Timo Juonoja

The Analysis of BlueCheese Mobile Peer-to-Peer Middleware

Master's Thesis
in Information Technology
July 28, 2006

University of Jyväskylä

Department of Mathematical Information Technology

Jyväskylä

Author: Timo Juonoja
Contact information: timo.juonoja@cc.jyu.fi
Title: The Analysis of BlueCheese Mobile Peer-to-Peer Middleware
Työn nimi: Mobiilin vertaisverkon BlueCheese väliohjelmiston analysointi
Project: Master's Thesis in Information Technology
Page count: 66
Abstract: BlueCheese is a mobile peer-to-peer middleware that was produced in the MoPeDi student software project at the University of Jyväskylä. BlueCheese operates in Symbian OS mobile phones as a transmission system for mobile peer-to-peer applications. The transmission technology is Bluetooth. This master's thesis describes first the related technologies and functionality of BlueCheese. Emphasis in the thesis was to analyze BlueCheese and to develop a working prototype.
Suomenkielinen tiivistelmä: BlueCheese on Jyväskylän yliopiston MoPeDi-sovellusprojektissa tuotettu väliohjelmisto. BlueCheese toimii Symbian OS -matkapuhelimissa tiedonsiirtovälineenä mobiileille vertaisverkkosovelluksille. Tiedonsiirto matkapuhelinten kesken on toteutettu Bluetoothilla. Tämä Pro Gradu -työ kuvaa ensin BlueCheeseen liittyvät teknologiat sekä BlueCheesen toiminnan. Pääpaino Pro Gradu -työllä oli analysoida BlueCheesen toiminta sekä kehittää siitä toimiva prototyyppi.
Keywords: Bluetooth, BlueCheese, middleware, mobile peer-to-peer, Symbian OS, middleware analysis
Avainsanat: Bluetooth, BlueCheese, väliohjelmisto, mobiilit vertaisverkot, Symbian OS, väliohjelmiston analysointi

Glossary

ACK	Acknowledge packet
ACL	Asynchronous Connection-Less
API	Application Program Interface
BPSK	Binary Phase Shift Keying
CCK	Complementary Code Keying
CRC	Cyclic Redundancy Checksum
CSMA-CA	Carrier Sense Multiple Access with Collision Avoidance
DS	Direct Sequence
DSFH	Direct Sequence Frequency Hopping
DSSS	Direct Sequence Spread Spectrum
EDR	Enhanced Data Rate
FCC	Federal Communications Commission
FFD	Full Function Device
GFSK	Gaussian Frequency Shift Keying
GSM	Global System for Mobile Communications
GPCS	Gasoline Price Comparison System
GPRS	General Packet Radio System
GTS	Guaranteed Time Slot
HDR	High Data Rate
HDTV	High-Definition television
IrDA	Infrared Data Association
IEEE	Institute of Electrical and Electronics Engineers
ISM	Industrial Scientific Medical band
L2CAP	Logical Link Control and Adaptation Protocol
LAN	Local Area Network
LDR	Low Data Rate
Li-ion	Lithium ion
LMP	Link Manager Protocol
MAC	Medium Access Control
MBOA	MultiBand OFDM Alliance
MP2P	Mobile Peer-to-Peer
OFDM	Orthogonal Frequency Division Multiplexing

OQPSK	Offset Quadrature Phase Shift Keying
OS	Operating System
OSI	Open Systems Interconnection
PAN	Personal Area Network
P2P	Peer-to-Peer
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RFCOMM	Radio Frequency Communications Protocol
RFD	Reduced Function Device
RFID	Radio Frequency Identification
SCO	Synchronous Connection-Oriented
SDK	Software Development Kit
SDP	Service Discovery Protocol
SIG	Special Interest Group
TDD	Time-Division Duplex
UMTS	Universal Mobile Telephone System
USB	Universal Serial Bus
UWB	Ultra WideBand
Wi-Fi	Wireless Fidelity
WAN	Wide-Area Network
WBAN	Wireless Body Area Network
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WUSB	Wireless Universal Serial Bus

Contents

1 Introduction

This master's thesis was done as a part of the Cheese Factory research project. Cheese Factory is a peer-to-peer research project located at the University of Jyväskylä. The emphasis of the Cheese Factory project is to study peer-to-peer communication and the behavior of peer-to-peer networks concentrating on distributed search of resources and their efficient use [7].

BlueCheese was produced in the MoPeDi student software project at the University of Jyväskylä. BlueCheese is a mobile peer-to-peer middleware that operates in Symbian OS mobile phones as a transmission system for mobile peer-to-peer applications. The transmission technology is Bluetooth, which is now supported by many Symbian OS mobile phones. Another possible choice could have been Wireless Local Area Network (WLAN), but it was not supported in many smartphones at the time of implementation. However WLAN is becoming more widespread little by little, so in the future it might be a possible transmission technology for BlueCheese.

1.1 Research Problem

All applications contain errors even in the final version of the product. One of the thesis' purpose was to develop a working prototype of BlueCheese mobile peer-to-peer middleware. The stability of the application is an especially challenging task in Symbian OS applications. Mobile phones have little memory compared to personal computers, so the applications must have a strict memory handling. If the memory handling is poorly done, the system crashes.

Another main point was to analyze the functionality of BlueCheese. One of the most significant keys was to consider the suitability of Bluetooth transmission technology for mobile peer-to-peer communications. BlueCheese was tested with various methods by computing the power consumption, connections, data transfer and fault-tolerance.

1.2 Related Work

One of the most used mobile peer-to-peer middleware might be Nokia Proximity Toolkit. The middleware itself may be unknown but the applications, which exploit the services of Nokia Proximity Toolkit, are familiar to some people. For example Nokia Sensor [22], the application for providing portable personality and communication with file sharing in short range area, uses Nokia Proximity Toolkit as transmission system.

Nokia Proximity Toolkit provides nearly the same services as BlueCheese with some dissimilarities (see table 1.1). BlueCheese works mostly independently whereas the Proximity Toolkit is handled mainly by the application or the user of the device. BlueCheese is designed for spreading the information automatically whereas the Proximity Toolkit is used mainly according to the user's wish to communicate. Therefore the connection establishment procedures in BlueCheese differ totally from the Proximity Toolkit. BlueCheese informs the application for available connections whereas the connections in Proximity Toolkit are mainly created by the user.

	BlueCheese	**Nokia Proximity Toolkit**
Application sessions	YES	YES
Data transmissions	YES	YES
Automatic device search and connection establishment	YES	YES
Manual device search and connection establishment	NO	YES
Location service	YES	NO

Table 1.1: Features of BlueCheese and Nokia Proximity Toolkit.

BlueCheese and Nokia Proximity Toolkit are mobile peer-to-peer middlewares with the same purpose: to provide a simple way to communicate with other Bluetooth devices. Several applications can use simultaneously both systems and the data is supplied between similar applications of the two devices. BlueCheese provides location service for estimating physical locations of the mobile device as an extra feature. Section 3.4 treats the comparison of BlueCheese and Nokia Proximity Toolkit more precisely. [25]

1.3 Structure of the Thesis

The structure of the master's thesis is as follows. Chapter 2 describes the main involved architectures and technologies of BlueCheese. The functionality and the features of BlueCheese are represented in chapter 3. Chapter 4 contains the measurements and analysis of BlueCheese and chapter 5 considers future development. Chapter 6 draws the conclusions of the BlueCheese's functionality.

2 Architectures and Technologies

This chapter describes the architectures and technologies used in BlueCheese.

2.1 Distributed Systems

A system in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages is defined as a *distributed system*. With this definition, the distributed system includes three main characteristics: concurrency of components, no global clock and independent failures of components. [8]

Distributed systems are everywhere. The Internet enables users to access its services all over the place. Organizations have their own intranets to provide interior services. Small distributed systems can be constructed for example by mobile devices. [8]

Resource sharing is the main reason for constructing distributed systems. Resources like printers, files, web pages are managed by servers and accessed by clients. For example, a web server manages and a browser accesses web pages. [8]

The construction of distributed systems causes many challenges: [8]

1. *Heterogeneity*
   The network has to be constructed from a variety of diverse networks, computer hardware, operating systems, programming languages and implementations by different developers. Communication protocols are the solution for difference in networks and the middleware for other differences.

2. *Openness*
   Distributed systems should be extensible. Open systems are characterized by the fact that their key interfaces are published. But the real challenge is the interoperability of the components written by different programmers.

3. *Security*
   An important thing for shared resources is security, which includes confidentiality, integrity and availability. Encryption has to be used to provide pro-

tection of shared resources and to maintain information secret when passed through a network.

4. *Scalability*
   A system is described as scalable if it remains effective when there is a remarkable increase in the number of resources and users. The algorithms for accessing shared data prevent performance breakdowns and the hierarchical data structure enables efficient access times.

5. *Failure Handling*
   Computer systems fail sometimes. The fault may be aroused by a process, a computer or the network. Therefore each component needs to be appropriately designed for handling the failures for providing sufficient availability.

6. *Concurrency*
   In a network of computers, shared resources have to provide simultaneous processing. So each resource must be designed to be secure in a concurrent environment e.g. avoiding deadlocks and livelocks.

7. *Transparency*
   Transparency hides some features of distribution from the application programmers so that they need only be worried about the design of the particular application.

