Dmytro Kovtun

MOBILE SERVICES FOR WIRELESS PRIVATE AREA NETWORKS BASED ON AD HOC CONNECTIVITY

Master's thesis

Mobile computing

27.12.04

University of Jyväskylä

Department of Mathematical Information Technology

Author: **Dmytro Kovtun**

Contact Information: e-mail: dmkovtun@cc.jyu.fi

Title: Mobile services for wireless private area networks based on ad hoc connectivity

Work: Master’s Thesis

Number of Pages:

Study Line: Mobile Computing

Department: University of Jyväskylä, Department of Mathematical Information Technology

Keywords: WPAN, Sensor Network, AD HOC, Short Range Mobile Device, Routing Protocols, EAI, Application Server, XML, Web-Service

Abstract: Wireless connectivity is not novelty in IT industry already. New technologies and standards are available to build wireless infrastructure in real market and they penetrate into our everyday life. More and more various approaches, different mobile devises are rushing in market to propose and provide diversity of services.

However, among others techniques of wireless connectivity, WPAN is quite new direction in IT industry which appeared recent years. There some research groups and projects which are working in the field of private area networks and are aimed to elaborate new standards and approaches concerning various issues such as connectivity, integration, routing, power consumption, security, quality of service.

As the result we can see already ready-made products and systems based on wireless ad hoc connectivity. It is expected rapid development of this technology and implying it into different market segment. Therefore it assumes the impetuous growing of diversity mobile services, networks integration issues and elaboration new standards.

This Master Thesis represents the pilot WPAN distributed system which was developed to show the possibility of utilization of WPAN technology and possible ways of WPAN integration with distributed environmental. Developed systems covers such aspects as providing some services with as example of obtaining temperature and lightness, the possible distributed architecture. Also it given some examples of use-case of what king of applications might be build upon this architecture.

Acknowledgements

The main practical base of this master thesis was obtained in the Kaustinen research lab of Chydenius institute. So I would like to thank to all co-workers for their warm and kind attitude to me. I am very thankful to my master's thesis supervisors – professor Vagan Terziyan and professor Ismo Hakala for their help, advises and support during the writing this Thesis.

I also would like to express my deep gratitude to Vagan Terziyan and Helen Kaikova for all their invaluable efforts in supporting me during my studding and living in Finland.

Abbreviations

API – Application Programming Interface

EIS – Enterprise Information System

FFD – Full Function Device

GSM – Global System for Mobile communications

GPRS – General Packet Radio Service

HTTP – Hyper Text Transfer Protocol

JSP – Java Server Page

IP – Internet Protocol

LAN – Local Area Network

MAC – Media Access Control

MVC – Model View Controller

RFD – Reduced Function Device

SLIP – Serial Line Internet Protocol

SOAP – Simple Object Access Protocol

TCP – Transmission Control Protocol

UDDI - Universal Description, Discovery and Integration standard

WSDL – Web Services Description Language

WAP - Wireless Application Protocol

UDP – User Datagram Protocol

WLAN – Wireless Local Area Network

WPAN – Wireless Private Area Network

XML – eXtensible Markup Language

Contents

1 Introduction

1.1 Background

Nowadays the use of different wireless devises is considered as traditional way of our everyday life. Penetrating into our houses, offices, business mobiles networks show impressive evolution of wireless communication. As assumed, in 2003 – 2005 the market of various mobile services will experience of large grow, especially driven by Internet-based WEB services.

We can point the great achievement in the evolution of mobile networks moving from second generation (2G) to the third generation (3G) systems. It shows the evolution of digital mobile cellular systems which mainly based on GSM family standards and specifications. It covers different aspects such as data rate transition, communication protocols, data formats and so on.

The next step of improvement of GSM was introduction GPRS and EDGE as transitional data technologies for the evolution of GSM [2]. GPRS uses packet mode extension, to support data applications and exploiting the already existing network infrastructure in order to save the operator's investments.

From another hand one of the very important aspects of wireless network evolution is represented by wireless local area networks (WLAN). WLAN systems are family of standards and technologies that provide high data rate applications and individual links and represent the attractive way of network communication. These systems represent the combination of well-elaborated LAN technique and mobile computing.

Whereas many WLANs need the infrastructure network that provides access to other networks, ad hoc wireless networks do not need any infrastructure. In such systems mobile node (devise) may act the role of intermediate node during multi-hop transmission of data from source node to destination node. Such kinds of systems play a complementary role to extend coverage for low power systems with arbitrary dynamic topology. The main challenges in design and deployment of ad hoc networks are issues of designing and development dynamic routing protocols and device's power consumption. One of the research groups which are dealing with these questions is MANet [7] working group which has been formed within Internet Engineering Task Force (IETF) to develop a routing framework for IP-based protocol in the ad hoc networks [8].

WLANs without need of an infrastructure with the very limited (up to 10-20 meters) coverage and consisted from tiny devices with low data rate transmission and power consumption could be as the new direction in wireless connectivity. This new emerging architecture is indicated as wireless personal area network (WPAN). In practical view such kind of network is aimed to establish interaction between various mobile and portable devises such as PDAs, tablet PCs, laptops, digital cameras, cell phones and many others. Bluetooth technology is the striking example of WPAN implementation.

On the Figure 1.1 is depicted the division of all wireless spectrum and the range of interaction for particular clusters. It is also showed what standards and technologies could constitute particular cluster.

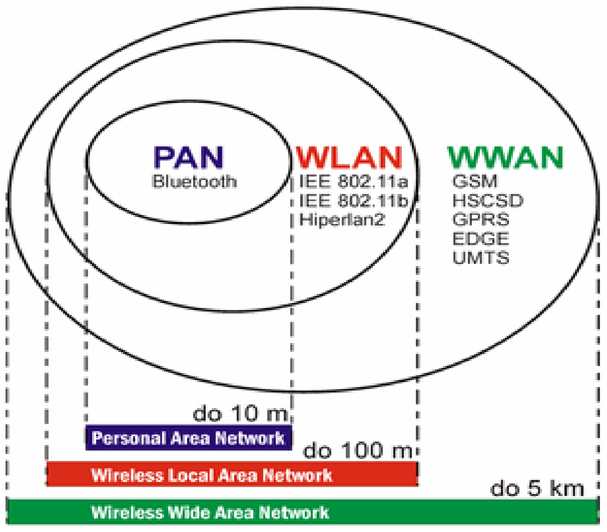


Figure 1.1 Wireless spectrum division

Involving wireless connectivity there is other type of networks – sensor networks. A wireless sensor network consists of number of sensors spread over particular (geographical) area. As pointed in [9] each sensor has wireless communication capability and some level of intelligence for signal processing and networking of the data. Sensor networks are getting spreading in the military, environmental monitoring, traffic control, security issues and others areas. Sensor network could be exploiting for: determine the values of some parameters at a given location; detect the occurrence of events of interest and estimate parameters of the detected events or event; classify a detected objects; track an object.

1.2 Related work

Related work concerning to the wireless ad hoc connectivity, different aspects of mobile communication, some issues of sensor networks and their integration, definition of short range devises can be found in [1], [9], [6].

These works coverers and explain main questions of ad hoc technology itself, challenges which could be faced during designing and building WPAN networks, trends and directions for close future in further evolution, review of related technologies which cooperate with WPAN.

1.3 Research problem statement

The questions which were related to research activity during composing this thesis describe the problem of sensor networks integration in general. Some questions are related to technical investigations such as design of stack of network protocols, design of application protocol, design of physical interface for sensor network.

Another part of question go to problem of sensor networks compatibility, how to manage particular kind of network, and were considered some question about possible uses cases for sensor networks.

I would like to point that the question of sensor networks integration and their utilisation is quite new in the research society and IT industry but this quest could be considered as main issue in the trend of development.

1.4 Contribution of the thesis

This thesis first of all tries to cover quite new problem which arise from context of wireless network integration, especially integration of WPAN networks. Since WPAN network enclosed itself there is need to combine it with legacy systems or infrastructure in someway. So we fist get the question of integration.

The second point is attempt to show possible utilisation such kind of network to provide some service based upon their functionality. It was developed pilot WPAN control system which has functionality to operate on sensor network, send different commands to nodes and get the result from sensors.

1.5 LR WPAN project

The main research and practical context of this thesis comes from my practical training which was passed in the term of LR WPAN project in Chydenious Institute, Kokkola. The LR WPAN project is driven by research team of 6 people who are working in the Kaustinen research lab.

The main activities of the project consist of:

- Research investigation in the are of wireless communication based on ad hock connectivity
- Design and implementation ad hoc routing protocols
- Design and development stack of network protocols
- Elaboration of methods for power consumption decrease
- Sensor networks integration
- Design and manufacturing mobile sensor networks nodes

1.6 Structure of the thesis

Chapter 2 introduces some concepts about wireless technologies. In particular is describes different trends in wireless connectivity, makes a review of various standards, introduce to the WPAN technology and shows directions and challenges there.

Chapter 3 covers the main idea of distributed computing. It introduces into multi-tires architecture which used in the pilot WPAN system.

Chapter 4 introduce into WEB services. It shows main standards and parts which constitute this technology.

Chapter 5 shows the pilot WPAN system as the main result of this thesis. It covers questions of system's architecture, network stack protocol and application protocol specification, describes functionality of the system, and shows some assumption about practical implementation.

Section "Conclusions" summarizes the context and done work of the Master Thesis.

Section "References" shows all used literature and sources

Section "Appendix A" includes listings of sources codes of developed system.

2 Wireless Private Area Network Domain

2.1 General description

Low rate WPAN is the standard described by the IEEE P802.15 specification and is the research question of The IEEE P802.15 Low Rate Study Group for Wireless Personal Area Networks. A Low Rate WPAN can be considered as a simple, low-cost communication network consisted of different mobile devices which based on the wireless connectivity with limited power consumption and operating range. The main objectives of an LR-WPAN are extremely low –cost of mobile devices (up to several $ per unit), ease of devices installation, reliable data transfer, short-range operation, and a reasonable battery life, while maintaining and exploiting a simple and flexible communication protocols.

According to [10] some of characteristics of an LR WPAN are:

- Over air data rates of 250 kb/s, 40 kb/s, and 20 kb/s.
- Operational range 10 – 100 meters
- Star or Peer-to-Peer operation.
- Allocated 16-bit short or 64-bit extended addresses.
- Allocation of guaranteed time slots.
- CSMA-CA channel access.
- Fully acknowledged protocol for transfer reliability.
- Low power consumption.
- Energy detection.
- Link quality indication.

- 16 channels in the 2450 MHz band, 10 channels in the 915MHz band, and 1 channel in the 868MHz band.

As pointed in draft standard specification [10] there are two different device types that can participate in an LR-WPAN network, a full function device (FFD) and a reduced function device (RFD). Devices related to the FFD type can operate in three various modes serving either as a PAN coordinator, a coordinator or a device. An FFD can communicate with RFDs or other FFD devices, while an RFD can only talk to an FFD. An RFD is intended for applications that are extremely simple, such as a light switch or temperature sensor; they do not have the need to get and end large amounts of data and may only associate with a single FFD at a time.

2.2 WPAN components

Different mobile devices with wireless communication can constitute WPAN networks, but from structural point of view they can be classified as an RFD or an FFD type as pointed in section 2.1. Two or more devices within a personal communicating area and on the same physical channel constitute a WPAN. However, a network shouldl include at least one FFD, operating as the PAN coordinator. Usually PAN coordinator has more important role in the WPAN network and as the rule it has PC connection interface that allow being a sink for that network. All ingoing and outgoing communication for WPAN network is fulfilled through the PAN coordinator in such case.

2.3 Network issues and topologies

WPAN networks can have both static and dynamic topologies, but mostly mobile WPAN networks are formed dynamically. According to descriptions in [1] nodes in WPAN networks which exploit ad hoc connectivity are free to move in arbitrary way and organize themselves in random topology therefore such kind of network’s wireless topology may change rapidly in time and unpredictably.

Due the dynamic topology of WPAN networks we face the main challenge of WPAN technology – the data routing and ways to resolve this task. Routes between nodes in an ad hoc network could exploit different routing protocols and may include as single hop as multiple hops. Ad hoc networks with multi- hop communication is called as ‘‘multi-hop wireless ad hoc networks’’. Nodes which placed beyond the range of communication need to use intermediate nodes to transfer the data hop by hop. This principle of dynamic multi-hop topology is depicted on the Figure 1.2

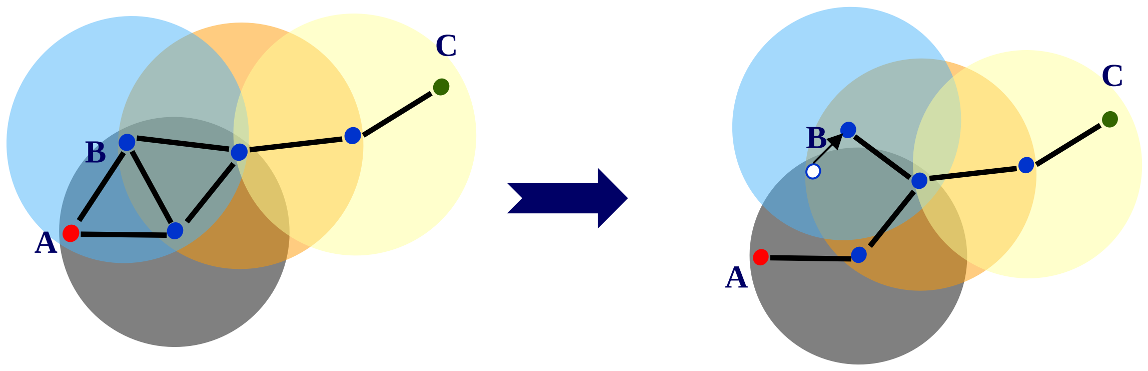


Figure 1.2 Dynamic multi-hop ad hoc topology

From point of typology’s shape and depending of applications requirements, the LR-WPAN may have and operate in ether of two kinds of topologies: the star or peer to peer topology.

