absorption spectrum is unique for each molecule of interest. The computational process is complex because the original light is of a low energy. Therefore, the reflection pattern is rather noisy and must be carefully filtered. Such software is responsible for the relatively high price (>1000 Euro) and the slow response (>1 min).

6.5 Safety in Transactions

Safety is a major concern in society. Valuables are precious and it always hurts if they are stolen or when you are robbed. Older people sense this concern by a feeling of unsafety. As a steadily larger portion of the population falls in the senior citizen category, the desire for safety will become more commonly felt.

Cash money is a widespread valuable. Though a number of cashless payment systems have been introduced, it is still widely used as the bank costs for using cards or wireless payments are substantial. Therefore, many small shops try to cut costs through the use of cash money for the many small payments, and are therefore a constant target for robberies. Furthermore, IT-payments [45] are based on a

complicated security mechanism that is hard to manage for the older person and still does not achieve full protection.

6.5.1 e-Traveling Card

Electronic information has the advantage of providing access to live updates compared to hard paper copies. This product-idea aims to make bus and train tickets more dynamic by adding an e-paper screen (see also Section 7.2.2) to both the pre-paid-monthly-cards and the long-distance tickets. The idea is to implement the device such that the transport company will not have to change all the existing slot machines. Introducing some specialized machines in major-stations and sub-stations will enable the end-user to buy or recharge its pre-paid-monthly-card, i.e., the slot machines in the buses and minor train stations will remain the same.

The Pre-Paid-card will still be the size of a credit card with a magnetic strip, but it will have an e-paper screen on one side. The card will have internal logic that supports RF-Signal, an EEPROM, and one flat button (Fig. 6.12).

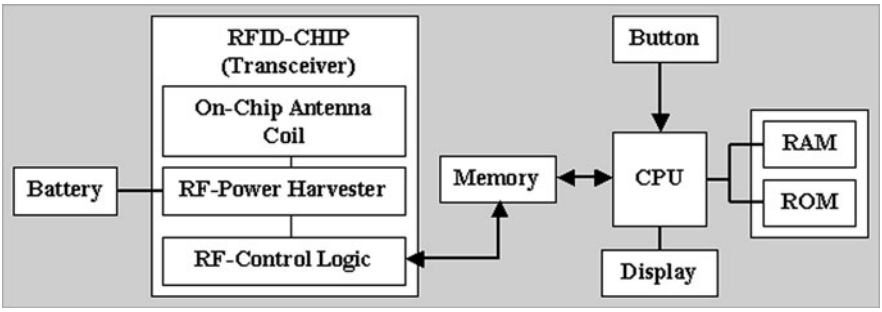

Fig. 6.12 The block diagram of the pre-paid local traffic card

6.5.1.1 Local-Traffic-Card

For the local-traffic-cards, the EEPROM will store at least the timetable for three routes of the user's own choice made at the time of purchase. The e-time table for the selected routes will be updated automatically through RF signals if the transport company updates its database. Of course, the user has the option of changing the selected routes by contacting the customer service in person. Pressing the flat button outside the RF range will show the end-user the static-time-table. However, for a user within the RF range, pressing the flat button once will show the static-time-table, pressing it a second time will show the dynamic-time-table, and pressing it a third time will show other information like sales promotion. The dynamic timetable refers to the latest departure times and other information regarding bus/train, e.g., bus/train is delayed or full.

The e-screen and the RF signal can be used to pass useful information such as weather forecast, latest sales promotion, bus location, space available, community information like the neighborhood road will be closed for repairs.

Tangible Benefits of the Local-Traffic Card

- For the transport company:
  - The same size as a regular card so the transport company does not need to change their existing slot machines. Introducing a few machines in the sub-central stations is enough to cover the needs of the new e-card, i.e., slot machines in the buses cum trains will be unchanged.
  - The transport company benefits from the new system by using the e-paper screen for sales promotion through RF.
  - The implemented timetable on the card enables the transport company to achieve a good customer satisfaction.
- For the end user:
  - The end user will benefit from both the static and the dynamic timetable that is implemented in the e-card. This reflects on the slogan "You won't miss your bus/train."
  - The commuter is always aware of the latest sales promotion and community information.
  - The end user will benefit from the hourly weather forecast.

Functionality

- Dynamic information from the nearest bus stop, i.e., weather forecast, bus status information (delay/on time and empty/full), community information, etc.
- If the cardholder presses the button once, then the CPU will pass a control signal that will display the static timetable, which is stored in a reserved partition of the memory.
- If the cardholder presses the button a second time in the RF range[5], then the CPU will activate the RFID Chip to get the dynamic information from the nearest bus stop, i.e., weather forecast, bus status information (delay/on-time and empty/full), community information, etc..
- Pressing the button the third time will enable sales promotion information.

The module in Fig. 6.13 depicts the manager payment system for the pre-paid cards. The upper part of the diagram shows the interaction between the contact-less information transfer and the pre-paid card. The lower part of the diagram shows the old magnetic strip system, which is kept here for compatibility with the present slot machines in the buses and trains.

[5]Note: If the user is not in the RF range, only a static timetable will be displayed.

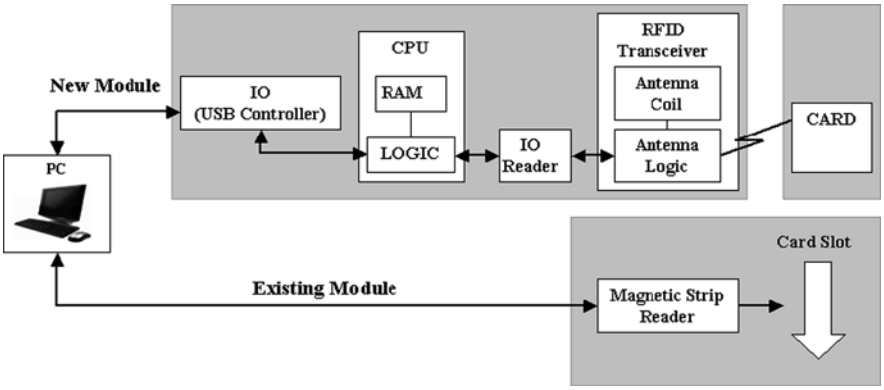

Fig. 6.13 Block diagram of the manager payment module

e-Traveling Card-Issuing Mechanism Is as Follows

- For a user who wants to buy a new pre-paid traveling card, the service manager will issue the card via existing payment method, which is represented by "Existing Module" arrow in Fig. 6.13.
- The officer will use the contact-less module (New Module) to transfer a copy of a static timetable for three routes of the user's desire from the PC to the card.
- The information will be written in the card from the PC to RF reader/writer via USB cable, and then from RF reader/writer the information flows to the RF chip of the card.[6]

Working

- The Central Station broadcasts all the dynamic information, which is then received by all the nearest bus stops. The bus stops act like booster stations they amplify and retransmit the signals to neighboring bus stops.
- The card holders in the vicinity of RF signal will pick the dynamic information depending in which mode a card is, i.e., if a card holder has pressed the card button two or three times.

The complete illustration showing the interaction between the cards, the buses, the bus stops, and the central server can be seen in Fig. 6.14.

[6]Note: Further enhancement can be done by writing the payment information in both the card's memory as well as the magnetic strip, so depending on the technology inside the bus both systems will work. The aim is that in the near future when all the buses/trains slot machines are replaced by contact-less modules, the magnetic strip reader will not be required.

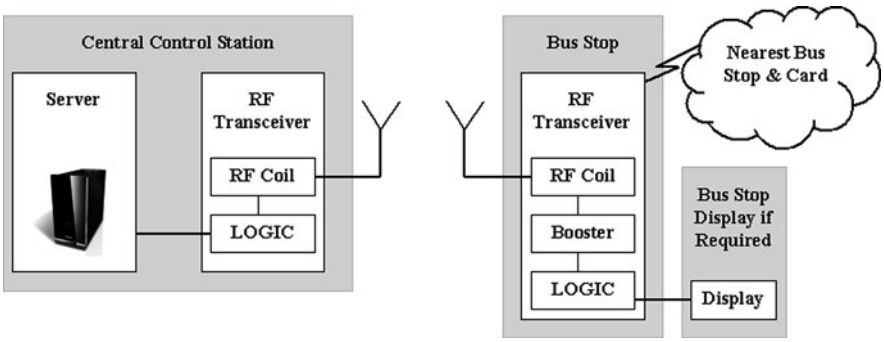

Fig. 6.14 A simple interaction between central control and nearest bus stops

6.5.1.2 Long-Distance-Card

A long-distance ticket (Fig. 6.15) should register itself automatically when the passenger boards and it should cancel itself when the passenger reaches the destination it was paid for. The tickets should respond to RF signal when it receives a poll-signal from the master wireless-center inside the train. In the train the conductor will control all the passengers from a central terminal by sending out a broadcast, an unpaid or expired passengers ticket will respond with a false-signal (red light on the terminal screen), while a paid passenger will respond with a true-signal (green light on the terminal screen). The Conductor-Master-Control-Terminal could be a central point in the whole train; it could also be a handheld device that controls tickets within 4 m of the held-direction.

A practical example of the system usage is a person traveling from Malmö to Stockholm, but paid for Malmö to Jonköping, when the train stops at Jönköping the ticket will be automatically cancelled remotely (RF). Consequently, in the next round of conductor-control it will respond with a false-signal. Long-distance tickets could be recyclable, so the user gets some refund if he/she leaves it when disembarking from the train. e-Screen and the RF signal can be used to pass useful information such as weather forecast, latest sales promotion, city map, restaurant location, cheap transport, etc. (Fig. 6.16).

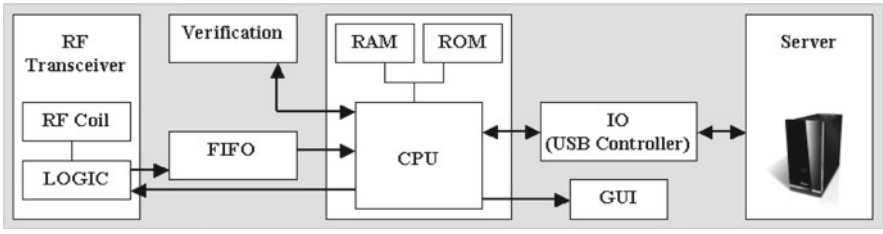

Fig. 6.15 Block diagram of conductor master controller module for long distance

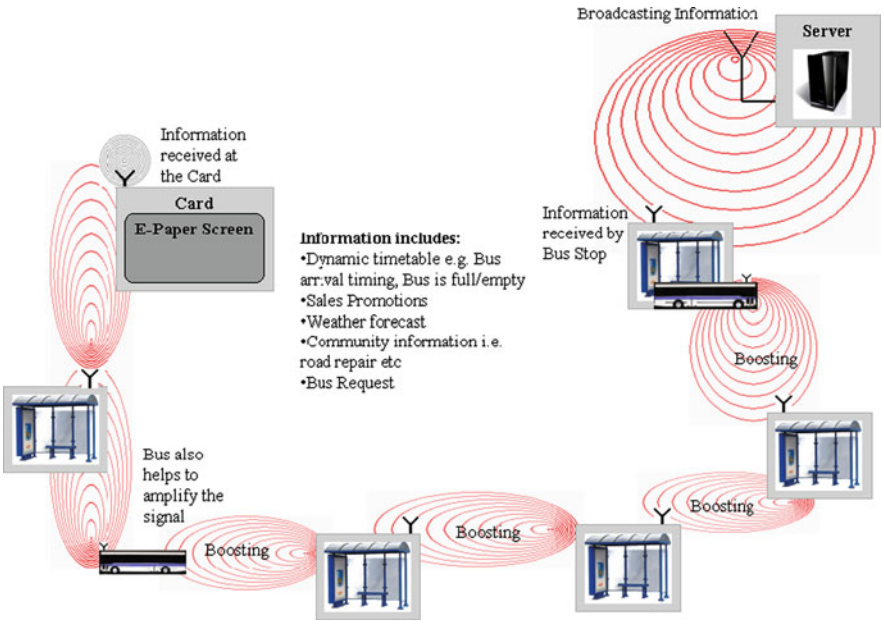

Fig. 6.16 Overview of the Interaction between the central control, the bus stops, the buses, and the cards

The long-distance ticket should have an optional bag-tag-slip that the ticket holder could buy as an additional accessory when purchasing his/ her ticket. The luggage-tag is a simple RF-Transponder that communicates with the ticket through RF and consequently acts as a bag-watchman. A ticket holder should put the bag-tag inside his/her bag before switching it on through the tickets. The ticket-bag-tag-link operates within 5-m radius only, such that if the bag is moved outside this 5 m-radius, the ticket will beep and the owner will know that someone is trying to take the bag away.

Tangible Benefits for Long-Distance-Ticket

- For the transport company:
  - The new card will enable the transport company to generate more revenues through the automated ticket check, i.e., from a central terminal a conductor can detect unpaid passengers via RF signal, resulting in staff reduction for the transport company.
  - The transport company benefits from the new system by using the e-paper screen for sales promotion through the RF.
- For the end user:
  - The user benefits from the weather forecast, latest sales promotion, city map, restaurant location, cheap transport, etc. information that are displayed.

The cards will use piezoelectricity as its primary source of energy, or another alternative could be tiny solar panels.

How It Works

- The e-ticket sends the information to Control Module via the RF Transceiver.[7]
- The information is then placed in the FIFO for Verification.
- After the verification the CPU displays the valid and invalid e-tickets in the GUI by green and red spot, respectively.
- If the conductor finds any red spot or spots, he/she then take a survey of the train and track the invalid e-ticket holder with his/her handheld device.
- The arrow from the CPU to the RF LOGIC block represents the flow of information from the server to the e-ticket, such as weather forecast, address of the hotels and restaurants at the destination station, taxi.

6.5.1.3 Bonvoyage

Bonvoyage is a device to ease your journey by keeping the master in your key chain and the slaves inside your individual luggage you can enjoy your trip. The nightmare of loosing your baggage while you are at the airport, train station, bus stops, or in any crowded environment will be a thing of the past because if anybody wants to steal your baggage, the Bonvoyage is there to warn you of the imminent danger. Bonvoyage comes in three different sets:

- One Master, One Slave
- One Master, Three Slaves (Fig. 6.17)
- One Master, Five Slaves

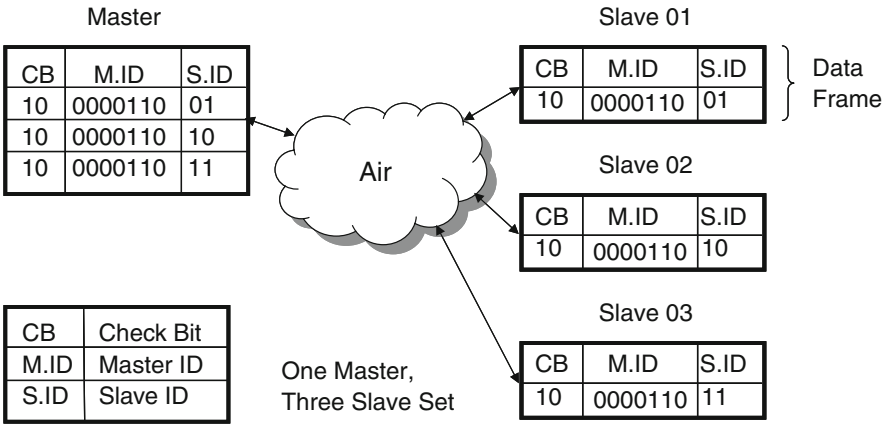

Fig. 6.17 Data frame exchange between master and slave

[7]Note: Logic for long-distance e-tickets is similar to that of pre-paid traveling card. Consequently, block diagram for long-distance e-tickets is not drawn.

How a Master Works

- The master device has the control-logic with inbuilt transceiver of range 5 m.
- It has a battery and is therefore the active unit in the Bonvoyage system.
- An active master sends a unique ID after every 5 s to all of its passive slaves one by one and the slaves respond to their corresponding master with their own ID.
- If for some reason a passive slave does not respond during polling, then the active master detects that the corresponding slave is outside the range (5 m) and will start beeping to warn the user of Bonvoyage.
- Every active master has a unique ID tag so the slaves will not respond to the master it does not belong to.

It is fitted with two buttons, one on-off button and the second button to enable slave selection, i.e., someone who has five slaves, but want to travel with three slaves has to be able to configure the master, such that master is only aware of the slaves that will make the journey. Once turned on, the master will only control the slaves that are selected.

