Version 1.0, date 14-03-2007

Project Number:33826

CREDO

Modeling and Analysis of Evolutionary Structures for Distributed Services

Deliverable D6.1
User driven requirements

Due Date: 14-03-2007
Submission Date: dd-mm-yyyy

Start date of project: 01-09-2006 **Duration:** 3 years

Lead Participant name here **Revision:** Draft

Project funded by the European Commission
within the Sixth Framework Programme (2002-2006)
Dissemination Level: PU Public

Project Participants

Role	No	Name	Acronym	Country
CO	1	Stichting Centrum voor Wiskunde en Informatica	CWI	NL
CR	2	Universitetet i Oslo	UIO	N
CR	3	Christian-Albrechts-Universität zu Kiel	CAU	DE
CR	4	Dresden University of Technology	TUD	DE
CR	5	Uppsala Universitet	UU	S
CR	6	United Nations University, International Institute for Software and Technology	UNU-IIST	JP
CR	7	Almende B. V.	ALMENDE	NL
CR	8	Rikshospitalet - Radiumhospitalet HF	RRHF	N
CR	9	Norsk Regnesentral	NR	N

C0	=	Coordinator	CR	=	Contractor
NL	=	Netherlands	N	=	Norway
DE	=	Germany	S	=	Sweden
JP	=	Japan			

Document History

Principal Contributors:

Names	Affiliation
Ilangko Balasingham	RRHF
Marcel Kyas	UIO
Wolfgang Leister	NR
Xuedong Liang	RRHF
Bjarte M. Østvold	NR
Anne van Rossum	Almende
Alfons Salden	Almende
Martin Steffen	UIO
Jeroen M. Valk	Almende
Enter name here	CWI

Revision History:

Version	Primary Authors	Description of Changes	Date
0.1	Tom Chothia	Template creation	16/10/2006
0.2	Bjarte M. Østvold	Minor syntactical changes	05/02/2007
0.2.1	Bjarte M. Østvold	Proposed master document structure	05/02/2007
0.3	Bjarte M. Østvold	Clarified the division of labour	20/02/2007
0.4	Jeroen M. Valk	Included: CS 1 (Sections 3 and A), Section 5	22/02/2007
0.5	Alfons H. Salden	Reviewed: Sections 1, 2 and 3	22/02/2007

Contents

1 Summary

Dynamically reconfigurable systems which involve a large amount of interacting processes are difficult to manage in a coordinated way. This specifically holds if component updates and network reconfigurations are allowed to be carried out by local authorities without any centralized coordination mechanisms. Modern societal forces, e.g., globalization, are demanding next-generation ICT systems that allow dynamic system reconfigurations even without centralized service level agreements (SLA).

This document presents the high-level user requirements of two case studies concerning the design and implementation of next-generation ICT systems as described above. The first case study concerns the ASK community system: a system that supports coordination and communication of human resources. ASK consists of a collection of distributed components that interact using a variety of network technologies. Each component represents and/or maintains a simple autonomous functionality: e.g., a component could represent a single user in the system or could be responsible for finding an appropriate responder for a given request. The second case study concerns biomedical sensor networks (BSNs). A BSN consists of small, low-power and multi-functional sensor nodes that are equipped with biomedical sensors, a processing unit and a wireless communication device. The equipment in a biomedical sensor network may be produced by many different manufacturers. The challenge is to provide adequate service level agreements for critical health surveillance applications.

After a brief description of the technical details and use-cases of the two case studies, a high-level description of the user requirements for verification and testing tools developed in the Credo project are given. Both case study descriptions provide a clear idea of the modelling requirements that next-generation dynamically reconfigurable systems should satisfy. These requirements provide a solid guidance in the development of Credo tools: they help specifying the corresponding technical Credo tools requirements .

2 Introduction

Despite many technologies for system- and application integration and management (e.g., IBM MQSeries, MS Biztalk, service-oriented architectures) the integration and management problem is far from being solved. Due to increasing globalization, organizations must be able to cope with ever changing environments, and in their role in a supply chain, become more dependent on other players. As a result, interorganizational coordination and cooperation becomes more important. This not only holds for the activities of people, but also for ICT. Many technological innovations find their way to the market: intelligent sensors and actuators (e.g., surveillance cameras, road sensors, medical equipment), communication technology (e.g., mobile phones, PDA's), etc.

All these technologies can be used most effectively if they can co-evolve and interoperate in a coordinated way. This is a difficult task however. A component from a single manufacturer is often thoroughly tested and sometimes partly verified using formal methods (e.g., assert statements, model checking). But if combined together in a heterogenous network of components, verification of the components alone is not sufficient. Especially, if multiple interactive processes are active, it becomes difficult to maintain sufficient quality of service. Reasing about quality of service delivered by interacting components is a challenge even if, at an abstract level, interactions remain relatively simple.

Traditional verification techniques identify “normal” patterns of interaction, e.g., in the form of assert statements. Correctness then consists of verifying that all components behave normally. Here, it is assumed that all components are able and willing to cooperate. In dynamic environments, however, this assumption no longer holds. Due to, e.g., software errors, unexpected situations can occur which may cause errors to propagate throughout the whole system. To cope with error propagation, components should not only minimise their own errors, but must also be able to correct errors in the environment.

The application domains presented in this document have dynamic properties in that both the network and the components should be allowed to change while applications are running. Runtime reconfigurations are incompatible with current techniques for the verification of service levels. Verification and testing has traditionally been applied offline. Environments where components and network connections are allowed to co-evolve and reconfigure at runtime, however, call for online verification. Using runtime tests, a component can constantly monitor its interaction with the environment. Formal verification allows a component to monitor the quality of other components in the vicinity. These testing and verification results can, for instance, be used to generate feedback for the environment. This allows components to adapt their behavior in order to meet the appropriate quality standards.

This document describes two concrete case studies in which the above considerations are important. Case study 1 concerns the ASK community system. The ASK system organizes and coordinates human communication and activities. The system is said to support self-organization by humans and systems within an organization in favor of more hierarchical management of business processes. To achieve its goal, the ASK system is highly concurrent, communication resources are represented by agents which interact in a dynamic way.

Case study 2 concerns biomedical sensor networks (BSNs). A BSN consists of small, low-power and multi-functional sensor nodes that are equipped with biomedical sensors, a processing unit and a wireless communication device. Typically many different sensors are involved in patient monitoring. The challenge is to collect and process the information from all sensors in a coordinated way. Usually, this is done by collecting all data at a central place for processing. But there is an ongoing trend to distribute intelligence over the network. Systems for early warning of heart attack, for example, process data at the sensor node and should, in case of alarm, trigger the appropriate qualified personnel.

3 Case study 1: ASK community systems

This section describes the ASK community system to be modelled in Case study 1, and outlines high-level requirements for the modelling process. We start by giving a technical description of ASK in Section 3.2.1. Next, in Section 3.2, we describe some possible scenarios in which the ASK system can be applied. Finally, Section 3.3 concludes this section about Case study 1 with an overview of the requirements that the CREDO tools have to satisfy.

3.1 Technical ASK description

The ASK community system is a distributed system for connecting people or for providing automated response services (e.g., interactive voice response IVR and DTMF). Thereto, it performs management of distributed and heterogeneous concurrently active organizations, application services and communication network services (e.g., ISDN, GSM, VoIP, email, sms). In particular, it does so by dynamic composition of highly reconfigurable component-based software systems, communication channels and its application components in line with service level agreements existing amongst organizations and service providers. Changes in organizations, application component services and communication network services, however, impose very high demands on the dynamic reconfiguration capabilities of the ASK system. Those changes may involve new constellations of organizations, increase in the number, customization (differentiation e.g. personalization and integration) of applications and communication network technologies, and the opening up of existing platforms like MSN, ICQ, AOL messenger. Evidently, the CREDO compositional modeling and verification techniques have to meet stringent high-level requirements in order to support sensible dynamic reconfiguration of the ASK community system. In the sequel we further detail use cases of ASK community systems that ground those high-level CREDO requirements.

3.1.1 ASK components

The ASK community system connects people and systems in the “right” way by means of a message-based system. The functional architecture of the ASK community system consists of five components (see Figure 1):

Reception. The Reception is responsible for handling incoming calls. When somebody calls for a certain service, the user gets connected to the reception. This component tries to acquire information about the caller, his request, etc., through, e.g., an IVR (Interactive Voice Response). The reception then sends a request for service to the Matcher.

Matcher. The Matcher is responsible for connecting the “right” persons within the ASK community system. Based on information provided by the Reception, the Matcher has to come up with a list of the best available

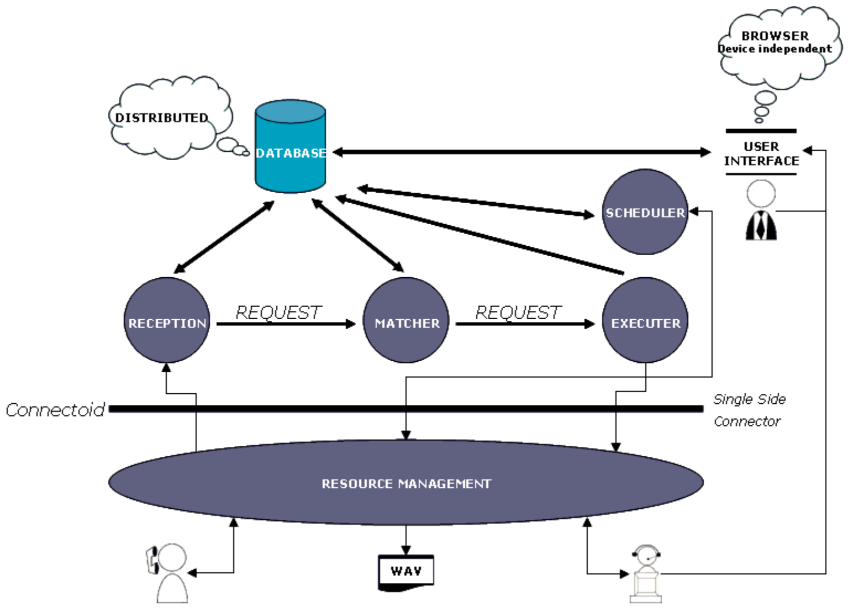


Figure 1: Overview of the ASK architecture

persons to answer this specific request. The matcher sends the candidates on this list to the Executer.

Scheduler. The Scheduler is responsible for maintaining availability information and collecting feedback from the user. The component keeps track of the agenda of responders and knows when somebody is available and through which communication channel. When the availability level drops below a certain threshold, the scheduler actively starts calling potential responders to ask them to be available for the moment. Additionally, the scheduler actively retrieves this feedback after a service has been performed (responder to requester). This information about the success of a connection is used by the Matcher to organize the next connection in the system.

Executer. The Executer is responsible for handling outgoing calls while taking into account availability of the callee. The Executer calls the best available person and connects him/her with the requester. Availability information is obtained from the Scheduler.

Resource Management. The Resource Management has the responsibility to handle every type of communication channel such as telephone, mobile phone, VoIP, email, sms, etc.

3.1.2 The user interface

There are several users involved in the phases of design, implementation, deployment and usage of the ASK system. Hence, most often more specific terms than "user" will be used, to indicate their roles. The person using the ASK system is the caller. The person writing the ASK software itself the programmer. The person constructing the IVR (Interactive Voice Response) menus the menu constructor. The person designing ASK, the ASK designer.

The caller communicates with the ASK system by a computerized system, IVR (Interactive Voice Response). This system allows the caller to navigate through a menu by selecting options from a voice menu by entering numbers. Which options are available to the caller and which sound bits are played when a caller chooses an option, is defined by the menu constructor. The menu constructor is guided with his task by the "menu constructor interface", see figure 2.