2.3.1 Star topology

The communication networks which based on star topology are conducting between network’s devices and single central network controller called PAN coordinator. A PAN coordinator may also have a specific application but it can be used to initiate, terminate or route communication around the network [1]. In the sensor networks such central network controller plays the role of sink for the whole network or its part.

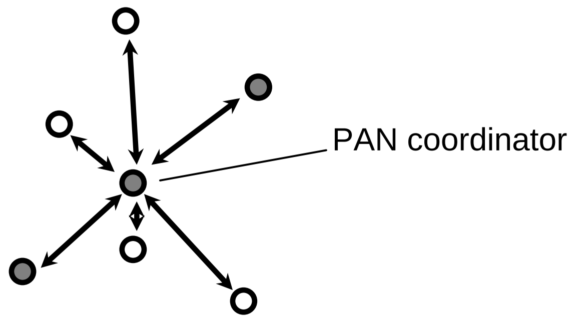


Figure 1.3 Star topology

Usually PAN coordinator drives all ingoing and outgoing network traffic and the main loading, that’s why it may has permanent power supply, while the other devices have most likely the battery power. Some examples of applications which use star topology include home automation, PC peripherals, toys and games, and telemedicine [10].

2.3.2 Peer to peer topology

The same approach of PAN coordinator is used in peer to peer topology but unlike star topology any device can establish communication one to other as long as they are in the range of communication. Peer to peer topology allows designing and implementing more complex network formation to solve more wide range of tasks. These tasks could relate to applications such as industrial control and monitoring, wireless sensor networks, asset and inventory tracking, intelligent agriculture, and security would benefit from such a network topology as exampled in [10]. A peer-to-peer network can be ad hoc, self-organizing and self-healing. Also it may use multiple hops communication to route data from source to destination on the network.

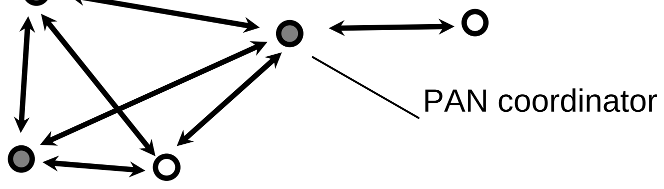


Figure 1.4 Peer to peer topology

2.4 Routing protocols

The nature of WPAN networks, the limitation of its resources put on strict conditions for designing and efficient implementation reliable routing strategy and since that is being very challenging task. Routing strategy has to fit limited resources, use intelligent behaviour and at the same time being reliable and adaptive to the dynamic topology of WPAN networks. At the same time with this, routing strategy has to provide sufficient level of QoS up to request of various applications and users.

In the solutions of choosing routing strategy it is used three approaches and types of routing protocols: table driven routing or proactive routing in which all routes are already stored and are ready for use; source initiated on-demand routing or reactive routing in which a route is discovered only when there is data packet to be routed; hybrid routing – the combination of proactive and reactive routing strategies.

2.4.1 Proactive routing protocols

Proactive routing protocols use approach in which each node keeps routing information to every neighbouring node in the same network. The information about route is usually kept in the different routing tables. These tables are periodically updated and/or if the network topology changes. The difference between proactive-family protocols exists in the way the routing information is updated, detected and the type of information kept at each routing table [2].

2.4.1.1 Destination Sequenced Distance Vector (DSDV)

DSDV - is based on the classical Bellman-Ford routing algorithm. Each node in this protocol maintains routing tables of all destinations and number of hops to each destination. It uses full dump or incremental packets to reduce network traffic generated by route updates. The full dump packet carries all the available routing information and the incremental packet carries only the information changed since the last full dump [2].Only improvement which is implemented in this protocol is avoidance of routing loops in a mobile network of routers. With this improvement, routing information can always be readily available, regardless of whether the source node requires route or not.

2.4.1.2 Wireless Routing Protocol (WRP)

WRP exploits a path-finding algorithm with the exception of avoiding the count-to-infinity problem by forcing each node to perform consistency checks of predecessor information reported by all its neighbours. Each node according to this protocol keeps 4 tables - Distance table, Routing table, Link-cost table and Message retransmission list table. Link changes are propagated using update messages sent between neighboring nodes. This protocol avoids count-to-infinity problem.

2.4.1.3 Source Tree Adaptive Routing (STAR)

STAR - the protocol which based on the link state algorithm. Each router maintains a source tree, which is a set of links containing the preferred paths to destinations [2]. This protocol does not require periodic routing updates nor does it attempt to maintain optimum routes to destinations.

2.4.2 Reactive routing protocols

The family of reactive routing protocols were designed to reduce that overheads and drawbacks of proactive protocols by maintaining information for active routes only. In [2] pointed that it means that routes are determined and maintained for nodes that require sending data to a particular destination.

Reactive routing protocols can be divided into two main groups: sources routing and hop-by-hop routing. In Source routed on-demand protocols each data packets carry the complete source to destination address and each intermediate node which takes part in transition forwards this packets according to the packet’s destination [2]. Therefore these intermediate nodes do not need maintain the up-to-date information of routing to forward the packet correctly to the destination.

Group of hop-by-hop protocols uses the method when packet holds addresses of destination and next hop only. Thus during packet transmission every intermediate node uses its own routing table to determine the path to the destination. The advantage of this strategy is that routes are adaptable to the dynamically changing of topology of network.

There are number protocols which constitute the reactive routing protocols have been designed to increase the performance of routing strategy are:

2.4.2.1 Dynamic Source Routing (DSR)

DSR belongs to source group routing protocols. For this protocol, mobile nodes are required to maintain route caches that contain the source routes of which the mobile is aware. Entries in the route cache are continually updated as new routes are learned. There are 2 major phases of the protocol - route discovery and route maintenance Route discovery uses route request and route reply packets. Route maintenance uses route error packets and acknowledgements.

2.4.2.2 Ad Hoc on Demand Distance Vector Routing (AODV)

AODV uses routing algorithm which based on the DSDV algorithm and the improvement is on minimizing the number of required broadcasts by creating routes on an on-demand basis. A path discovery is initiated when a route to a destination does not exist. Broadcast is used for route request. Link failure notification is sent to the upstream neighbors and this algorithm requires symmetric links

2.4.2.3 Temporary Ordered Routing Algorithm (TORA)

TORA is highly adaptive, loop-free, distributed routing algorithm based on the concept of link reversal. It is proposed to operate in a highly dynamic mobile networking environment. It is source initiated and provides multiple routes for any desired source/ destination pair. This algorithm requires the need for synchronized clocks. There are three basic functions of the protocol, namely route creation, route maintenance and route erasure.

2.4.2.4 Relative Distance Microdiversity Routing (RDMAR)

RDMAR - This type of routing estimates the distance, in radio loops, between two nodes using the relative distance estimation algorithm. It is source initiated, having features found in ABR. This routing protocol limits the range of route searching in order to save the cost of flooding a route request message into the entire wireless area.

The summarizing of the various features of both reactive and proactive routing protocols is listed in the follow tables.

Table 2.1 Proactive routing protocols

Protocols	**RS**	**Tables #**	**Frequency of updates**	**HM**	**Critical nodes**	**Features**
STAR	H	1 and 5 list	Conditional	No	No	Employes LORA and/or ORA. Minimize CO
WRP	F	4	Periodic	Yes	No	Loop freedom using predecessor info
DSDV	F	2	Periodic and as required	Yes	No	Loop free

RS - routing structure; HM - hello message; H - hierarchical; F- flat; CO - control overhead; LORA - least overhead routing approach; ORA - optimum routing approach

Table 2.2 Reactive routing protocols

Protocols	**RS**	**Multiple Hops**	**Beacons**	**Route metric method**	**Route configuration strategy**
DSR	F	Yes	No	SP, or next available in RC	Erase route the SN
AODV	F	No	Yes	Freshest & SP	Erase route then SN or local route repair
TORA	F	Yes	No	SP, or next available	Link reversal & Route repair
RDMAR	F	No	No	Shortest relative distance or SP	Erase route then SN

RS - routing structure; H - hierarchical; F - flat; RT - route table; RC - route cache; SP - shortest path; SN - source notification

2.5 Power consumption

Devices which constitute WPAN network are small and limited in resources and then rely on limited battery energy to perform their activity. There are number of groups and research projects, which are working in the area aiming to reduce energy consumption using various approaches. These are issues of energy reducing at the hardware level [3], at the levels of protocol stack [4] in particular on routing and transport protocols [5]. These approaches can be described in follow:

2.5.1 Power-Aware Routing

Power-aware algorithms choose the routing of data according to some predefined power cost functions [6]. As described in [11], they introduce the term of minimum transmission energy (MTE) routing scheme, which selects the route using the least amount of energy to transport a packet from source to destination. The purpose of using such function is to maximize the network lifetime, which is defined as the period from the point of time when the network starts functioning to the point of time when the first node loses its energy supply. In [13], the problem of finding the most beneficial source rate allocation and flow control strategy, given a required network lifetime, as defined in [12], is posed. Each source is associated with a utility function that increases with the traffic flowing over the available source-destination routes.

2.5.2 Transmission Power Control

The questions of transmission power control (TPC) have been quite good investigated in the context of GSM cellular networks such as TDMA/FDMA. This technique uses the algorisms aimed to decrease the effect of mutual interference instead preserving the energy. That’s why, most of the solutions using TPC rely on a centralized control are it is not presently reflected in wireless ad hoc networks. Some works in this area show some theoretical studies [14] and simulations, that by applying TPC in ad hoc packet networks, concerning to the reducing of energy consumption both in a single-hop environment [16] and in a multi-hop environment [17] can be obtained. That works are dealing how TPC is employed to control the topology of wireless ad hoc networks. In [55, 56], power control is proposed as a part of the multiple access protocol for the class of CSMA/CA protocols and, more in general, for contention-based multiple access protocols. For specification of IEEE 802.11 standard for WLAN, they concerning so called distributed coordination function (DCF). This standard defines an optional point coordination function (PCF) also. Power control is undesirable for DCF when the number of hidden terminals is likely to increase, which, in turn, results in more collisions and in more energy consumption. From the other side it can be effective in the PCF access mechanism, since there is no hidden terminal problem [15].

2.6 Conclusions

WPAN is a quite promising technology which can change the view of ubiquitous computing and future heterogeneous communication. Possible application scenarios for these short-range communications, together with the existing and emerging technologies to support them, have been presented. Low power consumption, operation in the unlicensed spectrum, coexistence between the WPAN and the WLAN, low cost, and small package size are some of the most important technical challenges to be faced.

The great volume of research work in ad hoc routing protocols should be put in the context of concrete MAC layer realization. The future work on the short-range wireless network must apply the routing protocols in a way that is adapted both to the channel access/transmission conditions and the application requirements.

3 Distributed system architecture

3.1 History of Distributed Computing

The history of distributed computing began around 1970 with the emergence of two technologies:

- Minicomputers, then workstations, and then PCs
- Computer networks (eventually Ethernet and the Internet) [23]

Mini-computers such as Digital's PDP-11 and mainframes could be considered as a beginning of the distributed computing. Mini-computers worked on timesharing operating system such as Multix, Unix, RSX, RSTS. Users used the same machine for their work and it looked to the users as if they worked each with the own machine.

Mini-computers were slower than their Big Brother mainframes made by IBM, Control Data, Univac, etc [Distributed system]. When they became popular they had not spread over large number of users as the mainframes had. The way to scale mini-computers was to buy more of them. The trend toward cheaper machines made the idea of having may minis a feasible replacement for a single mainframe and made it possible to contemplate a future computing environment where every user had their own computer on their desk, that is a computer workstation.

Work on the first computer workstation began in 1970 at Xerox Corporation's Palo Alto Research Centre (PARC). This computer was called the Alto. Over the next 10 years, the computer system's group at PARC would invent almost everything that's interesting about the computer workstations and personal computers we use today.

3.2 Multi-tier distributed architecture

Multi-tier computing allows to developers to write programs that are distributed across a different network. It allows also to build applications that run on different application servers, and that can be called by client programs. The application server can resides on one machine, the client application on another machine. The two programs work in tandem, each using the resources available on their own machine, and neither impinging significantly on the resources available on the other machine.

One of the necessary functionality for multi-tier computing is the ability to make remote function calls. A remote function call is a call to a method on a server that resides on a different machine than the one you are currently using. If you are running program A on machine X, and you call a function on program B that is running on machine Y, then you are doing multi-tier computing.