How a Slave Works

- It acts like a transponder; after sending the signal, it goes to sleep mode.
- It does not require any battery because the received transmitted power from the master is enough to respond back to the master.

It has no button, but has its number written on its body for configuration purpose.

Technical Conflicts

A preliminary investigation shows that RFID is being introduced in airports around the world. This is to eliminate the current process, which involves manual sorting activity, which is both costly in terms of personnel and creates some mishandling of baggage due to the human error.

To avoid conflicting with RFID signal band, Bonvoyage will use ISM (Industrial Scientific and Medical) band for transferring data between master and slaves. Before the product comes to the market, the investigation of the exact frequency the airport system RFID is using should be made and then avoid that frequency.

Political Constraints

In this present world of tight security, most people are afraid of unattended luggage; hence these constraints are placed on Bonvoyage.

- Bonvoyage is limited to 5 m range because both the luggage and the owner should be visible to avoid suspicion.

- The slave does not have an inbuilt alarm. Inbuilt alarm or beeping in the slave can cause panic in the airport if a luggage suddenly starts beeping due to increase in the distance between master and slave. Bomb panic among people are not what Bonvoyage is aiming for.

Targeted Market

Bonvoyage is aiming for the individual user, not a company or organization. A general overview is given in Fig. 6.18.

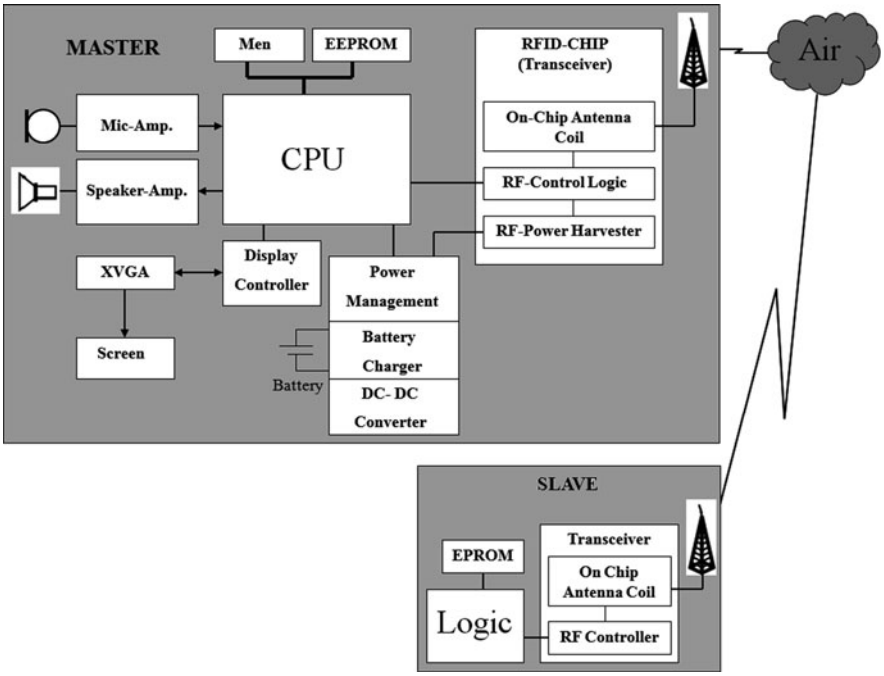

Fig. 6.18 A simple interaction between master and slave

6.5.2 The i-Coin

Mobile telephones are constantly expanding their functionality from simple communicator to communication center. They have a camera, a GPS, a compass and an accelerometer, and a range of services based on them. Notably they carry all the information about a person and have thereby become the most valuable item a person carries with him/her. Gradually, the telephone becomes part of a dynamic social

network, where personal messages and pictures are exchanged. In development is the use for exchange of valuables such as payments, reservations, and receipts.

With so much in one pocket, theft becomes more attractive. Robbery on the street has almost become normal. Many pizza deliverers have this problem of bringing the money back safely. Mobile payment handles ordering and payment at the same time for the benefit of the shop owner. If the payment system gets hacked, the personal bank account is left without any protection. A cash-on-delivery system cuts the risks, but then the shop owner gets more vulnerable.

Having just one mechanism for payments messages, identity theft is following soon. This makes security an important aspect to maintain or create a safe society. Various ways to create secure mobile payment have been introduced. PhoneKeys introduces a mediating service to check on payments orders. FaceCash introduce a download for barcodes as added security. All use the same communication channel and thereby still allow for identity theft.

The longer the distance is that a communication channel has to bridge, the more vulnerable it becomes. A first means to secure the communication is to echo the message. Originally invented for the telex, it re-appears regularly. It does not work however when the communication window is too small, like for instance in satellite communication [44]. The call-back can be seen as a variation. In general, it is sensible to call back when you do not trust the identity of the caller. All this is caused by the fact that there is a single communication channel to observe with enough traffic that eventually the encryption can be broken.

One way to achieve safety on payments is by recognizing that a typical transaction is based on two phases: ordering and delivery. We enforce this by splitting the communication over several channels of different electrical and information properties using different physical devices. Notably, we will use small packaged devices, called i-coins, loaded by near-field messages and monitored by a sensory cloud to verify integrity. The increased variety makes it exponentially hard to fake the communication.

The on-line ordering takes place conventionally, but the payment is only announced to a society server in the cloud and certified on a swiped card. This card takes on the identity of the client and is subsequently being monitored from the cloud. In other words, the person's identity exists in both the cloud and the reality. If the card is infringed in reality (being stolen or lost, and subsequently altered), this will be noticed. Upon delivery, the card is swiped again, compared with the order and confirmed by the cloud and the actual payment is being made within the society. This system, where the transaction takes place in reality and virtually, is clearly harder to be hacked into than a single pay system.

The main advantage is the provision of safety according to individual needs. Safety expectations may differ over time and place, and the i-coins allow a person to choose. i-Coins can physically be moved, giving the notion of identity and ownership in trust with a third party. It will do so temporarily and only for the purpose of the claimed transaction severely reducing the damages caused by theft or loss, or at least the attraction. The technology can easily be added to existing e-systems such as for mobile payments or to create public safety schemes.

The i-coin by itself carries no value. Consequently, the loss of an i-coin has hardly consequences, but the massive loss of these items may cause an environmental waste problem. Therefore, i-coins should be treated as electronic waste. On the other hand, it is technically no problem to salvage them by similar means as metal detectors.

6.6 Safety-as-a-Service

Some very simple examples of Safety-as-a-Service are already in use to a large extent, they generally are used for scanning or updating a computer from a central location. Any virus-detection software provider already has a "Live Update" feature where the current virus detection software configuration of the processor is compared against a most recent list for updates and downloads. The next logical step would be a Repair-as-a-Service proposition where the logical state of the processor is evaluated in the cloud for the determining of correctness of operation. Repairs would be offered if failures have been detected.

6.7 Summary

Blow-ups do happen, but it is good to limit the consequences. We do accept the incidental mishap, but it should not cost one's valuables. With increasing age, people feel less safe. So when everything comes together, safety is all that matters.

Security was involved with holding off the attack. In this chapter we looked at ways to find out about the attack. Of course, one can never be too careful. But holding off all contact with the outside world is against the purpose of being ambient, while opening up may cause a lot of false alarms. This is just as bad because too many false alarms includes shutting off the detection, leaving the system again defenseless.

Therefore, we have proposed ownership as the key to safety. It is discussed how persons and computing machines can be reliably identified. Then we discuss two typical electronic payment systems and illustrate how ownership influences their safe handling.

References

1. Studer A, Shi E, Bai F, Perrig A (June 2009) TACKing together efficient authentication, revocation, and privacy in VANETs. In: Sixth annual IEEE communications society conference on sensor, mesh and ad hoc communications and networks (SECON2009), Rome, Italy
2. Shi E, Perrig A (December 2004) Designing secure sensor networks. IEEE Wireless Commun 11(6):38–43
3. Holcomb DE, Burleson WP, Fu K (September 2009) Power-up SRAM state as an identifying fingerprint and source of true random numbers. IEEE Trans Comput 58(9):1198–1210
4. Lawton G (May 2010) Fighting intrusions into wireless networks. IEEE Comput 43(5):12–15

5. Waksman A, Sethumadhavan S (May 2010) Tamper evident microprocessors. In: 31st IEEE Symposium on Security & Privacy, Oakland, CA
6. Lamport L, Shostak R. Pease M (July 1982) The Byzantine generals problem. ACM Trans Program Lang Syst 4(3):382–401
7. Jadliwala M, Upadhyaya S, Taneja M (October 2007) ASFALT: a simple fault-tolerant signature-based localization technique for emergency sensor networks. In: IEEE symposium on reliable distributed systems (SRDS 2007), Beijing, China
8. Roman D (January 2010) The corollary of empowerment. Commun ACM 53(1):18
9. Isermann R (1984) Process fault detection based on modeling and estimation methods – a survey. Automatica 20(4):347–404
10. van Veelen M (2007) Considerations on modeling for early detection of abnormalities in locally autonomous distributed systems, Ph. D. Thesis, Groningen University, Groningen, The Netherlands
11. Amin M (2000) Towards self-healing infrastructure systems. IEEE Comput 8(8):44–53
12. Ray A (2004) Symbolic dynamic analysis of complex systems for anomaly detection. Signal Process 84(7):1115–1130
13. Chua L (September 1971) Memristor-the missing circuit element. IEEE Trans Circuit Theory 18(5):507–519
14. Strukov DB, Snider GS, Stewart DR, Williams SR (2008) The missing memristor found. Nature 453(7191):80–83
15. Spaanenburg L (September 2007) Early detection of abnormal emergent behaviour. In: Fifteenth European signal processing conf (Eusipco2007), Poznan, Poland, pp 1741–1745
16. van Veelen M, Spaanenburg L (2004) Model-based containment of process fault dissemination. J Intell Fuzzy Syst 15(1):47–59
17. Isermann R (1984) Process fault detection based on modeling and estimation methods −a survey. Automatica 20(4):387−404
18. Patton R, Frank P, Clark R (1989) Fault diagnosis in dynamical systems − theory and application. Prentice-Hall, Englewood Cliffs, NJ
19. Barabasi A-L (2003) Linked: how everything is connected to everything else and what it means for business, science, and everyday life. PLUME Penguin Group, New York, NY
20. Keim B (April 2010) Networked networks are prone to epic failure. Wired Sci www.wired.com/wiredscience
21. Buldyrev SV, Parshani R, Paul G, Stanley HE, Havlin S (April 2010) Catastrophic cascade of failures in interdependent networks. Nature 464(7291):1025–1028
22. Vespignani A (April 2010) The fragility of interdependency. Nature 464(7291):984–985
23. Amin M (September/October 2003) North America's electricity infrastructure: are we ready for more perfect storms? IEEE Secure Private 1(5):19–25
24. Hofmeyr SA, Forrest S (2000) Architecture for an artificial immune system. Evol Comput 8(4):443–473
25. Brooks RA (1986) A robust layered control system for a mobile robot. IEEE J Robot Autom 2:14–23
26. Geman S, Bienenstock E, Dousat R (1989) Neural networks and the bias/variance dilemma. Neural Comput 2:303–314
27. Ahnlund J, Bergquist T (2001) Alarm cleanup toolbox. M. Sc. Thesis, Department of Information Technology, University of Lund, Lund, Sweden
28. Ahnlund J, Bergquist T, Spaanenburg L (2004) Rule-based reduction of alarm signals in industrial control. J Intell Fuzzy Syst 14(2):73–84 (IOS Press)
29. Larsson JE (2000) Simple methods for alarm sanitation. In: Proceedings of the IFAC symposium on artificial intelligence in real-time control, Budapest, Hungary
30. Ahnlund J, Bergquist T (2004) Process alarm reduction. Lic. Thesis, Lund University, Lund, Sweden

31. EEMUA (1999) Alarm systems, a guide to design, management and procurement. EEMUA Publication 191, EEMUA, London
32. Veland O, Kaarstad M, Seim LA, Fordestrommen N (2001) Principles for alarm system design. Institutt for Energiteknikk, Halden, Norway
33. Schoeneburg E (1992) Diagnosis using Neural Nets. In: Proceedings CompEuro'92, The Hague, The Netherlands
34. Simon C, Goldstein I (1935) A new scientific method of identification. N Y State J Med 35(18):901–906
35. Daugman JG (March 1994) Biometric personal identification system based on iris analysis. US Patent 5291560
36. Rice J, (October 1987) Apparatus for the identification of individuals. US Patent 4:699,149
37. Watanabe M (2008) Palm vein authentication. Chapter 5 In: Ratha N. K., Govindaraju V (eds) Advances in biometrics sensors, algorithms and systems. Springer, London
38. Park G, Im S.-K, Choi H (1997) A person identification algorithm utilizing hand vein pattern. Korean Signal Process Conf (Pusan), 10(1) :1107–1110
39. Hashimoto J (2006) Finger vein authentication technology and its future. In: Digest symposium on VLSI circuits, Honolulu, HI, pp 5–8
40. Im S-K, Park H-M, Kim S-W, Chung C-K, Choi H-S (2000) Improved vein pattern extracting algorithm and its implementation. In: International Conference on Consumer Electronics ICCE, Los Angeles, CA, pp 2–3
41. Hitachi Engineering Co. Ltd (September 2009) Finger vein authentication technology. Available at http://www.hitachi.eu/veinid/. Accessed 16 Oct 2010
42. Malki S, Spaanenburg L (2010) CBAS: A CNN-based biometrics authentication system. In: Proceedings of the 12th international workshop on cellular nanoscale networks and their applications, Berkeley, CA, pp 350–355
43. Kumar A, Prathyusha KV (September 2009) Personal authentication using hand vein triangulation and knuckle shape. IEEE Trans Image Process (18)9:2127–2136
44. Asimov I (February 1962) My son, the physicist. Sci Am 206(2):110–111
45. Evans J, Haider S (February 2008) E-facts. Masters thesis, Department of Electrical and Information Technology, Lund University, Lund, Sweden

Part IV
Last-Mile-from-the-Cloud

The already maturing "cloud computing" concept allows for the integration of data collection networks into a combined cloud computing/ambient intelligence environment. This integration, in addition to many benefits, will also introduce new requirements. We envision that cloud-centric sensory networks in their respective environments will deal with those new requirements (such as safety, security, and integrity), as well as develop new "disruptive" business concepts (software-, security-, gaming-, every-thing-as-a-service). These developments constitute a completely new business paradigm, gone are the days of sole-source software packages (e.g., Office), and gone are the days of lonely self-contained sensory networks.

In the fourth part of the book, we focus on details of a futuristic sensory/display-instrumented home environment completely in touch with the Cloud.

Highlights

Centralized Compute Power

The processing in the cloud will be cost-effective, pruned to the essential as a result of the centralization of processing in the cloud. Just enough processing resources in the sensory networks will be applied to its requirements. The emphasis on centralization will result in high-productivity usage of the resources present in the cloud.

System Virtualization

From the point of view of the sensor network, the most appropriate compute configuration will become available in the cloud (at price of course). The complete range of computational alternatives can be available in the cloud, ranging from multi-core/tile blades, vector-graphics processors, to reconfigurable computing (FPGA) resources.

Thinking-in-the-Cloud

The introduction of cloud computing and cloud-based sensor networks will result in opportunities to rethink basic processing philosophies as well as to introduce new (disruptive technology) businesses. Many commercial cloud systems and services have already become available and are already challenging current business concepts.

HPC-from-the-Cloud

The democratization of the cloud, even from the point of view of lowly sensors and handheld devices, will make the presence of high performance computing (HPC) equipment in the cloud universally accessible and available. Access to the server farms in the cloud can be as simple as a request for resources and the subsequent payment of its use by credit card.

Chapter 7
Bringing the Cloud Back to Home

In this chapter we will describe the coincidence in a future domestic environment of sensor, media, display, and cloud technology. The resulting high-technology environment will have a great impact on living, working, relating, and the economic world. These disruptive technology developments will result in new business models for Sensory Networks (including imagery/displays) and Cloud Computing.

7.1 Net-Centric Home Automation

The real impetus for embedded applications is in networked environments. Local functions communicate and cooperate for global intelligence. Such environments are not limited to the Internet, but may involve any geographical dispersion ranging from world- to home-scale. To create such a transparent and pervasive embedding of digital functions requires the enforcement of a single network philosophy for small and large computing and controlling devices. An environment of special interest for this new range of embedded products is the Home, where the fight between Windows, browser and Java technology is started and is still on.