2.2 Peer-to-Peer

Peer-to-Peer networks (P2P) are instances of distributed systems. A distributed network may be called a Peer-to-Peer network, if the participants share a part of their own hardware resources (for example computing power, storage capacity, network bandwidth, printers, file sharing, etc.). Every peer in the network has a direct access to other peers without passing via intermediary entities. The participants of such a network are thus resource providers as well as resource requestors. So the peers are capable of acting as a server and a client at the same time (alias servents). [28]

In contrast to Client-Server networking, Peer-to-Peer networking does not use a centralized server for message passing making the absence of servers the biggest difference between these two architectures (see figure 2.1). Peer-to-Peer is an architectural model, where all the peers are equal and function as a server and a client at the same time. Peer-to-Peer networking enables faster information spreading because there is no server as a bottleneck for information diffusion. [28, 15]

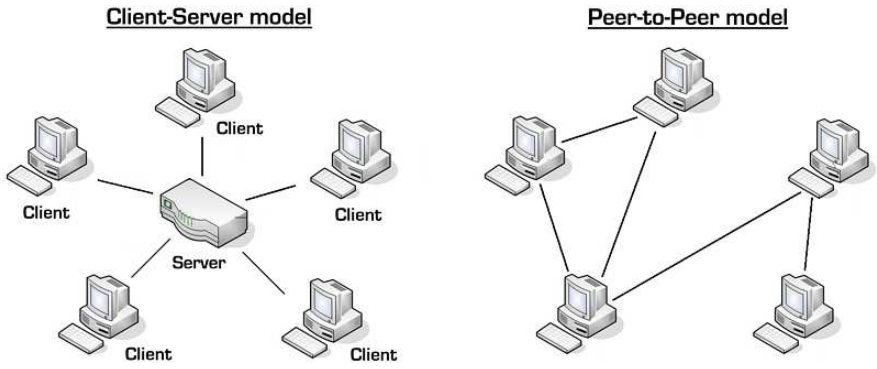


Figure 2.1: Client-Server and Peer-to-Peer models.

The definition of Peer-to-Peer networking can be divided into two sub-definitions. They are known as *Pure Peer-to-Peer* and *Hybrid Peer-to-Peer* networking. The network is Pure Peer-to-Peer if it is firstly a Peer-to-Peer network and secondly if any single peer can be removed from the network without having the network suffer any loss of network service. If the network is a Peer-to-Peer network and needs a necessary central entity to provide parts of the offered network services, it is called a Hybrid Peer-to-Peer network. [28]

2.3 Mobile Peer-to-Peer

Mobile Peer-to-Peer network is a Peer-to-Peer network where at least one peer is a mobile device. The structure of the network is dynamic since the mobile devices continuously change their physical location and establish peer-to-peer communications with peers based on their proximity. Unlike in a wired network, every mobile peer has a limited transmission range. Therefore direct communication between two mobile peers is possible only if the peers are in the transmission range of each other. [13, 15]

Kimmo Haukimäki [15] defines two structure models of Mobile Peer-to-Peer networks: Partial MP2P and Pure MP2P. These models are represented in figure 2.2.

A Peer-to-Peer network is a Partial Mobile Peer-to-Peer network, if it consists of both mobile and immovable devices. Then at least one of the mobile devices has a wireless connection to the immovable device. Therefore a mobile device can connect and share services more widely than in a Pure Mobile Peer-to-Peer network. [15]

If there are only mobile devices in the Peer-to-Peer network, it is called a Pure Mobile Peer-to-Peer network. Then all the devices are connected to each other with a wireless technique like for example Wireless Local Area Network (WLAN) or Blue-

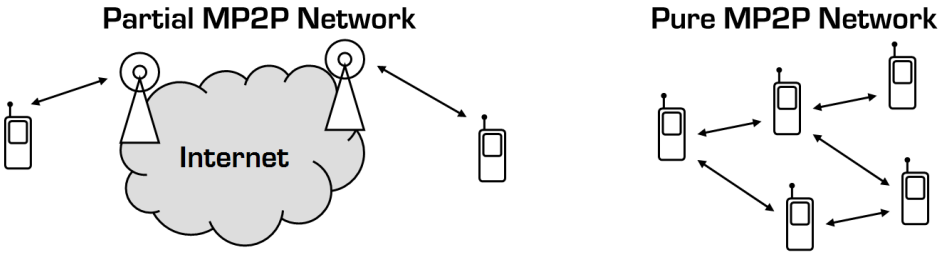


Figure 2.2: Mobile Peer-to-Peer structure models.

tooth. Because wireless techniques mostly have a small-scale functional range and the physical locations of devices fluctuate, the network is very dynamic. Limited range and mobility also bring challenges for efficient data communications. [15]

2.4 Middleware

The term *software architecture* refers to the layered structure of software in the same device or services provided and requested between local devices or different devices. This definition can also be expressed in terms of *service layers*. [8]

The lowest-level service layer is *platform*, which includes (computer and network) hardware and an operating system. The operating system provides problem-oriented abstractions of the underlying physical resources such as the processors, memory communications and storage media. For example operating systems like UNIX and Windows offer files rather than disk blocks or sockets rather than raw network access. Both operating systems are examples of *network operating systems*, which have a networking capability but not for example distributed process execution. [8]

A heterogeneous network generates a great challenge for the compatibility of distributed systems. Middleware resolves the heterogeneity with shared interfaces and protocols and simplifies the system development process by providing scripted functionality. Middleware is defined as a layer of software between network operating systems and application components (see figure 2.3). [3, 10, 21]

Middleware is a piece of software that provides a programming model above the basic building blocks of processes and message passing. The aim of middleware is to provide application engineers with high-level primitives that simplify distributed system construction. [8, 10] The typical operations of middleware are providing data connection, disconnection, segmentation, reassembly, etc. [15]

Remote procedure calling packages like Sun RPC and group communication

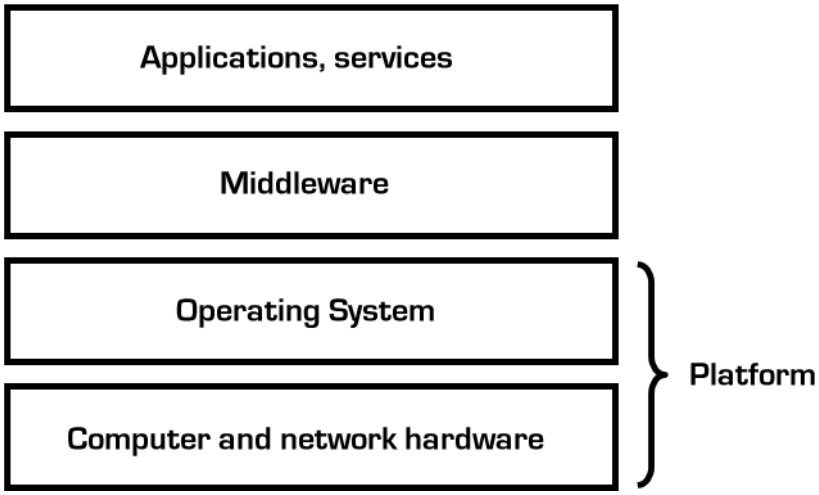


Figure 2.3: The positions of application, middleware, operating system and hardware components.

systems such as Isiswere were the earliest instances of middleware. Nowadays object-oriented middleware products and standards such as Object Management Group's Common Object Request Broker Architecture (CORBA), Java Remote Object Invocation (RMI), Microsoft's Distributed Component Object Model (DCOM) and ISO/ITU-T's Reference Model for Open Distributed Processing (RM-ODP) are widely used. [8]

2.5 Bluetooth

A few years ago it was recognized, that a low-cost and low-power wireless transmission technology could be feasible to put into practise. The goal was to provide a short-range radio-based ad-hoc network for portable devices with effortless service. [24]

A special interest group (SIG) was formed in the beginning of 1998 [24]. The purpose of SIG was to develop, publish and promote the preferred short-range wireless specification for connecting mobile products, and to administer a qualification program that fosters interoperability for a positive user experience [5].

At the beginning there were five promoters of the Bluetooth technology: Ericsson, IBM, Intel, Nokia and Toshiba [24]. Later on Agere Systems, Microsoft and Motorola joined as promoter companies. Promoter companies are highly engaged in the technical development of Bluetooth wireless technology. The Bluetooth SIG has over 2,000 member companies, which can use Bluetooth technology [5].

Architecture

Bluetooth is a short-range radio technology for data transmission between portable devices. It operates in the 2,4 GHz worldwide unlicensed Industrial Scientific Medical (ISM) band . The channel is represented by a pseudo-random hopping sequence through 79 or 23 RF channels [9]. Bluetooth uses fast frequency hopping for low interference and fading, Time-Division Duplex (TDD) scheme for full duplex transmission and transmits using Gaussian Frequency Shift Keying (GFSK) modulation [30].

The Bluetooth system supports both point-to-point and point-to-multipoint connections. Two or more devices sharing the same channel form a *piconet*. There is one master device and up to seven slave devices in a piconet. Multiple overlapped piconets form a *scatternet*. Figure 2.4 illustrates piconets and a scatternet. [30] However, scatternets are not widely implemented in current Bluetooth devices.

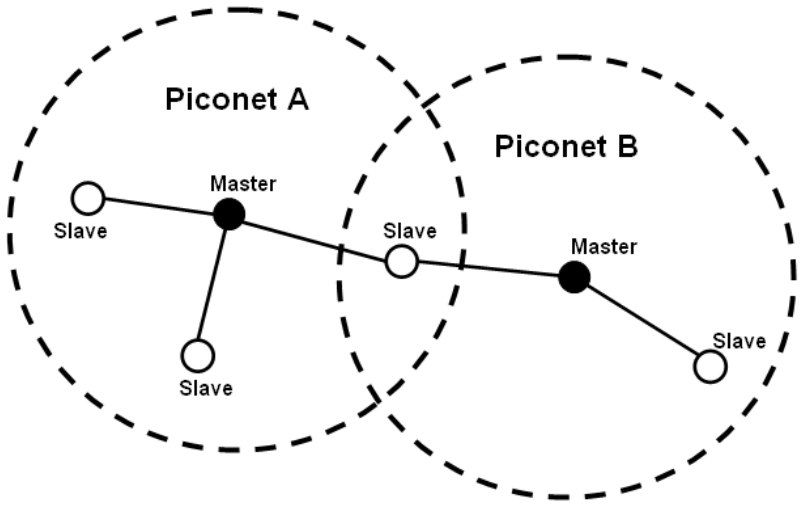


Figure 2.4: The combination of piconet A and B forms a scatternet.