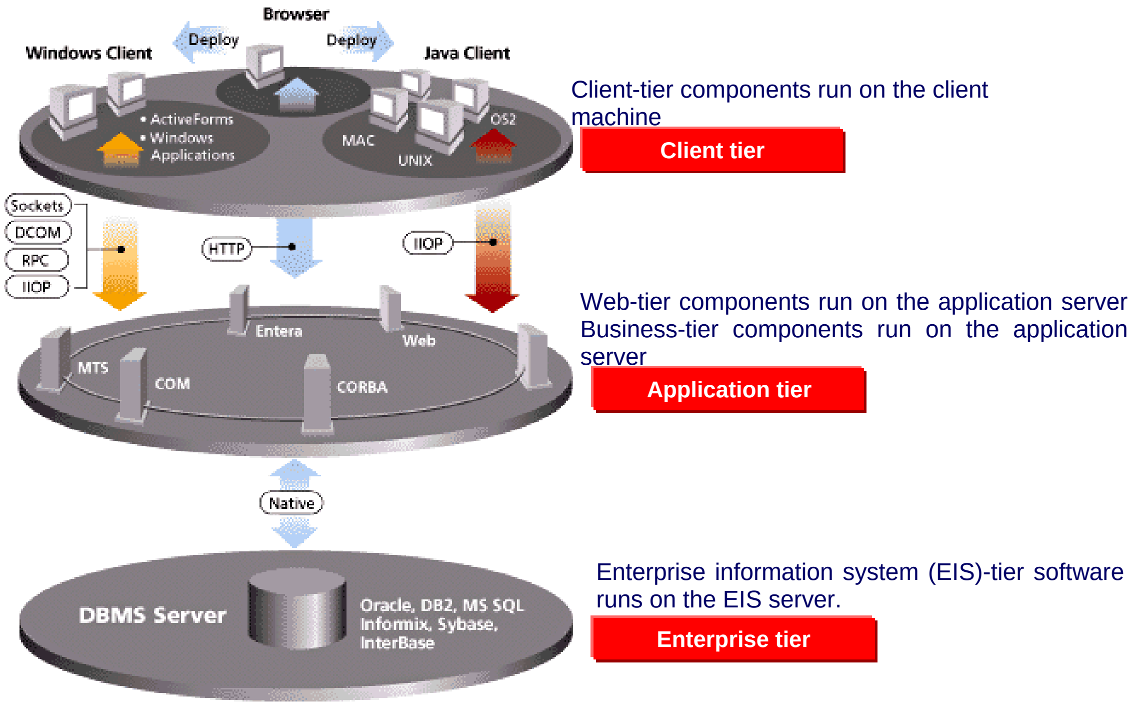
In brief description, multi-tier distributed computing, in its simplest possible form, is about making remote function calls. If you have a client program on one machine that calls a function or method on a binary file located on a second machine, then you are doing distributed computing. The crucial point here is that you don't load the server program or DLL into the memory space of your own machine; instead, the function call executes on the remote machine.

On the Figure 3.1 is depicted the multi-tired architecture of distributed computing. The distributed platform uses a distributed multitiered application model for enterprise applications. As the rule, application logic is divided into components according to purpose and function, and the various application components that make up a distributed enterprice application are installed on different machines depending on the tier in the multitiered environment to which the application component belongs [25].

3.2.1 Tires of distributed architecure

As the rule, enterprice applications are distributed over three locations: client machines, the application server machine, and the database or legacy system machines at the back end this distributed architecture is called three-tired distributed architecture and consists of:

- Client-tier components run on the client machine
- Business-tier components run on the application server
- Enterprise information system (EIS)-tier software runs on the EIS server



Browser
Windows Client
Deploy
Deploy
Java Client
• ActiveForms
• Windows Applications
MAC
UNIX
OS2
Client-tier components run on the client machine
Client tier
Sockets
DCOM
RPC
IIOP
HTTP
IIOP
Entera
Web
MTS
COM
CORBA
Web-tier components run on the application server
Business-tier components run on the application server
Application tier
Native
DBMS Server
Oracle, DB2, MS SQL Informix, Sybase, InterBase
Enterprise information system (EIS)-tier software runs on the EIS server.
Enterprise tier

Figure 3.1 Multitiered Applications Architecrure

Three-tiered applications that run in this way extend the standard two-tiered client and server model by placing a multithreaded application server between the client application and back-end storage. One of the interpretation of three tires distributed system can be compared with MVC pattern. According to this approach we divide data, logic functionality and representation into three separate parts. Data are stored as they are in their own model. Controller has logic and makes the appropriate functionality upon data. View describes in what way data should be represented. So we can see close relationship between three tires distributed architecture and MVC pattern.

3.2.1.1 Client tire

A client tire can be a Web client or an application client. A Web client consists of two parts: dynamic Web pages containing various types of markup language (HTML, XML, and so on), which are generated by Web components running in the Web tier, and a Web browser, which visually represents the pages received from the server. A Web page received from the Web tier also can include different embedded application components like Java Applets or ActiveX components.

An application client runs on a client machine and provides a way for users to handle tasks that require a richer user interface than can be provided by a markup language. It typically has a graphical user interface (GUI), for instance Java Swing API, but a command-line interface is certainly possible [25].

Application clients directly access enterprise beans running in the business tier. However, if application requirements warrant it, an application client can open an HTTP, Sockets or DCOM connection to establish communication with a servlet running in the Application tier.

3.2.1.2 Application tire

Application tire is the tire of application servers where all business logic are stored. Application server servers and manage that logic which solves or meets the needs of a particular business domain of an enterprice or organisation such as manufactoring, banking, or finance, is resided on the papllication tire. For example in J2EE technology this logic is encapsulated into EJB – enterprice java beans. An enterprise bean receives data from client programs, processes it (if necessary), and sends it to the enterprise information system tier for storage.

3.2.1.3 Enterprise tire

The enterprise information system tier (or just enterprice tire) handles EIS software and includes variouse enterprise infrastructure systems such as enterprise resource planning (ERP), mainframe transaction processing, different database systems (Oracle, Sybase, MS SQL), and other legacy information systems. For example, distributed application components might need access to enterprise information systems for database connectivity.

3.3 Java Connector Architecture

The Java Connector Architecture defines a standard architecture for connecting the J2EE platform to heterogeneous EIS systems such as ERP, mainframe transaction processing, database systems, and legacy applications whis is not written in the Java programming language [26]. It defining a a set of scalable, secure, and transactional mechanisms, the J2EE Connector architecture enables the integration of EISs with application servers.

The J2EE Connector architecture describes standards and provides its specification that enables vendors of EIS to provide standard resource adapters for its EIS. The resource adapter plugs into an application server, providing connectivity between the EIS, the application server, and the enterprise application. If an application server vendor has extended its system to support the J2EE Connector architecture, it is assured of seamless connectivity to multiple EISs. An EIS vendor needs to provide just one standard resource adapter which has the capability to plug in to any application server that supports the J2EE Connector architecture [26].

Multiple resource adapters (that is, one resource adapter per type of EIS) are pluggable into an application server. This capability enables application components deployed on the application server to access the underlying EIS systems [26].

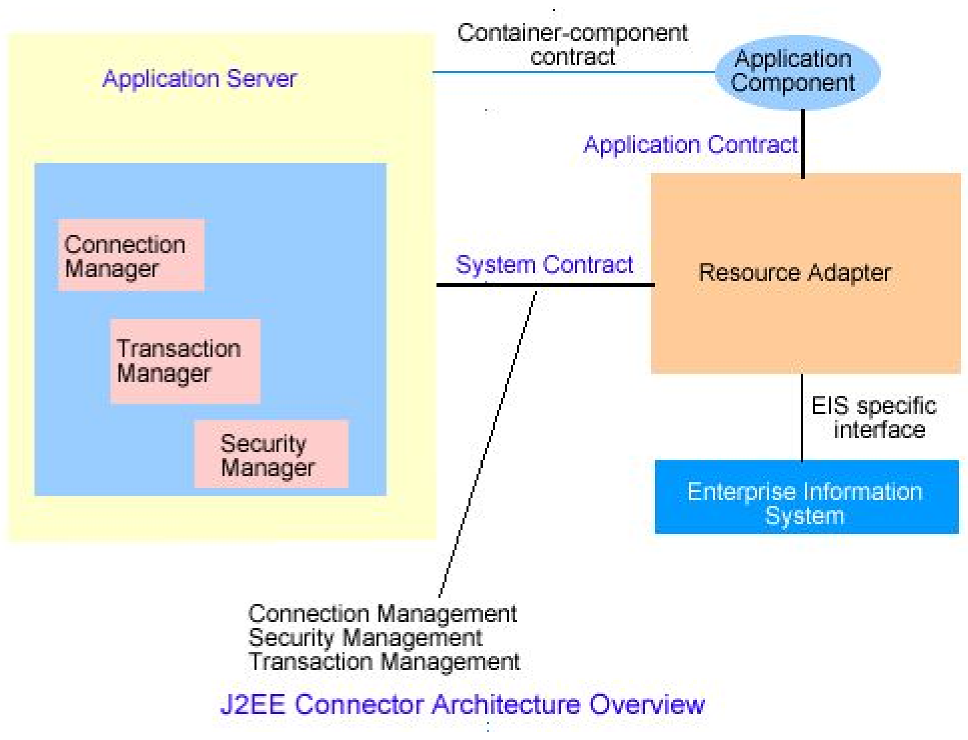


Figure 3.2 Common schema of Java Connector Architecture [26]

The approach of JCA helps to think about sensor network integration and combination of WPAN with current legacy systems. JCA describes specification of resource adapters for heterogeneous resources. We entirely can consider WPAN network as some kind of resource which could be integrated with others resources.

So to integrate WPAN network with some legacy system or other distributed application we need to provide resource adapter for this WPAN network. Resource adapter is developing according JCA specification and allows to system to be integrated into J2EE distributed platform.

Since JCA covers such issues as connection, transaction and secure management; common client interface (CCI) and has built-in implementation of them, there is no need to develop such functionality from the scratch. All what you need is to provide the new resource adapter or to enhance current one. For example if the connection interface will be change from RS232 to Bluetooth you need to provide new connection factory for Bluetooth connection.

4 WEB services technique

First of all it should be mentioned that WEB services are the big part of distributed computing paradigm which was considered in the previous topic. But WEB services technique is so big and complicated part of computer science that it might be distinguished into individual area.

4.1 Service-Oriented Architectures for WPAN networks

Service-oriented architectures are all about connections and data interaction [20]. A web service is a network accessible interface to application functionality, built using standard Internet technologies, which is opened for other applications, usually with not human interaction. In other words, if an application can be accessed over a network using a combination of protocols like HTTP, SMTP, or Jabber, then it is a web service. Web Services are tightly exploited with the use of XML. Web services using XML are the most common connections in a service-oriented environment.

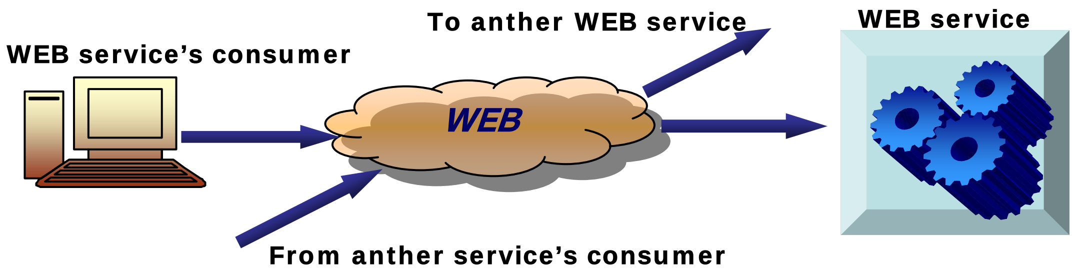


Figure 4.1 WEB service network communications

Mostly, the web services which are deployed on the Internet today are HTML based web sites. Thus, the services which are publishing, managing, searching, and retrieving content—are accessed through the use of standard protocols and data formats: HTTP and HTML [20]. Client applications (web browsers) that understand these standards can interact with the application services to perform tasks like ordering books, obtaining data from environment monitoring system, sending greeting cards, or reading news, planning the business route and many others.

Web services are a messaging framework. Web services have the requirement to obey to the set of standard Internet protocols to be capable of sending and receiving messages. The most common form of web services is to call procedures running on a server, in which case the messages encode "Call this subroutine with these arguments," and "Here are the results of the subroutine call."[21]

4.2 The Web Service Technology Stack

As showed in [21] the web services architecture similar to network stack protocol is implemented through the layering of five types of technologies, organized into layers that build upon one another Figure 3.2

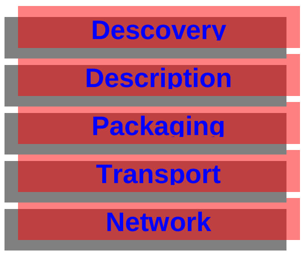


Figure 4.2 The web service technology stack

Because the architecture of the web services are divided on the separate parts stack it reflects separate business problem, and you only have to implement those pieces that make the most sense at any given time. It allows reusing designed parts and when a new layer of the stack is needed, you do not have to rewrite and rebuild your previous infrastructure and just to provide the support of a new form of exchanging information or a new way of authenticating users.

Even in such architecture, the layered stack of web services does not provide a complete solution to many tasks of business. For example they don't cover such issues like security, identity, trust, workflow and other business concerns.

Web services can be easily implemented with the combination of the WPAN technology. Actually you need to wrap existing sensor network by the web service. Such solution allows incorporating and integrating heterogeneous networks into global interchanging environment. It also allows collaborating with different legacy systems, and combining new WPAN solutions with present business systems.

4.3 Web Services Explained

When the Web services were announced they were described as the connection and interchanging technology of the future. In [22] described that Web services are software applications that can be discovered, described and accessed based on XML and standard Web protocols over intranets, extranets, and the Internet. The techniques which constitute the Web service's layer stack are WSDL, UDDI, SOAP, XML and describes bellow.