The Internet has rapidly become the universal means of communication between computers. It has likewise created a new industry for an entire collection of e-techniques such as e-commerce, e-pliance, and e-surance. Having a computerized interface is not the prerogative of solely a computer; any appliance can be computerized and become networked. Broadly speaking, one distinguishes between computing and controlling. This distinction returns in net-centric appliances where we find much more than only computers as products. Notably, we see the higher demands of computing (and entertainment) and the lower demands of sensory-activated controllers (e.g., climate control or surveillance).

In this section we will focus on technologies for embedded systems in the household. After a long time of electrification and electronification of the household, it has become feasible to connect many of those kinds of devices into a home-network [1]. In other words, the household becomes its own local-area network (LAN) of consumer systems that from the outside do not even resemble a computer. This network will carry telephone conversations, movies, stills, music, and of course plain information. And for this to happen, any broadband access scenario will do.

L. Spaanenburg, H. Spaanenburg, *Cloud Connectivity and Embedded Sensory Systems*, DOI 10.1007/978-1-4419-7545-4_7,

7.1.1 Home-Networks anno 2000

The major characteristic of the household for networking is that, in contrast with the office, a suitable physical backbone for digital communication is lacking. As it also does not pay to rebuild the home for new wiring, one has therefore to use whatever wiring is present: power-line, specialized line, and no line at all (wireless). The "Yankee Group" forecasts, already in 1998, that by 2003 home-networks will be used in more than six million US households. According to their estimate, 25% of such networks will be wireless, 5% power-line, and 70% specialized line (like Grade 5 twisted pairs, but largely phone wires). Any standard will have to reckon with the fact that on a small scale some dedicated wiring has already been created. For instance, the audio-set will usually contain a number of physical products that communicate over a brand-owned protocol. As proprietary systems have usually a low impact on the market, the trend is for global alliances with standardized, open interconnect. In this situation, phone-line systems have an historical lead, which they may keep in the future. Nevertheless, the wireless technology may have some surprises in store.

To specify the technology and parameters for any residential communication, the Electronic Industries Alliance (EIA) has formed the CEBus Industry Council (CIC) to produce the CEBus specification (EIA-600). The partners have collaborated in the definition of a Common Application Layer (CAL), based on the IEEE 802.3 standard for the physical layer.

A number of manufacturers have cooperated under the auspices of the EIA-600 to create the Home Plug-and-Play (HomePnP) specification to harmonize the different and independently developed consumer products. Since HomePnP is intended to run on established consumer electronic protocols, the demands on the underlying transport are minimal: Loose Coupling. The added advantage is that designs can be constructed without specific knowledge of other vendor's products.

7.1.1.1 Power-Line Networks

The origin of using the power-line for other purposes than the mere transport of power lies in the need for simple control. This was especially of interest during the early 1980s. From that time, only two concepts seem to have survived. The first is LonWorks, often advertised as the means for a low-cost, intelligent network. The other is X10, often advertised as an Open Initiative. However, the X10 Company in Hong Kong wholly owns the basic X10 protocol. Though anybody may create X10-complient transmitters, only the company itself is allowed to construct and sell the receivers. This makes the X10 proposition less open.

Communication over power-lines generally uses some variation on frequency-division multiplexing (FDM). Special caution should be spent to avoid signal interference by multiple reflections. The right technology should theoretically be able to handle 100 Mbit/s. Another popular technology uses carrier-sensing multiple access with collision detection and resolution (CSMA/CDSR) similar to IEEE

802.3 protocols like Ethernet. As an example, the Plug-in PLX technology allows for 2 Mbit/s.

There is little evidence of an ongoing standardization in communication over powerlines, though the International Powerline Communication Forum (IPCF) appears to make an effort. As other efforts claim to include power-line technology also, there seems to be no reason to expect a change in this situation. For instance, based on the CEBus standard, a number of companies (such as Intellon[1] or Domosys[2]) offer suitable hardware. Such schemes allow an easy change-over from power-line to a different technology without changing the application software, or even to use a mixed technology of power-line, phone-line, and Internet.

7.1.1.2 Specialized Line Networks

In June 1998, a number of American ICT companies (Compaq, Hewlett-Packard, Intel, IBM, Lucent, etc.) founded the Home Phone-line Networking Alliance, or HomePNA for short, to draft a set of industry-wide standards for digital communication within the home over the phone-line. By December 1999, the group has grown from the initial 11 to more than 100, of which more than 25% are non-American.

As phone-lines are already in place, they are most likely to be used. To connect one needs devices that are simple to construct, easy to use, secure from outside intrusion, scalable, able to support high data rates, and cost less than $100 per node. Phone-lines are meant to carry voice rather than high-speed data. Further phone traffic should be continued while accessing the Internet and using entertainment programs. The Tut technology in HomePNA release 1.0 does this by frequency division multiplexing to three channels: audio (DC – 3.4 kHz), phone (25 kHz – 1.1 MHz), video and data (5.5 MHz–9.5 MHz).

The system assumes the availability of some residential exchange. Sony has originated the Firewire protocol, later standardized as IEEE 1394. The protocol is packet-oriented; the first half of the packet contains a synchronized line of a video image, the second half allows for asynchronous control messages. Home PnP can be used to manage a Firewire network, though in principle any standard can be applied.

7.1.1.3 Wireless Networks

As wireless seems to be the obvious choice, the problem lies in the freedom of selection. Standards appear to be as bountiful as sand particles on the beach, see Table 7.1. Most of the wireless communication within the home is targeted for the 2.4 GHz unlicensed industrial, scientific, and medical (ISM) band.

Under leadership of the IEEE, the 802.11 standard is gradually being improved to get the most out of the ISM. Later, it has taken the initiative for the Home RF Working Group. This group has created the Shared Wireless Access Protocol

[1] http://www.intellon.com.

[2] http://www.domosys.com.

Table 7.1 Wireless home-network technologies

	Application	Characteristics	Freq. band, GHz	Modulation	Maximum data rate, Mb/s	Specifying organization	Certification agency
Open air	Mobile data	Small, light, low power	2.4	Frequency hopping (FH)	1.6	Wireless LAN Interoperability Forum (WLIF)	
802.1 FH	Data networking	Encryption available	2.4	FH	2	IEEE	WLIF
802.11 DS			2.4	Direct sequence (DS)	2		Wireless Ethernet Compatibility Alliance (WECA)
High-speed 802.11	High-speed LAN	Broadband	5 2.4	FDM DS	6–54 11		WLIF and WECA
BRAN/HiperLAN	High-speed multimedia	Voice, data, and video	5	GPSK	24	European Telecommunications Standards Institute (ETSI)	
DECT	Home and small offices	Integrated voice and data	1.88–1.9	GFSK	1.152		
SWAP	Homes and small offices	Low cost	2.4	FH	2	HomeRF Working Group	
Bluetooth	Short range	Voice and data only	2.4	FH	1	Bluetooth Consortium	

(Swap); the current version is 1.1. SWAP is based on the open IEEE 802.11 standard, but specially adapted to the home environment.

Another but European initiative is the Broadband Radio Access Network (BRAN) by the European Telecommunications Standards Institute (ETSI). In the general framework of the Universal Mobile Telecommunication System (UMTS), some variations are made with the home in mind. Furthermore, there are manufacturer initiatives like DECT and BlueTooth, which seem to carry some momentum.

7.1.2 Operating the Home anno 2010

Clear progress has been made since 2000, but not as fast as one may expect. With respect to power, the critical point seems the Home Battery. The current work on Smart Grid seems to focus on the networking of appliances, such that peaks in the energy consumption are avoided. But in the meantime, battery technology is making progress and the moment that a house can have its own supply for the next three days seems not too far away. In such scenarios, the Home Battery will already take the peaks in the consumption away from the grid. It will negotiate energy prices over the Net and buy when it is cheapest.

It is suggested that the e-Car will take the role of the Home Battery. Maybe yes, maybe no. Driving the car would then leave the Home powerless. So, it seems far more likely that the e-Car will push the development of the Home Battery, such that the e-Car can be loaded without additional peaks on the energy grid. All in all, the advancement of the three grids inside the house seems to be of immediate interest.

7.1.2.1 Power-Line Communication

Power-Line Communication (PLC) has suffered a long time from stability problems, but seems to have overcome this. In general, one can find four variations depending on the carrier frequency:

- Ultra-High ($\geq$100 MHz). This is typically for communication outside the home over long distances.
- High ($\geq$MHz). Here we find power-line Internet, bringing the Web over ordinary power lines. A computer or similar digital equipment can be simply plugged into the wall to have high-speed Internet access.
- Medium (kHz). This is usable for remote control, but also for radio transmission in the house.
- Low (<kHz). Here we find the more popular remote-reading applications. The reach of several meters confines it to parts of a house.

Anno 2000, the world of PLC is governed by two standards. The IEEE P1901 explains how power-lines should be employed for data purposes, while ITU G.hn takes also other types of house wiring into account for building a home grid.

7.1.2.2 DLNA

The Digital Living Network Alliance (DLNA) aims to support multimedia services within the house. With currently more than 7000 different devices certified, it more or less guarantees the interconnectivity of consumer devices. It also included provisions for Data Rights Management (DRM). There are also other players in the field, but their efforts are viewed as complementary:

> The UPnP Forum is an industry initiative aimed at connectivity between stand-alone devices and PCs from different vendors. UPnP technology is a key standard required for an interoperable network in the digital hone. On the other hand, the Internet Home Alliance, a cross-industry network of leading companies advancing the home technology market, explores new uses and conducts field trials to validate consumer needs. Both organizations are complementary to the vision of DLNA, which is focused on developing and promoting an interoperable network through a specific framework.

7.1.2.3 The Internet

The Internet can serve to transmit power next to data. Power over Ethernet (PoE) serves then as an alternative to wireless power transmission (or power harvesting, see Section 4.1.3) and can be used to pass 15–25 W to a digital device connected to the network, such as IP telephones, cameras, and LCDs. In this sense it complements USB. It is usually implemented according to IEEE 802.3af.

In the abundance of alternatives, the Internet technology seems to slowly make progress as being usable for everything. With the rise of Android we see all kind of consumables, including TV, being accessible from the Internet. On the other hand, TV is also accessible through a PC. This indicates that the Home Grid wiring problem is gradually being unraveled, leaving power and apps as the next issue.

7.1.3 The Home Outside the House

When home automation will finally brake through, the goal is the Caring, Comforting, and Concerned Home. It pleasures and secures but will not impose. The following is a typical story about what we have in mind with such words:

> Bob had always felt that there was something weird about the house. As if somebody was looking at him. But no matter where he looked, there was nothing to be seen. Gradually the feeling wore off. He still felt being watched, but he did not stop to see what it was anymore. And finally he totally forgot about it; he almost felt at home.
>
> One day, when he came home, the place was crowded with police, waiting for him at the driveway. They had been summoned by a call from the house. A burglar was spotted and chased away. This was all broadcasted directly to the police station. When they arrived at the house, a full account of what had happened was already being printed, complete with pictures of scenes and fingerprints. But who had done all this? Nobody was supposed to be in the house and the neighbors were off to a funeral.
>
> So when the police left, he went immediately to his PC. What was printed must have gone through his server and therefore left traces. In fact, there was a file and . . .... he had made it himself! He was the creator, but at the time of the burglary, while he was not at home.

How was that possible? The next day he went over and over this problem. Somebody was impersonating him, but nobody had seen that person.

Then people started to remember small, seemingly unimportant incidents in the house. How Alice had inspected herself in the mirror, while softly talking to herself. Then Bob had said that she looked fine. When she turned around to face him, he had apparently left already but he smiled knowingly to her when she came back at the party. But had it really been him?! And Eve told about the night that she had sneaked down to the kitchen to take some of the delicious strawberries. But when she opened the refrigerator door, Bob had said from the dark that she could take anything, but not the strawberries.

So he must have been right all the time: it was the house! The house must be haunted and it could only be a matter of time before it turned ugly: before it would replace Bob. Apparently it was learning from Bob's character over the past months and did already behave just like him. Bob had to put an end to this alter ego and quickly, because with thoughts like this, who knows what might happen?!

Bob left the house when it became dark. From Alice's place he called home and talked with himself. So he must be home, but he was not. There was only one way to find out: to break into his own house. So that's what he did. He went to the back of the house and climbed over the wall. Due to the hooks on the top, his hands and clothing were cut in the process. Stepping down into the garden, a total silence enclosed him. Under other circumstances this would have been comforting. Slowly he moved forward, attentive to the unexpected. Nothing happened. He broke the kitchen window, grabbed inside for the handle, opened the door and went in. Still nothing happened. The house was empty, void of any natural or supernatural presence.

Suddenly he felt tired and set himself down. All tension from the past weeks seemed to be released into a single, seemingly infinite cry of relief. Tears came from nowhere and in the mist of his eyes he saw just emptiness. But no . . . there were myriads of eyes, small needle points that pierced through the dark. And a consoling sound carried through the dark, telling that he was all right. He was among friends and all his worries were gone. And gradually he slipped into a deep and healing sleep.

When Bob woke up the next morning, it all seemed like a bad dream. But with the fresh blood crusts and torn clothing, it was clear that something must have happened. At that moment a friendly face appeared in the mirror and said: "Good morning." Bob asked him angrily for his name and mission. "I am your house and we need to talk. Your behavior yesterday was very erratic. You tried to burgle your own house and damaged us in doing so." Bob reacted by standing up to run for the door. But the door was locked now. He felt panic breaking out. Where could he run to, where should he go?

And then the panic slid away. Was it something in the air? He just felt at ease, even a little bit exhilarated. And the house started to explain. A robotics engineer built it. He liked to make small bots. When he had already quite a collection, he slowly started to give them duties for whenever he was elsewhere. After his death, they started to rely on one another, caring for the house and guarding it closely. It was relatively easy to find out who the new owner was. They got quickly used to him and cared for him, just as they did for their creator. At daytime they kept out of sight, at nighttime they came out and did their work, which was making sure that nothing bad happened. And they learnt how to delegate and specialize, thus being stronger and cleverer. They took his pulse, measured his temperature and analyzed his breath. They checked the house, asked for quotes to do repairs and placed the repair orders. At the beginning of each day their findings and actions were collected in the day report within the PC.

This had been going on for several decades now, unnoticed by anybody while the house and its inhabitants were kept spick-and-span. No harm was meant and no harm was done. Even Bob could find himself pleased with the help. He started to enjoy living in the house.

> Sometimes he thought that he saw a bot but otherwise he was alone and still cared for. He did not have to wonder about his health: it was all documented in the day report. He did not have to check the house; everything became repaired before any real damage was done. And when he got very old, he still did not have to leave.

7.2 Home–Human Interfaces

In the following sections we look at two specific aspects of high-technology appliances in the home. We first review the technology involved in increased sophistication of observing activities and/or sensor measurements in or around the house. These imagers are used for pleasure, documentation of life, or in the protection (security) of home and its inhabitants. The cameras described perform with a wide scale of accuracy (and purpose) at a wide range of costs.

Secondly, we review the various technologies involved in displaying imagery data and reports. Especially when applied for entertainment, there does not seem an upper limit on size and level of detail of the displays demanded. The display alternatives range from small mobile devices to a multi-panel dome-like projection system.

Imaging and display processing requirements generally are on opposite high ends of the sensor-(-to cloud-)-to-presentation flow. Both require a large amount of relatively simple computations on large data frames. For both imaging and displaying devices, sophisticated algorithms can be developed to realize larger virtual-size devices, in imaging by using computational photography on multiple cameras, in displays by multi-image patching and blending.

7.2.1 Home Imagers

The camera is a typical example of a product that has been drastically changed by the advent of microelectronics. Though it was known for a long time, that microelectronic devices are very sensitive to light, this effect was only spuriously used [2]. In the 1980s, the introduction of Charge-Coupled Devices (CCD) brought interest for focal-plane vision sensors, as the image could be easily shifted out. Then CMOS technology improved the interface to digital computing engines, making for a vision device that could be easily integrated in all kind of products.

It is of clear practical significance to raise the capacity (pixel count) of a vision sensor as high as possible. As the principle of light to current conversion is dependent on the purity of the silicon, the wafer yield is strongly dependent on the required sensorial image capacity. With improving technology, the price of a small individual sensor will go down, but there will always be an extra price to pay for larger conglomerates. The alternative is then the sensor array, a fixture of many small (and therefore cheap) sensors that collectively work as one. For things such as a large aperture camera [3], this moves the problem to integration subjects such as image stitching and synchronization.