Bluetooth needs to support both synchronous services like voice and asynchronous services like internet access. Bluetooth provides Asynchronous Connection-Less (ACL) link for packet routing and up to three Synchronous Connection-Oriented (SCO) links for circuit switching. [14, 24]

The Bluetooth system uses packet-based transmission. In each 625 microsecond slot, only one packet can be sent. The packet consists of an access code, header and a payload (see figure 2.5). The access code is used for synchronization and identification. The header contains lower-level link control information and the payload carries the actual user information. There are several packet types in Bluetooth for paging, inquiry, polling, data transmission etc. [24]

Figure 2.5: Standard Bluetooth packet format.

Bluetooth is a lower-layer specification from the view of Open Systems Interconnection (OSI) protocol stack specification. Bluetooth specifies Physical, Data Link and partly Network layers of the OSI model. Figure 2.6 shows the main protocols of Bluetooth. [24]

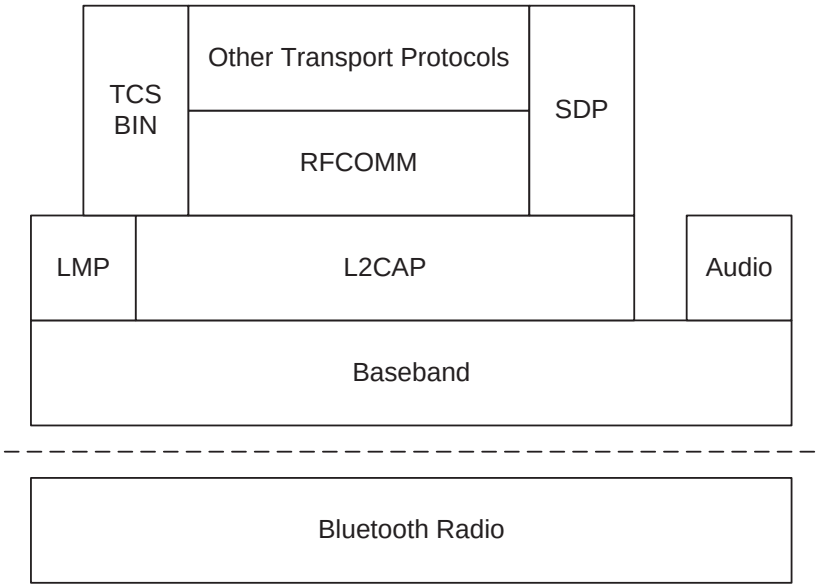


Figure 2.6: Bluetooth stack.

The Bluetooth Radio part defines the frequency bands, channel arrangement and receiver characteristics. Baseband defines packet format, physical and logical channels, channel control, etc. Link Manager Protocol (LMP) is used for link set-up and control. These three protocols are usually integrated in the chips to provide hardware platform for the higher layer application. [24]

Logical Link Control and Adaptation Protocol (L2CAP) supports protocol multiplexing, packet segmentation and reassembly, and the conveying of quality of service information (for example Cyclic Redundancy Checksum (CRC)). Service Discovery Protocol (SDP) is used for discovering, searching and browsing specific services from another Bluetooth device. Radio Frequency Communications Protocol (RFCOMM) is not a "core" Bluetooth layer. It is more like a layer above core layers providing an interface between existing services and the Bluetooth stack. To up-

per layers RFCOMM is an interface that looks exactly like an RS-232 serial line. In addition Bluetooth contains other protocols that are not described in this master's thesis.[24, 18]

There are several Bluetooth states for supporting functional operations like channel establishment of a piconet, adding and releasing units from a piconet. Figure 2.7 shows a state diagram of used states in Bluetooth. `STANDBY` and `CONNECT` are two major states and the others (`Page`, `Page Scan`, `Inquiry`, `Inquiry Scan`, `Master Response`, `Slave Response` and `Inquiry Response`) are sub-states. The sub-states are used for adding new slaves to the piconet. [24]

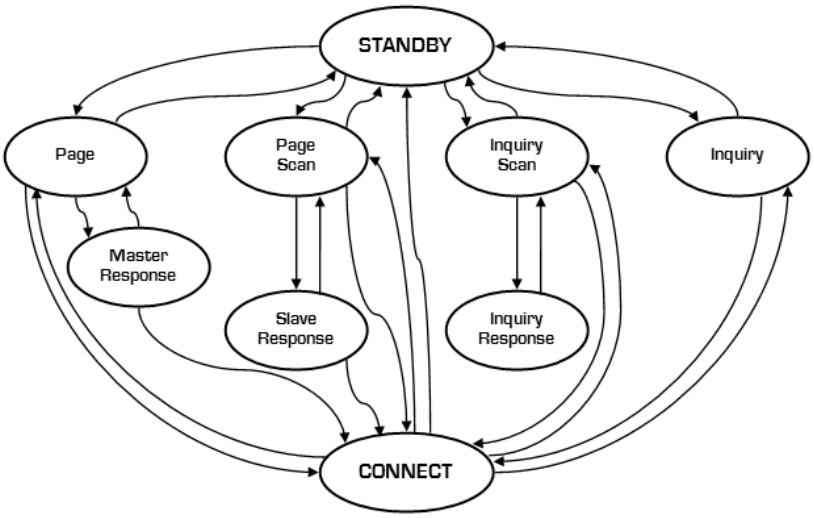


Figure 2.7: The state diagram of Bluetooth.

When the slave is in `CONNECT` mode, there are three different low power modes as regards of the purpose of the slave. `HOLD` may be used if there is no data to be transmitted. Units can thus be connected without data transfer, but in low power mode. `SNIFF` is also a low power mode in which the re-transmission can only be done by master in specified time slots, so-called sniff slots. Actually, the number of devices in a piconet is unlimited but the master can only have seven **active** slaves. `PARK` mode is used when the slave does not need to participate on the piconet channel, but still wants to remain synchronized to the channel. The time between `CONNECT` and `PARK` is only approximately 2ms, so the master is able to connect more than 7 slaves in the piconet. Figure 2.8 illustrates the connection procedures of Bluetooth. [24]

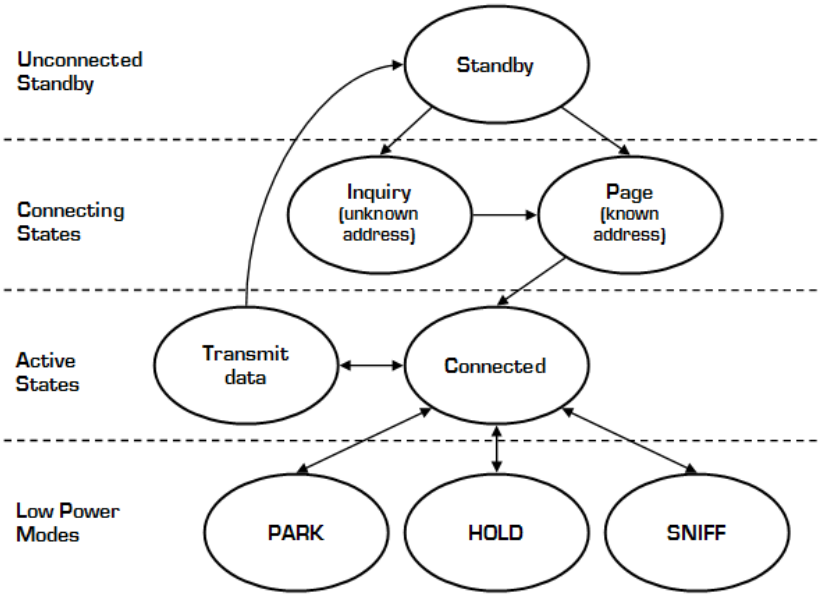


Figure 2.8: The connection procedures of Bluetooth.

2.6 Symbian OS

Symbian OS is an open operating system for mobile devices. It is licensed by the world's leading mobile phone manufacturers such as Nokia, Siemens, etc. It is designed for the functional demands and specific requirements of advanced 2G, 2.5G and 3G mobile phones. Symbian OS combines the power of an integrated applications environment with mobile telephony, bringing advanced data services to the mass market.

Mobile phones have a limited amount of memory, so the operating system must be compact and still provide a rich set of functionalities. The five key characteristics are the basis for the design and development of Symbian OS:

1. *Small, but always accessible mobile devices*
   Small mobile devices create high expectations. The device must provide many hours of operation to assure availability. The operating system must be designed efficiently to ensure quick boot sequences when turned on and long-lasting usage.

2. *Addressing mass-market*
   Reliability is a prime issue for mass-market phones. Mobile phones should never lock up or come with a significant defect. Reliability requires good software engineering and error-handling framework. Memory leaks should also be avoided due to a mobile device's limited amount of memory.

3. *Handling intermittent connectivity*
   Mobile connectivity is occasional and intermittent. Incomplete coverage and fade-outs while moving requires the operating system to provide solid control in connectivity. The operating system must offer multi-tasking, communication-capable real-time performance and a rich set of communication protocols.