4.3.1 Using the Web Services Description Language (WSDL)

WSDL is the way to describe the communication details and the specific of message intended to service's application. Figure 4.3 illustrates the use of WSDL. At the left is a service provider. At the right is a service consumer. The steps involved in providing and consuming a service are:

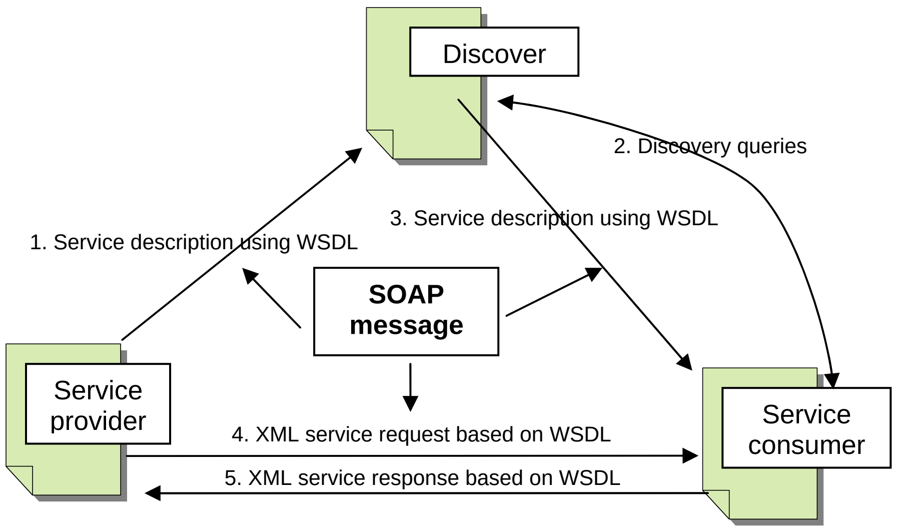

Discover
2. Discovery queries
3. Service description using WSDL
1. Service description using WSDL
SOAP message
Service provider
Service consumer
4. XML service request based on WSDL
5. XML service response based on WSDL

Figure 4.3 WEB service descriptions

- A service provider uses WSDL to describe describes the specific of its service. This definition is published to a directory of services. The directory could use Universal Description, Discovery, and Integration (UDDI).
- A service consumer which needs to exploit the service functionality sends one or more queries to the directory to locate a service and determine the way of communication with that service.
- Part of the WSDL provided by the service provider is passed to the service consumer. This tells the consumer of service what format the requests and responses are for the service provider.
- The service consumer uses the WSDL to send a request to the service provider.
- The service provider provides the expected response to the service consumer.

So, as showed in [20] WSDL describes the operational information – the location of service, the functionality of service and how to talk (or invoke) the service.

4.3.2 Using Universal Description, Discovery, and Integration (UDDI)

UDDI might be considered as the “phone book” of Web services. It has the register for storing essential business information about services. The idea is that the UDDI registry can be searched in various ways to obtain contact information and the Web Services available for various organizations. The information provided in a UDDI business registration consists of three components:

- white pages of company contact information
- yellow pages that categorize businesses by standard taxonomies
- green pages includes the technical information about services

The figure 4.4 demonstrates this concept.

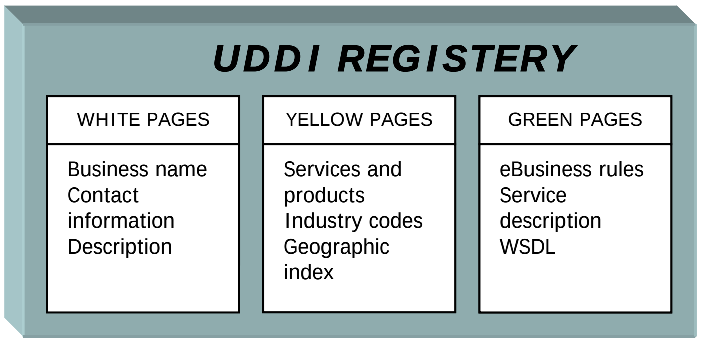


Figure 4.4 UDDI Registry

4.3.3 Using SOAP

All the messages shown in Figure 4.3 are sent using SOAP. SOAP at one time stood for Simple Object Access Protocol. SOAP essentially provides the envelope for sending the Web Services messages. SOAP generally uses HTTP, but other means of connection may be used. HTTP is the familiar connection we all use for the Internet. In fact, it is the pervasiveness of HTTP connections that will help drive the adoption of Web Services.

So as described in [21], SOAP's place in the web services technology stack is as a standardized packaging protocol for the messages shared by applications. The specification defines nothing more than a simple XML-based envelope for the information being transferred, and a set of rules for translating application and platform-specific data types into XML representations. SOAP's design makes it suitable for a wide variety of application messaging and integration patterns. This, for the most part, contributes to its growing popularity.

4.4 Conclusions

To utilize the functionality of the WPAN system which is showed in the details in chapter 5 by other application it should be developed WEB services according to standards described in this chapter. For example it possible to use some tools for WEB services creation which have wizards, as the result there will be generated WSDL description files and WEB services archive for deploying it to the application server.

5 Mobile services

5.1 Introduction

This Master Thesis uses mostly all materials and practical results which were obtained during practical training passed in frame of research project which is going in Chydenius Institute – Kokkola University Consortium. The project is aimed to construct the prototype of wireless private network which node's communication is based upon ad hoc connectivity.

The first phase of project was to construct mobile WPAN node form electronic components and program it to communicate with other same devises. The main feature of these devises is they communicate on short range distances with low data transmit range. The main crucial features are these devisees have limited computation power (the capacity of Flash RAM of microcontroller is only 128 Kb), low power consumption, small size.

The next phase is to implement one of the AD HOC routing protocols for node's communication. This will allow building real prototype of WPAN network with dynamic topology that is nodes could change their place in real time.

Simultaneously with building WPAN network it was intended to develop WEB-based interface for external communication with WPAN network. One of the purpose of this interface to have connection to WPAN network through Internet, which allow to administrate WPAN network, incorporate different heterogeneous WPAN networks, development various services.

Upon this task pilot WPAN system was developed to show possible ways of sensor network integration with distributed environment. It utilizes the WPAN node's functionality and provides it to the user.

5.2 WPAN services architecture

By the meaning of "WPAN service" in this chapter we assume that functionality which were originally built into mobile nodes and provided to the end user. By end user we assume any device which is interact in standard way (HTTP, WAP) with WPAN application and exploits this functionality. As mentioned previously this master thesis is based on the results of practical training and the initial task for WPAN project was the development of simple service to show the possibility of idea of services. There were no any requirements for the choice of platform, computer design facilities, the architecture of system, so this was one of the project's task.

During the analysis of problem domain and requirements for the system's functionality we got the follow architecture of system which is depicted on the figure 5.1.

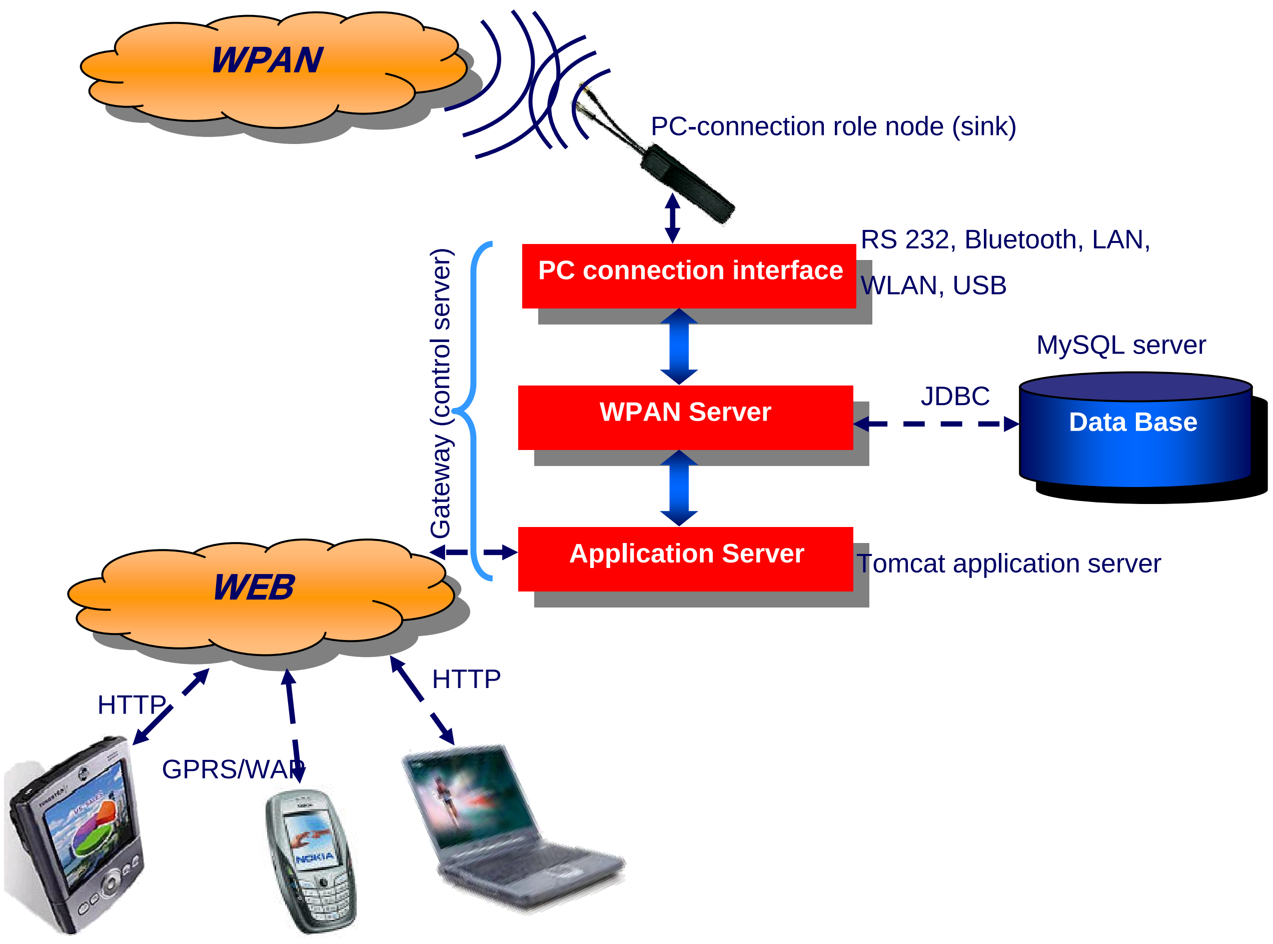


Figure 5.1 Architecture of WPAN distributed system

The main idea of this system is to integrate self-enclosed WPAN network with Internet using the approach of distributed architecture. Such approach allows utilizing all power of distributed systems architecture.

It is shown that whole system consists of four main parts:

- WEB application
- Application server
- WPAN server
- Data Base

The interaction scenario of above architecture is follow:

End user sends the standard HTTP (POST/GET) or WAP request through the any web browser from the Internet. Application server gets the request and involves appropriate servelet to process it. Then this request sends to the WPAN server via open socket connection. WPAN server codes the request into SLIP format and sends it via physical connection to the WPAN network and starts to wait for the response. When it gets the response from WPAN it transfers it back to the user.

The PC connection interface layer has a program code for the physical connection to PC-connection role node. It uses one of the connection factories to establish connection (RS232 in our case). The factory method easy allows to add new factories if the connection interface will be changed. For example we need to obtain Bluetooth driver, wrap it to the factory and add this factory to the connection method.

5.2.1 WPAN distributed system architecture description

The core element in the whole system is the WPAN server which plays role of the bridge between the distributed environment and WPAN network. From one side WPAN server exploits the low level API for sensor network communication; from another side it opens socket communication for incoming requests. Besides the physical communication, WPAN server fulfils functionality of management and transaction of incoming and outcoming responses and requests. As the program framework for the whole architecture it was decided to use J2EE platform with appropriate API.

All core logic of WPAN server is encapsulated in the WpanServer class. The main class which can start the WPAN server is WpanServerStart. This class includes swing elements which represent to administrator the WPAN server window with all information about server starting and server working such as: server’s IP address and port, information about server’s starting stages, information about all requests going through server.

5.2.1.1 Physical sensor network connection

To realize the physical communicating with WPAN sensor network it was chosen the Java Communications API. The Java Communications API can be used to write platform-independent communications applications for technologies such as voice mail, fax, and smart cards. Two versions of the Java Communications API implementations -- 2.0.3 for Solaris/SPARC, and 2.0 for Windows and Solaris x86 -- are available for use [24].

The Java Communications API contains support for RS232 serial ports and IEEE 1284 parallel ports. With updated functionality, developers can:

- Enumerate ports available on the system
- Open and claim ownership of ports
- Resolve port ownership contention between multiple applications
- Perform asynchronous and synchronous I/O on ports
- Receive Beans-style events describing communication port state changes

Since the sink of sensor network has RS 232 interface to be connected to the PC (gateway) it was enough to use Java Communications API, which provides methods for serial port connection. But WPAN server is not limited by this connection method only and preserves connection factory for new types of connections. For instance it might be USB, Bluetooth or WLAN types of connection.

5.2.1.2 Datagram encapsulation

In paragraph 5.2.1.1 described that WPAN server use serial connection as the physical connection with “PC-connection role” node. So server listens for incoming datagram on RS-232 port and fires DATA_AVAILABLE event if new datagram comes from sensor network. By the network stack protocol specification, datagrams are wrapped by the SLIP protocol.

SLIP (Serial Line IP) is the simple form for encapsulation of IP datagrams which are transferred via serial port. SLIP is very used especially for connection of home computers to the Internet via fast-speed modems.

The wrapping of the SLIP datagram is adhered by the follow rules:

- At the tail of IP datagram it is added the special symbol END (0xC0). For reliability the same symbol is added at the begin of frame

- If there is the END byte code in the body of IP datagram it must be replaced by the sequence of 2 bytes: 0xdb, 0xdc. The special symbol 0xdb is called as ESC symbol of SLIP frame

- If there is the ESC symbol in the body of IP datagram it must be replaced by the sequence of two bytes: 0xdb, 0xdd

On the figure 5.2 is depicted the example of encapsulation of IP datagram by the SLIP protocol. We can see the “worst” case where we involve all three rules of SLIP protocol.