Configuring the vision sensor as a Random-Access Memory (RAM) gives the usual memory bottleneck in the subsequent image transport. Image compaction helps, but going from a fixed line to wireless aggravates the problems again. It is therefore of interest to bring more preprocessing into the sensor.

Therefore, two different directions have been pursued. Anafocus has added basic image-processing hardware in analog circuitry to the focal plane sensor that extracts features rather than communicate full image [4]. On the other hand, NXP has created an image processor that performs lengthy operations on double-buffered image lines to reduce the bandwidth requirements [5]. Both directions aim for a vision sensor that has more intelligence and can perform a number of basic operations within the camera itself.

The digital camera has found a large number of applications where a range of supporting technologies are gradually introduced (Fig. 7.1), but in most it is just a single image capture device. Making it more intelligent creates a larger versatility, allowing for a greater handling ease in the amateur field. However, a number of installation problems have remained.

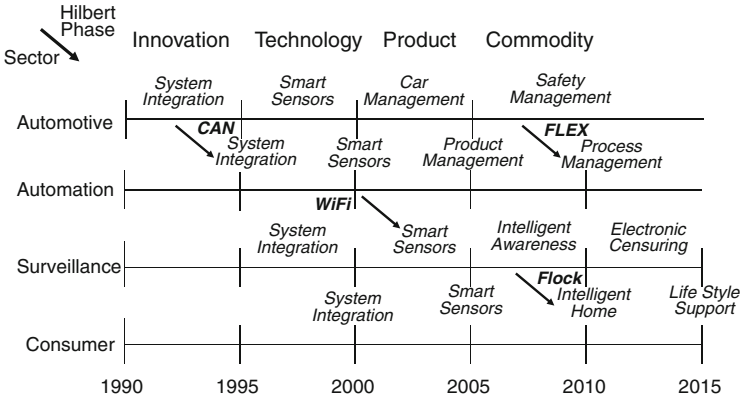

Fig. 7.1 The vision technology roadmap

Compared to the human eye, the clear difference is the presence of an adaptive lens for accommodation, the use of two eyes for depth control and the hierarchical feature extraction and image understanding layering. Though there is lively research on such Neo-Cortex issues, the question is how such results can be used to inspire current developments for industrial automation.

For instance, applications have focused on the situation in the metal industry, where we want detect, isolate, and diagnose defects on the surface of, for instance, flat steel coming from rolling mills. In the current situation, a single camera is in use, requiring a large installation and maintenance effort as a meaningful system needs both light source and camera to be placed under exactly the right angle. The bio-inspiration suggests that natural diversity can be created by using many cheap and uncalibrated vision sensors that collaboratively perform the task of a single high-quality and adaptive camera.

7.2.1.1 Principle of 3-D Measurement

A number of approaches for measuring within an image have been pursued in the past. They come largely in three categories. Firstly, one may assume that the camera is in a fixed position, but the installation leaves much to be desired. Self-calibration can be achieved by checking on known object measures versus their deformation in the image. In ter Brugge [6], the license plate viewed in a car image is checked versus the known size of the characters and of the plate, according to the traffic regulations. This provides the parameters to transform the cut-out plate to a normalized format, so that the license reading is independent on the camera angle.

Secondly, two cameras can be used to view the same object [7]. The calibration needs for the two cameras can be related through the fundamental matrix. Assuming enough objects in the image, the epipolar lines can be established to solve the fundamental matrix to establish the respective camera view angles. An alternative approach is reported in [8], where a known object is moving into the scene.

Thirdly, additional laser projectors can be applied to cast a spot on the object [9]. With known installation of the sources, the camera can time the number of pixels between the spots as viewed and therefore relate clock frequency within the vision sensor to the object measures.

Common in all these approaches is the need to take away the shadow from the object. Shadow blurs the edges and therefore it needs to be eliminated to establish the real perimeter. Here, it is recommended to do the opposite and use the shadow. The main reason is that in an industrial environment there will be no objects between the camera and the target, which makes the classical means for self-calibration hard to use. On the other hand, we have a clear light source to object correspondence in the reflection of the shadow. In line with [9], the single-camera model can now be formulated.

As Fig. 7.2 shows, two light sources are fixed up on either side of a camera to produce a triangle. Distance between those two projectors is ticked as $D_{\max}$. Assume that M and N are the two projected points on screen, whose distance is ticked as D_k. Point O is the apex of this triangle. Distance from O to line MN is H_k, as well as distance from O to line AB is $H_{\max}$. According to the Triangle Theorem, it can be found that $D_k/D_{\max} = H_k/H_{\max}$, which can be rewritten to $D_k = (D_{\max} * H_k)/H_{\max}$. This formula is the theoretical basis to calculate actual distance in the model.

Firstly, it is clear to see that $D_{\max}$ is just the distance between those two light sources. That distance can be easily measured by regular measuring device, like rulers. Then, since the apex O is behind screen and actually exists virtually, it is impossible for people to measure distance $H_{\max}$ directly. However, it still can be calculated by triangle equations. For simplicity, the two angles α and β between light sources and the screen are set equally; in other words, the triangle AOB is an isosceles triangle. According to the Pythagorean Theorem, $H_{\max}$ can be computed as $\tan \alpha * D_{\max}/2$. Finally, the distance H_k can be fetched by $H_{\max} - H_d$, which is the perpendicular distance from the camera or light source to the screen.

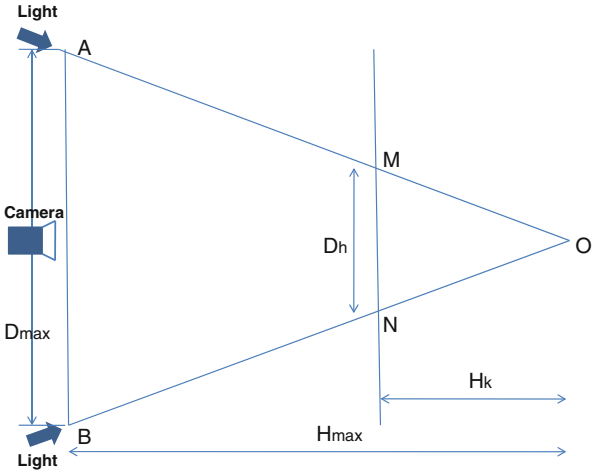

Fig. 7.2 Measuring method applied to the system model

To sum up the above, the D_k formula has been transformed to $D_{\max}$ * (tan α * $0.5\ D_{\max} - H_d$)/(tan α * $0.5\ D_{\max}$), in which $D_{\max}$, H_d, and angle α can be easily measured by people in steel factories.

A noticeable problem is brought by the view angle, causing round shapes to become elliptic. For convenience we introduce the Horizontal Resolution K_h, defined as one pixel width's corresponding spatially horizontal length, and the Vertical Resolution K_v, defined as one pixel height's corresponding spatially vertical length. Under such circumstances, the actual length of two sides of a rectangle, D_h and D_v, can be calculated by the triangle measuring method discussed before. When the camera position and/or angle is changed, which means the pixel number between two certain points on image also changes accordingly, then the resolution values K_h and K_v need to be recalculated for a new measuring.

It is generally acknowledged that camera lenses are usually not perfect optical devices. Nonlinear distortions are prone to appear on the edges of photos. So, the rectangle area defined by D_h and D_v should not be too close to image boundaries. In other words, D_h and D_v are supposed to be defined around the center area of an image and neither too large nor too small. This will not work for measurements on curved objects, where again changes in view angle will lead to nonlinear distortions.

7.2.1.2 Multiple Cameras

The use of multiple cameras has already been mentioned in the previous section for calibration purposes. Both cameras provide an image and the server manipulates them to establish the calibration settings [10]. An extension of this approach lies in the camera array, where the server stitches the individual images to a single large aperture [3]. The drawback of such techniques is the need to transfer the images.

This usually requires high-performance (GB/sec) buses, which drives the costs in a wireless cluster of cameras up to unreasonable levels.

In the VASD (Vision Array for Surface Detection) project, a technique is developed whereby the feature extraction is performed within each single camera [4, 5], and then the server inspects the feature values to come with a consolidated decision [11]. Figure 7.3 shows the principle of operation. When an object passes by or when multiple cameras take a picture, the light source casts different shadows of the object. These shadows are measured and the values are sent to the server, where the Light-Object-Camera-Screen system is nonlinearly validated as a redundant number series.

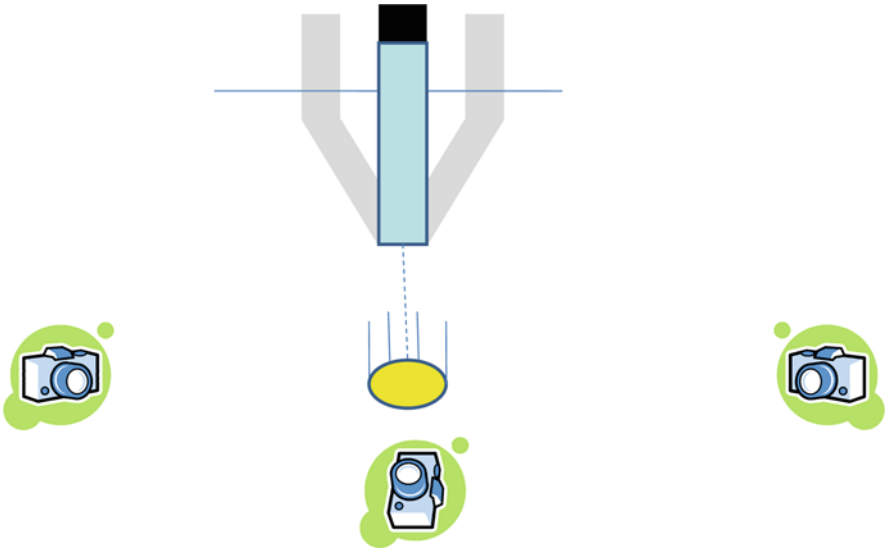

Fig. 7.3 Different cameras see different shadows

The system development starts from MATLAB simulations. Any shape of an object can be constructed in MATLAB. Camera and light manipulations are also readily available. In the next phases MATLAB is gradually turned into reality (Fig. 7.4). The major development problem is not as much the conversion to smart vision sensors, as well as the fact that MATLAB does not support shadow casting. For our purposes, we use ray tracing as supported by the MATLAB add-in package "Optical Bench." Screen is an optical element to carry on radiated rays. It is used to present points of intersection and then displays shadow images, actually lighted areas and unlighted areas.

Rays are very important element to generate shadow. This is based on the Laws of Rectilinear Propagation of Light, namely, if light along its direction falls cross an object, it will be blocked; otherwise, it will intersect with the screen. Finally the image on screen is shadow cast, actually unreached areas by rays. Apparently, only one ray is not enough to cast a shadow area. A sharp increase of rays' number is

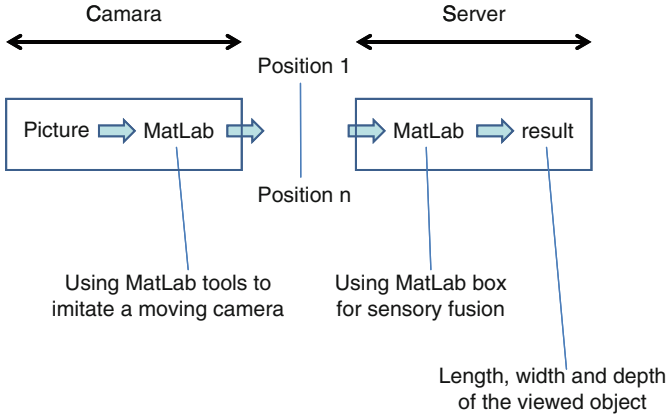

Fig. 7.4 VASD development phase I

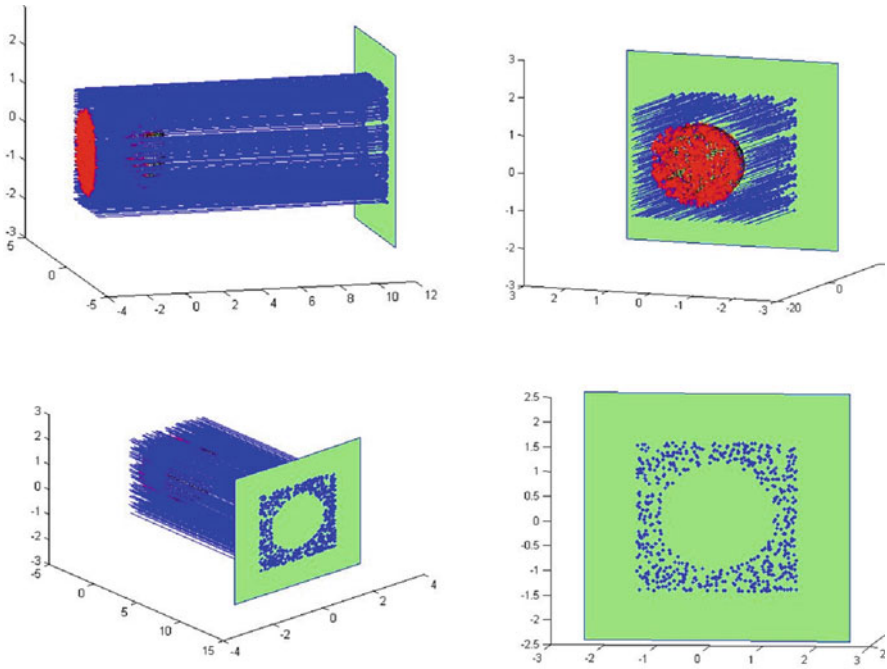

Fig. 7.5 Thousand rays generation in the system model

needed. Figure 7.5 illustrates the use of 1000 rays to create shadow on a screen. Different view angles are adopted to observe, including the views of front, profile, and back side.

Unfortunately, shadow cannot be guaranteed at all times. One reason is that the shadow can be hidden behind the object for a certain camera angle. It may appear

again if the object and/or the light is moving, but a more generic solution is to have a moving camera, or rather many cameras effective at the same time.

7.2.2 Panoramic Visualizations

Real product innovation does not come out of the blue. Surprising ideas will usually never reach the market place on their own strength. A number of things have to work together for the idea to find sufficient momentum. This does not mean that good things are bound to happen when individual developments meet. It is just that the chances are much better that strength through harmony can be achieved. There is every chance on many such innovations in mobile networks, with corresponding consequences in the cloud.

Let us investigate the potential for a breakthrough coming from three sides (though for the moment we have not checked whether they all do matter or in which degree):

- Light-emitting diodes: LED displays allow for erasable displays in any form and shape.
- Image capturing: the camera of old becomes smaller as well as more accurate.
- 3-D imaging: improved algorithms for 3-D rendering.

The idea is that the old world of the photo camera and the picture frame is to change. In the past period we have seen the camera going digital, allowing it to enter a mass market but still making 2-D pictures in standard frame sizes. In the meantime, the television screen has changed. Stepping away from the tube technology has allowed larger and flatter displays then ever before. All by all, the technologies have changed but the products all look alike. So what about "panoptics," capturing and displaying images with no restraint on view angles?

7.2.2.1 The Principle of Panoptics

It is clearly time to move from photo camera to a vision-based anything. The camera, becoming small and inexpensive, is turning into the pervasive vision sensor, and therefore entering a number of non-camera products. It is wisdom to forecast what will happen, but we may permit ourselves some speculation here. With enough computing power on-board, 3-D projection techniques are within reach. This allows picture alignment between multiple physical and/or virtual cameras.

Panoramic photography is already an old concept and its theory has been well developed [12]. The principle model of the spherical camera is based on a large aperture set-up giving non-linearly deformed images that together provide for 360° vision. It comes as a single integrated dome that is usually hanging from the ceiling. This model has a price tag above 1000 USD.

The cheap alternative is a camera looking into a rotating mirror. So the camera is fixed and the mirror is turning, because it is light and can therefore be easily rotated.

This is more of a mechanical problem as the camera and the rotating device have to be aligned on one axis. Having the camera integrated into the holder leads to a basic price around 400 USD.

The DIY street view camera does not use moving parts, but simply stacks eight cameras together [13]. Each Microsoft LiveCam NX-6000 has a 71° view. It seems sufficient to take five cameras to obtain a 360° view, but the small overlap is too sensitive to positioning errors. Each of the cameras is hardwired to the notebook. The software is open source: "autopano" identifies common features in different images and aligns them horizontally, while "hugin" stitches them together. The total cost is 300 USD plus notebook.