4. *Product diversity*
   Mobile devices diverge in input mechanisms, shapes, size of the screen, etc. To support distinct devices, Symbian OS includes a common core with functionalities like multi-tasking, user interface framework, data service enablers, application engines, personal information management (PIM) and wireless communication. Phone manufacturers are active to extend Symbian OS and create highly differentiated products.

5. *Open platform*
   An operating system for the mass-market must be open for third-party development. Symbian is committed to open standards and is actively working with existing standards.

Symbian OS is very strict in memory control to preclude memory leaks. The allocated memory has to be released by the application programmer when the memory is not needed anymore. if the memory, is not released it could become depleted and the operating system runs out of memory. The need for strict memory handling comes from the fact that the handheld devices have little memory and are almost never rebooted, so each piece of memory is precious.

3 BlueCheese

BlueCheese is a mobile peer-to-peer middleware that is used between Bluetooth's transmission protocols and mobile peer-to-peer applications. BlueCheese contains interfaces and functionalities for mobile peer-to-peer applications (Application Programming Interface (API) of BlueCheese is shown in appendix A). The applications can connect to other devices and transfer data via BlueCheese. This chapter is based on the Software Design [1] and Specification [2] documents of BlueCheese.

3.1 Protocols and Features

BlueCheese utilizes Bluetooth in data transmission procedures. L2CAP and RFCOMM are used as data transmission protocols and SDP for service discovery (see figure 2.6 on page 10). The architecture of BlueCheese is presented in figure 3.1.

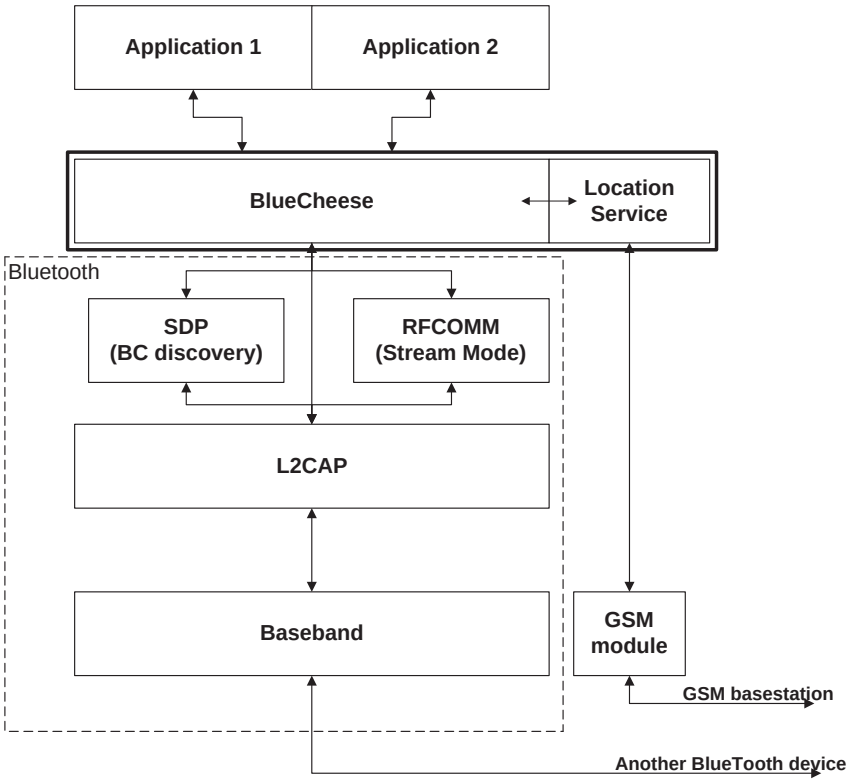


Figure 3.1: The architecture of BlueCheese.

BlueCheese can only be connected to one other BlueCheese at a time. Other devices are put into queue to wait until the prevailing connection is terminated. However several peer-to-peer applications may run on the connected devices and exchange data at the same time.

Location Service is also a feature of BlueCheese. It opens up a possibility to request current location information for applications that have a session to BlueCheese. The service can also compare two different locations with a rough estimation of distance.

The BlueCheese class diagram is represented in figure 3.2. There are following classes:

- *CBcSession* is used for an application session.
- *CBcSessionServer* handles application sessions.
- *CReceiveQueue* is the queue for incoming data.
- *CBcGsmLocator* is used for GSM locating.
- *CBcCore* is the central class coordinating the whole system and classes by passing messages between them.
- *CBcNotifier* discovers the nearby devices.
- *CDeviceQueue* maintains all discovered BlueCheese devices.
- *CBcCommunicator* is the central class for handling Bluetooth operations.
- *CTransmitQueue* is the queue for outgoing data.
- *CBcListener* listens for other devices' connection requests.
- *CBcConnector* is used for establishing connections.
- *CBcReader* receives incoming data.
- *CBcWriter* sends outgoing data.

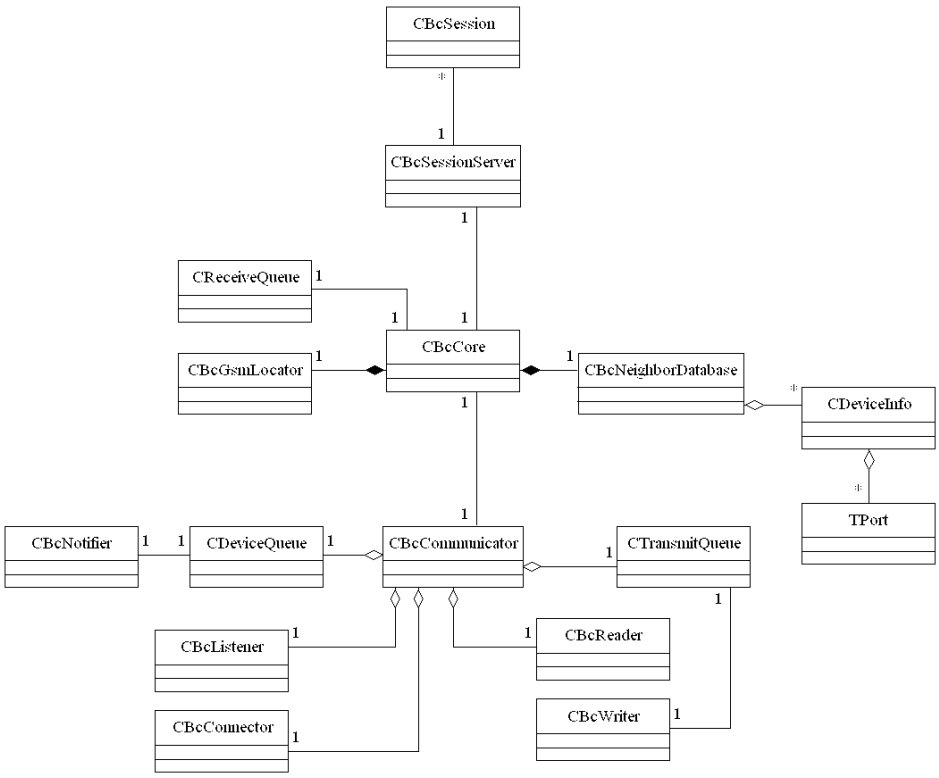


Figure 3.2: The class diagram of BlueCheese.

3.2 Functionality

This section describes the functionalities of BlueCheese.

Connections and Sessions

The terms *connection* and *session* mean different things in BlueCheese. Connection represents the connection between the same layer of two devices and session between the layers of the same device. Figure 3.3 clarifies the difference between connection and session.

When a mobile peer-to-peer application is started on a mobile device, it must first establish a session to BlueCheese. Every application has unique ID in BlueCheese (like a porting system), which allows several applications to have a session to BlueCheese simultaneously.

The interaction between BlueCheese and applications is implemented by events. The application can request the offered service by a function call that is transformed into an event. The event is handled by BlueCheese as soon as possible. Respectively the BlueCheese can inform the application by events such as incoming data

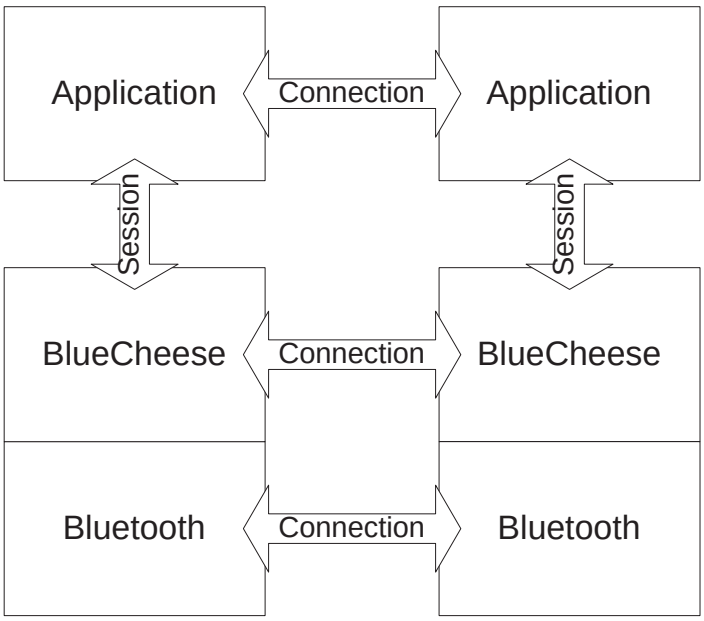


Figure 3.3: Connections and sessions.

or disconnection. Event-typed interaction requires event handling class in the application.