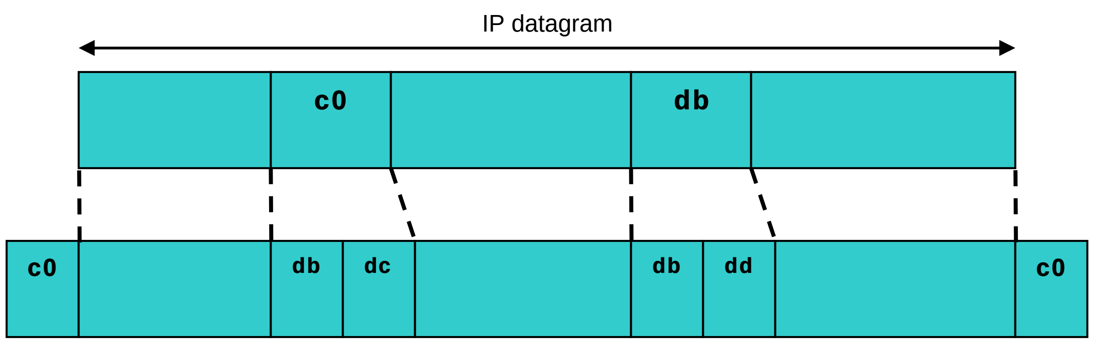


Figure 5.2 Encapsulation of IP datagram in SLIP protocol

WPAN server fulfils the logic of coding/decoding IP frames by the specification of SLIP protocol. The main logic of coding/decoding by the SLIP protocol is done with the SlipFrame class.

5.2.1.3 Traffic storage

Another task for WPAN server is the storing all incoming and outcoming traffic in the local data base for the further processing these data. For this purposes it was designed simple database schema and chosen MySQL DBMS as data base server.

WPAN delegates the connection to the data base to MYSQLConnect class which includes all logic for connection to MySQL server. After established connection MYSQLConnect class returns the Connection object to the WPAN server and then WPAN server is able to perform storing all traffic data.

5.2.1.4 Sockets connection

To provide the standard way for connection to WPAN server from other applications it opens socket connection. Actually WPAN server opens multi sockets and listens for requests on the port 3045. The logic of sockets connection is encapsulated in two classes – SocketServer and MultiSocketClient. Such way of WPAN server allows to interact to any application with the sockets API.

On the Figure 5.3 is depicted UML class diagram of WPAN server with all classes which constitute it.

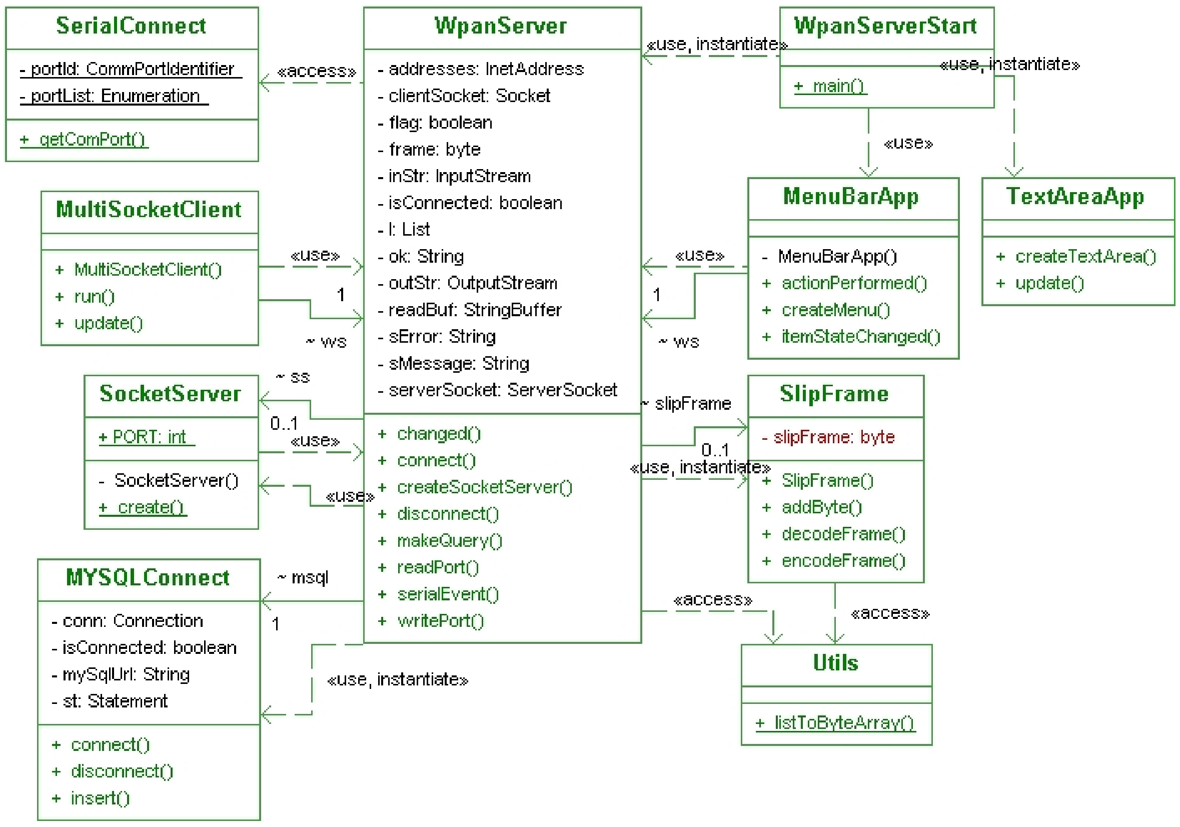


Figure 5.3 UML class diagram of WPAN server

5.2.1.5 WPAN control application

As the pilot application for WPAN network which provides simple service it was developed control application for getting two parameters from the WPAN sensors. This application just tries to show possible implementation of such approach to provide different mobile services.

Control application is build with the distributed paradigm and allows getting parameters of temperature and lightness controlled by WPAN nodes. Also it allows to carry out some management tasks such as setting and getting different node’s parameters. The interaction between user and application is taking place by “thin client” user interface which is compatible with any WEB browser.

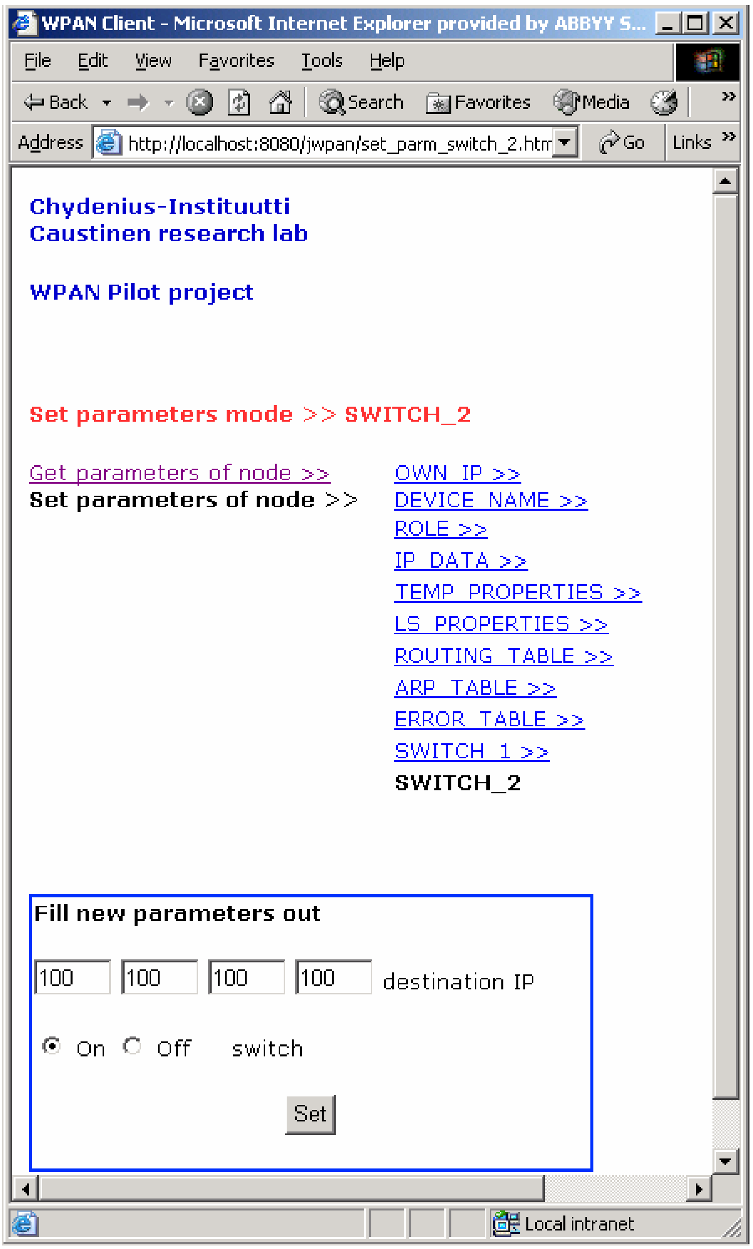
User interface of application based on simple HTML language and include HTML form to enter all necessary data. Forms use standard POST/GET HTTP methods to send requests to the application server. On the application server side there are different servlets which are involved upon every browser request.

It was chose Java JSP/Servlets technique to build server part of this system and Apache Tomcat as the application server. All servlets are resided in the *Tomcat_Home*\Tomcat 5.0\webapps\jwpan\WEB-INF\classes\ where *Tomcat_Home* is the root directory for Tomcat server installation.

Table 5.1 Description for servlets functionality

Servlet name	Description
GetParameterAll.class	This servlet returns result upon GET method for all twelve parameters (own IP, device name, temperature, lightness, etc.)
DeviceNameSetParameter.class	This servlet returns result upon SET method for set up DEVICE_NAME
OwnIpSetParameter.class	This servlet returns result upon SET method for set up OWN_IP
RoleSetParameter.class	Set the PC-connection role for the mobile node
Switch1SetParameter.class	Set on /set off the switch #1 parameter of the mobile node
Switch2SetParameter.class	Set on /set off the switch #2 parameter of the mobile node

On the figure 5.4 is depicted the screenshot of the user interface of the control WPAN application. This is the one of the example where user can control WPAN’s nodes. In this case user can change the parameter SWITCH_2 of the mobile node, send this request to the sensor network and in such way switch-on or switch-off the switch on the board of node with appropriate IP address.


WPAN Client - Microsoft Internet Explorer provided by ABBYY S...

Address http://localhost:8080/jwpan/set_parm_switch_2.htm

Chydenius-Instituutti
Caustinen research lab

WPAN Pilot project

Set parameters mode >> SWITCH_2

Get parameters of node >>
Set parameters of node >>

OWN_IP >>
DEVICE_NAME >>
ROLE >>
IP_DATA >>
TEMP_PROPERTIES >>
LS_PROPERTIES >>
ROUTING_TABLE >>
ARP_TABLE >>
ERROR_TABLE >>
SWITCH_1 >>
SWITCH_2

Fill new parameters out

100 100 100 100 destination IP

[✓ On] Off switch

Set

Local intranet

Figure 5.4 Control application – example

On the picture 5.5 and 5.6 are depicted UML class diagram of the distributed WPAN application which resides on the application server.

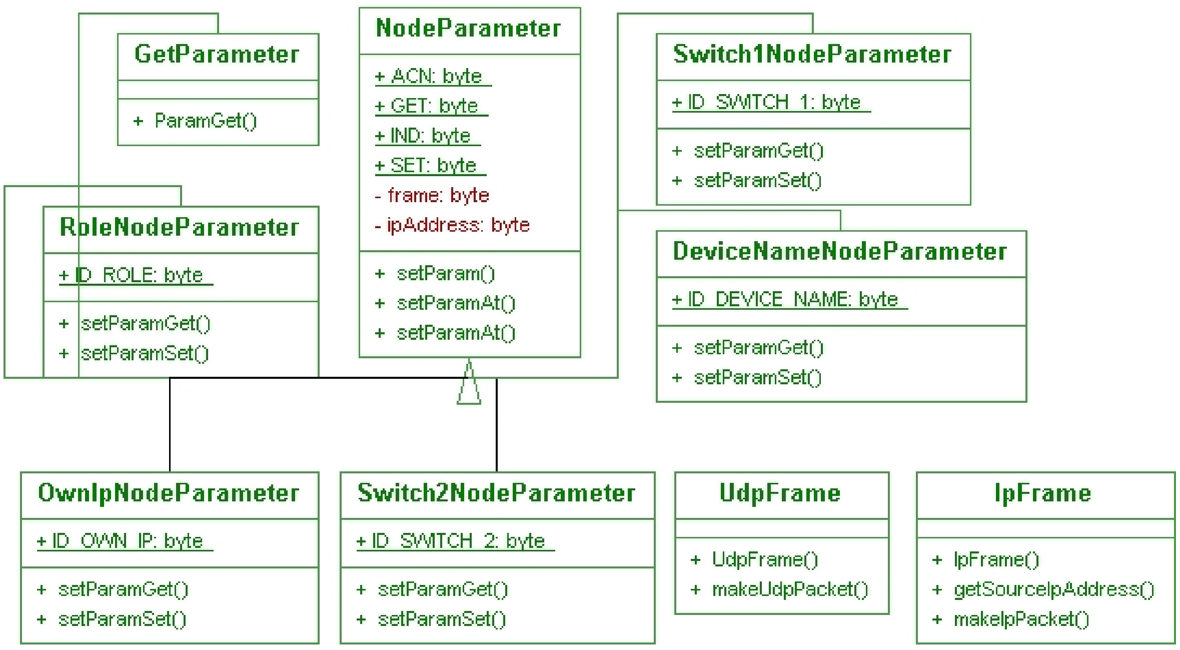


Figure 5.5 UML class diagram of the WPAN application

The Figure 5.5 shows the classes of Java classes which encapsulate logic for control WPAN network. Every class's name corresponds to that operation which can be done with mobile node.