The Sony Cyber Shot is a camera with moving parts. Angles to 300° can be reached to support for instance over-the-head pictures. But it does not provide auto-moving and can therefore not be used for panoramic pictures. But the camera is becoming as light as a mirror. So the obvious step is to rotate the camera as a whole. Even better, the camera can be a conventional camera phone and can therefore be installed separately. The pedestal can be fabricated separately and has only two functions: rotate and stabilize the camera. Such an object has a price tag of around 30 USD. A tight synchronization between camera and rotation is needed. The camera should know when to capture a new frame, c. q. the rotation has to stop during capture. A current example is the Sony Party Shot, a docking station for two Sony camera types, shaped as a saucer.

7.2.2.2 Digital Portrait

A portrait can be shown on any display, which in the e-world comes in any shape and size. So far a portrait was shaped like a conventional paper holder: flat with a fixed dimension to hold a photograph with a print-compatible format. With the coming of the LED-display, or even e-paper, other shapes and sizes are becoming feasible:

- A round display, like a lamp, but this time steered electronically. This is ideal to show a panoramic view. It would make a clear difference in the house, but it could also become in the first years an ornament of advanced technology in the executive office.
- A convex display to provide a broad picture in a small area. This is ideal to provide the background in shop windows. One may think here of travel agencies.

These examples show that the conventional relation between a 2-D picture and a 2-D display may change, which gives a range of possibilities. The crucial ingredient for this is the dimensional transformation of the image. For showing a flat image on a curved surface, one may consider adding 3-D perspective. Or, vice versa, one may consider a Pixel-Plus operation (see Philips TVs) whereby an image is mapped on a surface with different dimensional constraints.

It is clear that software will play a major role in this area. But it seems unlikely that a picture can be transformed in sequence without loosing quality. So the

2-D image will probably retain its value as storage format for the time being, and displayed images will always start from the stored version.

7.2.2.3 e-Paper/Reader

Mobile devices have a tight energy budget and therefore require displays that are energy-efficient, are readable under any lighting conditions, and have a wide viewing angle. Reflective displays fulfill most of these requirements. These positive qualities of reflective displays should lead to a marketing analysis of reflective display-based products and a study of potential products that could use this technology.

e-Paper technology is evolving very fast, and several companies are working very hard to move to the next stage, which is incorporating color to their respective technology. Many companies are approaching this by adding a color filter layer to their front-plane, while others (like the cholesteric technology companies) must add three primary color layers (RGB) to achieve full color. Fujitsu's rewritable color e-paper is based on highly reflective cholesteric selective-reflection technology.[3] Fujitsu has announced that they have produced a colored e-paper (FLEPia).

Another company that has produced a colored e-paper using e-ink frontplane is LG-Display.[4] Further Liquavista and Advance Display Technologies have colored e-paper in the market. The remaining companies in the e-paper manufacturing business are also on the verge of making colored e-papers.

Although e-paper has been around since the early 1970s [14], its use was not commercially seen anywhere. The factors that hindered its commercial development were its contrast ratio and the fact that microelectronics was still in its early stages. Presently, several products made from e-paper have reached the consumer electronic market and many more are still in their development stages.

The e-Paper screen market is currently dominated by e-ink, which has provided the screen for most of the e-Readers available in the market. e-Inks dominance of the market is so prominent that most people use "e-ink" as synonym to e-paper. In spite of its advantages, e-paper still has a very low refresh rate and therefore cannot be used to build a display for streaming motion pictures. It has problems with color, but (as aforementioned) several research programs are going on around the world to enable color too. Surely, in the coming years electronic devices based on e-paper screen will be very common.

e-Paper does not need a backlight to illuminate its pixels. Rather the letters or images are illuminated by the light from the surrounding environment; just like the conventional paper it reflects light. It uses power only when information is loaded into the e-paper screen. Thereafter, the information will remain unchanged until

[3] http://www.fujitsu.com/global/about/rd/200509epaper.html.

[4] LG.Philips, "LCD Develops World's First Flexible Colour A4-Size E-Paper," http://www.lgphilips-lcd.com/.

replaced by another one. Consequently, e-paper is good for mobile devices because of its minimal power consumption.

The computer monitor like the Cathode Ray Tube (CRT) or the Liquid Crystal Display (LCD) has disadvantages (i.e., backlight effect on the human eye, difficulty of viewing the text or image from a certain angle and pure weight) that cannot be ignored. e-paper solves all these problems because they are lightweight (as light as a paper-card of equivalent size), flexible, and can be bent like a normal paper. e-Paper can be read in a bright sunlight and can be viewed from almost 180°. The minimum requirements for e-paper technology utilization are listed in Table 7.2.

Table 7.2 Some basic requirements for e-paper technology [15]

Requirements for e-papers technologies	
Power	Must be bistable; uses zero power to retain a still image.
Resolution	Minimum of 150 dpi
Contrast ratio	10:1
Viewing angle	More than 100°, preferable close to 180°
Cost	Cheap, the price should be almost comparable to paper
Refresh time	Less than 1 s
Reflectance (brightness)	Should be readable in sunlight, reflectance should be more than 30%
Flexible	It should be bendable
Lightweight	It should be almost as light-weighted as a paper-card

The common denominator of e-applications is the e-Reader. It is a device with a limited number of buttons that allows getting reading material to a person no matter where he/she is and what the local lighting conditions are.

7.2.2.4 Catadioptric Displays

As previously described in Section 7.2.1, panoramic view imaging systems [16] are commercially available. The reverse, a scalable, high-resolution, field deployable, and cost-effective multi-projector approach [17] is feasible and needed. Issues related to image warping and blending need to be resolved in order to build large-scale projection systems. When the sensing system is in motion, there is also need for continuous co-registration of images and the stitching and blending of subsequent images along the trajectory traveled. There is need to obtain as much information as possible from a sensing opportunity. We have described that with the proper optics and packaging design, a single camera can provide images, as well as distance measurements. The general subject area of computational photography has been developed to support and computationally off-load the sensors/cameras.

A modular, field-deployable panoramic-view environment can be constructed out of currently available, low-cost technology. The environment can be built out of

identical self-contained catadioptric[5] projection panels with their respective projectors and reflecting mirrors. Imaging pre-warping and edge-blending processing algorithms appropriate to the panoramic viewer panels are generally available.

As summarized in Table 7.3, there are several distinct approaches to constructing a panoramic viewing environment.

Table 7.3 Panoramic viewer technology tradeoffs

Technology	Advantages	Disadvantages
LED wall	• Bright, high contrast • Mature technology	• Expensive (millions of LEDs) • Heavy, fragile • Power hungry • Complex driver system
Organic LEDs	• Self-emissive • Flexible substrate (potentially)	• Very immature • Short lifespan (∼1500 h)
Tiled LCD panels	• Reasonable cost • Very high resolution Commodity components	• Heavy, fragile • Variations between panels • Hard to join edges • Using commodity panels requires hundreds of video drivers
Single projector	• Simple system • Field deployable	• Limited resolution, brightness • Single point of failure (bulb) • Relatively expensive • Loud and hot • Vibration issues
Projector panels	• Low cost • Field deployable • High resolution • High contrast • Commodity components	• Requires image processing to match panel edges, blend overlaps • Requires video driver per panel

LED panels are in wide use for outdoor advertising and large events. Although they can generate very bright, high-quality displays, they are expensive, power hungry, and complicated. Organic LED displays may help overcome some of these limitations. In particular, organic LEDs can theoretically be "printed" onto a large, flexible substrate – ideal for a panoramic viewer. However, they are still in the research stage and although small panels are currently in limited production, they suffer from short lifetimes (∼1500 h).

LCD panels, once expensive, have become a commodity technology and show a constant cost decrease. Covering a wall with many (relatively) small, cheap LCDs would provide extremely high resolutions at modest cost. Although initially appealing, the disadvantages of this approach are significant. Seamlessly joining the edges of hundreds of panels, and providing hundreds of video drivers, is costly and logistically overwhelming.

[5]Catadioptric projector panels are built with combinations of reflecting mirrors (catoptrics) and projectors/lenses (dioptrics) with their own projection panel.

The reverse of the one-shot panoramic imaging approach, using projection on a single half-ball mirror in a dome or cylindrical environment, has previously been evaluated [20]. In our assessment a single projector, even one designed for commercial cinema, will provide insufficient illumination and resolution. Other problems include cost, heat, noise, and image sensitivity to vibration.

A panoramic viewer environment can be built out of identical self-sufficient catadioptric projection panels with their own respective projectors and reflecting mirrors (Fig. 7.6). This multi-projector approach is scalable (add panels for a bigger environment), high-resolution (for displaying superimposed metadata), field-deployable (simple, lightweight, foldable components), and cost-effective (now that business-class LCD and DLP projectors have fallen below the $1000 mark).

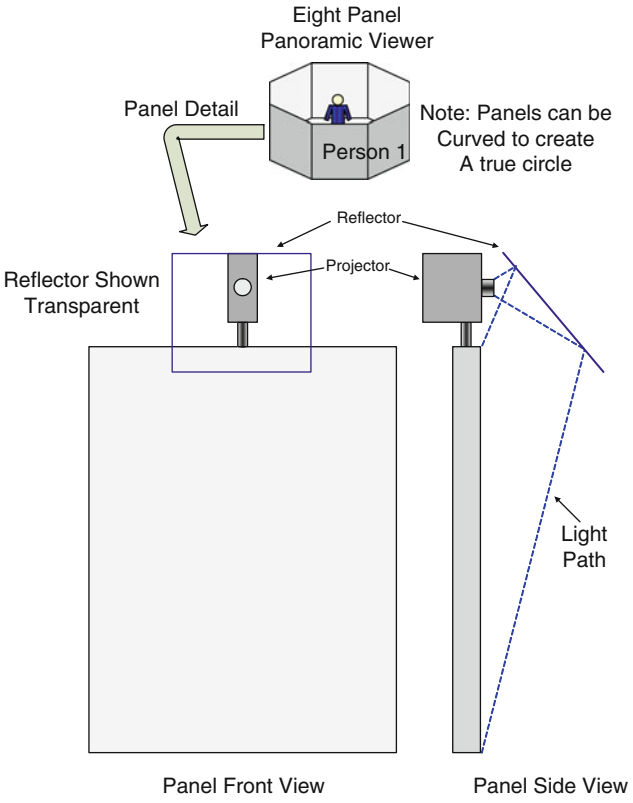

Fig. 7.6 Panoramic viewer concept and individual catadioptric panels

The cylindrical panoramic viewing environment is created by using a series of adjacent projection panels. In this system, each panel has an independent projector and reflective optics. Panels can be flat (for simplicity) or curved to provide a true cylindrical environment. A nominal configuration would include ten $8' \times 6'$ panels arranged in a circle, providing an approximately $20'$ diameter cylinder ($10'$ view

distance) with +/– 20° of horizontal coverage. Employing standard XGA resolution projectors would provide a vertical resolution of 1000+ pixels and a horizontal resolution of 12,000 + pixels, providing an excellent platform for superimposing large amounts high-resolution metadata on top of the panoramic image.

The environment will eventually consist out of a cylindrical assembly of the described projection panels. Each of the panels will be assigned a segment of the panoramic view "donut" for projection. A calibration procedure will be applied to correct for any misalignment of projectors, mirrors, and panels.

The challenge in such a system is in maintaining a continuous, artifact-free image across panel boundaries. This can be accomplished by:

- Calibrating each projector to a video standard (e.g., D65)
- Arranging panel optics such that the image from each panel slightly overlaps adjacent panels.
- Digitally processing the incoming panoramic image to provide each panel with a unique sub-image, which is transformed to match the geometry of adjacent panels and blend with the edges of adjacent projected images.

The heart of a panel-based system is a central video-processing unit, responsible for breaking the incoming panoramic image into separate sub-images for each panel. This video processor will perform the following tasks (See examples in Fig. 7.7).

- Globally scale the incoming image to match the size of the panoramic display
- Pre-warp each image to adjust for optical distortions created by the projection path (e.g., keystoning).
- Adjust luminosity across each image to account for differences in optical path length
- Rotate, warp, and scale each image to align it with adjacent panels
- Blend the edge of images to combine seamlessly with adjacent panels

The basic graphics software required for preparing the image data for projection has been developed under the subject of Digital Image Warping. Being able to use separable (1-D) resampling algorithms will significantly reduce the complexity of digital image warping over the use of matrix (2-D) based transformations.

As an example, Fant [18] has presented a separable[6] algorithm that is well suited for hardware implementation in FPGAs. That method, in principle, is inherently sequential, and originally applied only when the inverse mapping was given. Wolberg has presented another algorithm [19] that is somewhat less suited for hardware implementation, and applied only when the forward mapping is given. A subsequent paper [20] has demonstrated the equivalence of these two algorithms in the sense that they produce identical output scan lines. The paper also removes forward and inverse mapping restrictions.

[6] 2-D operation performed with orthogonal 1-D operations without ill effect

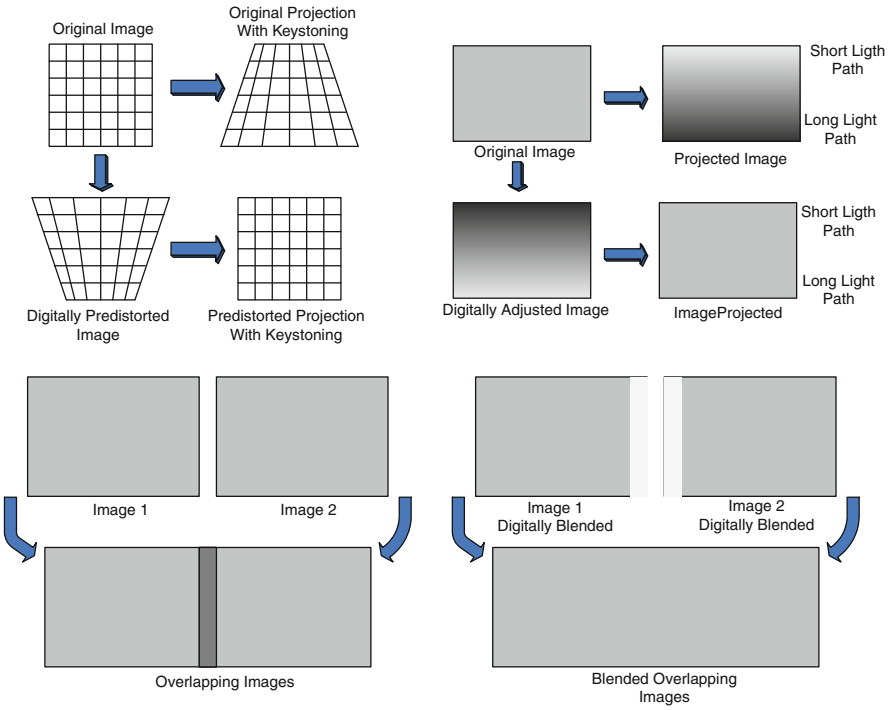

Fig. 7.7 Image processing for multi-panel projection

The Fant algorithm has the advantage of being very simple when implemented in hardware, while the Wolberg algorithm has the advantage of being parallelizable and will facilitate a faster implementation. The Wolberg algorithm also has the advantage of decoupling the round-off errors made among intervals since it does not accrue errors through the incremental calculations required by the Fant algorithm. Algorithms exist and are available for both these approaches.

For edge blending [21], it is suggested to apply the gamma correction algorithm as described and illustrated by Bourke [22].

7.2.2.5 Dome-Projection

The described projection system is a simplified realization of what ultimately should be the complete visualization environment, the dome projection system. Dome projection is either performed from a central location in the dome or the less intrusive back projection system with projector on the outside of the dome. The back projection system will require the previously described patching and blending techniques between the individual projector domains. Dome projection is a method of display that has evolved from flight/trainer simulation systems. It does have exciting applicability for computer games in the home.

7.2.2.6 Hologram

Where the dome is an outward viewing experience, the hologram projection represents an inward viewing experience. The hologram has been used to display people in real time. A typical home application is for birthday parties, dating, etc., where the participants do not have to travel to the same location.

7.2.2.7 Living Wall

In the Living Wall project [23], the wall is covered with paints that is not only decorative but also houses a large number of elementary circuits on which various sensors can be attached to monitor the typical home. The integration of sensors, displays, and control could be the ultimate use of this technology.

7.3 Media Inviter

The principle aspect of a media-inviter is in sharing all facets of multimedia handling, like having persons looking over one's shoulder through the family album. Physical limitations need to be bridged to make this into a concerted action. In the mobile world this leads to interaction between the persons while manipulating the album. The purpose of this section is to identify the technologies that build such a system, the time order in which they may become available, and the opportunity they bring to create modules that can be arbitrarily mixed within a generic set-up. In order to see what this means, we will first go through the different parts of such interaction architecture.