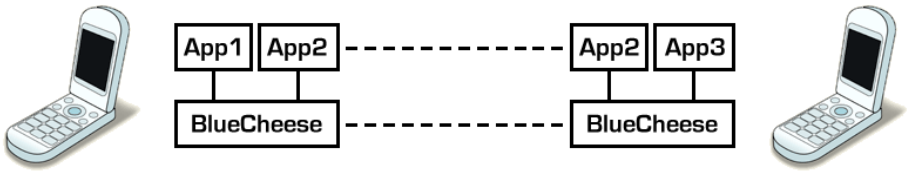
When two BlueCheese devices meet each other, they establish a connection if they have not recently met. In connection establishment, the information is being exchanged regarding to applications that have a session to BlueCheese. Those applications that are running in both devices are being informed that the connection is established and the data transfer is possible. Applications can exchange their data and the connection is terminated when all data has been exchanged and each application has informed BlueCheese about it. Figure 3.4 illustrates connections and sessions between two devices and the functionality of BlueCheese.

Device Handling

In all relationships between two devices, one is a master and the other one is a slave. The master establishes a connection and the slave waits for the requests of the master. The master also handles the device queueing. One device can be a master to one connection and a slave to another connection, because the master is always the device with a bigger Bluetooth address. A Bluetooth address is a 48-bit integer.

(a) Two applications (App1 and App2) make a session to BlueCheese.

(b) BlueCheese makes a connection to another device with BlueCheese. Information about applications is exchanged and the applications that both devices have are informed (in this case App2) about the connection establishment.



(c) App2 exchanges data and informs the BlueCheese that all the data has been exchanged.

(d) The connection is terminated by BlueCheese.

Figure 3.4: Example of connection and session establishment.

Data Exchange

The content of a BlueCheese header packet is illustrated in figure 3.5. The `Version` field contains the version of BlueCheese and the data type is specified with `Type`. The `OpCode` field could include a specific operation code such as the information for BlueCheese that the data includes the other devices' ports (applications having a session). The data for certain application is specified with `Port` and the length of the data in the `DataLengthBytes` field.

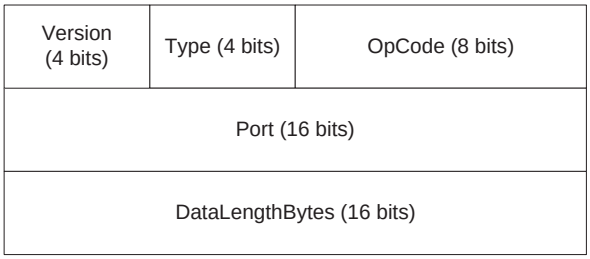


Figure 3.5: The content of the header packet of BlueCheese.

Location Service

Location Service is an optional feature of BlueCheese. It provides location information to the applications that have a session to BlueCheese. Location information is based on the Global System for Mobile Communications (GSM) network infrastructure. A GSM network consists of local areas, which include cells (see figure 3.6). Location Service can also compare locations and estimate their distances. Locations based on different networks cannot be compared.

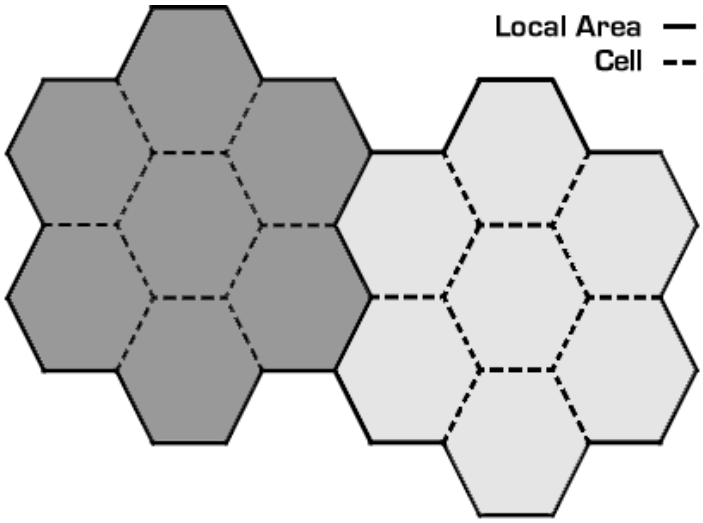


Figure 3.6: GSM network infrastructure.

The estimation is made as follows:

- *Unknown*, if locations have different networks.
- *Far away*, if locations have the same network but different local areas.
- *Quite close*, if locations have the same local area but different cells.
- *Very close*, if locations have the same local area and the same cell.

Reliability and Security

To allow the data to spread widely, BlueCheese does not use authentication or authorization in connection establishment. There is no security risk, because BlueCheese is designed for transferring data, not for executing the received data.

3.3 Modifications and Improvements

Sections 3.1 and 3.2 introduced the BlueCheese as it was meant to be according to the requirement specification of the MoPeDi software project at the University of Jyväskylä. During this thesis work it was realized that all of the requirements cannot be implemented to BlueCheese due to limitations of technologies, especially restrictions of Bluetooth.

One significant restriction of the tested devices (Nokia N-Gage, Nokia 6600 and Nokia 6630) was that they could only execute one Bluetooth operation at a time. Therefore the following removals or improvements have been made:

- *Leaving out of RFCOMM*
  L2CAP is used as a connection establishment protocol, so the data transmission is also committed to L2CAP because there cannot be two different protocols working at the same time.

- *Removal of Master/Slave definition*
  The Master/Slave definition was planned only for preventing duplicate connections between two devices, so it is not needed anymore.

- *Interference between transmission and device search*
  In the MoPeDi project it was planned to do the device search every 5 seconds. Consequently the device search consumes all of Bluetooth's capability since the devices are capable of doing one Bluetooth operation at a time. Approaches to solve this kind of interference problem were:

  - to do the device search at long intervals for improving the connection establishment.
  - to put the device search for a sleep during the data transmission.
  - to randomize the interval for preventing devices to search in the same phase interfering connections between them.

- not to do the device search if some other device is already searching devices.

- *Dropping multi-hopping*
  BlueCheese was considered to support multi-hopping, but that was left out of the implementation due to the complexity of execution and the absence of scatternet capability in the devices. Scatternet inability is illustrated in figure 3.7 wherein a connection cannot be established with a connected master (in the left) nor a connected slave (in the right). In spite of all, it is possible to have multi-hop by using point-to-multipoint support found in Nokia Symbian Series 60 devices. Then the master device could function as a router for data transmission to the slaves. This idea of multi-hopping is basically a "two-hopping", so the benefit of implementing it is small. The figure 3.8 illustrates the point-to-multipoint connection.

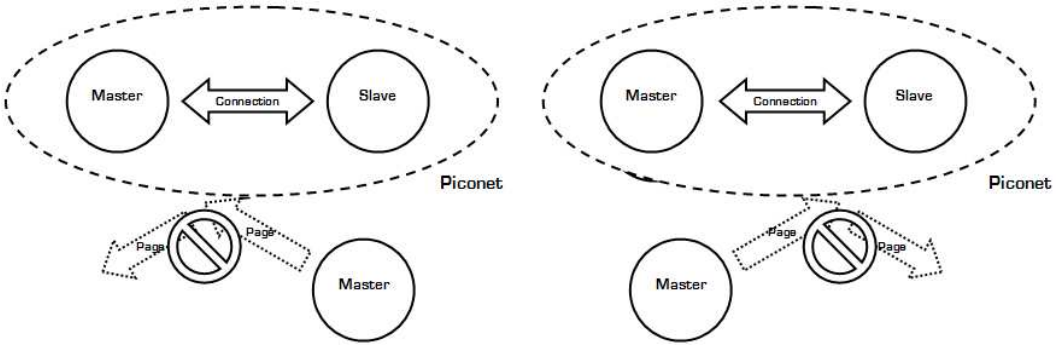


Figure 3.7: Failed scatternet initiations.

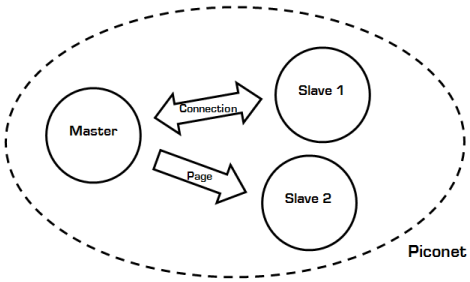


Figure 3.8: Point-to-multipoint connection.

3.4 Comparison of BlueCheese and Nokia Proximity Toolkit

Mobile Peer-to-Peer applications are divided into two general areas:

- *Active Collaboration*
  User activity is needed in active collaboration, which focuses to exchange information by using non-intrusive user notifications. Nokia Proximity Toolkit is well suited for active collaboration applications. BlueCheese on the other hand is not adequate enough for active collaboration applications due to the lack of manual device search and connection establishment for providing user actions.

- *Passive Collaboration*
  Passive collaboration aims to collect and pass information to the users without user interaction. BlueCheese is mainly designed for passive collaboration applications such as Gasoline Price Comparison System. Nokia Proximity Toolkit can handle passive collaboration applications, but not very effectively due to long (20 minutes) intervals of automatic device search and connection establishment. Consequently, the user action is needed almost every time for connection establishment.

Building blocks for Mobile Peer-to-Peer applications include common functionalities. These features are described as services. [29]

- *Presence Awareness Service* provides the information of active users in communication range.

- *Message Exchange Service* allows sending and receiving messages between nearby users.

- *Information Filtering Service* provides a filtering mechanism for preventing SPAM or other irrelevant information.

- *Information Distribution Service* offers three choices for sharing the received information; sharing straight away, choosing when and with whom to share or not sharing at all.