According to the network stack protocol specification described in the paragraph 5.3 WPAN application wraps the original requests into UDP frame and then into IP frame. The classes UdpFrame and IpFrame encapsulate appropriate logic to build UDP and then IP frame from user request correspondingly.

The Figure 5.6 shows the classes of Java servlets, which process the user's requests. Servlets use appropriate classes, showed on Figure 5.5 to build request frames and redirect them to the WPAN server via socket connectivity.

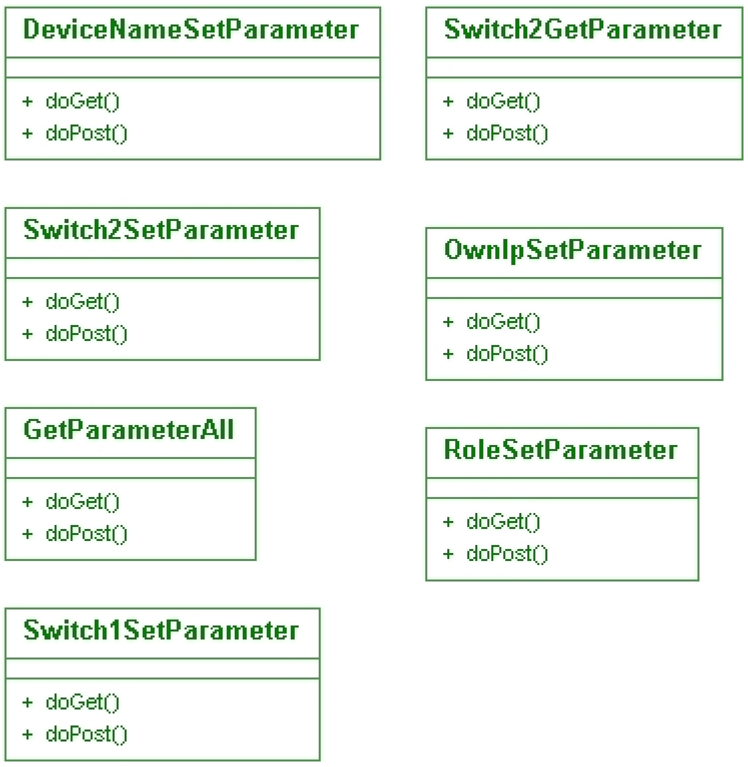


Figure 5.6 UML class diagram of the servlets

5.3 WPAN mobile device specification

As was mentioned above one of the main task of the research activity in the Kaustinen Lab was developing WPAN mobile device. The first prototype of such device is showed on the Figure 5.7

The node has follow technical specification:

- RS 232 serial port connection
- Light and temperature sensor controls
- Two switch controls
- CC2420 radio chip
- ATmega32 microcontroller

- B&W display

It should be mentioned here that this schema and functionality of this device was developed from the scratch in laboratory, it was build first prototype and successfully tested its work.

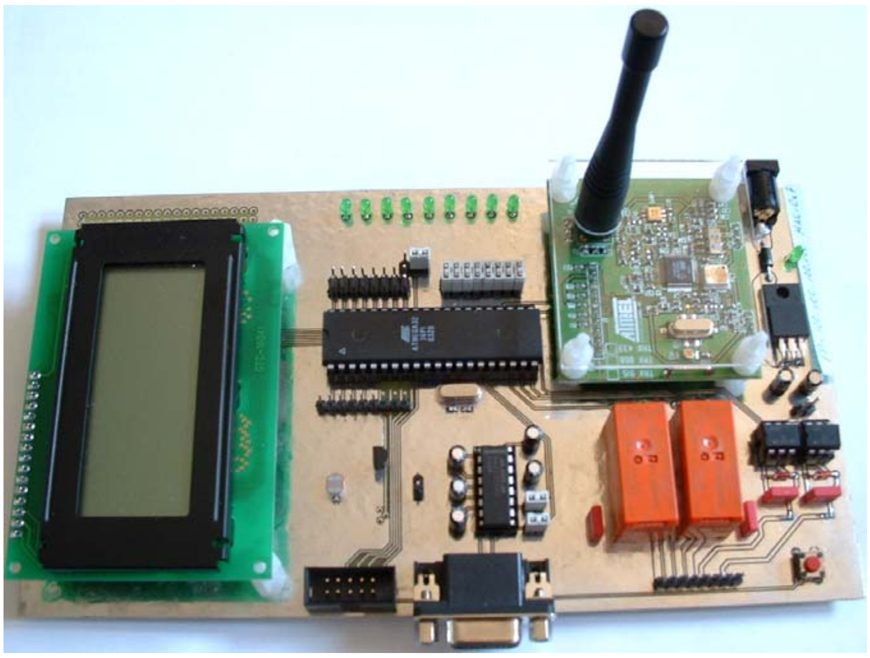


Figure 5.7 The WPAN mobile device

Device has two chips - CC2420 radio chip and ATmega32 microcontroller chip. The bellow lists summarize their characteristics:

CC2420 radio chip

- True single-chip 2.4 GHz IEEE 802.15.4/ZigBee(tm) RF transceiver with MAC support
- Low current consumption (RX: 19.7 mA, TX: 17.4 mA)
- Low supply voltage with internal voltage regulator (2.1 V - 3.6 V)
- Packet handling with 128 byte (RX) + 128 byte (TX) data buffering

- Hardware MAC encryption and authentication (AES-128)

ATmega32 microcontroller chip

- 2-wire Serial Interface (I2C compatible)
- Full Duplex Serial Peripheral Interface (SPI)
- In-System Programming via JTAG port
- Flash memory 32kB
- Analog-to-Digital Converter (10-bit)

5.4 Network stack protocol

Figure 5.8 shows the network stack protocol which was designed for WPAN network.

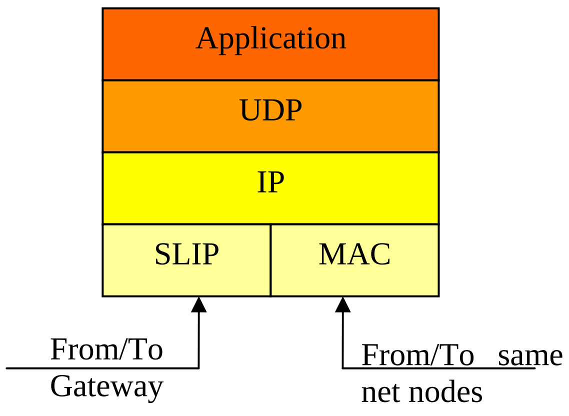


Figure 5.8 Network stack protocol layers

There are two flows of ingoing/outgoing traffic: from/to gateway (PC) and from/to mobile nodes which are in the same WPAN network.

Communication of WPAN network with external environment goes through gateway (PC). For that purpose one of net's node has PC connection role (PAN coordinator) and it is connected to the PC with one of the standard interface such as USB, LAN, Bluetooth, etc (in the project’s case by the RS 232 serial port).

All mobile nodes have built in application which performs some functionality with some data structures. These structures were defined just for test and research purposes. The following table indicates predefined functionality of node:

Table 5.2 WPAN node memory structure

Name	**ID**	**Memory location**	**Read/Write**
ID_OWN_IP	0	eeprom	R/W
ID_DEVICE_NAME	1	eeprom	R/W
ID_ROLE	2	eeprom	R/W
ID_IP_DATA	3	eeprom	R/W
ID_TEMP_PROPERTIES	4	eeprom	R/W
ID_LS_PROPERTIES	5	eeprom	R/W
ID_ROUTING_TABLE	6	sram	R/W
ID_ARP_TABLE	7	sram	R/W
ID_ERROR_TABLE	8	sram	R/W
ID_TEMPERATURE	9	-	R
ID_LIGHT_SENSOR	10	-	R
ID_SWITCH_1	11	-	R/W
ID_SWITCH_2	12	-	R/W

5.5 Protocol specification

5.5.1 GET command

Get command is one of the most used commands in the application protocol. It used when upon the request to devise to read values of data structures or memory from any device.

- **Function**

A mobile device can generate GET command if the application needs to send a request to another device. GET command can be also generated by the "PC role connection" device if it gets the request form PC application. The GET request includes all necessary parameters according to application protocol. When the device's application receive GET request form PC connection interface it sends acknowledge immediately and check destination IP address. If IP address does not match to device's IP then this packet must be sent forward, in such case application forms GET command frame according to the protocol specification. This frame is transferred as the primitive to the transport layer. The protocol stack manages the message, routing, etc. and sends it to the destination address.

The application with the destination IP address receives frame and indicate the GET command as DATA indication primitive. If the application is able to process the intended data it fulfills all necessary logic and end IND frame back to the initiator of GET command. If the right answer can not be sent or any error happens then application sends ACK command with the error information. The initiator of GET command handles the answer and transfers it to the PC via available connection interface.

After sending by the PC application the GET command it waits until the answer is received or timeout is occurred. Program should report ether any errors or timeout occurring or right answer to the user.

- **Command frame**

The data frame according to the Figure 5.9 is sent to the protocol stack. The specification of the GET frame is describing in the Table 5.3

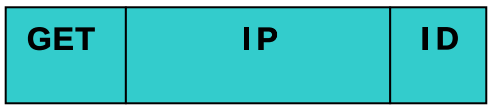


Figure 5.9 GET command frame

Table 5.3 GET frame specification

Field	**Length/ octets**	**Purpose**
GET	1	Type of frame: 0x01
IP	4	IP address of sender
ID	1	ID of data structure or memory

5.5.2 SET command

Some data structures and memory location are stored in the local device's memory. The purpose of SET command is to change some entries of these data.

- **Function**

SET command is used when the control application needs to change the value of memory location. The wireless device can get the SET request from the PC through available connection interface or this command can be retransmitted by intermediate device to the destination device.

If device's application gets the SET command from PC, the acknowledge is sent back immediately. If the destination IP address in request is intended to other device, the SET command frame is built and sent to the destination device.

When SET command reaches the destination device it goes to the application layer as the DATA.indication primitive, the application program save the data and sends ACK command as an acknowledge. If saving is done without the errors then the ERR field include zero value, otherwise the error code is placed to the ACK frame. The ACK frame is sent back to the initiator of SET command ether device or PC. The originator expects to receive acknowledgement in defined timeout, otherwise the error is happened. In PC case this error should be reported (displayed) to the control application.

- **Command frame**

The SET data frame according to the Figure 5.10 is sent to the protocol stack. The specification of the SET frame is describing in the Table 5.4

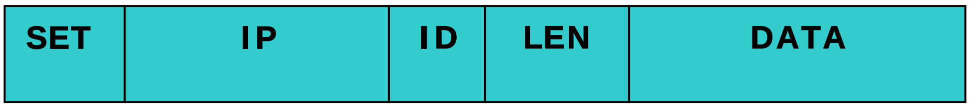


Figure 5.10 SET command frame

Table 5.4 SET frame specification

Field	**Length/ octets**	**Purpose**
SET	1	Type of frame: 0x02
IP	4	IP address of sender
ID	1	ID of data structure or table
LEN	1	Length of data frame
DATA	variable	Data value

5.5.3 IND command

IND command is used when values of data structures or single memory location are delivered from a one device to another one.

- **Function**

IND command is used as a response to GET command or it can be generated by a device itself when frame data must be sent via network. For example, the result of measurement may be sent through the definite time interval to data acquisition device and forward to the control application on the PC.

- **Command frame**

The IND data frame according to the Figure 5.11 is sent to the protocol stack. The specification of the IND frame is describing in the Table 5.5

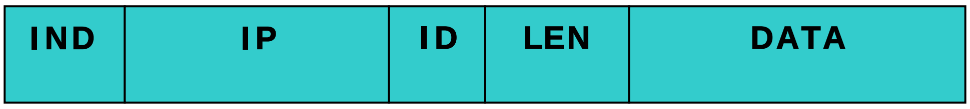


Figure 5.11 SET command frame

Table 5.5 IND frame specification

Field	**Length/ octets**	**Purpose**
IND	1	Type of frame: 0x03
IP	4	IP address of sender
ID	1	ID of data structure or table
LEN	1	Length of data frame
DATA	variable	Information

5.5.4 ACK command

ACK command is used as a response to the any command or as a error message.

- **Function**

ACK command is sent when a device or control application needs a confirmation that a sent message has reached the destination device. It is used also when special error message has to be sent. The error codes are defined elsewhere. ERR field of ACK frame has zero value if no error occurred.

- **Command frame**

The ACK data frame according to the Figure 5.12 is sent to the protocol stack. The specification of the ACK frame is describing in the Table 5.6

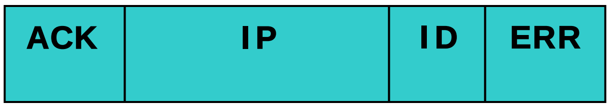


Figure 5.12 GET command frame

Table 5.6 ACK frame specification

Field	**Length/ octets**	**Purpose**
ACN	1	Type of frame: 0x04
IP	4	IP address of sender
ID	1	ID of data structure or memory location
ERR	1	Error code (0 – no error)

5.6 WPAN Services

In the previous chapters was showed the pilot WPAN application and described its architecture and features. Even with such functionality (obtaining two sensor's parameters), this simple application gives an idea for building real complex systems which involve WPAN technology. It should be noticed that the market of WPAN technologies quite young and unexplored.