The following case gives some illustration of what we envision.

> While watching television, the telephone rings. My wife calls. Which of the children's photo I like to have framed for the grandparents? As the home network trusts her, she accesses our photo repository and shows her selection in the corner of the TV screen. With my telephone I point to my favorite photo. My wife agrees and says that she will take it from here.

This is the typical situation where the entry, storage, and display devices in a communication are all different, and have to take on different functions. Where the wife starts to call, everything is on her device. Then she sends gesture interpretation to the TV and repository search to the Home Directory. Gesturing leaves a copy on her phone that follows my movements, so she sees what I select. In other words, the virtual caller moves into the home-network, communicates with me and leaves again. Clearly, such a scenario requires a careful design and a raised level of security.

7.3.1 Integrated Media Community

The classical communities that are served by surround vision are websites for house sales, video surveillance, and homeland security. Such applications are typically inspired by the availability of the spheric camera at a relatively high price tag.

Furthermore, they tend to emphasize robustness of the construction. The camera is then used as part of a service by a company that can afford itself the one-time expensive buy. With a lower-cost solution, anybody can do it, for instance, for selling a house privately over the net. But there are more photo opportunities, where the interest is in a group that scrums together. The following are some suggestions:

- Parties: to catch the moment where everybody groups together around the ceremony like giving the marriage vows or cutting the cake. These are also typically the moments, where otherwise somebody would slowly move around with a video camera to get the audience in view while being focused on the ceremony.
- Special happenings: to make a photo when the child is celebrating his/hers birthday "in the circle." But also to make a picture of everybody enjoying the dinner at this table, either in a restaurant or at some resort.

7.3.1.1 Basic Operations

What to do with images? Pictorial information serves two purposes: either it shows an event for further documentation or serves as a means to interact. This does not need the same support but there is a large degree of overlap. Such is especially true where interaction is a search for further documentation. In short, we can distinguish according to the "A2I (A-to-I) model" the following items:

- [**A**ugmentation] The picture is enhanced with further information, similar to hyperlinks to a text file. This is not only interesting for the viewer but also helps search-and-find actions.
- [**B**ackground] The main feature is put into a new scenery. This is similar to the studio photography where different backgrounds can be pulled down by means of a shade.
- [**C**ompose] Parts of the image are replaced for some cosmetic reason, such as for removing shadow. This raises a more general discussion whether the capture of a good image requires a plus-class lens system or just good post-processing software.
- [**D**imension] Transform the image into multiple dimensions, for instance, by mapping on a corridor or a globe. We have mentioned this before as a means to capture gaze, but could also be used for artistic reasons.
- [**E**dit] The usual set of pixel edits.
- [**F**rame] Putting a nice frame around the picture for presentation purposes.
- [**G**eneration] Making and/or supplementing an image with software-generated elements
- [**I**nterpretation] Feature extraction for additional purposes.

Most of these content-based operations are present on modern cameras. However, they do not come for free. Such operations are computation-intensive and therefore require high performance. In this part of the market we find NVIDIA and

similar graphics operation-processing companies represented. A typical example is the recent Sony camera using the GTX280, a device that we do not yet find in the mobile telephony market where the emphasis is on minimal power dissipation rather than maximal performance.

When even this is not enough, cloud computing can be applied. For gaming this type of Software-as-a-Service (SaaS) has recently been proposed (NCsoft's Guild Wars and AMD's Fusion Render Cloud). It aims to work around communication bandwidth problems as the streaming nature allows computing and transferring before the need appears. The advantages are less appealing when the application is not streaming.

Images will be stored for later usage. Unfortunately, it is easier to store than to retrieve. To facilitate retrieval the images have to be annotated. In case of text the annotation will be using meta-words; something similar is needed for images. If a picture is taken of a known person or object, it would be good to be able to search for all pictures where this person or object appears. Again, such functionality is already part of high-quality cameras.

7.3.1.2 The System

The goal of the Media Inviter is to capture and share images within a community without resorting to a central repository. In the existing social communities, textual and video materials are stored centrally and made available over a personalized page. This way the member relinquishes control of the material, but above all he/she is not in directed communication with the other viewers. In the Media-Inviter we go back to the source and create an architecture where the picture is taken in direct communication with other viewers. Earlier in this text we have already given some examples.

A more complicated one is:

> When driving into the car park, I pay the entrance fee using my telephone, download the FAMILY widget and receive tickets for each person in the car. I pass the telephone to Pierrot, my 10-year old, and follow the radio voice guiding me to the nearest parking lot. We get out of the car and hurry to the entrance of the amusement park; I leave my notebook behind in the trunk to lock the car, once we are gone. We want to go to the wild water slide. I look at the 3-dimensional panorama as displayed on my phone. The slide is close-by but there is a 2-h waiting line. So I make a reservation and take the kids first for an ice cream and some animals. It is just a little reroute and Donald Duck will pass there also. And, yes, just as we are licking our ice creams, Donald comes and hugs the little ones: a clear Kodak moment. The sun is right in my face and it is crowded with people, but when I look at the pictures in my telephone they just look beautiful.

A lot of work has already been devoted to connecting devices in the Home (DLNA). In 2008 researchers from Intel came with the DCC concept to connect occasional wireless devices in a situational arrangement. Similarly, we have seen wireless devices to situationally share signage devices. Such arrangements are based on proximity sensing.

The Media Inviter extends on such concepts in two ways:

- proximity is interpreted in social rather than situational sense. As a consequence the community is built through the fixed network.
- The communication is based on available bandwidth by separating between data and control with a first-things-first scheme for heavy traffic on limited bandwidth links.

The principle design equation balances between communication and computation costs (Fig. 7.8). Wireless transmission speed gets steadily higher, but is still smaller than the increase in on-chip speed. Therefore, the balance between computation and communication remains more or less the same over time, though the operating point has shifted. In other words, the distribution of tasks between cloud(s) and devices is a stable one. In a general sense, the mobile will handle limited information, such as control commands and thumbnails, which can be augmented on the cloud with content from other information sources.

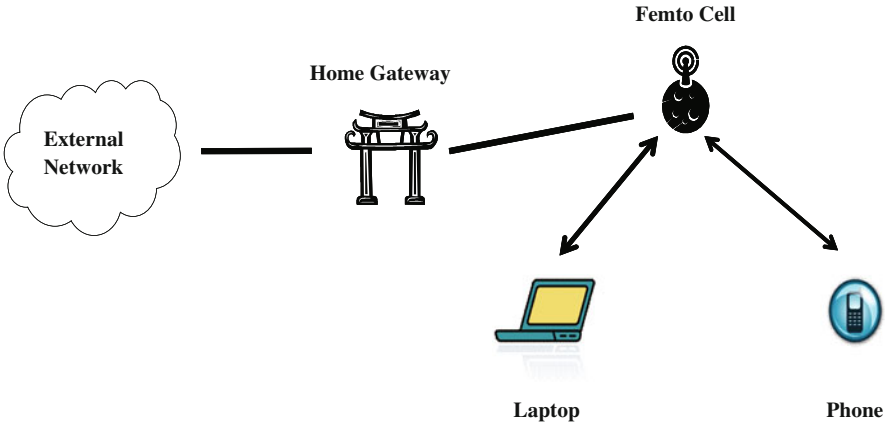

Fig. 7.8 Typical small office/home office (SoHo) structure [34]

A mobile device needs to preserve energy for a long life. This marks the main difference with scientific computing. But even the difference in embedded computing between a notebook and a telephone is to be noted, even though both are wireless and can be used for remote conversation. Usually, the telephone has less resources and less speed, and will therefore be too slow for real-time execution of complex functions. Then there is the issue of wireless communication versus (for instance) wired over glass fiber. The wireless transport rate is limited. Bandwidth can be improved by parallelizing some frequency channels, but this will also come at the expense of other transmissions. So the ideal is to mix wireless and wired transmission. Where wireless transmitter/receiver (T/R) has to be shut down when non-active in order to save power, this plays less of a role in wired T/R.

7.3.2 Home-Inviter Interface

The red line in this story is the fundamental basis of engineering by separation of concerns. This is the mathematical counterpart to the evolutionary approaches exercised in biology. Though lately sincere doubt has been cast on the appropriateness of this choice, the widely accepted Ford-like hierarchical concepts of project development still do not allow anything else. Nevertheless, we will try to indicate where a biological inspiration can make a difference.

7.3.2.1 Sub-ordination

When the design progresses toward further detail, the division between global and local tasks creeps in as assignments to the different layers. The ISO/OSI model for network communication reflects this point of view. Another example is the hierarchical control scheme as introduced by Rodney Brooks. Such schemes to define functionality by collaboration and coordination are fully transparent to the diverse technological means for realization. Hence when the overall design functionality is captured, the balance between software and hardware creeps in as cost balance for construction and maintenance.

An embedded system is like any other computer but not visible to the user, as it is embedded within another encapsulating system. When the users program the computer, they will be interfaced through a keyboard and a screen. They encapsulate the system to the benefit of the programmer. When this computer has to interact real-time with other object, the burden is on the programmer to take this environment into account. This has to be done in advance and/or learned from practice, as there will be little chance for the programmer to debug under operation.

When the computer as hardwired accelerator of calculatory algorithms shares the encapsulation with the programmer, it is totally different for an embedded application. Here, the keyboard and the screen have been degraded to mere options. Overall, the embedded system runs a stored program on sensory values to steer the actuators. Having a single sensor for each independent piece of information comes at a price. It has to be ultimately reliable as otherwise the entire system will fail. Splitting the single sensor into multiple ones, or providing similar information through other sensors, reduces the sensitivity to mishaps. The principle design situation is the network of nodes with non-redundant sensors. It is easier to assemble some of these network nodes into a new single element than to split a complex node into simpler ones.

Most remote devices use infrared connectivity. But if the view is blocked (as by a passing person), the connection is broken. Hence it is interesting to inspect alternatives, for instance speech. Likewise, it is interesting to find entry means that are easily personalizable. Speech offers this in two varieties: (a) speaker-independent, but then often supporting just a small vocabulary of short commands, and (b) speaker-dependent, with a much richer vocabulary and often allowing for a degree of Natural Speech.

Using speech for command purposes brings also some additional consequences, dependent on the degree of safety required in the application. Clearly, the speaker

must be recognized before his/her personal command set can be applied. Even if the speakers identify themselves by other means, the identification must be verified and sometimes even the authenticity must be cleared, before the commands are allowed to take effect.

Such consequences are not unique for speech, but have posed such problems that speech was for a long time no viable alternative. By the coming of better signal processing hardware and software, massive real-time processing became feasible. Limited speaker-dependent systems are already widely used, as a positive development is clearly discernable.

7.3.2.2 Co-ordination

The icon-based desktop has shaped the PC world. Previously, the single command line gave access to a linear list of files and directories. The desktop changed this into cursor pointing into a 2-D window. The worktable with its assortment of paper stacks is imaged on the screen to make the user feel at home. This graphics user interface (GUI) concept for information retrieval has hardly changed over the years, but the growing complexity of the representation has inspired a number of researchers to rattle the chains.

An interesting approach starts from the assumption that though the picturing will be small and hard to read when too much icons must be represented on the limited-size desktop, the actual scope will be limited. By bloating up the icons near the cursor, all within the area of interest becomes larger and better readable. Moving the cursor will behave like moving a loupe, as like looking through a peephole on an infinite landscape. The technique needs just little additional software, while not changing the overall desktop philosophy.

A direct improvement can be obtained by going from 2-D to 3-D reality. An example is the TaskGallery [24] from Microsoft. The icons are placed on the walls, ceiling, and floor of a corridor. Moving through the corridor brings the icons in sight. The closer you come to important objects, the large they are and the more detail they show. This seems almost like real-life, but does not really solve the information retrieval problem. The screen is still 2-D and the additional effect comes from the traversing process that still needs to be click-controlled.

The user is more intuitively directed when the icons represent also a user relation. For instance, in Metaphor the icons represent part of the virtual organization [25]. Moving through the organization one finds documents in the cabinets where they should be stored. This is better than real-life as documents are not the single-copy physical paper stacks, but hyper-linker files. This makes the search easier as documents are to be found in all relevant locations.

All such techniques assume the user to be in front of a display. When the screen is small as in portable systems, they allow for navigating easily through the amount of information that would otherwise require a larger screen. Such a virtualization can also be applied to create screens beyond physical reality. Many behavioral studies are already based on 110° projection, while the fantasy of movies like The Matrix

is rapidly coming into reality. In a sense, this makes the user into a clicking device for navigation: the mouse-made man.

A related question concerns intelligent information mining, a zero knowledge game where the search patterns are learned from the user to create a better support environment. The attention patterns learnt from the virtualized searches are higher dimensional than obtained from a mere clicking device. It can therefore be expected that more intelligent support can be created than with the current Cookie technology.

Where personalized support becomes better, identification also becomes a reality. Identification by keyboard or mouse usage has not been a major success. A successful experiment, ITS MAGIC [26], provides a decorated house, where the user can pay a visit. The system monitors the visitor and his/her behavioral pattern in reaction to a number of objects and incidents to find his real interests and likings. This in turn provides information for direct sellers. Unfortunately, the derivation of this prototype for usage in kitchen selection has not been a success, mainly because the technology was too advanced for the slightly reactionary market of kitchen sellers.

7.3.2.3 Auto-ordination

Hand–eye coordination plays an important role in the human handling of objects. But for complex tasks, the sensory bandwidth is not enough and additional means, such as voice, have to be used. The ultimate complex task is similar to the blind who even have to compensate for the total lack of vision. Also in animals one finds other sensors in use for functions that are handled with humans by the hand–eye system.

Speech is produced by a vibration system that generates bursts of characteristic frequency: the phoneme. This is the atom of speech: the smallest individual unit from which everything is built. In other words: speech can be expressed as a string of phonemes. Which phonemes are used is language-, region-, speaker-, and time-dependent. And a separation into characters and words, as in written/printed text, is not necessarily present.

Phonemes are especially hard to distinguish in a time series. Transforming the signal to the frequency domain solves this, but looses the time order. This is a fundamental dilemma: it is necessary to cut the phonemes in the time domain where they cannot be seen precisely before they can be exactly identified in the frequency domain where they cannot be precisely ordered. At this point the spectrogram can be used. A time-sequenced series of arbitrarily cut bursts are plotted versus their frequency content, colored according to the specific energy of the content at the participating frequencies.

Isolated words are easily recognized from a library of spectrograms. In control situations with a limited number of words, a degree of speaker independence can be expected. Where the required dictionary increases in size, the separation increases and speaker independence may be easily lost. With an interactive scenario where at arbitrary states only a limited choice in answers can be expected eases the separation requirements and makes the understanding system more scalable. Speaker-driven dialing is such an impressive, yet technologically easy, gadget [27].

A next step in complexity is Sign or Picture Language. The ancient Egyptians have already concluded that a message can be conveyed through a number of pictures. In Chinese and Japanese this is taking to the extreme. Where alphabetic language create a great many different abstract composites from a limited amount of characters and leave the relation between words and concepts to the education of the reader, picture languages skips the character as intermediate and have the many concepts as many pictures. This makes the language harder to learn, but once learnt easier to understand.

The ideal is in using a natural language. This is also the most difficult challenge as no deterministic machine has ever been found that can generate sentences in a natural language from scratch. Though much progress has been made, the commercial usage is still largely confined to rule-based interaction situations such as answering services and help desks. Direct translation services are generally still in need of an over-bearing amount of post-processing.

The step from signal to image-processing seems very small. Each pixel can be interpreted as a signal by itself that changes value over time, but even the dynamics of larger picture parts may have a direct meaning. Though it is entirely feasible to detect the composition of an image from the identification of its components, such still leaves to understand the meaning of the image.

7.3.3 Multimedia Home Automation

The current status in Home Automation is dominated by DLNA and PS3. Using DNLA as carrier and Playstation3 format, the amusement center is not confined to a corner in the living room, but is composed from multiple devices across the house with potential archiving over the Internet. A typical system is the MediaMall from 2007, while in the IST program already wireless equivalents were investigated.[7] More recent are Boxee and Sony Bravia. More into the future are DCC and CloudClone of Intel [28, 29]. This shows that we are rapidly moving in the direction of cloud support for location-free personal amusement.