- *Security Service* offers sign and encrypt operations for data communications.

- *Identity Management Service* offers identification management for the system. Users may appear on THE system anonymously, under a pseudonym or with assigned identifies.

- *Service for Incentive Schemes* offers stimulative bonus for individual users for using the application.
- *Reputation Service* provides the build of reputation in the application. Other users might value the received information based on the reputation of the sender.
- *User Notification Service* informs the user of incoming information that may require instant response.

Table 3.1 summarizes the common services and their conceptual usage in BlueCheese and Nokia Proximity Toolkit.

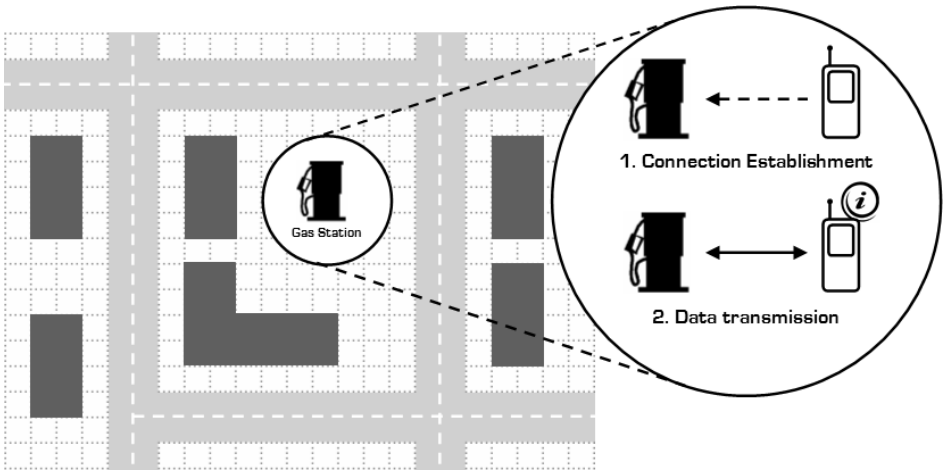
	BlueCheese	**Nokia Proximity Toolkit**
Presence Awareness Service	YES	YES
Message Exchange Service	YES	YES
Information Filtering Service	NO	NO
Information Distribution Service	NO	NO
Security Service	NO	YES
Identity Management Service	YES	YES
Service for Incentive Schemes	NO	NO
Reputation Service	NO	NO
User Notification Service	YES	YES

Table 3.1: Services of BlueCheese and Nokia Proximity Toolkit.

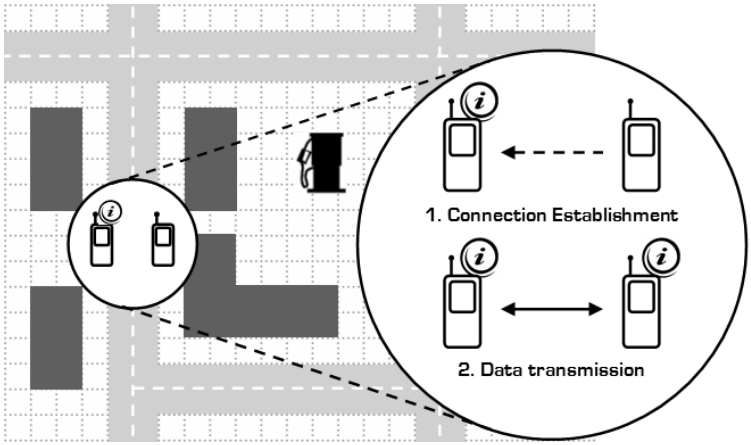
3.5 Gasoline Price Comparison System

The idea for the Gasoline Price Comparison System (GPCS) application emerged from professor Jarkko Vuori from the University of Jyväskylä. GPCS is a mobile application, which can be run in the Symbian OS mobile devices. Its task is to collect information about the gas prices of different gas stations and then make decisions on where to refuel. Gasoline Price Comparison System has been devised by Oleksiy Volovikov [32].

In the system a mobile device gets the information of the gas price from the gas station where the car is refuelled. Then it starts to spread the information to other mobile devices. Figure 3.9 represents the data spreading in Gasoline Price Comparison System.



(a) In the future it is possible to make payments with a mobile device, so the gas station could give the gasoline price information in the bill when the fuel is paid.



(b) When the mobile device with gasoline price information meets another device with GPCS, the information is exchanged between them. Thereby the data can spread to other devices as well.

Figure 3.9: Example of information diffusion in Gasoline Price Comparison System.

4 Measurements and Analysis

In the test cases there were three different phones, which each had a different version of Symbian OS:

- *Nokia N-Gage* with Symbian OS 6.1 and Bluetooth v1.1
- *Nokia 6600* with Symbian OS 7.0 and Bluetooth v1.1
- *Nokia 6630* with Symbian OS 8.0 and Bluetooth v1.2

Tools for test cases were Symbian 6.1 and 7.0 SDKs, Nokia Connectivity Framework 1.2, Bluetake BT009X Bluetooth USB Adapter and the phones mentioned above (see figure 4.1).

Figure 4.1: Phones and adapters in the test cases.

All the tests were made indoors at a room temperature of approximately 20 degrees celsius. There may have been disruptive factors that may have affected the test cases like for example a possible use of WLAN networks, other Bluetooth devices or microwave ovens.

4.1 Battery Power

Battery Power was measured in three different ways for all three phones: standby time, continuous device discovery and continuous data transfer.

By using these measurements we can estimate the Bluetooth power consumption, which is in a significant role in the analysis of Mobile Peer-to-Peer middlewares.

4.1.1 Standby Time

Standby time was measured simply by using the phone's power on without having any data or network connections or using the power consuming applications like for example camera. The test was made ten times for all the phones. Table 4.1 shows standby times for each phone with ten measurements t_1 - t_{10}.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
Nokia N-Gage	70	60	142	88	96	126	82	136	86	78
Nokia 6600	127	126	114	122	110	118	125	120	114	122
Nokia 6630	220	178	186	204	208	194	176	202	192	180

Table 4.1: Standby times of Nokia N-Gage, 6600 and 6630 in hours.

Now we can calculate descriptive statistics for measurements. The formula of mean is represented in equation 4.1 and equation 4.2 shows the formula of standard deviation. Table 4.2 shows the minimum, maximum, mean and standard deviation of standby times for each phone.

$$\bar{t} = \frac{1}{n}\sum_{i=1}^{n} t_i \tag{4.1}$$

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(t_i - \bar{t})^2} \tag{4.2}$$

	min	max	$\bar{t}$	σ
Nokia N-Gage	60	142	96.4	28.4
Nokia 6600	110	127	119.8	5.7
Nokia 6630	176	220	194.0	14.5

Table 4.2: Descriptive statistics of phones' standby times.

Nokia informs that Nokia N-Gage has a standby time of 150-200 hours. As can be seen from the chart, 150 hours is nearly achieved two times. The measured standby times vary a lot, showing signs of unsuitable battery or incomplete charging.

Nokia 6600 has even 240 hours of standby time according to Nokia, but the measured times are only half of that. Although the standby time does not vary much, the maximum capacity cannot be reached. The battery might be worn out and not as good as a new one.

Nokia 6630 is the newest phone in the test. Nokia advertises almost 11 days of standby for 6630. The measured values reach 9 days which is very customary.

4.1.2 Continuous Device Discovery

Standby time with continuous Bluetooth activity was measured by having the device discovery in action until the depletion of the batteries. Table 4.3 shows the power consumption for each phone when there is continuous device discovery going on.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
Nokia N-Gage	6	8	7	7.5	6.5	7	7	8	7	6.5
Nokia 6600	8	9	8.5	8	8	9	8.5	8	9	8
Nokia 6630	11	10	10.5	11	11	10	10	9.5	11	10.5

Table 4.3: Standby times of Nokia N-Gage, 6600 and 6630 in hours during continuous device discovery.

Table 4.4 shows the minimum, maximum, mean and standard deviation of standby times for each phone during continuous device discovery (see formulas for calculations in equations 4.1 and 4.2 on page 26).

	min	max	$\bar{t}$	σ
Nokia N-Gage	6	8	7.05	0.64
Nokia 6600	8	9	8.40	0.46
Nokia 6630	9.5	11	10.45	0.55

Table 4.4: Descriptive statistics of phones' standby times during continuous device discovery.

4.1.3 Continuous Data Transfer

Standby time was measured also by transferring data by turns without acknowledgement between devices all the time. Table 4.5 shows the power consumption for each phone when there is continuous data transfer going on.

Table 4.6 shows the minimum, maximum, mean and standard deviation of standby times for each phone during continuous data transfer (see formulas for calculations in equations 4.1 and 4.2 on page 26).

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
Nokia N-Gage	8.5	8	8	7.5	8	8	8.5	8	8	8
Nokia 6600	9.5	10	9.5	9.5	9.5	9	9.5	9	9.5	9
Nokia 6630	9.5	10	10	9.5	10.5	10	9.5	9.5	10.5	9.5

Table 4.5: Standby times of Nokia N-Gage, 6600 and 6630 in hours during continuous data transfer.

	min	max	$\bar{t}$	σ
Nokia N-Gage	7.5	8.5	8.05	0.28
Nokia 6600	9	10	9.40	0.32
Nokia 6630	9.5	10.5	9.85	0.41

Table 4.6: Descriptive statistics of phones' standby times during continuous data transfer.

4.1.4 Summary

Bluetooth power consumption is a major factor in estimating the suitability of BlueCheese for Mobile Peer-to-Peer communications. Figure 4.2 illustrates Bluetooth power consumption compared to standby time. According to graphs, **a device with Bluetooth action needs over 10 times greater power than the device in standby mode**.