Such questions as sensor networks integration, combination with legacy systems, deploying different WEB services upon WPAN functionality, ubiquitous computing are quite actual and need detailed research and practical implementation.

As the example of possible implementation it could be showed cases of the WPAN systems deployed in the hothouse and in the modern building to maintain its infrastructure. In the first example the WPAN sensors could be easily deployed to control temperature, lightness and humidity of the microclimate in the hothouse during crop growing. The distributed application with appropriate logic provides to operator remote control for all hothouse parameters. Picture 5.13 shows this idea.

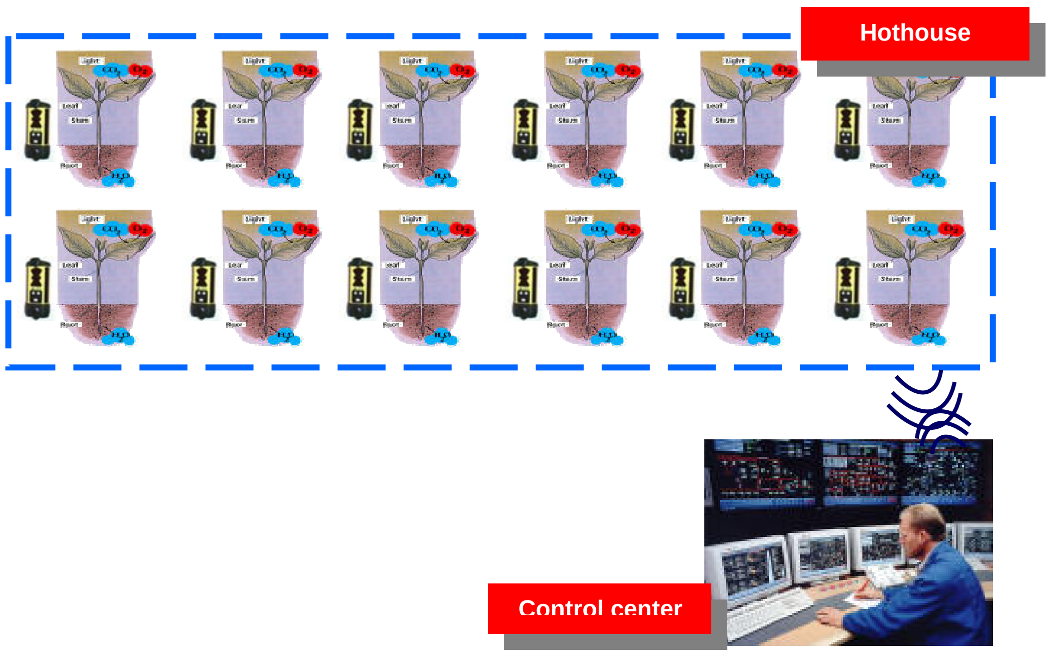


Figure 5.13 “Hothouse” WPAN system

Another view of implementing of the designed architecture could be embodied in the “Modern Building” infrastructure maintenance system. There is a rich branched out infrastructure in the modern building and it needs to be controlled and maintained. Using designed WPAN devices it is possible to build in their in different electronic building’s devices such as locks, fire sensors, light switches and so on. Using developed WPAN system it possible to provide the monitoring for that devices. Picture 5.14 shows this idea.

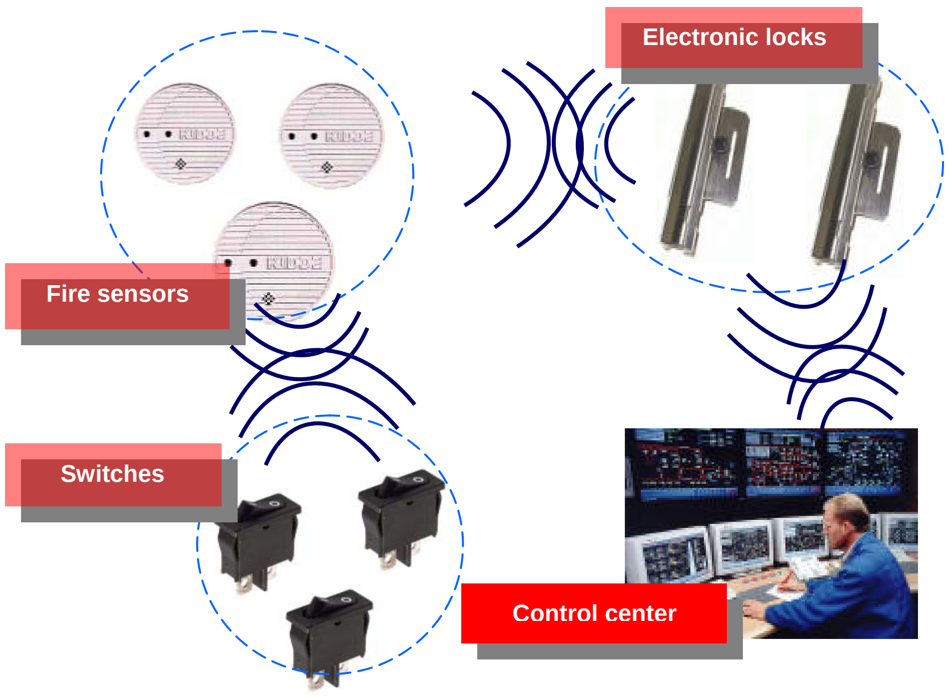


Figure 5.14 “Modern Building” infrastructure maintenance WPAN system

Conclusions

The main idea of this master thesis was to show possibility of practical implementation of WPAN technology with conjunction of distributed computing and mobility. Itself WPAN technology is quite young in IT industry and all works are fulfilled mostly in research field. However it's possible already to use practically that approaches and methods which were elaborated for resent years.

By its architecture WPAN or sensor networks are closed on itself. So it necessary to have at least one sink or PAN coordinator or "PC-Connection" device. As usual some PC application fulfills the coordinator of this network.

But further improving of such approach gives an idea to incorporate   WPAN network for the world of distributed computing. In the opinion of the author of this thesis such kind of combination could lead to the impressive results.

Today many companies involved into IT industry, having their legacy systems. WPAN might be the part of one's business and need to be implied into it. It might increase the company's benefits of such combination.

There are two essential technology aimed on the corporative business – distributed computing and WEB services. The first describes how to integrate heterogeneous pieces of Software second one allows to provide functionality.

This work tries to cover booth technologies and propose its implementation with WPAN. As a result in the chapter 4 is showed pilot system which realized such idea. From one side this simple application developed for WPAN network based on ad hoc connectivity fulfills trivial task – obtains temperature and lightness from arbitrary mobile node. From other side this system introduces the concept of new approach – it opens the standard interface for communication and provides interoperability between WPAN and other systems.

It should be mentioned here that this thesis is completely based on the results of practical training, but proposed and consider future views how to improved developed system with new functionality. In particular it's mentioned above about possible incorporation with some legacy systems using Java Connector Architecture and deploying WPAN network as a WEB service. Some examples are given in this thesis as applicable use-cases: “Hothouse” control system and “Modern Building” maintenance infrastructure system.

To summarize the whole work it should be mentioned that this thesis has more practical trend rather than research. It tries to show a solution for some new aspects concerning WPAN technology.

References

[1] Mobile ad hoc networking: imperatives and challenges; Imrich Chlamtac, Marco Conti, Jennifer J.-N. Liu; Ad Hoc Networks 1 (2003) 13–64; School of Engineering, University of Texas at Dallas, Dallas, TX, USA; Istituto IIT, Consiglio Nazionale delle Ricerche, Pisa, Italy; Department of Computer Science, University of Texas at Dallas, Dallas, TX, USA

[2] A review of routing protocols for mobile ad hoc networks; Mehran Abolhasan, Tadeusz Wysocki, Eryk Dutkiewicz, Telecommunication and Information Research Institute, University of Wollongong, Wollongong, NSW 2522, Australia; Motorola Australia Research Centre, 12 Lord St., Botany, NSW 2525, Australia

[3] - Simunic, T., et al., "Energy Efficient Design of Portable Wireless Systems," Proceedings of the 2000 International Symposium on Low Power Electronics and Design, Rapallo, Italy, July 25–27, 2000.

[4] - Girling, G., et al., "The Design and Implementation of a Low Power Ad Hoc Protocol Stack," Proceedings of IEEE Wireless Communications and Networking Conference, Chicago, Sept. 2000.

[5] Agrawal, S., and S. Singh, "An Experimental Study of TCP's Energy Consumption over a Wireless Link," in 4th European Personal Mobile Communications Conference, Feb. 2001.

[6] Technology Trends in Wireless Communications; Ramjee Prasad, Marina Ruggieri; Artech House; ISBN 1-58053-352-3

[7] Mobile Ad Hoc Networking working group, http://protean.itd.nrl.navy.mil/manet/manet_home.html

[8] The Internet Engineering Task Force, http://www.ietf.org/

[9] Wireless Ad Hoc Sensor Networks, http://w3.antd.nist.gov/wahn_ssn.shtml

[10] Draft Standard for Part 15.4: Wireless Medium Access Control (MAC) and Physical layer (PHY) specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)

[11] Shepard, T., "Decentralized Channel Management in Scalable Multihop Spread Spectrum Packet Radio Networks," Tech. Rep. MIT/LCS/TR-670, Massachusetts Institute of Technology Laboratory for Computer Science, July 1995.

[12] Chang, J.-H., and L. Tassiulas, "Energy Conserving Routing in Wireless ad hoc Networks," Proc. of INFOCOM 2000, Tel Aviv, Israel, March 2000.

[13] Srinivasan, V., et al., "Optimal Rate Allocation and Traffic Splits for Energy Efficient Routing in Ad Hoc Networks," in INFOCOM 2000, March 2002.

[14] Gupta, P., and P. Kumar, "The Capacity of Wireless Networks," IEEE Transactions on Information Theory, Vol. 46, No. 2, March 2000, pp. 388–404.

[15] Qiao, D., et al., "Energy-Efficient PCF Operation of IEEE 802.11a Wireless LAN," Proc. IEEE INFOCOM '02, March 2002.

[16] Wu, S.-L., Y.-C. Tseng, and J.-P. Sheu, "Intelligent Medium Access for Mobile Ad Hoc Networks with Busy Tones and Power Control," IEEE Journal on Selected Areas in Communications, Vol. 18, No. 9, Sept. 2000, pp. 1647–1657.

[17] Monks, J., et al., "A Study of the Energy Saving and Capacity Improvement Potential of Power Control in Multihop Wireless Networks," Proc. IEEE Conference on Local Computer Networks LCN, Nov. 2001, pp. 550–559

[18] Monks, J., V. Bharghavan, and W. Hwu, "A Power Controlled Multiple Access Protocol for Wireless Packet Networks," Proc. IEEE INFOCOM '01, April 2001.

[19] ElBatt, T., and A. Ephremides, "Joint Scheduling and Power Control for Wireless Ad Hoc Networks," Proc. IEEE INFOCOM '02, 2002.

[20] Web Services and Service-Oriented Architecture: The Savvy Manager's Guide, Douglas K. Barry , Morgan Kaufmann Publishers, 2003

[21] Programming Web Services with SOAP, Doug Tidwell James Snell Pavel Kulchenko, Publisher: O'Reilly, First Edition December 2001, ISBN: 0-596-00095-2

[22] The Semantic Web:A Guide to the Future of XML, Web Services, and Knowledge Management, by Michael C. Daconta,   Leo J. Obrst and Kevin T. Smith, Publisher: John Wiley & Sons 2003, ISBN:0471432571

[23] The University of British Columbia, Department of computer science, Lectures of Distributed Systems, http://www.ugrad.cs.ubc.ca/~cs416/X/

[24] Java communications, http://java.sun.com/products/javacomm/

[25] Java 2 Enterprise Edition, online tutorial, http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html

[26] Java Connector Architecture overview, http://java.sun.com/j2ee/connector/overview.html

Appendix A

Source code listing of WPAN server classes

Listing 1. WpanServerStart class

```
import javax.swing.*;

public class WpanServerStart{

        public static void main(String args[]){

                // specify the Java look and feel
                JFrame.setDefaultLookAndFeelDecorated(true);   // create instance of model

                WpanServer ws = new WpanServer();
                MenuBarApp mb = new MenuBarApp(ws);    // main window

                JFrame frame = new JFrame("Communication Server");

                StringBuffer sb = new StringBuffer("WPAN Communication Server,
Version: 1.0\nResearch Lab, Chydenius-Instituutti\n");

                //Text area as View in MVC

                TextAreaApp ta = new TextAreaApp();
                JTextArea  output = ta.createTextArea();

                //Scrolling element
                JScrollPane scroller = new JScrollPane(output);

                frame.getContentPane().add(scroller);
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setJMenuBar(mb.createMenu());
                output.setText(sb.toString());

                // register model and viewer in MVC
                ws.addObserver(ta);

                frame.pack();
                frame.setVisible(true);
                ws.connect();
        }
}
```