The menu constructor interface highlights functionality that exist under the hood in the ASK system. The caller has - for example - an option to press the buttons 1, 2, 3 or 6. Each option depicted by a coloured box. Within each option a certain action is taken, that can differ from connecting to somebody, listening voicemail, to asking for feedback. It is also possible to jump to another IVR menu. The subitems in other menus are not addressable. The menu can be regarded as a finite state machine that can only be linked to others using its start state.

3.1.3 Generic model of the abbey

The mentioned components, Scheduler, Matcher, Executor, conforms to a specific model, that is called an *abbey*. An abbey is a *coordinated* community of *workers* where each worker can be understood as a separate thread of execution. In the abbey paradigm are those workers called *monks*.

There are several flavours of abbeys thinkable. The one that underlies the mentioned ASK components, is very limited. It can be described in terms of what it does:

- There is a fixed set of task slots available. A new activity is dispatched by calling a dispatch method with as parameters: a pointer to a function and its context/parameters (and optionally a description). The dispatch function searches for an empty task slot, and stores the received data over there;
- There is a fixed set of threads in a thread pool that all run a "monk" method. This method contains an infinite loop where the monk inspects all tasks that are waiting to be executed. One of those is (not randomly) picked and executed: the function evoked with context as parameter;
- The dispatch method and the monks all try to access the same task array. This task array is therefore protected by a mutex. The ?dispatch thread? does not have a higher priority than a "monk thread".

The ASK designer is, however, not that much interested in the current specifications. It is more important to sketch the directions in which robustness, and other characteristics yet to be sketched can be improved. Hence, it is more important to describe what an ASK abbey currently lacks, with respect to:

- Reusability: There is no communication between monks. So, there is redundancy in behaviour. There is no monk interface, so it's undefined how to speak against a monk.
- Performance: There is no specialization of monks tailored to certain tasks. So, there is no performance gain of monks executing on hardware that fits better with respect to a certain task.
- Performance: There is no coordination mechanism. Monks cannot delegate tasks to other monks. Monks cannot cooperate, they just work independently, hampered by the existence of other monks, not profiting of it.
- Interoperability: There is no communcation with external entities. So, e.g. no coordination with monks in other abbeys.
- Reliability: There is no entity that encaptures responsability. Each monk is allowed to perform whatever task. There are no escalation mechanisms to warn about malicious tasks or malfunctioning monks.
- Adaptivity: There is no dynamics in the amount of threads. There is a fixed amount of monks and there are no mechanisms that observe how many monks are actually needed.
- Emergence: Tasks are pointers to C-functions that do not have return values. Tasks do not have a behavioral description that the monk just implements in its specific way (using a DNA in the form of instructions). There is no competition between monks performing worse and better.

Again, it would be extremely valuable to test the consequences of certain policy decisions with respect to e.g. the definition of the monk interface, types of escalation mechanisms, types of thread dynamics. In what sense do they increase reliability, etc.

3.1.4 Resource Management

The Resource Management communicates with the Reception, the Matcher and the Executer. In the Resource Management a specific abstract concept, namely the connectoid is covenient. A *connectoid* is the software equivalent of cable plugs and jacks. A connectoid can be attached to a sound file at one hand, and to a listener at the other hand. A model that respects loose channel ends, makes it easy to tackle reconfigurations in the network.

For each hardware or software system used for communication, like the (sound files on the) file system, the Asterisk telephony toolkit, upto a Teles

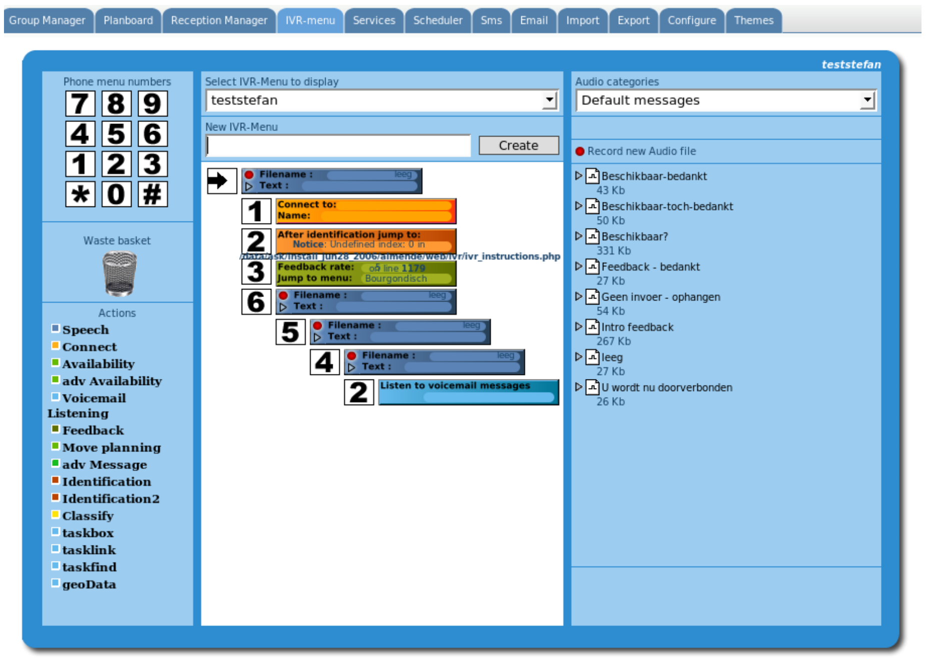


Figure 2: ASK Menu Constructor Interface

ISDN card, there is a module - say a *Hardware Equipment Operator*- within the Resource Management that knows how to create, connect, disconnect and destroy connectoids of the proper type. The main module in the Resource Management is able to connect connectoids across the entire system. Within each connectoid a protocol is embedded that they use to negotiate. A connection between connectoids is always possible, but when certain constraints specified by the mentioned protocol are not fulfilled, no data transmission will happen.

The Resource Management communicates with the Reception, Matcher and Executer by receiving requests. A *request is a C struct* that contains information like identifiers, status fields, requester and responder specifics, time stamps, etc. The requester and responder are C structs themselves that exhibit different fields depending on perspective. The same field may be a phone number in a phone context, an IP number in a VOIP context, an email address in an email context, etc.

Design decisions in regard to threading are delegated to each Hardware Equipment Operator. So, for example the Asterisk Operator may use threads for every call, while the Teles Operator uses only one or two threads to handle all calls.

Requirements concerning the Resource Management take the form of:

- Functional and temporal demands with respect to coupling different hardware communication equipment within restricted time limits and providing sufficient feedback in the case modules are coupled that are not conform;

- Communication load demands that pose restrictions upon the information carrier (like the mentioned request struct);
- Modularization demands that need flexibility in distributing (parts of) the Resource Management;
- Anticipation demands that need flexibility in regard to the way for example requests are defined. The Resource Management should gracefully accept changes to defined formats.

3.1.5 Request, menu operator and virtual user

The Request object is the entity that is communicated between the Resource Management, Reception, Matcher and Executer. It has the following structure (using BNF syntax: [] means optional, | means OR):

1. request := general_info requester responder reception_address disconnect_reason timestamp
2. general_info := request_id request_cycle_id request_status {etc}
3. requester := participant
4. responder := participant
5. participant := user_info | service_info | virtual_user_info
6. reception_address := phone | voip | email | scheduler | chat
7. disconnect_reason := reason role keys_pressed connectoid_id
8. virtual_user_info := menu_id

The Request is generated by the Resource Management, that in that sense functions as a Request factory. The Reception receives the Request from the Resource Management. It examines general information, like the address of the requester and the responder. There are situations in which the request is "recycled" through the entire ASK system. For example when the Scheduler decides to call out to a person to receive feedback, it sends an appropriate signal with appropriate information to the Resource Management. The Resource Management creates a new Request with the aforementioned information in the disconnect_reason struct. Subsequently (part of) the entire chain of Reception, Matcher and Executer will process this new Request object.

The Menu Operator and the Virtual User cooperate to let the user navigate through the menus. Each of them is a state machine. The Virtual User has states that are similar to that of a normal user, that navigates through a menu.

The Menu Operator has states that correspond with each box in the menu. For example a sound box, that plays a sound. Or a connect box that connects to a person or service. Or a message box that sends a message to a voicemail or email address. Within each box are states and transitions defined that specifies the sequence of actions to be undertaken. A real receptionist would replace the Menu Operator as well as the Virtual User.

3.1.6 Reception

The Reception will be explained in detail. The Matcher, Executor and Scheduler are designed alike. The Reception has a main method: It initializes workspace folders, database settings, port settings and an abbey, and it establishes a port on a socket.

A task "reception hostess" is launched. In the reception hostess task the socket is continuously polled until a tagged entity is retrieved. This tag can be of an acknowledgment, request and identification type. In the case of a request, the load is mapped to a request entity. And subsequently a new "handle request" task is launched.

In the "handle request" task the request entity is inspected. In the case of an "disconnect reason" one of the ASK components, like the Matcher or Executor, was very disappointed about the previous request, and demanded the Resource Management to create a new request. The reason about this behavior is encapsulated in the "disconnect reason" field.

The requester type is inspected. If it's a request from the Virtual User, it can involve voicemail. If it is a Service request it can concern a call that didn't got answered, that is stored and marked for feedback. Subsequently the different "disconnect reasons" are inspected. The reason can differ from: new call initiated, key pressed, phone hangup to phone not answered. In the case of a new call the "service group type" tells us about the used medium or initiator type. It can concern a voip, phone, email or scheduler group. Obeying those requests often the Virtual User becomes running again, for example to call out (initiated by the Scheduler). Or the Virtual User is put into a new (reconnect) state, while the Menu Operator remains in the same state (of playing a sound e.g.).

Communication from the Reception to the Matcher, or whatever entity is performed by changing things to the Request. Setting its state to "archive me" puts it somewhere in a database. Setting its state to "recognized" indicates that it can be eaten by another component. After the tasks have been executed, the request is send to a method that opens a socket and sends a tagged entity with the request to the appropriate component using the request state.

The interface of the Reception therefore contains two channels over which tagged entities enter and leave. Certain tags indicate that what kind of actions have to be performed. The most important communicated entity is the Request struct.

It is important for the ASK designer:

- To keep track of the request trace. Can it not disappear accidentally?
- To appropriately reformulate Reception code into (an right amount of) tasks;
- To be able to change the model of the Reception while keeping the old Matcher, Executer, Resource Management intact;
- To verify compositions of multiple Reception's on different machines;
- To change the type of channels used between ASK components from local to remote, from lossless to lossy, from synchronous to asynchronous, from fifo1 to fifoN.

3.1.7 Matcher, Executer, Scheduler

The Matcher does have - like the Reception - the following tasks. The task "matcher hostess" receives like the "reception hostess" messages from a socket. The task "handle request" handles the request. Depending on the participant, a real participant, a virtual user, or a service participant is matched. The user address (e.g. telephone number, email address) is retrieved from the database. On failure the Virtual User is targeted again, and put into a new (error) state. The actual matching routine searches for a match in the same (service) group, that is available, using an algorithm from feedback rating, round robin to last spoken.

The Executer and the Scheduler contain also those tasks. Although there functionality is entirely different, that what they actually do can be seen as a side-issue. They communicate with each other using the mentioned Request object. Interests of the ASK designer concerns (besides the same compositional and intercommunication points as for the Reception) different matching policies, different scheduling policies, different couplings between service groups.

3.2 Use cases for ASK

In our modern society, interorganizational interaction, communication and coordination becomes increasingly important. Organizations become more dynamic and constantly have to coordinate their activities with other players in the supply chain. ICT systems are struggling to keep up with these new ways of doing business. Integrated ICT solutions that are based on unified information sources (e.g., enterprise resource planning (ERP) applications) cannot handle the increased complexity of globalization.

As integrated solutions are reaching their limits, ICT must be organized in independent applications each covering only a single business aspect. This seems like a step back to the old pre-ERP days where each department used its own application. Only human intervention could make these isolated applications interact with each other. Humans are very flexible to translate information from one system to another without having to rely on a comprehensive understanding of all possible contingencies.