Obviously the battery cannot provide the maximal capacity so the electric current cannot be calculated without knowing the current capacity. There are various factors that dictate the capacity of the battery like for example temperature, interior resistance of battery, all kinds of memory effects, charging situation, charging voltage, etc.

Capacity is proportional to electric current ($P = RI^2$; wherein P is capacity, R is resistance and I is electric current), so Bluetooth increases significantly the loss of power caused by the interior resistance of the battery and decreases the battery life. High power consumption can also turn off the phone before its time, because the terminal voltage decreases during burdening the battery ($U = E - RI$; wherein U is terminal voltage, E is source voltage, R is resistance and I is electric current). The device can fuse, even if there could be current remaining, because of the need of minimum voltage for keeping it alive.

All tested phones use a lithium ion (Li-ion) battery. Li-ion batteries do not suffer from the memory effect. They also have a low self-discharge rate (approximately 5%

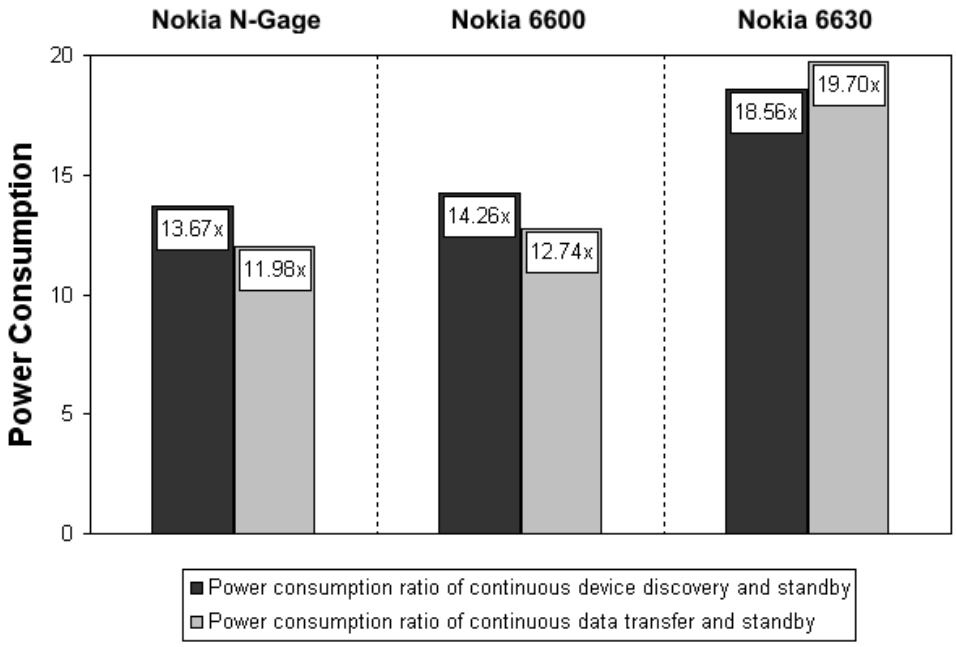


Figure 4.2: Bluetooth power consumption in contrast to standby time.

per month) and their lifespan remains relatively unaffected if they are kept plugged in after they have been fully charged. A unique drawback of the Li-ion battery is that its lifespan is dependent upon aging from time of manufacturing regardless of whether it was charged, and not just on the number of charge/discharge cycles. So the battery loses irreversibly capacity in the course of time. [34]

4.2 Data Transfer

There were three different tests for estimating data transfer capabilities: device discovery rate, connection establishment rate and data transfer rate.

4.2.1 Device Discovery Rate

Device discovery rate was measured in three circumstances wherein the number of devices around changed. Measurements, which are shown in table 4.7, were made with 0, 1 and 2 devices in the Bluetooth operation range. Table 4.8 shows the minimum, maximum, mean and standard deviation of device discovery times.

Nokia N-Gage	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
0 device around	13.5	13.5	13.8	13.5	13.7	13.5	13.5	13.7	13.6	13.5
1 device around	15.7	15.6	15.3	15.6	15.7	15.8	16.0	16.0	15.8	15.4
2 devices around	17.0	16.9	16.4	17.1	16.9	17.0	17.2	16.9	16.6	16.3

Nokia 6600	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
0 device around	14.0	13.6	13.5	13.8	13.7	13.7	13.9	13.6	13.7	13.6
1 device around	15.5	15.0	14.8	14.9	15.0	14.5	14.8	14.4	14.6	14.9
2 devices around	16.8	16.4	16.0	16.6	15.7	15.4	16.0	15.6	15.7	15.8

Nokia 6630	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
0 device around	13.2	13.4	13.3	13.4	13.3	13.4	13.3	13.4	13.3	13.4
1 device around	14.5	14.7	14.0	14.6	14.3	14.1	14.4	14.7	14.0	14.1
2 devices around	15.0	15.4	17.4	17.3	15.4	15.2	17.2	17.6	15.0	17.3

Table 4.7: The device discovery time in seconds for each phone with 0, 1 and 2 devices around.

Nokia N-Gage	min	max	$\bar{t}$	σ
0 device around	13.5	13.8	13.6	0.11
1 device around	15.3	16.0	15.7	0.23
2 devices around	16.3	17.2	16.8	0.30

Nokia 6600	min	max	$\bar{t}$	σ
0 device around	13.5	14.0	13.7	0.15
1 device around	14.4	15.5	14.8	0.31
2 devices around	15.4	16.8	16.0	0.46

Nokia 6630	min	max	$\bar{t}$	σ
0 device around	13.2	13.4	13.3	0.07
1 device around	14.0	14.7	14.3	0.28
2 devices around	15.0	17.6	16.3	1.15

Table 4.8: Descriptive statistics of phones' device discovery times.

Measurements show that **device discovery takes about 15 seconds** to complete regardless of the phone model. Search rate also depends on how many devices are around. Measurements illustrate that if the number of devices around grows by one, the device discovery time grows about 1 - 2 seconds.

According to the specification of Bluetooth version 1.1 (in Nokia N-Gage and Nokia 6600), the device search takes 10.24 seconds in an error free environment. An enhanced inquiry requires 5 seconds and interlaced inquiry 2.5 seconds in Bluetooth 1.2 (in Nokia 6630), but 10 seconds is often used due to compatibility reasons. SDP uses L2CAP for transporting protocol providing service search and PAGE - procedure (aka connection establishment) lasts 0.64 - 2.56 seconds in theory (average 1.28 seconds). Now we can calculate theoretical values for device searches. [19]

$$t = t_{INQUIRY} + n \times t_{SDP} \tag{4.3}$$

0 device around

$t_{min} = 10.24 + 0 \times 0.64 = 10.24$ s
$t_{mean} = 10.24 + 0 \times 1.28 = 10.24$ s
$t_{max} = 10.24 + 0 \times 2.56 = 10.24$ s

1 device around

$t_{min} = 10.24 + 1 \times 0.64 = 10.88$ s
$t_{mean} = 10.24 + 1 \times 1.28 = 11.52$ s
$t_{max} = 10.24 + 1 \times 2.56 = 12.80$ s

2 devices around

$t_{min} = 10.24 + 2 \times 0.64 = 11.52$ s
$t_{mean} = 10.24 + 2 \times 1.28 = 12.80$ s
$t_{max} = 10.24 + 2 \times 2.56 = 15.36$ s

The theoretical mean values differ from the practical values by about 3 - 4 seconds. There is no guarantee of successful inquiry in a noisy or error-prone environment since the packets may corrupt in those circumstances. Therefore the inquiry time may far exceed the theoretical time of 10.24 seconds. Figure 4.3 illustrates the device discovery time in all three cases compared to the theoretical values.

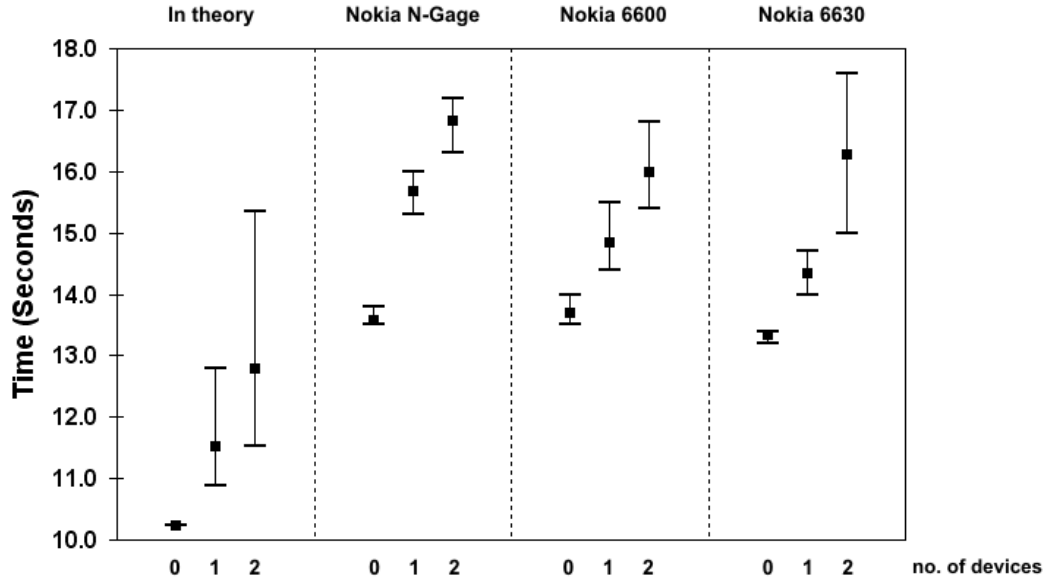


Figure 4.3: Min, max and mean of device discovery times with 0, 1 and 2 devices around in theory and in practice.