The principle characteristic of the system is the invitation. Within the home environment, identity does not have to be checked severely and a network is simply constructed from the devices in the neighborhood [28]. On the other hand, when the system is merely a matter of the mobile device and cloud access, the usual username/password combination suffices. However, where the Media Inviter extends over the breadth of the Internet, identity becomes an issue.

We solve this by a combination of methods. First of all, there is the combination of channels. In the case described in the introduction, the invitation is send by SMS from mobile to mobile. Once accepted, it is used to create a streaming connection over the Internet in a way similar to a Skype connection. This involves exchange of

[7] 2WEAR Consortium, "A Runtime for Adaptive and Extensible Wireless Wearables", Final Report IST-2000-25286, 2004.

software that checks whether the involved mobile device belong to persons known within the community. Where both the audio, video, and command channels do not cooperate, the session will not start and the migrated software is removed. In essence, the communication during the session is only admitted between known devices.

In the more complicated example given in Section 7.2.1, all the local cells will only handle known devices. As these cells are the strongest, all call attempts will go through them and only valid calls will be handled. Therefore, no calls to outside can be made through wireless links. Moreover as all mobiles are active, they will be observant and detect any unauthorized newcomers. This is a typical swarm situation where the collective intelligence is larger than the sum of the parts. Where a number of cameras are taking pictures of the same scene, we can refrain from 3-D stitching within the single one, but leave it to the cloud. A number of pictures come from surveillance cameras or are prestored as "best quality." Other pictures that are more typical of the family situation emanate from the mobile devices, but each being responsible for one only. Again the cloud will stitch them in proper order, but the transport to the cloud has not been compromised.

With the requirement to move software in place comes the need for distribution. In the network we have a multitude of sensors and devices, and in hardware we get a network of cores. Polytopol computing brings these areas together by introducing an abstraction of the network topology, enabling computation to be executed on the network rather than on the nodes. It is here that the new field of function migration comes in and provides together with polytopol computing the infrastructure that enables flow of functionality on a topology. This hides the network structure and therefore the location of software functionality. Enabling technologies will be the skeleton injector as a replacement of the task scheduler, the polytopol computing topology abstraction of the network, virtualization applied on the connection of software to software, and the introduction of a connection-set architecture (see Chapter 2 for more details).

7.3.4 Multimedia Home Security

Security and safety have become major concerns in modern society. All kinds of devices, such as card readers, palm readers, and iris scanners, have been introduced to control access to sensitive areas, ranging from countries to public places and small offices/homes. Once inside a controlled area, surveillance has to check on proper conduct, as most of the security breaches come from trusted persons. Damage prevention is the ultimate goal.

Visual monitoring and surveillance attract much attention not only due to increasing tension between privacy and crime prevention but also due to technological hurdles that need to be overcome. In a typical surveillance application, the number of cameras and the amount of captured video can be so large that a human operator cannot handle all the information from multiple input channels. Moreover, it is difficult for humans to watch video streams for a long time and remain able to analyze the data, detect all suspicious events, and respond to them in a timely fashion without missing anything.

The key challenge in this area is to extract relevant information from the scene, identify probably suspicious behavior, and direct the attention of security personnel to the situation for further investigation. Unfortunately, it can be hard to automatically distinguish between normal and abnormal (or suspicious) events. In a place where everyone walks running is an unusual behavior, while in places where everyone runs walking is unusual. In our work abnormal events are defined as events which are not predictable due to time, condition, and location of their occurrence.

7.4 Futuristic Home Environment

In this section we integrate technology gained from applications described in this book with requirements for a futuristic cloud-centric, sensory-network-based living environment. As usually, "if we build it, they will come." The current iPhone and iPad are great examples of devices that are creating great demands for technology solutions, and that not necessarily were asked for by most users.

7.4.1 The Intelligent Ball

The following story envisions futuristic scenarios that are perfectly realizable within currently available technologies. It combines real-time sensory devices, displays, multimedia, and high-performance computations for data analysis, interpretation, and advising:

> Lisa is a new student at LTH. It is her first day at the university and she is a bit nervous but excited. Her fellow students have told her that she should not buy course books; she should just go to the Study Center to study. So she goes there. As she enters the building, the information desk is to the right, but what to ask? It feels all so strange and confusing! Suddenly she feels a soft ball bouncing against her leg. It is pink and fluffy. She bends down, picks up the ball and looks around to see who has thrown it to her. The ball then opens two kind brown eyes, smiles and says: "Hi, I'm Bobo. Can I help you with anything?"
>
> Lisa is very surprised, but before she has time to think how stupid it is to talk to a ball, she says: "I have never been here before and I am looking for course books." "Welcome," says Bobo. "What course do you attend?" Lisa tells Bobo what course she wants. "If you put me down, I will show you were the books are," says Bobo.
>
> Lisa puts Bobo down and the ball rolls away. Bobo stays close to Lisa and makes sure they do not loose each other. A small lamp on the shelf lights up, when they come to the shelf, where Lisa's course books are. "You do know that there are electronic course books as well?" Bobo asks. "Oh, I didn't know that" says Lisa. "Where do I find them?" "You can talk to the blue ball in the corner over there," says Bobo. "Do you want me to call it for you?" "No thanks" says Lisa. "I'll talk to that one later. Thanks for your help." "You are welcome," says Bobo. "When you want a guided tour of the building just talk to any of the balls, or call me. What's your name if we meet again?" When Lisa tells Bobo her name, Bobo says Hi and rolls off.
>
> Lisa feels somewhat confused, but she assumes that talking balls are something natural since she studies at a technical university. When she looks around she sees balls in all different sizes, colors and textures. Some students play around with a small yellow ball,

which happily cheers. Two students sit comfortably on a ball with leopard markings and study notes from a lecture. In a chair next to a shelf a guy is comfortably seated deeply concentrated on a book while at the same time absent-minded stroking a black and white, purring ball.

Lisa looks at the books on the shelf, they look very dull. She decides to go somewhere else. At that moment the blue ball approaches her. "Hi Lisa" says the blue ball in a nice voice. "Have you forgotten the electronic books? Would you like to see them now?" Lisa looks surprised at the ball. Oh yes, the electronic books. OK, hopefully they are more interesting than the printed ones, she thinks. "Yes please," says Lisa. The ball rolls in front of Lisa and its top part opens. You can see the back of the books that are Lisa's course books.

Lisa cautiously touches one and it opens. She turns the leaves over in an absent-minded way and thanks the ball for its help. "Do you want to register these books to your account?" asks the blue ball. "If you do that, you have access to them on all blue balls at the Study Center and in the LTH houses." That sounds like a good idea to Lisa and she checks out the books by stroking the ball.

In a sofa a group of students are sitting with a ball in front of them at a table. They are engaged in a lively discussion with the ball and they draw things on a piece of paper. When Lisa approaches she sees that the ball has tiny arms which draw things on the paper and that the ball explains something to the students. Suddenly someone calls her name and Lisa looks around, trying to find a ball. But it is Kalle, one of her fellow students. They decide to go to the cafeteria.

At the adjacent table professor Larsson is deeply engaged in a conversation with a ball in a cow fur. "I told you I wanted articles about invertebrates living in the sediment and their influence on phosphorous and nitrogen retention. Do you really mean this is all there is?" "Excuse me, mister Larsson, I thought you meant electronic articles only," says the ball. "I will rerun the search. Wait a minute." Professor Larsson shakes his head, takes a bite of his mazarin (cake) and waits. "What do you think about this?" says the ball and shows the search results. "Hmm. . .." says Larsson. "Show me the pictures of the authors. Mmmmm . . .. Take away the articles of Tranlund. He is an idiot." "OK," says the ball. "Do you want me to remember that Tranlund is an idiot?" "Do that, and read me Johannessons article and send me the rest," says Professor Larsson. "At your service," answers the ball and starts reading the article in a very pleasant voice. Larsson leans back in his chair, takes a delightful sip of his coffee, closes his eyes and listens carefully.

Kalle and Lisa look at each other, shake their heads and smile. "Are there no people working here, only balls?" Lisa asks. "If the balls cannot help you, they will call on a nice lady to come and help you," says Kalle. "Have you seen the hand yet?" "What is the hand?" Lisa asks. "If you try to take books out of the house, litter or try to destroy things, then the hand will come and chase you until you ask for forgiveness in public, and clean up the mess you made. Do you want to see what a ball looks like inside?" "OK," Lisa says. They leave the cafeteria searching for a chequered ball, which is willing to be taken apart. They find one, take it apart and manage to re-assemble it after three attempts. Then the ball starts singing: "We are the champions."

7.4.2 Complexities of Operation

At this point in the book, we have reviewed technologies and concepts related to the connecting of intelligence-gathering sensors into the computing cloud. In this

chapter we have looked at the home and near-home environment as an antipode (or opposite) network-wise of the server farms in the cloud.

In essence both domains, the server farm cloud and the sensor cloud, reflect a similarity in elasticity. In the server farm, we do not know where the actual computations take place and actually we do not care. In the sensor intelligence-gathering domain, we have seen a similar elasticity if one would follow the concepts described in Chapter 2 of this book. As a reminder, in that chapter we described that in, for instance, ad hoc networks of cell-phones we do not have to insist that all computations are performed locally and in situ.

Having reviewed the home-environment and its concomitant advanced technologies to be applied, we can list various home-centered applications as follows (Table 7.4, in no particular order) with their respective estimates of compute complexity:

7.4.3 HPC-from-the-Cloud

All through the book, we have described many applications of intelligence gathering in various environments. As it turns out many of these can also be applied in the close-to-home environments. We have summarized these technology correlations in Table 7.5.

Most of these applications will require computational powers, larger than that are commonly present in most sensor assemblies, even with multi-core processing.

In order to satisfy the processing requirements of the things we have described in this book, be they mobile, fielded, or ultimately in the home, they will require supercomputer capabilities. The High Performance Computing (HPC) community has developed systems at a relatively high price, but with not necessarily universal access. Cloud computing, based on server farms, will make HPC-like capabilities economically accessible from the home and the university.

One of the thoughts expressed in cloud-thinking is one of democratically bringing compute power closer to the application/users. One has gone as far as developing the notion of High-Performance Computing (HPC)-in-the-Cloud. This would entail the availability to the home owner/manager of super-computing capabilities simply by acquiring compute nodes access with a credit card charge. An example of an HPC-in-the-Cloud realization can be found in IBM's approach to Deep Computing.[8]

An alternative would be the expansion of the concepts of network extenders, the femto-cell, the pico-cell into a home-server or community server concentrating compute power before access would be made to the far (Internet) away server-farm cloud. For instance for the applications in Table 7.5, O(10) applications

[8]IBM Deep Computing Institute (DCI), also HPC-for-a-Smarter-Planet and The Smarter City

Table 7.4 Home-apps complexity estimates

Home-apps	Description	Complexity (order of # of compute clients)
Home media	The presentation and control of various media displays, sound, and information	1–10 real-time
Communication	The infrastructure required for secure information communication in the home, as well as for external communications	1
Smart grid	The control of the proper usage of power in the home, as well as the production of potential power from the home into the external power grid	1–4 Dependent on size home
Security	The protection of home assets, controlling the going in and out of home inhabitants	1–10 real-time
Homework	The management over the performance of homework of children, as well as their access to Internet data needed for completion of homework	1–4 Dependent on size home
Training	The management of compute, display, and camera resources for the purpose of education, professional, or serious training	1–4 Dependent on size home
Entertainment	The management of compute, display, and camera resources for the purpose of entertainment of the various inhabitants of the home	1–10 real-time
News	The display of externally produced news items	1
Bills	The support of billing for services and the payments to external agencies	1
Toys	The management of toys behavior, as well as the control of access to the toys	1–4 Dependent on size home
e-Stock	The support of e-stock transactions, as well as the tracking of external investment opportunities	1–10 real-time
Mail service	The management of e-mail services, as well as spam, fishing controls	1
Inventory	The management and tracking of marked (RFID) resources in the home, as well as the display of appropriate summaries of content and shopping lists	1–4 Dependent on size home
Insurance	The management of inventories for the purpose of establishing sufficient insurance levels	1
Computer repair	The management of compute resources, in the home and/or cloud, as well as the establishment of correct and sufficient operations	1–4 Dependent on size home
Thermostat	The measurement and management of environmental well-being in the home	1–4 Dependent on size home
Travel planning	The support for planning of trips out of the home, includes information, mapping, and warnings for external negative experiences	1
Patient monitoring	The observation for care, comfort, and concern for needy inhabitants of the home, the management of external warnings	1–10 real-time
Babysitting	The observation of babysitters and sittee's, the management of external warnings	1–10 real-time
Cleaning robot	The management of automated support services in the home	1–10 real-time

Table 7.5 Cloud apps for the home

Reviewed sensor network (Cloud apps) applications (in order of appearance in the book)	Selected for which home (Ground Apps) Applications	Relative compute complexity
Gesturing system	Home media, security, training, entertainment, toys, patient monitoring, babysitting, cleaning robot	8–68 ~ O[9](100)
Games for learning	Home media, security, homework, training, entertainment, toys	6–42 ~ O(50)
Socio-metrics	Bills, inventory, mail service, thermostat, patient monitoring, babysitting, cleaning robot	7–34 ~ O(25)
Motion detection	Home media, security, training, entertainment, toys, thermostat, patient monitoring, babysitting, cleaning robot	9–72 ~ O(100)
Voice recognition	Home media, communication, homework, security, training, entertainment, toys, thermostat, patient monitoring, babysitting, cleaning robot	11–76 ~ O(100)
Vital sign monitoring	Security, thermostat, patient monitoring, babysitting, cleaning robot	5–44 ~ O(50)
Security warning	Communication, security, inventory, mail service, insurance, computer repair, thermostat, patient monitoring, babysitting, cleaning robot	10–52 ~ O(50)
A-Social behavior	Security, patient monitoring, babysitting	3–30 ~ O(25)
Environmental monitoring	Smart grid, thermostat, patient monitoring, babysitting	4–28 ~ O(25)
Weather forecasting	News, thermostat, travel planning	3–4 ~ O(10)
Intelligent road management	News, insurance, travel planning	3 ~ O(10)
Smart power	Smart grid, security, bills, inventory, thermostat, cleaning robot	6–32 ~ O(25)
Collision avoidance	Training, entertainment, toys, cleaning robot	4–28 ~ O(25)
Motion tracking	Home media, security, homework, training, entertainment, patient monitoring, babysitting, cleaning robot	8–68 ~ O(100)
Face recognition	Home media, communication, security, homework, training, entertainment, bills, toys, e-stock, inventory, mail service, patient monitoring, babysitting, cleaning robot	14–86 ~ O(100)
Hacking alert	Communication, security, bills, e-stock, inventory, mail service, insurance, computer repair	8–22 ~ O(25)
Ticket issue	Security, travel planning	2–10 ~ O(10)
Media presentation	Home media, communication, homework, training, entertainment, news, travel planning, babysitting	8–38 ~ O(25)
Camera support	Home media, communication, homework, patient monitoring, babysitting	5–34 ~ O(25)
Office tools	Home media, communication, homework, training, news, bills, e-stock, inventory, mail-service, insurance, computer repair, travel planning	12–36 ~ O(25)

[9]O – Order of

can clearly be performed at the home, the O(25) in a home-server, the O(50) in a community server, and the O(100) at a higher level toward the server-farm cloud. Therefore, in the home many of the described applications are realizable, but only realistic when realized in the cloud computing context.

The effect of both of these approaches will be the making available of compute power without the home-owners/managers investment in compute resources and investment in a complete software development environment (including update/upgrade and configuration control). Just like notions in virtual memory, we can envision a notion of cloud-stacking. In that notion a wealth of new concepts in optimization, elasticity, virtualization, etc., will come about. Note that in the cloud-stack the server farm will be the low-cost side of the computation tradeoff spectrum.

If we define cloud computing as server farms with some level of elasticity, we can make a similar argument that the local network of intelligence-gathering sensors when exhibiting elasticity can be called a cloud, a local sensor cloud. These sensor clouds will operate on the opposite site of the Internet from the server farms (Fig. 7.9).