ASK provides a communication platform that supports human communication and makes human machine interaction more easy. The systems aims to accomplish this through a collection of distributed components that communicate with each other using UDP sockets. Each component is only concerned with a specific functionality which makes it easier to change aggregate system behavior in order to adapt to user preferences. In Section 3.2.1, we set out how we intend to achieve this desirable situation.

ASK never stands on itself however. The system aims to support human communication in order to better coordinate business processes. Business processes typically involve both people and ICT. Therefore, ASK not only needs to communicate with people but also with ICT systems. This is where we enter the realm of system integration: a topic which we will briefly discuss in Section 3.2.2.

ASK components are intended to offer global coordination of many ASK communities worldwide. This is called multi-ask: an idea that will be described in Section 3.2.3.

3.2.1 Dynamic ASK components

Currently, the five components mentioned above are distributed in that every component is represented by its own unix process. The components communicate using UDP sockets and they communicate via the database. Each component listens on the socket for incoming requests. In principle, these requests could come from different other components. However, correctness of the system can only be guaranteed if certain restrictions are satisfied. E.g.: only the Reception sends requests to the Matcher; the Executer only receives requests from the Matcher, etc. As a result of these restrictions ASK is always deployed in a fixed configuration with only one instance of each of the five components.

However, the components in the ASK system has been designed to be highly composable. This allows the ASK system to adapt, e.g., to the communication infrastucture, or to specific user requirements. If, for example, an Asterisk PBX and a TELES ISDN card would be available then two resource managers would be set up to represent these two communication resources. If one logical ASK application is assigned the exclusive use of these resources then a single executer would be configured to interact with the two resource managers. The Executer could do load balancing or select the resource that is best for providing the desired quality of service.

The above scenario is far from current practise however. If ASK components would be set up to freely communicate with each other then the complete system would become unstable. It could no longer be predicted what the aggregate behavior would be. We believe however that this can be changed if the interfaces between the components are better defined. The communication technology currently used is too heterogeneous. We can identify at least three forms of communication. UDP sockets, SQL queries, and interaction via the file system.

3.2.2 System integration

Dealing with contingencies is something which traditional software is not very good at. System integrators only allow applications to interact if their combined behavior can be fully understood. From this perspective the step to ERP was a logical one: introducing a common view on all business processes and replace all systems which are not compatible. The advantage is that ERP allows different departments to coordinate, but at the expense of forcing change on perfectly organized local activity.

Even more modern approaches suggest addressing the interoperability problem by introducing a comprehensive understanding of everything that is communicated between systems. A good example is the semantic web which aims at the development of shared ontologies and languages for expressing knowledge. These languages should allow data and knowledge to be presented in a formalized way that is understandable and processable by machines. However, not everything need to be understood in a network of interacting components. If something is not understood in one part of the network it might become useful when passed on to another part of the network.

To allow our systems to interact in a dynamic and flexible way, much research in computer science aims at new methods for building software: e.g., multi-agent systems, self-organization and self-management. Gradually, these research ideas find their way to ICT practice. A good example is the concept of service-oriented architectures (SOA). SOA offers a new approach of organizing software in terms of services that communicate with each other. Currently, web services are frequently used to provide the communication technology for interconnecting services. SOA brings the idea of building distributed interactive applications; web services provide open protocols and standards to run these applications on machines owned by different organizations; web services provide open protocols and standards to run applications on machines owned by different organizations. However, no thorough guideline is provided by SOA about how interactive applications should be build. Often only simple predictable interaction patterns are implemented to maintain control over the application. This issue has been recognized by Gartner Inc. and has led to the idea of event driven architectures.

Event driven architectures (EDA) support complex event processing that can be found in operating systems and system management tools. From a scientific point of view, EDA can best be compared with a trend in computer science to recognize interactive computing as a new paradigm.

Interactive computing is a new understanding of computation which allows interaction with an external world during computation. This is fundamentally different from the traditional definition of computation and computability in terms of Turing machines. The Turing machine model is only concerned with the question what computability of functions means. In other words, a simple interface is assumed: the Turing machine asks a question (its input) and generates an answer (the output). During computation, which may take an unbounded amount of time, the Turing machine is disconnected from its environment. Interaction machines are different in this respect: while computing they have a (possibly unbounded) ongoing interaction with their environment.

Interaction machines are more powerful than Turing machines. Interaction opens up the possibility to harness the complexity of the environment to produce complex input/output and internal behavior. Although this has been recognized by Alan Turing himself[1], it was recognized only recently that adequate models of interactive computing are necessary. Not only theoretically do we need such adequate models of computation, but there is also a strong practical need. At the moment, many companies are including web services into their ICT infrastructure to support interoperability. In this respect, web services are only an enabling technology. System integration requires more advanced SOA techniques like EDA, or self-organizing agent technology.

3.2.3 Multi-ASK

Dynamic ASK components not only aim at providing robust and redudant support of the technical communication infrastructure, but they also aim to support the way human communication and coordination is organized. It is impossible to represent human communities worldwide in a single ASK system. A single ASK system is typically designed to support a community with respect to a single service. Examples of such services are: request for a repairman, registering for a tennis competition, mobilizing a fire department in case of an emergencency. In practise, humans are often involved in multiple of such services. Moreover, the activities of different services must be coordinated; e.g., adquate emergency response requires the fire department to cooperate with ambulance personnel and the local police.

It is the intention that multiple ASK systems may coordinate the activities of many partially overlapping communities. This can be achieved through dynamic reconfiguration of the existing ASK components and possibly adding new components that are the responsiblity for new inter-ASK coordination functionalities. Typically, one call to multi-ASK might invoke several services at the same time in order to mobilize the required people. ASK should navigate a responder in such a way through the available services that the right responder(s) can be found.

Clearly, cooperation of overlapping ASK systems may lead to conflicts. More ASK systems for the same person may connect to a different activity at the same time. Due to the dynamic nature of the creation of ASK systems and their communities it is not possible to predefine the overall system behavior under all circumstances. Conflict resolution should be made possible using a mechanism providing the necessary combined awareness and tools to augment the situation.

[1]In 1939, Turing showed that Turing machines with oracles are more powerful than Turing machines without oracles. The oracle allows the Turing machine to harness an incomputable sequence of events.

3.3 High-level requirements

Normally each component in the ASK community system will contain software system and communication channel relicts that have no formal programming language representation at all. Consequently, ASK community system implementation errors, their types and their relations due to those relicts cannot be predicted and will be hard to handle in upcoming releases.

Sustaining the ASK community system will be hampered besides by those legacy problems also by evolutionary pressures. The ASK community system will have to serve different organizations becoming ever more complex. It has to do so by aligning individual people, according ever more detailed preference schema, by means of ever more advanced applications and ever diverse communication or collaboration means. Furthermore, the ASK community system service will need to be heterogeneously distributed and concurrently be provided in a decentralized way by a coalition of (hardware/software) components or agents that interact with each other in a message-based environment. Last but not least, both current and future ASK community systems require a variety of protocols (e.g., TCP/IP, SQL, HTTP), APIs and scripting code (e.g., PHP) for communication or collaboration. This makes it difficult to exactly pinpoint the behavior of the ASK community system with respect to its environment. As a result, programmers hardly understand the ASK community system and experience changing the system to be very cumbersome.

The legacy problems and evolutionary pressures related to the dynamic adaptation of ASK community systems, i.e. runtime composition of communication channels and run-time update and upgrade of related application specific adaptable software components, imply that the CREDO system, in particular its compositional modeling and verification techniques, has to satisfy the following requirements.

3.3.1 Usability

The users of the CREDO system - mainly technical engineers - need not have a profound understanding of formal programming languages. Therefore, the CREDO system should comprehensibly support modeling and verification of dynamic reconfiguration of a new ASK community system:

- The CREDO system interface should allow users automatically assessing the effectiveness of a single ASK community system (configuration) developed and deployed by the user, based on whether entities that have been tested to work in isolation still work correctly in a newly composed ASK community system in which the entities are interconnected.
- The CREDO system interface should allow users automatically observing the efficiency of a single ASK community system (configuration) developed and deployed by the user, based on a representation of non-functional aspects of the running isolated network entities.

3.3.2 Interoperability

The CREDO system should have an intuitive visual interface for building ASK community system configurations. Not only should it be possible to create these configurations from scratch, but it must additionally be possible to import existing ASK community system component specifications and implementations that use different specification languages. These languages may vary from industrial programming languages to more academic formalisms.

3.3.3 Scalability

The CREDO compositional and verification techniques should be scalable with respect to an increase in the complexity and dynamics of the ASK community systems. One may distinguish many not necessarily independent types of scalability issues for ASK community systems:

- Load scalability ensuring neither delay nor exaggerate resource consumption or contention by the ASK community system at various levels of workloads by distributing and scheduling in a proper way concurrent tasks over available resources.
- Space scalability ensuring that the storage memory requirements of the ASK community system do not grow to intolerable levels as the number of objects it supports increases.
- Space-time scalability ensuring ASK community system performance as the number of objects it encompasses increases by several orders of magnitude; indexed data structures and system components could sustain ready operation irrespective system size.
- Structural scalability ensuring growth of the number of objects that a ASK community system encompasses at least within a chosen time frame.
- Distance scalability ensuring a system - an algorithm or protocol - to perform well over long and short distances across IP/TCP.
- Speed-distance scalability ensuring a system to perform well over long and short distances at high and low speeds across IP/TCP.

In practice this means that the CREDO system should provide computationally efficient techniques that enable tractable and robust compositional modeling, verification but also prediction of those various ASK community system scalability aspects at various levels of aggregations or scales. This implies that the CREDO system should also be able to reason about uncertainties due to incomplete or unknown formal specifications of legacy subsystems or acquired test data from the ASK community systems.

3.3.4 Reliability

The CREDO system's reliability concerning its ability to handle uncertainties in distributed object and interface specifications due to system and environmental dynamics is important in particular for large scale systems like ASK community systems. This implies that the CREDO system should provide very sophisticated compositional modeling functionalities that are capable of handling structural and behavioral uncertainties. With respect to the verification, the speed and robustness of the CREDO system in finding valuable information is much more important, then 100% correctness of this information.

3.3.5 Inspectability

The CREDO system should allow for concurrent validation, verification and performance assessment of ASK community systems. This holds in particular for use of ASK community systems in critical application domains, like healthcare and defense and security.

3.3.6 Real-time operation

CREDO system should operate real-time in order to meet ASK community system service level agreements (SLAs) at run-time. This manageability heavily depends on CREDO system scalability features and ability to provide up to date performance indicators concerning running and alternative ASK community system configurations.

3.3.7 Sustainability through Self-organization

The CREDO system should sustain ASK community system service level agreements despite the presence of evolutionary structures for supporting distributed services. This sustainability heavily depends on self-organizing capabilities of the CREDO system in learning how to validate, verify and analyse performance of the ASK community system in changing environments. That is, the CREDO system should be capable of checking large-scale distributed evolving ASK components for specific changes in functional requirements. The CREDO system should be capable of analyzing non-functional requirements such as performance, robustness and scalability of ASK components. This analysis provides a basis for recommending alternative management strategies; in particular when dynamic characteristics and types of communication channels and software components change over time.

4 Case study 2: Biomedical sensor networks

This section, and subsections, describe the application domain to be modelled in Case study 2, and defines the requirements for the modelling process.

The goal of the sections is to give user-driven requirements for the case study on biomedical sensor networks—Case study 2 of *Credo*. We start by giving a technical background on biomedical sensors and networks in Section 4.1, and then we present specific use-cases for such network relevant to Case study 2 in Section 4.2. The actual requirements appear in Section 4.3.

We introduce the domain of biomedical sensor networks in some detail. We give requirements mostly as informal text, but we supplement with numeric values for certain parameters, for example, time and bandwidth. While eliciting requirements we have been concerned both with the needs of the users and with what the *Credo* project can deliver. Since *Credo* has as one goal to build tools based on research we consider both these concerns to be relevant.