Listing 2. WpanServer class

```
import java.util.*;
import javax.comm.*;
import java.io.*;
import java.net.*;
```

```
public class WpanServer extends Observable implements SerialPortEventListener{

        SerialPort sp = null;
        CommPortIdentifier portId = null;
        StringBuffer readBuf = new StringBuffer();
        String sError, sMessage;
        String ok = "\nok";
        InputStream inStr;
        OutputStream outStr;
        InetAddress[] addresses;
        ServerSocket serverSocket = null;
        Socket clientSocket = null;
        boolean isConnected = false;
        SocketServer ss = null;
        boolean flag = false;
        List l;
        SlipFrame slipFrame;
        byte[] frame;
        MYSQLConnect msql = new MYSQLConnect();

        // method of Wpan Server in which we create Soket Server and pass
        // into it reference of Wpan Server's object itself(trick!!!)

        public void     createSocketServer(){
                ss = new SocketServer(this);
        }

        // method for notifying observers (views) that model is changed
        public void changed(Object o){
           setChanged();
                notifyObservers(o);
        }
        public void connect(){
                // try to get port ID from driver
                portId = SerialConnect.getComPort();
                if (portId == null) {
                        sError = "\nCan not get COM1 port object";
                        changed(sError);
                        return;
                }
                //try open port for listening
                try{
                        sMessage = "\nTrying to open COM1 port...";
                        changed(sMessage);
                        sp = (SerialPort) portId.open("JWpan", 2000);
                        changed(ok);
                }catch(PortInUseException e){
                        sError = "\nCan not get port,\nPort is used by other
application";
                        changed(sError);
                }
                // try to get event listener to the port
                try {
```

```
                                        sMessage = "\nTrying to add listener...";
                                        changed(sMessage);
    sp.addEventListener(this);
                                        changed(ok);
                               } catch (TooManyListenersException e)
                               {
                  }
   //try get input stream from serial port
                               try{
                                        sMessage = "\nTrying to get I/O stream...";
                                        changed(sMessage);
                                        inStr = sp.getInputStream();
                                        outStr = sp.getOutputStream();
                                        changed(ok);
                               }catch(IOException e){
                                        sError = "\nCan not get stream from port";
                                        changed(sError);
                  }
   sp.notifyOnDataAvailable(true);
                               //try to set up paremeters for serial port
                               try{
                               sMessage = "\nSetup port params...";
                                        changed(sMessage);
                                        sp.setSerialPortParams(9600,

                  SerialPort.DATABITS_8,SerialPort.STOPBITS_1,
                  SerialPort.PARITY_NONE);
                                        changed(ok);
                  }catch(UnsupportedCommOperationException e){
                  sError = "\nCan not to set up parameters for serial port";}

                               // connect to data base
   changed("\nConnecting to MySQL server...");
                               msql.connect();
                               if (msql.isConnected()){
                                        changed(ok);
                               }else {
                                        changed("\nCan not connect to MySQL server");
                               }
                               // obtain server IP
                               try {
    addresses = InetAddress.getAllByName(InetAddress.getLocalHost().getHostName());
                               for (int i=0; i<addresses.length; i++) {
                                                 changed("\nServer IP: " +
addresses[i].getHostAddress());
                                        }
                               } catch (UnknownHostException e){
                                        changed("\nCan not abtain server IP");
                               }

                               sMessage = "\nWPAN server is started\n";
                               changed(sMessage);
                               isConnected = true;
                               createSocketServer();
                  }
```

```
public void serialEvent(SerialPortEvent event) {
            switch(event.getEventType()) {
                case SerialPortEvent.BI:
            case SerialPortEvent.OE:
                case SerialPortEvent.FE:
            case SerialPortEvent.PE:
                case SerialPortEvent.CD:
            case SerialPortEvent.CTS:
                case SerialPortEvent.DSR:
            case SerialPortEvent.RI:
                case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
              break;
                case SerialPortEvent.DATA_AVAILABLE:
                        readPort();
              break;
            }
}
            public void disconnect(){
                        sp.close();
                        try {
                                    clientSocket.close();
                                    serverSocket.close();
                        } catch (IOException e) {
                                    changed("\nCan not close sockets");
                        }
                        this.isConnected = false;
                        changed("\nServer stoped\n");
            }
            public void readPort(){
                        int size = 0;
                        byte[] readBuffer;
                        try {
                                    size = inStr.available();
                                    readBuffer = new byte[size];
                                    while (inStr.available() > 0) {
                                                int numBytes = inStr.read(readBuffer);
                                    }
                        makeFrame(readBuffer);
              } catch (IOException e) {}
            }
            public void writePort(byte[] frame){
                        try{
                                    // code this frame according SLIP format
                                    slipFrame = new SlipFrame();
                                    byte[] b = slipFrame.encodeFrame(frame);
                                    // save this frame into database
                                    if(msql.isConnected()){
                                                msql.insert(makeQuery("o", frame));
                                    }
                                    outStr.write(b);
                                    slipFrame = null;
                        } catch (IOException e){
                                    System.out.println("Write exception");
```

```
				}
			}
			public void makeFrame(byte[] b){
				for(int i=0; i<b.length; i++){
					if(((b[i]&0xFF)==192)&&(!flag)){ // indicator of begin of 
slip frame
						flag = true;
						l = new ArrayList();
						l.add(new Byte(b[i]));
						continue;
						// add to frame
					}else if(((b[i]&0xFF)!=192)&&flag){ // normal byte
						l.add(new Byte(b[i]));
					}else if(((b[i]&0xFF)==192)&&flag){ // end of slip frame
						l.add(new Byte(b[i]));
					// create SLIP object to make transformation with slip frame
						slipFrame = new SlipFrame();
					frame = slipFrame.decodeFrame(Utils.listToByteArray(l)); 
// get decoded sequence of byte from slip frame
						// send to view
						changed(frame);
						// save this frame into database
						if(msql.isConnected()){
						msql.insert(makeQuery("i", frame));
						}
				// distroy all objects for the next frame
				frame = null;
						slipFrame = null;
						l = null;
						flag = false;
					}
				}
			}
			public String makeQuery(String q, byte[] b){
				StringBuffer sb = new StringBuffer();
				for(int i=0; i<b.length; i++)
					sb.append(b[i] + " ");
				String s = ("insert into wpan_data values(null, '" + q + "', null, '" + 
sb.toString() + "')");
				System.out.println(s);
				return s;
			}
}
```

Listing 3. TextAreaApp class

```
import javax.swing.*;
import java.util.Observer;
import java.util.Observable;

public class TextAreaApp implements Observer{
			JTextArea output = new JTextArea(19, 35);
			public JTextArea createTextArea(){
				//Text area for displaying processing
			output.setEditable(false);
```

```
                    return output;
          }

          public void update(Observable obs, Object obj){
                    if(obj.getClass().getName().equals("java.lang.String"))
                              output.append((String)obj);
                    if(obj.getClass().getName().equals("[B")){
                    byte[] ar = new byte[((byte[])obj).length & 0xFF];
                    ar = (byte[])obj;
                              for (int i=0; i<ar.length; i++){
                              output.append(ar[i] + " ");
                              }
                    output.append("\n");
                    }
          }
}
```

Listing 4. SocketServer class

```
import java.io.*;
import java.net.*;

public class SocketServer {
  // Choose a port outside of the range 1-1024:
          public static final int PORT = 3045;
          SocketServer(WpanServer ws){
                    try{
                              this.create(ws);// create instance of wpan server
                    }catch (IOException e){
                              System.out.println("error of Socket Server creation");
                              System.out.println("\n" + e);
                    }
          }
          public static void create(WpanServer ws) throws IOException {
                    ServerSocket s = new ServerSocket(PORT); // new server socket
                    ws.changed("Started listening requests on port: " + PORT + "\n");
                    try {
          // create multyple sockets
             while(true){
                                        Socket socket = s.accept(); // start listen for
client requests
                    try{
                              new MultiSocketClient(socket, ws);
                    }catch(IOException e){
                              socket.close();
                    }
                    }
                    }finally{
                              s.close();
                    }
          }
}
```

Lsiting 5. MultiSocketClient class

```
import java.io.*;
import java.net.*;
import java.util.*;
public class MultiSocketClient extends Thread implements Observer{

                private Socket socket;
                WpanServer ws;
                private DataInputStream in;
                private DataOutputStream out;

                // consructor for initialisation all fields
                public MultiSocketClient(Socket s, WpanServer ws) throws IOException{
                                this.socket = s;
                                this.ws = ws;
                                ws.addObserver(this);
                   in = new DataInputStream(socket.getInputStream());
                                out = new DataOutputStream(socket.getOutputStream());
                   start(); // Calls run()
                }
                public void run(){
                try {
                                // read from socket into byte array
                                int arraySize;
                                while (true) {
                                                arraySize = in.readInt();
                                                byte[] frame = new byte[arraySize];
                                                in.read(frame);
                                                if((new String(frame)).equals("END"))
                                                 break;
                                   ws.writePort(frame);
                                }
                }catch(IOException e) {
   }finally{ // close sokets in any cases
                                try{
                                                socket.close();
                                }catch(IOException e){
                                }
                }
 }
                // notify view about all changes
                public void update(Observable obs, Object obj){
                                if(obj.getClass().getName().equals("[B")){// [B indicates that this object is
byte array
                                                byte[] ar = new byte[((byte[])obj).length & 0xFF];
                                                ar = (byte[])obj;
                                                try{
                                                out.writeInt(ar.length);
                                                out.write(ar);
                                                out.flush();
                                                } catch(IOException e){}
                                                ws.deleteObserver(this);
                                }
                }
}
```

Listing 6. SlipFrame class

```
import java.util.*;

public class SlipFrame{
		private static final byte SLIP_END_FRAME = (byte)192; // The end marker of slip frame
0xC0
		// Esc characters for Slip frame 0xDB, 0xDC, 0xDD correspondingly
		private static final byte SLIP_ESC_CHARACTER_DB = (byte)219;
		private static final byte SLIP_ESC_CHARACTER_DC = (byte)220;
		private static final byte SLIP_ESC_CHARACTER_DD = (byte)221;
		private List l;
		private byte[] slipFrame;
		public SlipFrame(){
				l = new ArrayList();
		}
		public void addByte(byte b){
				// Add bytes method, adds one byte to the list, according to the rules of
				// SLIP frame, which repalace some bytes with others
				if(b==SLIP_END_FRAME){
						l.add(new Byte(SLIP_ESC_CHARACTER_DB));
						l.add(new Byte(SLIP_ESC_CHARACTER_DC));
				}else if(b==SLIP_ESC_CHARACTER_DB){
						l.add(new Byte(SLIP_ESC_CHARACTER_DB));
						l.add(new Byte(SLIP_ESC_CHARACTER_DD));
				}else{
						l.add(new Byte(b));
				}
		}
		public byte[] encodeFrame(byte[] b){
				// make up SLIP frame, adding END character to the begin and end of frame
   for(int k=0; k<b.length; k++)
						addByte(b[k]);
				l.add(0, new Byte(SLIP_END_FRAME));
				l.add(new Byte(SLIP_END_FRAME));
				slipFrame = null;
				slipFrame = Utils.listToByteArray(l);
				return slipFrame;
		}
		public byte[] decodeFrame(byte[] c){
				// growable list for decode array
		  List l = new ArrayList();
				// start to decode slip frame and strore it in the list
		for(int k=0; k<c.length; k++){
						if((c[k])==SLIP_END_FRAME){
						continue;
						}else if((c[k])==SLIP_ESC_CHARACTER_DB){
								k++;
				if((c[k])==SLIP_ESC_CHARACTER_DC){
						l.add(new Byte(SLIP_END_FRAME));
						}else{
				l.add(new Byte(SLIP_ESC_CHARACTER_DB));
								}
		}else{
						l.add(new Byte(c[k]));
}
                                }
                                // store result in the byte array and return it
                                slipFrame = null;
                                slipFrame = new byte[l.size()];
                                int j = 0;
                                Iterator i = l.iterator();
                                while(i.hasNext()){
                                        slipFrame[j] = ((Byte) i.next()).byteValue();
                                        j++;
                                }
                                return slipFrame;
                }
                public byte[] getFrame(){
                  return slipFrame;
                }
}
```

Listing 7. SerialConnect class

```
import java.util.*;
import javax.comm.*;

public class SerialConnect {
  static CommPortIdentifier portId;
  static Enumeration portList;
SerialPort serialPort;
  public static CommPortIdentifier getComPort() {
    portList = CommPortIdentifier.getPortIdentifiers();
    while (portList.hasMoreElements()) {
                                        portId = (CommPortIdentifier) portList.nextElement();
        if (portId.getName().equals("COM1")){
                break;
                }
                                }
                                return portId;
  }
}
```

Listing 8. MYSQLConnect class

```
import java.sql.*;

                public class MYSQLConnect {
                                String mySqlUrl = "jdbc:mysql://localhost/wpan";
                                Connection conn = null;
                                Statement st = null;
                                boolean isConnected = false;
                  public void connect(){
                    try {
                      Class.forName("com.mysql.jdbc.Driver");
                                                        conn =
                DriverManager.getConnection(mySqlUrl, "root",
                ");
                                        st = conn.createStatement();
```

```
                                    isConnected = true;
                              }catch(ClassNotFoundException e){
                                 System.out.println("Load Driver Error: " +
e.getMessage());
   }catch(SQLException e){
               // handle any errors
      System.out.println("SQLException: " + e.getMessage());
      System.out.println("SQLState: " + e.getSQLState());
      System.out.println("VendorError: " + e.getErrorCode());
                              }
               }
               public void insert(String query){
                              if(st != null){
                                    try{
                              st.executeUpdate(query);
                                    }catch(SQLException e){

               System.out.println("SQLException: " + e.getMessage());
               System.out.println("SQLState: " + e.getSQLState());
                    System.out.println("VendorError: " + e.getErrorCode());
                                    }
}
               }
               public void disconnect(){
                              if (st != null) {
                  try {
                  st.close();
                  } catch (SQLException sqlEx){
                                 st = null;
                                    }
                              }
                              if (conn != null) {
                  try {
                  conn.close();
                  } catch (SQLException sqlEx){
                                 conn = null;
                                    }
                              }
                              isConnected = false;
               }
               public boolean isConnected(){
                              return isConnected;
               }
}
```