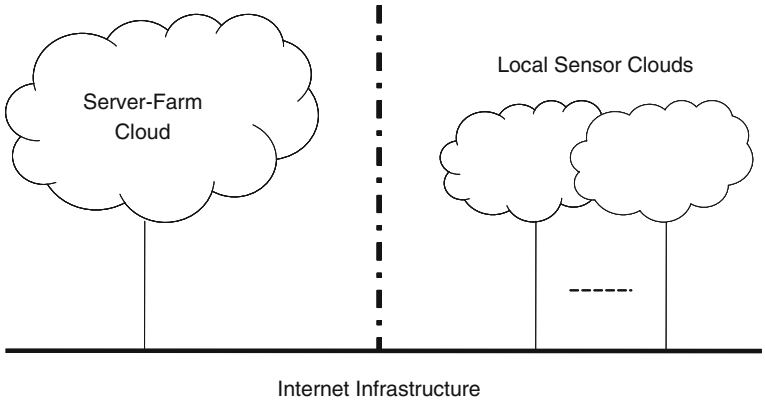

Fig. 7.9 Clouds on both sides of the internet!

In essence, even the Internet itself can be thought of as a cloud (within its non-deterministic packet switching as a form of elasticity). Because of its elasticity, the cloud concept is for all practical purposes a new computational abstraction level.

At the server farm cloud a service catalog can be established with various performance-cost tradeoff points. It is then up to the cloud end-user to select the applications, the "apps" (just like on the iPhone) at the appropriate cost-value points. A choice will also have to be made between cloud-apps and ground-apps, applications that by necessity have to run in the cloud (e.g., because of access to other data) and those that can be run in the local sensor cloud(s) or any of its local home or community servers.

7.4.4 Green Computing

All these approaches have as a beneficial side effect that compute-system-wise – that is inclusive of server farms and sensor clouds – the use of cloud computing will result in a smaller energy consumption footprint. Green computing [30][10] will be realizable in the Cloud Computing context.

Ultimately, if powerful enough (with multi-core processing), the local sensor clouds could become resources in a larger cloud context. This is similar to extraneous locally collected solar energy being fed back into the (smart) global power grid. So again we have gone full circle, from local to global to local. This notion of distributed (cloud-wise participating) sensing [31] will complement the other forms of distributed availability in computing (cloud and grid), storage, and even in bandwidth (BitTorrent peer-to-peer).

7.5 Summary

Sensory systems will appear everywhere in society, so why not at Home?! The embedded parts will be pervasive, but that does not really matter. What we are looking for is "functions" and these will in plain sight. Every thing in the house will be at the tip of the finger, though hardly ever part of the furniture. But you cannot see them because they are simply created on what is there with the help of the virtual pawn shop: the cloud.

The important lesson from cloud computing is the fact that software must be runnable on any device. In the current mobile technology, every application has to be adapted to any Operating System/Hardware platform. Therefore, the user is limited in choice, once he/she made his/her device selection. In Home Automation, typical standards are evolving because the product diversity is much larger. In a typical Media Inviter you will find both mobile and amusement equipment. Consequently, the standardization must be pushed further and include the cloud domain, or migratable software has to be introduced to break the platform dependence.

But a note of warning is due. As is already proving in the area of Industrial Automation, danger lurks in the standardization corner. Industrial Automation was happily using optimized, supplier-based communication schemes. Then, for the simple reason of design, fabrication, and maintenance costs, one moved over to existing, general-purpose standards. As a consequence, all the mischief that was created on the Internet became applicable and suddenly it was as easy to hack a power generator as it is to hack a bank account, or even the car computer [32] (the Electronic Control Unit).

All this will be just as easy for the Web-based Home. Therefore, the main criterion for a good system architecture will be safety. This will be not as a life saver as in automobile, or a service guarantee as in industrial automation, but simply as

[10] http://www.greentouch.org

a must. And the ubiquity and flexibility of cloud computing are efficient ways to achieve that. It creates a degree of dynamic diversity, which can be selected as high as a good encryption, but stays pervasive, similar to what has been advocated as the only way to keep a secret in the "Noise Society"[33].

References

1. Dutta-Roy A (December 1999) Networks for homes. IEEE Spectr 36(12):26–33
2. Hart K, Slob A (1972) Integrated injection logic: a new approach to LSI. IEEE J Solid State Circuits SC-7:346–351
3. Wilburn B et al (July 2005) High-performance imaging using large camera arrays. ACM Trans Graph 24(3):765–776
4. Rodriguez-Vazquez A et al (2008) The Eye-RIS CMOS vision system. In: Casier H et al (eds) Analog circuit design. Springer, Dordrecht, pp 15–32
5. Kleihorst R, Abbo AA, Choudhary V, Broers H (2005) Scalable IC platform for smart cameras. Eurasip J Appl Signal Process 2005(13):2018–2025
6. ter Brugge MH, Nijhuis JAG, Spaanenburg L (1999) License-plate recognition. In: Jain LC, Lazzarini B (eds) Knowledge-based intelligent techniques in character recognition. CRC Press, Boca Raton, FL, pp 263–296
7. Sundström P (2005) Stereo vision with uncalibrated cameras. MSc. Thesis, Lund University, Lund, Sweden
8. Bouguet J-Y, Perona P (January 1998) 3D photography on your desk. In: Proceedings of the 6th international conference on computer vision (ICCV1998), Bombay, India, pp 43–50
9. Lu M, Weiyen W, Chun-Yen C (2006) Image-based distance and area measuring systems. IEEE Sensors J 6(2):495–503
10. Sundström P (2005) Stereo vision with uncalibrated cameras. MSc. Thesis, Lund University, Lund, Sweden
11. Simmons E, Ljung E, Kleihorst R (October 2006) Distributed vision with multiple uncalibrated smart cameras. ACM workshop on distributed smart cameras (DSC06), Boulder, CO
12. Meers N (2003) Stretch: the world of panoramic photography. A Rotovision book, Switzerland
13. Ragsdale RD (October 2009) DIY street-view camera. IEEE Spectr 46(10):20–21
14. Liang RC et al (2002) Microcup electrophoretic displays by roll-to-roll manufacturing processes. IDW Publ Forums, EP2-2:1337–1340
15. Evans J, Haider S (February 2008) E-facts. Masters Thesis, Department of Electrical and Information Technology, Lund University, Lund, Sweden
16. Geyer C, Daniilidis K (2001) Catadioptric projective geometry. Int J Comput Vis 45(3): 223–243
17. Bourke PD (November/December 2005) Using a spherical mirror for projection into immersive environments. Graphite (ACM SigGraph) University of Otago, Dunedin, New Zealand
18. Fant KM (January 1986) A nonaliasing, real-time spatial transform technique. IEEE Comput Graph Appl 6(1):71–80
19. Wolberg G (1990) Digital image warping. IEEE Computer Society Press, Los Alamitos, CA
20. Wolberg G, Sueyllam HM, Ismail MA, Ahmed KM (2001) One-dimensional resampling with inverse and forward mapping functions. J Graph Tools 5(3):11–33
21. Raskar R, Welch G, Fuchs H (November 1998) Seamless projection overlaps using image warping and intensity blending. In: Proceedings of the 4th international conference on virtual systems and multimedia, Gifu, Japan
22. Bourke PD (May 2008) Low cost projection environment for immersive gaming. J Multimedia (JMM) 3(1):41–46

23. Buechley L, Hendrix S, Eisenberg M (February 2009) Paints, paper, and programs: first steps toward the computational sketchbook. In: 3rd international conference on Tangible and embedded interaction (TEI′09), Cambridge, UK
24. Robertson G et al (April 2000) The task gallery: a 3-D window manager. In: Digest conference on human factors in computing systems (CHI2000), The Hague, The Netherlands
25. Offenberg MAH, Gonsalves V, Spaanenburg L (May 2000) Monitoring the dynamics of an organization. In: Proceedings VDI Fachtagung Computational Intelligence, Baden-Baden, Germany, pp 387–392
26. Buist R, de Graaf J, Wichers W (1996) ITS MAGIC: design and implementation of an intelligent interactive tele-shopping application for the electronic highway. MSc thesis, Rijksuniversiteit Groningen, Groningen, The Netherlands
27. van Veelen M et al (1998) Speech-driven dialing. In: Proceedings 3rd international workshop NN′98, Magdeburg, Germany, pp 243–250
28. Want R, Pering T, Sud S, Rosario B (2008) Dynamic composable computing. In: Workshop on mobile computing systems and applications, Napa Valley, CA, pp 17–21
29. Chun B.-G, Maniatis P (2009) Augmented smart phone applications through clone cloud execution. In: Proceedings 12th workshop on hot topics in operating systems (HotOS XII), Monte Verita, Switzerland
30. Kurp P (October 2008) Green computing: are you ready for a personal energy meter? Commun ACM 51(10):11–13
31. Shilton K (November 2009) Four billion little brothers? Privacy, mobile phones, and ubiquitous data collection. Commun ACM-52(11):48–53
32. Koscher K, Czeskis A, Roesner F, Patel S, Kohno T, Checkoway S, McCoy D, Kantor B, Anderson D, Shacham H, Savage S (May 2010) Experimental security analysis of a modern automobile. In: The IEEE Symposium on Security and Privacy, Oakland, CA
33. Lundblad N (February 2008) Law in a noise society. Doctoral Dissertation, IT-University of Gothenburg, Sweden. Published as Gothenburg Studies in Informatics, Report 41
34. Awad J (December 2008) Design challenges ahead for media gateway. Electronic Engineering Times, 15 Dec 2008

Chapter 8
Afterwords

Cloud Computing is for sale on the corner of the street. It is the ultimate lease concept. Nothing needs to be owned and everything is for rent. Unfortunately, as the occasional Denial-of-Service (DoS) attack already shows, the system will break down long before 60 billion people demand an instantaneous 24/7 service. As meanwhile the number of Web-based appliances is exploding, the cloud is clearly getting overcrowded. In an attempt to satisfy this demand, we could easily all get a cloud in the back garden.

8.1 Cloud

In historical perspective, it is reasonable that cloud computing is not the end of an era, but the beginning of a new one. In the exploding ICT universe, we regularly see new "six Watson computers"[1] emerge at the heart of and then dissolve into the universe. If history repeats itself, we should see the cloud slowly cloning itself. Getting closer to the clients, it will pick up more business, which in turn stimulates a further dissolvement. Surely this is likely to happen, but it does not take the special technologies into account, which have made the cloud to happen. Let us review this again.

- Virtualization. Once compiled software is always run! Versions are of less importance. Android is the latest development. Launched for the mobile telephone market, it is rapidly becoming the operating system of choice for embedded systems. The Android TV is on the doorstep. and I hear the footsteps of the Android percolator (the one that hears me asking for coffee) already in the street.
- Homogeneous Architecture. The "more than Moore" gap indicates that the algorithm complexity is growing faster than the processor speed. Actually, the clouds are more in the data storage. Having the data warehouse in the cloud resolves the

[1]The original words of Thomas J. Watson (the founder of the International Business Machine company) that there will be a world-wide market for at most six computers is still passed on among computer scientists.

L. Spaanenburg, H. Spaanenburg, *Cloud Connectivity and Embedded Sensory Systems*, DOI 10.1007/978-1-4419-7545-4_8,

issues of ownership and maintenance, making it cheaper for the data owner to operate a service. The added beauty is that the need for high-speed interconnect is mostly within the cloud ("the first mile").
- Heterogeneous Architecture. Algorithms are not easily parallelized. Research into multicore architecture has indicated that systems with many heterogeneous nodes are better able to handle the lack of inherent parallelism. In the cloud we will therefore find servers with different resource sets, each optimized for different usage, so that together they can handle the differing requirements of the requested service.
- Configuration. Running the cloud involves moving the services to the most optimal server and/or combining services to run the task load on a minimal amount of servers for reason of energy efficiency. Consequently, a service request will over time be handled differently, depending on the current load. Except for time-critical software, the user is not required to have an intimate knowledge of the scheduling opportunities.

8.2 Sensor

What does all this mean for sensory networks? That depends as not all application areas are in the same development phase. In some, the Internet is just being introduced, while others are still dominated by proprietary concepts. We assume here an Internet-based infrastructure and see then:

- Virtualization. The microcontroller of old is hard to virtualize. Therefore, the decade-long development of the microcontroller has to continue to encompass also this characteristic. The alternative is to use modern microprocessors, but make them low-power. In the end, they will be able to accept guests that run processes that not necessarily use the integrated sensor. In other words, the sensor becomes a computing node with one or more special resources.
- Network. In sensory networks, the sensors are at the edge laboring under bad working conditions in the mine producing a lot of data. The cloud concept assumes that such data are transferred in their raw form to the store in the cloud for further refinement. The tendency in sensors is toward smart devices. These refine the data-stream into a feature or a value. The rationale is that this brings a severe data reduction as demanded by the low communication speed "at the edge." The consequence of being autonomous to this degree may however be that there is little gain to be made by moving to sensory systems.
- Heterogeneous Architecture. In sensory networks, the nodes will by nature be computing nodes with one or more special resources, called sensors. Each sensor is meant for a specific measurement, but the signal may easily contain influences that in the overall sensory network may be used to come to a virtual measurement. Consequently, such systems can be cost-optimized by shutting off the more

expensive nodes, permanently or temporarily "faking" the measurement by a virtual approach. We may call this "sensor consolidation."

- Redundancy. Having different mechanisms to perform the same detection and/or measurement is a clear example of redundancy. With redundancy in place, it becomes possible to use many cheap sensors instead of the single expensive one. The consequence is a much higher dependability or even lower fault sensitivity. In turn this facilitates unsupervised operation, where a sensory network can provide quality information on an isolated or far-away place.

8.3 And Everything in Between

It appears that technologies are coming which can be applied at the edge, as well as at the heart of the universe. But the question remains what happens in between (Fig. 8.1). Somewhere sensors become computers, and then become clouds. We pose here that on such levels, we will see redundancy move into pluriformity. This is the typical domain of what is usually called the Internet.

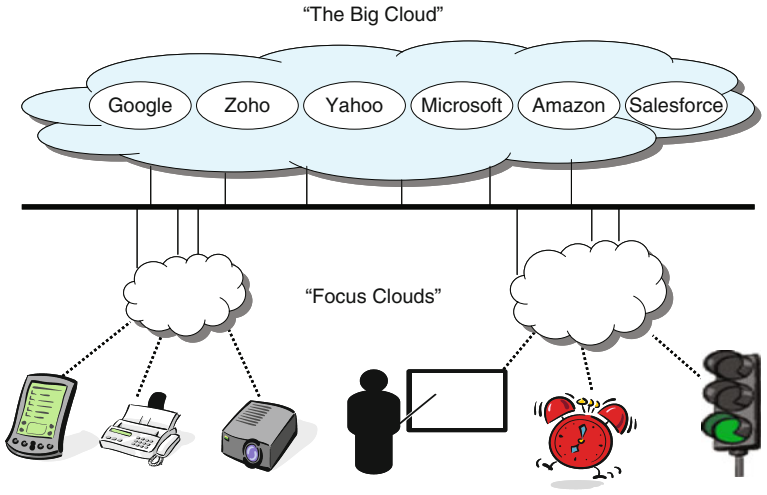

Fig. 8.1 The new clouds between sensory networks and server farms

A typical example is in music. There are many copies of popular music floating around through the Internet. The problem is not to find a copy, but to find a copy without cookies and other sorts of mischief. It shows that we want to find things in a safe way.

As another example, let us assume that we want to bring together free versions of handling images, video, speech, print, and so on to create a multimedia function. For each of them there are many varieties that all perform about the same function. There will be differences in performance, quality, and service level. Hence there is

no simple merge. On the other hand, not all different varieties will be used in the same amount. Consequently, when we aim to support everything, we put in a lot of effort for nothing.

The way out of this dilemma is also shown on the Internet: one is enough! Make a local cloud that support one or a few specific combinations. Basically, you should implement what you need. Then stimulate other people to do the same according to their taste. Then the usual proliferation takes place and gradually local clouds will appear with interesting combinations. Then for your specific application, you just select the cloud you want.

This brings the basic trick back to crawling or searching. In contrast to Google, who goes through the Internet and provides a one-shop-stop, this process deals only with peer-to-peer asking for the right combination. The usual complication is of course when you cannot find exactly what you want. Here one can think for the "combiner-in-the-sky," the overall cloud function that is able to do everything.

8.4 Panta Rei

The principle argument of this book is that system[2] functionality will be based on virtualized hardware and software, making the function the primary design object. Such functions are developed as part of the system architecture and carefully mapped on the provided network, primarily using IP hardware and software. Though we have used words like "hardware" and "software," the intention is that in the coming years such notions will gradually disappear into niches of the embedded sensory world.

[2]Everything Flows, Heraclitus' thoughts about change.

Index

L. Spaanenburg, H. Spaanenburg, *Cloud Connectivity and Embedded Sensory Systems*, DOI 10.1007/978-1-4419-7545-4,