A biomedical sensor network (BSN) consists of small, low-power and multi-functional sensor nodes that are equipped with biomedical sensors, a processing unit and a wireless communication device. Each sensor node has the abilities of sensing, computing and short-range wireless communication. Due to BSNs' reliability, self-organisation, flexibility, and ease of deployment, the applications of BSNs in medical care are growing fast. BSNs are used in a variety of scenarios in health care. Common scenarios are described in more detail in Section 4.2.

Light-weight and battery-operated sensor nodes could be worn by patients or elderly people who need continuous care. Physiological data and vital signs such as body temperature, blood pressure, ECG, S_PO_2 and heart rate are sensed and transmitted to medical centres in real time, where the data could be used for health status monitoring and further analysis.

The outcome of the case study is a framework useful for both the design of future sensor network solutions, and the evaluation of their suitability and correctness. The framework will be designed to meet the needs of RRHF[2] in their work towards better monitoring devices and network solutions. Apart from the tools developed, the framework should contain a library of reusable monitoring components and extensible models of biomedical sensor networks.

The users of the framework developed in Case Study 2 are the technical personnel responsible for biomedical sensor networks. The responsibility of these persons include planning future networks, acquiring hardware, setting up and maintaining networks, optimising network resources, and possibly writing special purpose software for the hardware. We emphasise the responsibilities that can be modelled by computer modelling and simulation, for example, planning and optimising networks; and we de-emphasise needs related to operating concrete networks.

In the following sections, a brief description of network architecture, typical application scenarios, platform, biomedical sensors, and user requirements are given. Definitions of biomedical terms appear in Appendix A.

[2]RRHF is an acronym for the Norwegian State Hospital, i.e., for its Norwegian name Rikshospitalet Radiumhospitalet Helse Foretak.

4.1 Biomedical sensors and networks

A wireless sensor network is a wireless network consisting of spatially distributed autonomous devices using sensors to cooperatively monitor physical, environmental or biomedical conditions, such as temperature, sound, vibration, pressure, motion, pollutants or biomedical signals at different locations. Sensor networks have been an active area of research for more than a decade, with applications in, e.g., medicine, military, oil and gas, and smart buildings. Biomedical sensors[3] are increasingly used to detect abnormal biological changes and monitor biological parameters in tissues and organs. Advanced hospitals are using an array of biomedical sensors for diagnostics, surgery, and monitoring in post operative phases. Patients undergoing complex surgical procedures like brain surgery, heart surgery, and organ transplantation surgery require that their physiological parameters are constantly monitored for titration of therapy. Biomedical sensors are used to monitor parameters such as blood gas, blood pressure, pulse rate, temperature, electrocardiogram (ECG), and electroencephalogram (EEG).[4]

4.1.1 Biomedical sensors

A biomedical sensor network consists of nodes, i.e., electronic devices which perform the tasks of sensing, processing, sending, and/or receiving biomedical data. From a data-flow perspective, a generic biomedical sensor node can be decomposed into the four *abstract* parts shown in Figure 3: sensor, receiver, processing unit, and transmitter. Each of these parts has its characteristics. The operation of one part may depended on that of another.[5]

Technically, a sensor node is built up of an MCU (CPU), memory (RAM, ROM), a (wireless) communication device, and the biomedical sensors. Since the power-consumption is an important issue, we add the power supply / battery as a separate building block. The technical building blocks of a biomedical sensor node are shown in Figure 4. The functionality of the biomedical sensor

[3]We use the term biomedical sensors for the sensors used in our case study. According to Wikipedia, the term biosensor refers a device for the detection of an analyte that combines a biological component with a physicochemical detector component. Biosensors consist of a sensitive biological element, a transducer, and a physiochemical detector element. Biosensors can be used as biomedical sensors.

[4]Another example of using biomedical sensors is in detecting ischemia, a phenomenon of low blood flow in tissues. Ischemia is still the most prevalent cause of death in the Western world. Except for the beating heart (ECG) and the conscious brain (EEG), most organs are silent when it comes to symptoms of ischemia.

In a clinical setting, we often do not detect organ dysfunction before the patient is already systemically affected. Most diseases are from the onset confined to one organ. In many cases the ischemia is reversible and an early detection could lead to appropriate treatment to save the organs and the patient's life. Thus, there is now increasing effort to bring about methods to detect organ ischemia in real time.

[5]As an example we mention that in some implementations the nodes cannot send and receive simultaneously, and therefore the total bandwidth is bounded by the sum of the sending and the receiving bandwidth. Note also that receiver and transmitter might be implemented technically as one device that performs both functions.

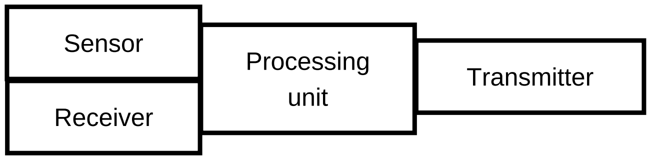


Figure 3: Generic block diagram of biomedical sensor node.

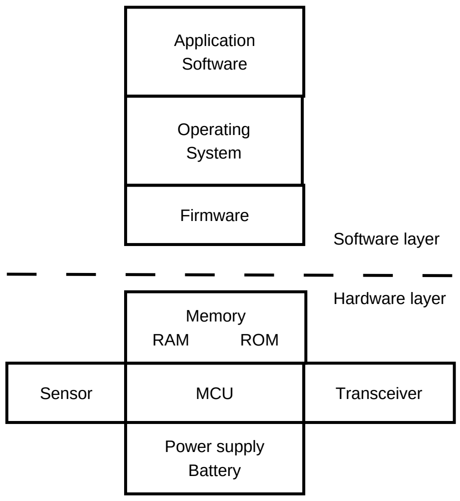


Figure 4: Technical building blocks of a biomedical sensor node.

is controlled by software, usually consisting of firmware[6], operating system, and specific application software for treating the biomedical signals, and their transfer.

4.1.2 Generic model of a sensor network

A sensor network may consist a large number of sensor nodes and sink nodes, which are connected by a wireless medium. The wireless communication of sensor nodes can be single-hop communication or multi-hop communication. The information flow is from the sensor to a sink node, which communicates the data to the second and third tier, as illustrated in Figure 6.

Each sensor node in a network has a unique identifier. It consists of the components described in Section 4.1. The capabilities of these nodes are limited due to size, cost, memory, and lifetime constraints. A single node has limited processing power, memory, and energy so that complex or computation intensive algorithms cannot be performed on an individual node.

On the link-layer the communication medium is wireless and broadcast is used as the basic communication primitive.[7] In the wireless broadcast model, messages are subject to fading and other propagation losses. Messages from nearby nodes may collide with each other if they are sent at the same time. This can also happen when the transmitting nodes are not in each other's communication radius: their messages can collide at the same receiver node and be lost [**?**].

The wireless medium is shared by all the sensor nodes, so a medium access control (MAC) mechanism is needed to avoid transmission collision. Distributed mechanisms are preferred due to the dynamic nature of sensor networks.

Sensor nodes employ short-range wireless communication devices, which implies that receiver nodes may be outside the direct transmission range. Therefore, a routing protocol is needed to direct messages from any node in the network to the sink node. The dynamic nature of the wireless communication medium often results in unstable long link-chains in the network, and can result in frequent route changes. Since sensor nodes can be mobile, dynamic routing protocols should be used.[8]

4.1.3 Embedding the biomedical sensor network

The sensor network is embedded into the information infrastructure of a hospital, which leads to a tier-structure. The biomedical sensor network is part of a network consisting of three tiers which facilitate the transport of biomedical

[6]In certain sensor node types the firmware does not exist as a separate entity. However, settings subject to national regulations, e.g., frequency spectra, are stored in a firmware.

[7]Broadcast implies a one-to-all relationship, where one sender transmits information which can be received by many receivers. Note that broadcast paradigms are in use in Layer 2 (see Section 4.1.6), and usually are not used in the upper communication layers.

[8]One approach to dealing with such links involves dynamic link estimation to determine the set of stable links. The routing protocol should be both reliable and lightweight in terms of communication, computation and memory requirements.

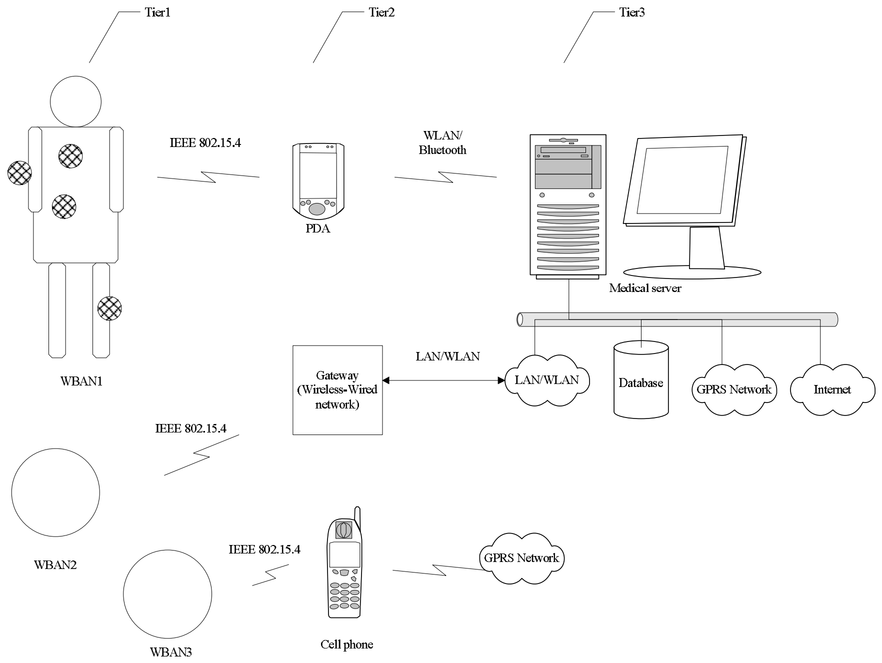


Figure 5: Tiered structure of biomedical sensor networks.

data from the patient to the health care personnel [**?**]. The biomedical sensor network for one patient is also referred to as a wireless body area network (WBAN) or wireless personal area network (WPAN) in the literature. Figure 5 shows a tier-structure in medical biomedical sensor networks. The three tiers are:

Tier 1 – WBAN: The WBAN consists of a number of sensor nodes equipped with different biomedical sensors. Each sensor is capable of sensing, processing, and communicating physiological data.

Tier 2 – Medical Gateway: The medical gateway collects data from sensor nodes of the WBAN, and forwards the data to the medical server. This functionality may be implemented on a sink node, a PDA[9], a dedicated gateway, such as the Tmote Connect gateway, or a personal computer.

Tier 3 – Medical server: The medical server is the centre of the health monitoring system. It interfaces users, medical personnel, and other wired and wireless networks. The medical server may service hundreds or thousands of users.

[9]PDA: Personal Digital Assistant, a handheld computer to assist the user with everyday tasks while the user is mobile.

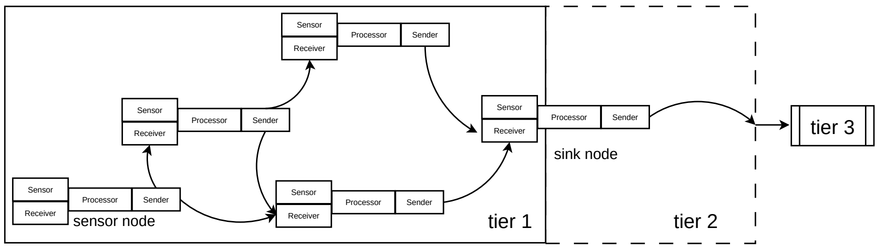


Figure 6: Structure of a biomedical sensor network, showing sensor node, sink node, and tier structure.

Biomedical sensor	*Sampling frequency* (Hz)	*Sampling resolution* (bits)	*Data rate* (bps)
ECG (3 channels)	250	16	12000
EEG	1000	12	12000
EMG	800	12	9600
S_PO_2	125	12	1500
Blood pressure	125	12	1500
Body temperature	1	12	12

Table 1: Specifications of the sensor data.

4.1.4 Properties of biomedical sensor data

Different biomedical sensors can produce measurements for different kinds of biomedical data, e.g., ECG, EEG, EMG, S_PO_2, blood pressure, temperature or sound. These sensors are widely used in medical applications, and the data can be used to evaluate patients' health status. The biomedical sensor data consist of one or several tracks of sampled measured values. A sensor could measure a physical entity which is converted from analogue to a digital representation, quantised, and sampled into a sequence of sampled values. The most important properties of medical data are sampling frequency and sampling resolution. Table 1 shows the properties of one-dimensional biomedical signals processed by the sensors.[10] Additionally, the raw medical data is supplemented with administrative data, e.g., a time-stamp and the identity of the sensor.

Some biomedical sensors process and communicate two-dimensional data, such as images. For our modelling purposes, this kind of sensors is treated similarly; however, these sensor work with much higher data volumes, need more processing capacity, and have some different threshold values.

[10]The sampling rates and resolutions are taken from the the following documents: IEEE P1073.1.3.6/D6.0 Draft Standard for Medical Device Communications Medical Device Data Language (MDDL) Virtual Medical Device, Specialised ECG [?], Biomedical Signal Processing Laboratory (http://bsp.pdx.edu/Data/), the Cognition and Brain Sciences Unit EEG Laboratory (http://www.mrc-cbu.cam.ac.uk/EEG/doc/eeg_intro.shtml), and the CodeBlue Project [?].

Platform	*Tmote Sky*
MCU	8MHz TI MSP430F1611
Raw data transmission rate	250kbps
Wireless transceiver	CC 2420 2.4 GHz, IEEE 802.15.4 radio
RAM	10K
ROM	48K flash ROM
ADC, DAC	12bit integrated
Communication range (m)	50 (in doors)/125 (outdoors)
Operating system	TinyOS
Wakeup from sleep	6 μs
external flash	1024 kbytes

Table 2: Characteristics of the Tmote Sky sensor node platform.

4.1.5 Platform

The platform includes a hardware platform and a software platform. In case study 2, we plan to use Tmote Sky[11] as sensor hardware platform, and Tmote Connect as gateway between sensor networks and TCP/IP-based networks.

The most important properties of the Tmote Sky are summarised in Table 2. The *Tmote Sky* is a low-power wireless module for use in sensor networks, equipped with both IEEE 802.15.4 and USB communication capabilities, an 8 MHz processor, and humidity, temperature, and light sensors[12]. Both the sensor nodes and the sink nodes use TinyOS as the operating system.

Tmote Connect[13] will be used as a gateway in our experiments. It can be used to bridge wireless sensor networks and wired local area networks, and provides bi-directional connectivity for data transfers to and from wireless sensor networks over TCP/IP sockets.

TinyOS, used by the Tmote Sky, is an open source component-based operating system and a platform for targeting wireless sensor networks. TinyOS is an embedded operating system.[14] TinyOS is developed by a consortium led by the University of California, Berkeley, in co-operation with Intel Research.[15] TinyOS employs a special C-dialect, called *nesC*.[16]

The Tmote Sky can be connected to a host computer[17] in order to commu-

[11]The data sheet for the Tmote Sky is available at http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf .

[12]The case study 2 intends to use a version of the Tmote Sky where the sensors for humidity, temperature, and light are not included on-chip.

[13]The data sheet for the Tmote Connect is available at http://moteiv.com/products/docs/tmote-connect-datasheet.pdf .

[14]TinyOS is intended to be incorporated into smartdust. Smartdust is a hypothetical network of tiny wireless micro-electromechanical systems (MEMS) sensors, robots, or devices, installed with wireless communications, that can detect anything from light and temperature, to vibrations, etc.

[15]More information on TinyOS can be found at http://www.tinyos.net/ .

[16]nesC is an acronym for "network embedded system C". A description is available from http://csl.stanford.edu/~pal/pubs/tinyos-programming.pdf .

[17]Drivers for the host computer are available for Windows, Linux, BSD, Macintosh, and

Current consumption	*Normal value*
MCU on, Radio RX	21.8 mA
MCU on, Radio TX	19.5 mA
MCU on, Radio off	1.8 mA
MCU idle, Radio off	54.5 μA
MCU standby	5.1 μA

Table 3: Typical current consumption of the Tmote Sky device.

nicate via a USB connector. The device is programmed through the on-board USB connector. The Tmote Sky supports re-programming over the radio link. The steps in this procedure are as follows:

1. Each node in the network receives a program image—containing new application software—via radio link.
2. Check/verify the program image.
3. A special program called bootloader load the new program image.
4. Reprogramming the micro-controller.
5. Reboot the node using the new program.

The same technique has been used in satellite software reprogramming. Obviously, this technique is not very reliable, especially in multihop sensor networks. However, it is the only way to re-program the sensor node when the node is not reachable physically, e.g., when it has been implanted into the human body.

Power consumption is an important issue for biomedical sensors. The Tmote Sky is powered by two AA batteries, the voltage supply should between 2.1 to 3.6 V DC. Table 3 describes the typical current consumption of sensor node platforms.

Transceiver. The transceiver, containing the functionality of transmitter (sender) and receiver, is IEEE 802.15.4 compliant, working in the 2.4 GHz band. In Case study 2, the *Chipcon 2420* (CC 2420) transceiver[18] is used in the sensor network. Note that the transceiver cannot transmit and receive simultaneously. The CC 2420 supports four states and switches between these four states when operating, in order to save energy [?]. Figure 7 shows the four states, and their respective energy consumption.

Transmitting: the transceiver is actively transmitting.

Receiving: the transceiver is actively receiving.

Idle: the chip is ready to receive instructions, while the clock is turned on.

Windows CE.

[18]Detailed information of CC 2420 is available in Chipcon's datasheet at http://www.chipcon.com/files/CC2420_Data_Sheet_1_4.pdf

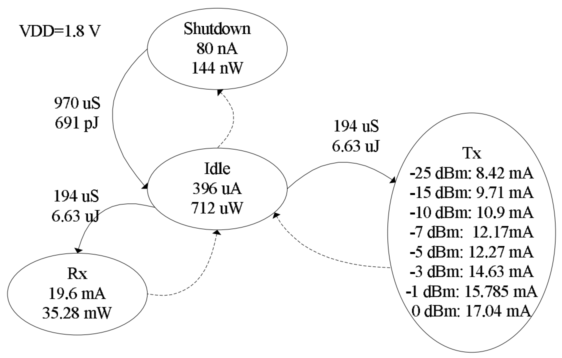


Figure 7: Steady state and transient power and energy measurement results for the Chipcon 2420.

Shutdown: the chip is deactivated, and waiting for a startup strobe; the clock is switched off.

The CC 2420 has programmable output power, which can be varied from -25 dBm to 0 dBm. The received signal strength can be obtained by reading a digital received signal strength indicator.

4.1.6 Communication in biomedical sensor networks

The biomedical sensor network is supposed to transmit the biomedical data wirelessly, using the IEEE 802.15 standard group[19], which specialises in Wireless personal area network (WPAN) standards.

The equipment used in our case study is based on the IEEE 802.15.4 standard (for the physical and link layers), and on the *ZigBee*[20] vendor standard for the upper communication layers.

The ISO Open Systems Interconnection Basic Reference Model (ISO/OSI Reference Model or OSI model for short) [?] is a layered, abstract description for communications and computer network protocol design, developed as part of the Open Systems Interconnection initiative. The following paragraphs refer to the layer structure of the OSI model.[21]

[19]See also http://www.ieee802.org/15/ .

[20]See http://www.zigbee.org/ [?] for information on ZigBee. The IEEE 802.15.4 task group 4 (Low Rate WPAN) works also on IEEE 802.15.4a (WPAN Low Rate alternative physical layer), which is providing communications and high precision ranging capabilities, high aggregate throughput, and ultra-low power, using either the 2.4GHz spectrum or the UWB Impulse Radio.

[21]Not all layers of the OSI model are relevant to Case study 2, and some of these are omitted in our overview.

Physical layer

The physical layer defines all the electrical and physical specifications for devices. The features of the physical layer are activation and deactivation of the radio transceiver, energy detection (ED), link quality indication (LQI), channel selection, clear channel assessment (CCA) and transmitting, as well as receiving packets across the physical medium.

In the CC 2420 a bandwidth of 250 kbps in the 2.4 GHz frequency band is given. Receiver sensitivity is -85 dBm for the 2.4 GHz band. The achievable range is a function of receiver sensitivity and transmitter power.

Typical radio propagation models are described as follows. The power of received signal, P_{rx}, is calculated as $P_{rx} = P_{tx} - pl$, where P_{tx} and pl represent the power of transmitted signal, and path loss, respectively.

Free space propagation. The free space propagation model assumes a transmit antenna and a receive antenna to be located in an otherwise empty environment. Neither absorbing obstacles nor reflecting surfaces are considered. The path loss is calculated as $pl = 32.5 + 20\log(d) + 20\log(f)$, where d is the distance in km and f is the frequency in MHz.

Additive white Gaussian noise (AWGN). In an AWGN channel model the only impairment is the linear summation of wide-band or white noise with a constant spectral density[22] and a Gaussian distribution of the amplitude. The model does not account for the phenomena of fading, frequency selectivity, interference, nonlinearity, or dispersion. However, it produces simple, tractable mathematical models which are useful for gaining insight into the underlying behaviour of a system before these other phenomena are considered.

The assumption of an AWGN channel is valid as long as the channel is coherent during the transmission of a packet (slow fading). With the maximum packet size of 133 bytes transmitted at the raw rate of 250 kbps, the packet transmission takes 4 ms, which is smaller than the coherence time encountered in the 2.450 GHz band without mobility issues [**?**].

Rayleigh channel. Rayleigh fading is caused by multipath reception. The mobile antenna receives a large number of reflected and scattered waves. Because of wave cancellation effects, the instantaneous received power seen by a moving antenna becomes a random variable, dependent on the location of the antenna. Deep fades occur occasionally. Although fading is a random process, deep fades have a tendency to occur approximately every half a wavelength of motion.

Rician channel. The model behind Rician fading is similar to that for Rayleigh fading, except that in Rician fading a strong dominant component is present. This dominant component can, for instance, be the line-of-sight wave.

[22]The spectral density is expressed in watts per hertz of bandwidth.

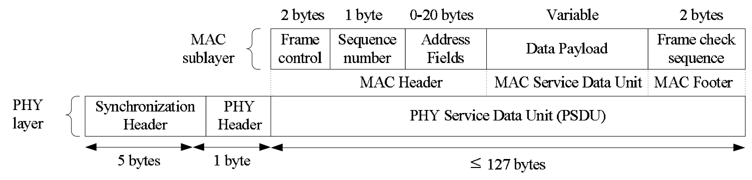


Figure 8: Schematic view of the IEEE 802.15.4 Frame Format.

The indoor wireless RF channel typically behaves as a Rician channel. If the line-of-sight is blocked, Rayleigh fading becomes an appropriate model.

Data link layer

The data link layer provides the functional and procedural means to transfer data between network entities, as well as the facility to detect and possibly correct errors that may occur in the physical layer. This layer may be split into sub-layers, such as the media access control (MAC) layer, depending on the used standard.

The features of *MAC sub-layer* are beacon management (optional), channel access, GTS management, frame validation, acknowledged frame delivery, association and disassociation.

Carrier sense multiple access with collision avoidance (CSMA/CA) is used as channel access mechanism. Both the physical layer and the MAC layer are defined in the IEEE 802.15.4 standard [?].

Frame format. The CC 2420 has hardware support for parts of the IEEE 802.15.4 frame format. Fig. 8 shows a schematic view of the IEEE 802.15.4 frame format. Specific frame formats (data, beacon, acknowledgement and MAC command frames) are defined for the IEEE 802.15.4 standard.

Wireless channel packet error rate (PER)/bit error rate (BER). In an IEEE 802.15.4 system, all communication is based on packets. It is more accurate to measure the PER than the BER since it mirrors the way the actual system operates. In the IEEE 802.15.4 standard, the PER is $\leq 1\%$, when the received signal $\geq$ -85 dBm. The Physical layer Service Data Unit (PSDU) length should be 20 bytes.

The link link layer discards packets that are recognised to be corrupted, and cannot be recovered by mechanisms of the IEEE 802.15.4 standard, like CRC. However, occasionally defectious packets are not recognised, and are passed to the upper layers.

Network layer

The network layer provides the functional and procedural means of transferring variable length data sequences from a source to a destination via one or more networks while maintaining the quality of service requested by the transport layer. The network layer performs network routing functions, and might also perform segmentation, de-segmentation, and report delivery errors. In sensor networks, the network layer mainly performs network routing functions.

The following two kinds of routing protocols are supported in a *ZigBee* network.

Ad hoc on demand distance vector (AODV). AODV is an on-demand route acquisition algorithm: Nodes that do not lie on active paths neither maintain any routing information nor participate in any periodic routing table exchanges. Further, a node does not have to discover and maintain a route to another node until the two need to communicate, unless the former node is offering services as an intermediate forwarding station to maintain connectivity between two other nodes.

Cluster-tree algorithm. The cluster-tree protocol is a protocol of the link and network layers that uses link-state packets to form either a single cluster network or a potentially larger cluster tree network. The network is basically self-organised and supports network redundancy to attain a degree of fault resistance and self-repair. Nodes select a cluster head and form a cluster according to the self-organised manner. Then self-developed clusters connect to each other using the designed device.

Fixed routing schemes often use routing tables that dictate the next node to be routed to, given the current message location and the destination node. Routing tables can be large for large networks, and cannot react to events in real-time, such as failed links, nodes with backed up queues, or congested links [**?**].

Other routing protocols may also be designed and employed, depends on specific user applications.

Transport layer

The transport layer provides transparent transfer of data between end users, thus relieving the upper layers from any concern while providing reliable data transfer.

Application layer

The application layer provides a means for the user to access information on the network through an application. This layer is the main interface for the user(s) to interact with the application and therefore the network. The application layer interfaces directly to and performs common application services for the application processes. Sensor data sampling, compression and aggregation take place in the application layer.

4.1.7 Existing simulation software

Several simulation systems have been developed to analyse the performance of communication networks. Some of these can be applied to the evaluation of sensor networks, by analysing performance parameters such as such as latency, packet loss rate, network throughput, and other metrics. Most of these systems use discrete event simulation. Examples for such simulation systems include Opnet[23], OMNeT++[24], and NS-2[25]. In the following we give a short overview of the discrete event simulators that are free for academic use.

OMNeT++. OMNeT++ is a public-source, component-based, modular and open-architecture simulation environment with GUI support and an embeddable simulation kernel. Its primary application area is the simulation of communication networks, and has been successfully used in areas like the simulation of IT systems, queueing networks, hardware architectures and business processes as well. Several open source simulation models have been published, in the field of Internet simulations (IP, IPv6, MPLS, etc), mobility and ad-hoc simulations and other areas.

Components include the simulation kernel library, a compiler for the NED topology description language[26], a graphical network editor for the NED files (GNED), GUI and command-line interfaces for the simulation execution, graphical output plotting and visualisation tools, and other utilities. OMNeT++ is supported for both Linux[27] and Microsoft Windows (XP, Win2K).

More information is available at http://www.omnetpp.org .

The Network Simulator – NS-2. NS-2 is a discrete event simulator targeted at networking research. NS-2 provides support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. NS-2 is developed from the *REAL network simulator* from 1989 onwards and has evolved substantially over the past few years.

Protocols and modules of NS-2 include application-level protocols (HTTP, telnet, CBR, etc.), transport protocols (UDP, TCP, RTP, SRM, etc.), routing (algorithmic routing, hierarchical routing, etc.), router mechanisms (scheduling, queue management, admission control, etc.) link layer mechanisms (LAN with CSMA/CD MAC protocols, etc.), and error modules. NS-2 is supported for both Unix-like platforms (FreeBSD, Linux, SunOS, Solaris), and Microsoft Windows (9x/2000/XP).

More information is available at http://www.isi.edu/nsnam/ns/ .

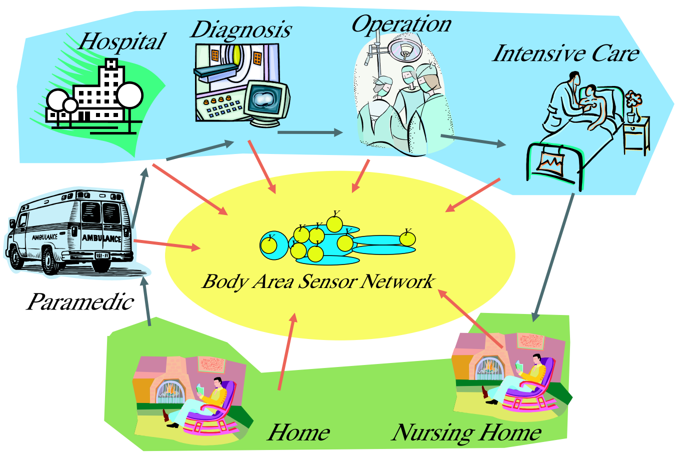
[23]See http://www.opnet.com/ .

[24]see http://www.omnetpp.org/ . OMNeT++ is licensed with the Academic Public License, see http://www.omnetpp.org/external/license.php .

[25]See http://nsnam.isi.edu/nsnam/ . NS-2 is licensed with a GPL-compatible license, see http://nsnam.isi.edu/nsnam/index.php/Developer_Information .

[26]NED (NEtwork Description) is the topology description language of OMNeT++. See http://www.omnetpp.org/download/docs/papers/cnds98-paramtop.pdf .

[27]and other Unix-like systems.



Biomedical Sensor Networking Transitions in Medical Scenario

Figure 9: Biomedical sensor network transitions.

4.2 Scenarios for biomedical sensors

The biomedical sensor networks will be used in several scenarios of a health care environment. These scenarios include surveillance of patients in home environments (e.g., at home, nursing home), paramedic scenarios (e.g., accidents), and hospital scenarios (e.g., diagnosis, operation, intensive care, surveillance). As patients and health care personnel enter and leave different networks, the networks must adapt automatically to these changes, and the units involved in the network must adapt their behaviour to control the relevant services. Also the activity level of the sensors may be controlled by other units in the network, automatically or manually by health personnel, and the network must be aware of sensors becoming weaker or failing, and adjust the information flow accordingly.

Figure 9 shows several biomedical sensors used in different applications. The sensors need to adapt to the dynamic nature of the network environment and applications. The mobility of the patient and their monitoring sensors in complex networking topologies will be a challenging task in order to guarantee a minimum set of quality of services (QoS).

With a few exceptions, the biomedical sensors are single devices with a single output. An array of biomedical sensors are needed for measuring multiple physiological parameters. Today, the individual sensors are often connected to the monitoring devices with wires, which may cause adverse events and restrict the mobility of the patients.

Use of wireless technology in biomedical sensors will become helpful for critically ill patients who are transported from one unit to another in the hospital and help in their recovery period. For patients living in their homes, this technology becomes helpful in providing the freedom of moment while ensuring that the patients are constantly monitored and cared for. Recent advancements made in electronic circuit design and wireless communication have enabled the development of low cost, low power, and multi-functional sensors. This will have a major impact on minimally invasive diagnostics.

The societal relevance of wireless biomedical sensors applies to a range of diagnostical, surgical and post operative phases:

- *Hospital.* As the regionalisation and specialisation of critical care become common, refined and safe methods for both inter hospital and intra hospital monitoring during transport becomes more important. The post operative and recovery period can be reduced if the patient can be mobilised faster. Wireless patient monitoring may help to eliminate the use of wires and may help the patient in recovery. This means that the patients can spend less time in post operative care at the hospital. This may become a potential element in reducing the operational cost in public hospitals, which are mostly funded by the tax payers. The hospital [**?**] and intra-hospital [**?**] monitoring during transport becomes more important. The patients too will benefit by returning to their normal lives faster. Digital data may help eliminate the use of paper and charts to record vital events. This may help the hospitals to move in the direction of paperless institutions. This will also help archive and retrieve data effectively using less human intensive work, thus cutting operational cost of the institutions.
- *Nursing and Senior Citizen Homes.* The patients are equipped with wireless biomedical sensors triggering alarms, which are sent to nearest hospitals and medical practitioners. Nursing homes, with facilities to provide beds to the nearest hospitals, may benefit from fast and flexible setup of equipment and wireless transmission of physiological data and alarms, avoiding costly cabling.
- *Paramedic.* Paramedic teams in ambulances dealing with emergencies need to gather and share physiological parameters with their bases. Accurate and quick diagnostics and immediate treatment are vital to save lives. As time is the most important factor, easy deployment of wireless biomedical sensors may be useful. Wireless solutions also allow for adhoc networking at the site and can provide the infrastructure for better information flow between hospital and emergency site.
- *Home Health Care.* Patient discharged early from hospitals can be monitored in their homes. The sensor data and alarms can be transmitted to medical practitioners, home health care nurses, or emergency dispatch units. This may increase the patients' mobility and safety and allow patients to live in their homes for longer periods while knowing that nursing home or hospital will respond in case of a crisis. Wireless systems will avoid costly cabling of patients' homes.

The case study investigates the three scenarios of *Post operative monitoring of patients with artificial heart*, *Early warning of heart attack and/or stroke*,

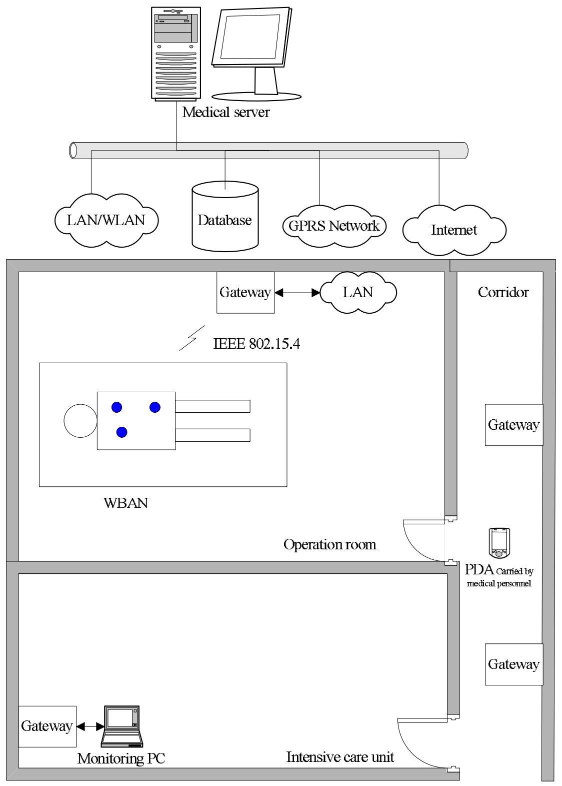


Figure 10: Post operative monitoring of patients with artificial heart.

and *Deployment of biomedical sensors networks at the site of an accident* in particular. These are described in the following subsections.

In particular, the output frequency of the sensors will be automatically adjusted when leaving and entering a net, in order to provide the required information flow, while saving battery life as much as possible. The case study will also define a minimum set of requirements for guaranteed QoS. Both automatic adaptions and explicitly enforced changes of sensor activity must be covered, including dynamic changes in the networks and dynamic changes of sensors. Thus, context awareness and network reconfigurability are central in the case study. While some of the scenarios involve only one patient, we note that the sensor network must be able to serve several patients in one sensor network, without compromising the security (confidentiality, integrity, availability) of the data.

4.2.1 Monitor patients with artificial heart

In the scenario shown in Figure 10 several biomedical sensors are used in a hospital environment, in which the patient is located in an operating room (OR) or intensive care unit (ICU) while he or she undergoes intensive monitoring of vital physiological parameters. Additional sensors might be required during this procedure in order to monitor other physiological parameters. The patient(s) may be moved between different rooms during the treatment, e.g., from the OR to the ICU, while monitoring must continue uninfluenced by this.

The sensor data may need to be transferred over different wireless networks. The system should be able to cope with breakdown in sensor nodes, new software updates, wireless network traffic congestion, and interferences with other wireless networks and biomedical devices.

Characteristics:

1. A fixed network infrastructure is available, such as a local area network, gateways, wireless access points (WAP), and Bluetooth. This network infrastructure can be used by the sink nodes of the biomedical sensor network.
2. The scenario includes a complex communication environment. Interference from co-existing wireless networks, such as WLAN, Bluetooth and various medical facilities are possible, and may reduce the performance of the transmission.
3. The scenario provides a relatively fixed network topology in the ICU or OR. Changes of the network topology, possibly causing handoffs, may happen while patients are moving or being moved from one place to another.

4.2.2 Early warning of heart attack or stroke

In the scenario shown in Figure 11 a few biomedical sensors are employed, within an environment in which the patient is at home or in a nursing home. Elderly people, chronic patients or patients discharged early from hospitals can be monitored by biomedical sensor networks. The sensor data and alarms can be transmitted to medical centres and emergency dispatch units.

In this scenario the sensors might not be monitoring or transmitting the physiological parameters continuously. This way of monitoring may be used for reducing battery power consumption. Depending on a predefined algorithm, abnormal sensor data from certain sensors may be used to activate other sensors autonomously before an alarm is triggered, and sent to a monitoring central unit.

Characteristics:

1. The ease of use is an important issue.
2. Very low power consumption, and thus a long life span of the batteries, is required.
3. A fixed network infrastructures is available, such as Internet, LAN, Gateway, wireless access point (WAP), GPRS/3G networks. This infrastructure can be used by the sink nodes of the sensor networks to transfer data to the monitoring unit.
4. Limited mobility, infrequent handoff.

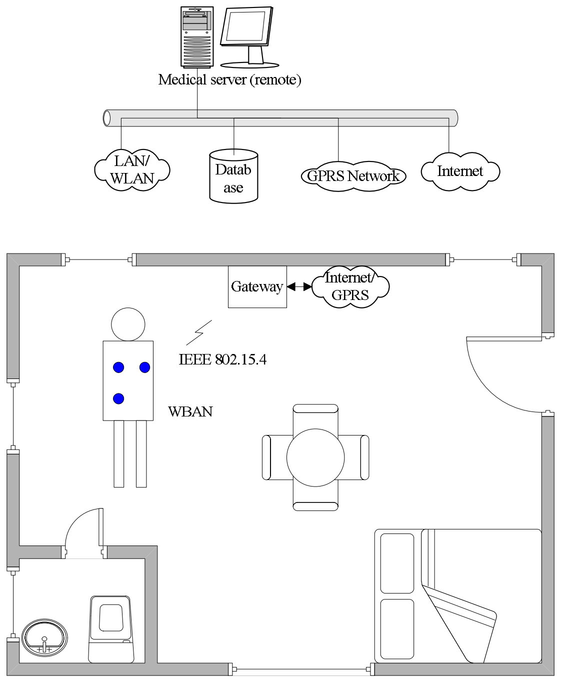


Figure 11: Early warning of heart attack and/or stroke.

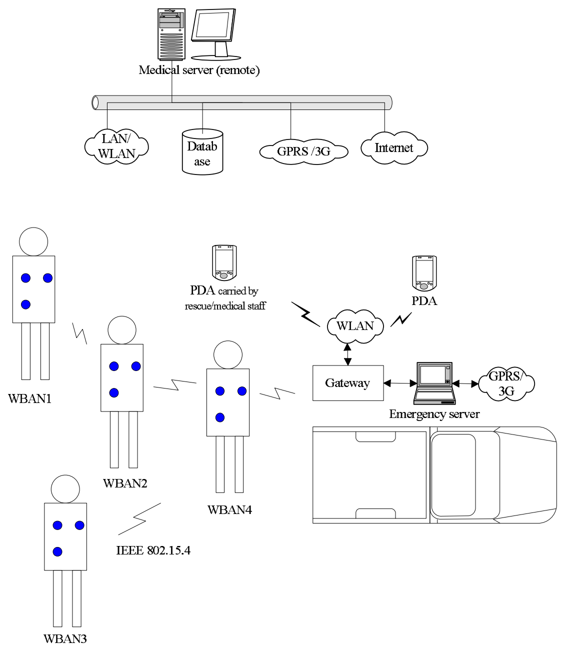


Figure 12: Deployment of biomedical sensors networks at the site of an accident.

4.2.3 Deployment at an accident site

In a disaster/accident response application scenario, such as fires, terrorist attacks, traffic accidents, the normal wired or wireless communications infrastructures may be damaged or destroyed, and a large number of severely injured overwhelm the emergency field personnel and hospital staff, and prevent them from providing efficient and effective emergency rescue. Biomedical sensor networks can be quickly deployed to monitor the vital signs such as blood pressure, temperature, pulse and ECG of the casualties. A large number of injured can be monitored simultaneously, thus limited emergency rescue resources can be allocated and managed properly.

In the scenario depicted in Figure 12 a few vital biomedical sensors are deployed, measuring values like blood pressure, temperature, pulse and ECG in an ad-hoc network at the site of an accident.

Characteristics:

1. The sensor network must operate autonomously, and needs a high degree of self-organisation. The network topology is highly dynamic. Therefore, sensor nodes should be able to discover each other, and setup a sensor network automatically.
2. No fixed network infrastructures is available; data transfered from the tier 2 (medical gateway) to the tier 3 (medical server) must use a mobile

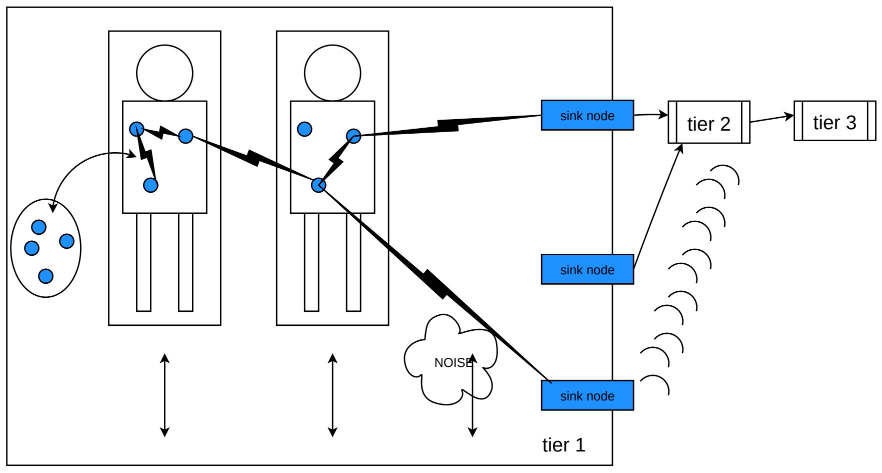


Figure 13: A generic scenario containing the important elements of the three scenarios previously described.

network, such as GPRS, 3G, satellite, or other specific wireless network (microwave, digital trunk communication).

3. In this scenario the radio link might be unstable, and the radio link quality might vary. Additionally, the communication environment is rather complex, since many sensor nodes may be deployed in a small area, possibly causing severe channel competition.
4. High degree of mobility.

4.2.4 A generic scenario

From the three scenarios presented above we develop a generic scenario, containing the relevant elements to be included in the modelling process. The generic scenario is presented in Fig. 13. The generic scenario is designed so that parts of it can be selected and modelled separately, e.g., using only one patient with one sensor, instead of several patients with multiple sensors.

The generic scenario includes the following elements:

- One or several patients are equipped with one or several biomedical sensors each. The biomedical sensors have the properties described in Section 4.1.
- The biomedical sensors communicate with each other, and with a sink node which acts as a gateway to tier 2. The communication between the sensors is facilitated with a suitable inter-sensor communication network, e.g., ZigBee.
- The patients can be moved, so that the topology of the biomedical sensor network changes. Additionally the connection to the gateway can change, and possibly biomedical sensors can get out of reach of each other. The connection to the gateway node can be interrupted.

- Biomedical sensor nodes can be added to a patient, or removed from a patients, thus changing the topology, and the data load of the network.
- Within the reach of the sensor network sources of signal noise can appear, move, and change in various properties.
- The gateway node can communicate with the tier 2 either wired, or wireless using a suitable network connection; e.g., WLAN, GSM, etc.
- The software of the medical sensor nodes can be updated while the sensor is active. The software is installed through the wireless communication network.
- The communication between tier 2 and tier 3 is beyond the scope of this scenario.

Concrete example of a biomedical sensor network

In order to show the concrete problem areas to be modelled in Case study 2, and to identify the concrete problem areas, we present a concrete scenario, where several issues are simplified.

We consider a scenario where the sensor network consists of four biomedical sensor nodes and one sink node (gateway node). Multi-hop communication in a tree-based hierarchical network topology is used. Each of the sensor nodes and the sink node is identified with a unique address[28].

For simplicity, we assume that a fixed routing scheme is used. The fixed routing scheme is described in Section 4.1.6. (Adaptive routing schemes can also be employed, depending on user applications.)

In order to achieve high network throughput and low latency, non-beacon mode[29] is employed due to less traffic transmission control overhead. 16 bit short addresses are used in the data packet. Since the wireless medium is shared by all nodes, CSMA/CA is used as medium access control mechanism.

Each sensor node is equipped with a 1-channel ECG sensor, the output data rate of the sensor is 4 kbps. Every 200 ms, each sensor node sends a data packet. The packet size is 119 bytes with payload size 100 bytes. The packet structure is shown in Figure 8. Some sensor nodes are responsible for packet forwarding.

Concrete work flow sketch

As a concrete work-flow sketch we present the work flow of a biomedical sensor network from a technical perspective.

1. The sink nodes are powered on, start booting procedure.
2. The sensor nodes are powered on, start booting procedure.
3. The sink nodes are ready to receive packets.

[28]Both sensor nodes and sink node use the address information to distinguish their identity.
[29]Defined in the IEEE 802.15.4 standard.

4. Whenever a sensor node wishes to send data packets, it waits for a random number of backoff periods, then checks the channel status. If the channel is found to be idle, the sensor node can start transmitting, if not, it waits for another random number of backoff periods before trying to access the channel again. (CSMA/CA is used as channel access control mechanism, detailed algorithms can be found in the IEEE 802.15.4 standard.)

5. The sensor nodes begin to send packets to the sink nodes or other intermediate sensor nodes, using a routing table, pre-planned by user application.

6. When a sensor node receives data packets which need forwarding, it forwards the packets to the next destination node, using routing table information.

7. When the sink node receives data packets, it acknowledges the packets have been received successfully by sending an ACK packet to the source sensor node (optional), and then it forwards the packets to a medical gateway in tier 2.

4.3 Requirements

We present the requirements for Case study 2 divided into two parts: *Domain requirements*, in Section 4.3.1, are general requirements of the domain of biomedical sensor network, including the capability of network nodes and the software they are running. Section 4.3.2 specify requirements to the *modelling language*. In this language users will describe their biomedical sensor networks. Section 4.3.3 lists *requirements to software tools*: How tools can help users build and reason about models.

4.3.1 Domain requirements and properties

The domain requirements describe general requirements of the domain itself, in particular requirements that stem from the domain technology and the intended use of the technology.

1. Flexibility

   (a) Software upgrade: Software running on the sensor nodes can be upgraded, either via the USB port from a PC, or by re-programming during network operation, using the wireless network. Note that some biomedical sensors solely can be upgraded through the wireless network connection, e.g., when they are placed inside a human body without extra wires.

   (b) Re-configuration: When being applied to different scenarios, the application software or parameter settings in the sensor node must be changed. Therefore, the node must be re-programmable and reconfigurable. For instance, a sensor network installed in a hospital can be deployed in an accident area for disaster response by adjusting parameters and re-programming the sensor nodes.

    Another application of re-configurability is that the functionality and role of a sensor may be controlled by other units within the network, in order to adjust the information flow according to variations in a patient's health conditions.

   (c) Self-organisation: After sensor nodes are turned on, they can discover each other and set up a network autonomically. This functionality is essential especially when a large number of sensor nodes needs to be deployed in an accident area.

   (d) Device mobility: Both patients and medical personnel are mobile, requiring that the communication layers adapt rapidly to changes of network topology, and radio link quality. Handovers on different layers must be supported.

2. Robustness and quality of service

   (a) Scalability: When many sensor nodes are deployed in a hospital or an accident area, the traffic load of the network may become high due to increasing number of sensor nodes and/or high date rate output of sensor nodes. This can cause severe wireless channel competition.

   (b) Robust wireless communication: In complex communication environments, mechanisms should be employed to mitigate interference from co-exist wireless networks and various medical applications, and robust error detection and correction algorithms should be used in packet transmission procedures.

   (c) Quality of service (QoS): A biomedical sensor network may be required on average to deliver a certain measurable performance regarding parameters such as the following: network bandwidth, data throughput, signal quality (signal-to-noise ratio), response times, latency, packet delivery rate, jitter.

3. Security and safety: The standard ISO/IEC 17799 [?] is relevant for security and integrity requirements. The key aspects of Information Security are to preserve the *confidentiality*, *integrity* and *availability* of an organisation's information and operations. All three properties are relevant for the biomedical sensor networks, since loss of one or more of these attributes can endanger the performance of a hospital. In addition, authentication is important in networked environments.

4. Power consumption: The power-consumption should be so low that battery changes are only necessary in a reasonable time frame, depending on the scenario. The main contribution of power consumption are wireless communication and computation. Detailed power consumption depends on the application scenarios, and can be calculated using Table 3.

Scenarios[a]	*Hospital*	*Home*	*Accident area*
Date rate (packets/s)	20–50	10–20	10–20
Number of sensor nodes	30	5	30
Communication hops	1	1	1–5
Contending senders[b]	2–10	2–5	10–30
Network setup time (s)	100	150	300
Network self-organisation ability	no	no	yes
End to end latency (s)	0.5	1	2
Packet delivery rate (%)	95	90	70
Throughput (Kbps)	20–40	15–25	15–25
Battery life (hours)	96	120	72
Transmitting power (dbm)	-1	-1	0
Receiving sensitivity (dbm)	-85	-85	-85
Power management	yes	yes	no

[a]The accuracy depends on the signal sampling resolutions and sensor specifications.

[b]Contending senders valid within CC 2420 radio communication range.

Table 4: Expected typical values to be used in the three selected scenarios.

Since power consumption is a major concern, one may update sensor software at runtime to improve battery life, and improve external dynamic control. In particular, the transmission of information from a biomedical sensor to the network is the main source of power consumption in the sensor.

4.3.2 Modelling requirements

From the generic scenario described in Section 4.2.4 and illustrated in Fig. 13, and from the properties of the application domain, we deduct that the modelling requirements should, if possible, be able to deal with the following:

1. (Discrete) objects that have an autonomous internal behaviour.
2. Communication. These objects communicate with each other and the communication can be described with certain properties and parameters.
3. Time.
4. Topology and geometry.[30] In the generic scenario patients, sensor nodes, noise sources, and receivers could be in motion, which will have an impact on the quality of the communication links. The model of geometry should be discretised in order to be applicable to *Creol* or *Maude*. This is possible

[30]The model expressed in two dimensions is supposed to suffice, e.g., movement of patients, beds, sensors, etc. A model of a three-dimensional scenario would only be necessary if the sensor placement in a three-dimensional space on the patients is of relevance.

using methods from computational geometry. Some of the properties can also be expressed in terms of topology.[31]

5. Statistics to express, e.g., the number of successfully transferred packets. Statistical modelling is relevant for, e.g., the lower layers in the communication model.

6. Quality of service as a dimension.

7. Trust and/or information security.

8. The OSI Layer structure as outlined in Section 4.1.6.

The modelling languages used in the *Credo* project are addressing models that have a discrete domain. Therefore, the properties of time, space, trust and service quality should be translated into a discrete description of the model.

On level of detail in models: The language of models should be specify roughly the following kinds of entities entities: biomedical sensor, transceiver (transmitter and receiver), wireless channel, communication protocol, mobility model (random direction, average walking speed, others) power consumption model.

4.3.3 Tool requirements

The tools are to help users analyse models of biomedical sensor networks in order to learn about the properties of those networks. This reduces the reliance on experiments involving real hardware and thus increases the efficiency of the network designer. Real-time modelling and probabilistic modelling are important for biomedical sensor networks. The tools for analysing biomedical sensor networks assist the user in two ways:

- Verification of properties of the model, via model checking, type checking, static analysis or exhaustive search (through all possible executions from a given initial state)

- Exploration of model behaviour, via systematic execution, testing and simulation.

The tool must have a means for specifying *properties*, perhaps as part of the models. Here are some examples of properties that the tools should be able to reason about:

- The absence of confusing dead sensors (due to internal failure) with silent sensors (i.e., sensor that do not transmit but await a triggering condition).

[31]Methods from the area of computational geometry can be applied for the case of geometry. An example would be the application of a Voronoï-diagram or similar techniques in order to subdivide space. This requires from the Creol language to be able to handle space, in addition to time.

- May an “undesirable state” or “desirable state” be reached by the model (over time).
- Are different user-required properties in conflict?
- Area sensing coverage or connectivity in networks.

Also, a tool should do more than just report that a verification or simulation yield a number below or above a threshold value or that there are conflicts between requirements. Users need help in *revising* their models. For example, how can a model be revised to satisfy (more) requirements or increase quality of service? By adding more sensor nodes or relay nodes, by removing sensor nodes and rearranging the remaining nodes, or by reducing the power consumption of nodes?

A Biomedical terminology

A.1 Sensor technology and radio technology

A *biomedical sensor* is a type of transducer, which can transform biomedical signals into electrical signals.

Wireless communication: The nodes in sensor networks are connected by wireless links.

Wireless medium: The medium used to implement the transfer of protocol data units (PDUs) between peer physical layer entities of a low-rate wireless personal network (LO-WPAN).

Coverage area: The area where two or more devices can exchange messages with acceptable quality and performance.

Packet transmission latency: The amount of time it takes a packet to travel from source device to destination device.

Data transmission rate: The number of bits that can be transmitted in a given time, measured in bits per second.

Battery life: How long the device can work without battery change or recharge.

Sample: The output of biomedical sensor is an analogue signal; it needs to be sampled and digitalised periodically.

CRC: A cyclic redundancy check (CRC) is a type of hash function used to produce a checksum which is used to detect data errors. In the IEEE 802.15.4 standard, CRC-16 is used to detect data transmission errors.

Power consumption: In sensor networks, the power consumption is usually the result of power used to perform the function of sensing, processing, and communication by nodes.

A.2 Medical terms and procedures

Physiological data: Indicate human health status, including ECG (heart activity), EEG (brain activity), EMG (muscle activity), heart rate, S_PO_2 (blood oxygen saturation), blood pressure, body temperature.

ECG: An electrocardiogram (ECG) is an electrical recording of the heart signals and is used in the investigation of heart disease.

EEG: An electroencephalogram (EEG) represents an electrical signal (postsynaptic potentials) from a large number of neurons.

EMG: An electromyograph (EMG) detects the electrical potential generated by muscle cells when these cells contract, and also when the cells are at rest.

S_PO_2**:** Blood oxygen saturation, the level can be measured by detecting the amount of light absorbed by hemoglobin in the blood at two different wavelengths (typically 650 nm and 805 nm).

PH: Ph is a measure of the acidity of a solution in terms of activity of hydrogen (H+).

Temperature: The patient's body temperature.

A.3 Network-related terms

Node: A *node* is a fundamental device in sensor networks, it consists of MCU (CPU), RAM, ROM, wireless communication device. It's the platform of various biomedical sensors.

Sensor node: A *sensor node* consists of MCU, RAM, ROM, wireless communication device, biomedical sensors. It is used to sense, pre-process and transmit the physiological data.

Sink node: A *sink node* consists of MCU, RAM, ROM, wireless communication device. Sink nodes may have more powerful computation and communication capacities, therefore, it is used to collect and aggregate data packets transmitted from sensor nodes and forward the data to a gateway or a PC.

Mobile device: A device that uses network communications while in motion.

Gateway: A *Gateway* is a bridge between wireless sensor networks and a wired local area network. Bi-directional connectivity involves data transfers to and from wireless sensor networks over TCP/IP sockets.

Packet: The format of aggregated bits that are transmitted together in time across the physical medium.

Payload data: The content of a data message that is being transmitted.

Self-organisation: A *self-organising* the ability of network nodes to detect the presence of other nodes and to organise them into a structured, functioning network without human intervention.

Communication protocol: A set of standard rules for data representation, signalling, authentication, and error detection required to send information over a communications channel.

TinyOS: An open-source operating system designed for wireless embedded sensor networks.

OSI seven layer model: The Open Systems Interconnection Basic Reference Model (OSI Reference Model or OSI Model for short) is a layered, abstract description for communications and computer network protocol design, developed as part of the Open Systems Interconnection initiative. It is also called the OSI seven layer model. An important property of this layered model is that each layer only uses the functions of the layer below, and only exports functionality to the layer above.

WBAN: Wireless body area network – a WBAN consists of multiple sensor nodes, each capable of sampling, processing, and communicating physiological data. Typical communication range is within 1 or 2 meters.

WPAN: Wireless personal area network, typical communication range is within 10 meters.

LAN: Local area network, is a computer network covering a local area, like a home, office, or group of buildings.

WLAN: Wireless local area network. A WLAN is a wireless local area network, which links two or more computers without using wires. WLAN utilises spread-spectrum technology based on radio waves to enable communication between devices in a limited area, also known as the basic service set.

GPRS: General Packet Radio Service, a mobile data service available to users of GSM and IS-136 mobile phones. GPRS is packet-switched which means that multiple users share the same transmission channel, only transmitting when they have data to send. The maximum speed of a GPRS connection (as offered in 2003) is 171.2kbps with all 8 time slots available. Latency is very high, a round-trip ping lasting typically about 600 $\sim$ 700 ms often reaching one second round trip time. GPRS is typically prioritised lower than speech, and thus the quality of connection varies greatly.

Tier structure: See Section 4.1.3.

Multi-hop: In multi-hop wireless sensor networks, communication between two sensor nodes or sink nodes may involve a sequence of hops through adjacent sensor nodes.

Single-hop: In single-hop wireless sensor networks, a sensor node can directly communicate with any other sensor node or sink node.

MANET A mobile ad-hoc network is a kind of wireless ad-hoc network, and is a self-configuring network of mobile routers (and associated hosts) connected by wireless links.

Wireless ad hoc network: Ad hoc connection is a network connection method which is most often associated with wireless devices. The connection is established for the duration of one session and requires no base station. Instead, devices discover others within range to form a network for those computers. Devices may search for target nodes that are out of range by flooding the network with broadcasts that are forwarded by each node. Connections are possible over multiple nodes (multihop ad hoc network). Routing protocols then provide stable connections even if nodes are moving around.