4.2.2 Connection Establishment Rate

Connection establishment rate was measured as a time that went from starting the Bluetooth inquiry to receiving a connection establishment event in the test program. Measurements, which are shown in table 4.9, were made with 2 devices wherein the devices were not in the same device discovery phase searching simultaneously. Thus we can estimate the connection establishment time of the device, otherwise we cannot be sure which one of the devices established the connection. Table 4.10 shows the minimum, maximum, mean and standard deviation of device discovery times.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
Nokia N-Gage	15.2	15.3	15.2	15.1	15.6	19.3	19.2	16.8	15.2	15.4
Nokia 6600	16.2	19.4	15.7	15.8	18.4	15.6	16.5	17.4	18.6	15.9
Nokia 6630	18.4	18.6	18.3	18.1	18.8	17.4	18.8	19.2	16.7	19.2

Table 4.9: Connection establishment time of phones.

As said in the previous section, average PAGE -procedure (connection establishment) lasts 1.28 seconds in theory. The range of paging fluctuates between 0.64 and 2.56 seconds.

	min	max	$\bar{t}$	σ
Nokia N-Gage	15.1	19.3	16.2	1.67
Nokia 6600	15.6	19.4	17.0	1.40
Nokia 6630	16.7	19.2	18.4	0.79

Table 4.10: Descriptive statistics of connection establishment rate.

The BlueCheese communication part, which controls Bluetooth operations, works like a state machine which tries to connect every 3 seconds. If there are no devices found in the device list, BlueCheese waits 3 seconds to check and try again. In that case, 0 - 3 seconds (t_{DELAY}) needs to be added for the calculation of theoretical values. Also there were no other Bluetooth devices around to distract the inquiry (number of devices around: $n = 1$). Now we can calculate the theoretical values for connection establishment rate.

$$t = t_{INQUIRY} + n \times t_{SDP} + t_{PAGE} + t_{DELAY} \tag{4.4}$$

$t_{min} = 10.24 + 1 \times 0.64 + 0.64 + 0 = 11.52$ s
$t_{mean} = 10.24 + 1 \times 1.28 + 1.28 + 1.5 = 14.30$ s
$t_{max} = 10.24 + 1 \times 2.56 + 2.56 + 3 = 18.36$ s

The theoretical mean values differ from the practical values by about 2 - 4 seconds. The cause of that is in an error-prone test environment. Although the theoretical maximum value was exceeded rarely in the tests (only approximately 37 % of test values). Figure 4.4 illustrates the connection establishment rate in all three cases compared to the theoretical values.

Time to establish an L2CAP connection is small compared to the time to perform device discovery. This lengthy discovery time becomes critical in certain situations like for example Peer-to-Peer communication wherein the devices are in motion. On this account, there are plenty of developed solutions that use additional technologies for performing the device discovery. For example Radio Frequency Identification (RFID), Infrared Data Association (IrDA) and visual tags (like a bar code to Bluetooth address) are used to improve performance. These solutions are not decent for BlueCheese, because they assume that devices know the presence of each other whereas the key factor in BlueCheese is in information diffusion among unknown devices.

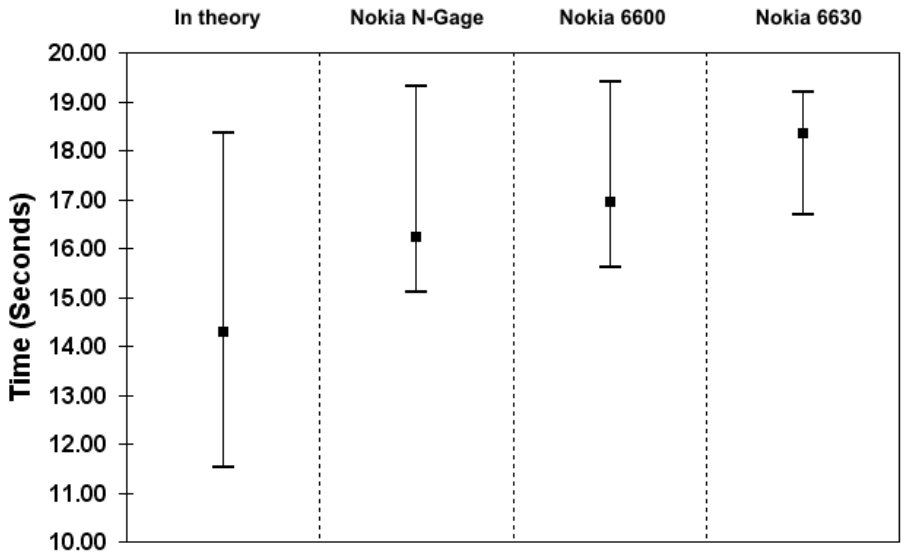


Figure 4.4: Min, max and mean of connection establishment times in theory and in practice.

4.2.3 Data Transfer Rate

Data transfer rate was measured between all the phones with two kinds of packets. Data transmission was done by turns without acknowledgements between devices measuring the amount of packets in one minute. The data in the packets were strings of 10 and 100 characters, so the payloads of the packets were respectively 80 bits and 800 bits. Measurements of data transfers are shown in table 4.11 and descriptive statistics (minimum, maximum, mean and standard deviation) are represented in table 4.12.

à 80 bit	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}
N-Gage ↔ 6600	88	90	88	90	90	89	85	88	87	90
N-Gage ↔ 6630	108	107	108	108	107	108	109	107	108	108
6600 ↔ 6630	100	96	96	98	99	98	98	100	97	98

à 800 bit	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}
N-Gage ↔ 6600	84	85	86	84	86	85	84	84	84	85
N-Gage ↔ 6630	101	103	102	101	102	103	102	102	103	102
6600 ↔ 6630	100	100	98	97	97	99	98	98	97	96

Table 4.11: Amount of packets transfered during one minute.

á 80 bit	min	max	$\overline{n}$	σ
N-Gage ↔ 6600	85	90	88.5	1.65
N-Gage ↔ 6630	107	109	107.8	0.63
6600 ↔ 6630	96	100	98.0	1.41

á 800 bit	min	max	$\overline{n}$	σ
N-Gage ↔ 6600	84	86	84.7	0.82
N-Gage ↔ 6630	101	103	102.1	0.74
6600 ↔ 6630	96	100	98.0	1.33

Table 4.12: Descriptive statistics of data transfer.

If the payload size of packet is grown 10 times from 80 bits to 800 bits, the time of transferring the packet to recipient is nearly the same. According to Bluetooth version 1.1 specification, the ACL link can handle the maximum of 732 kbps, so the time for transferring 80 and 800 bits payload should be equal. Consequently the measurements qualify sufficiently with the specification. Overall the **payload of 80 - 800 bits are transmitted approximately in one second between devices**.

BlueCheese uses two packets in one data transmission. A header packet is sent at first to specify the packet type and payload size. That feature must be taken into account when calculating the data transfer rate.

$$Bitrate(R) = \frac{number\ of\ bits}{unit\ of\ time} \tag{4.5}$$

80 bits payload

$$R_{N-Gage\leftrightarrow 6600} = \frac{88.5 \times (48+80)}{60} \approx 189 \text{ bps} = 0.189 \text{ kbps}$$

$$R_{N-Gage\leftrightarrow 6630} = \frac{107.8 \times (48+80)}{60} \approx 230 \text{ bps} = 0.230 \text{ kbps}$$

$$R_{6630\leftrightarrow 6600} = \frac{98.0 \times (48+80)}{60} \approx 209 \text{ bps} = 0.209 \text{ kbps}$$

800 bits payload

$$R_{N-Gage\leftrightarrow 6600} = \frac{84.7 \times (48+800)}{60} \approx 1197 \text{ bps} = 1.197 \text{ kbps}$$

$$R_{N-Gage \leftrightarrow 6630} = \frac{102.1 \times (48 + 800)}{60} \approx 1443 \text{ bps} = 1.443 \text{ kbps}$$
$$R_{6630 \leftrightarrow 6600} = \frac{98.0 \times (48 + 800)}{60} \approx 1385 \text{ bps} = 1.385 \text{ kbps}$$

4.3 Fault-tolerance

Fault tolerance was contemplated as general errors and data transfer faults.

4.3.1 General Errors

During the tests, BlueCheese crashed a few times. Crashes occured mainly in unnatural circumstances like for example:

- The test application crashed and was restarted
- The test application closed the BlueCheese session during the data transfer (not always)
- Bluetooth was turned off during the running of BlueCheese
- The phone was turned off during the data transfer

4.3.2 Data Transfer

Regarding to data transfer, there were no faults. The reason for that could be in L2CAP reliability.

L2CAP was used as a transmission protocol, which provides protocol multiplexing, packet segmentation and reassembly. In connection-oriented sessions, L2CAP sends data frames that are sequenced and numbered. Data frames must be delivered in order, so the L2CAP layer provides reliability.

Because the baseband layer does not distinguish data streams from separate upper layers, the L2CAP layer provides protocol multiplexing between the various types of the upper layer protocols. It gathers the upper protocol types and presents one data stream to the baseband layer.

Data stream segmentation and reassembly provides the freedom for the upper layers to make packets that are larger than the baseband layer will handle. L2CAP makes this possible by segmenting large upper layer data frames into frames that the baseband can handle. L2CAP also reassembles the frames from the baseband into the appropriate frames for the upper levels. This allows to the upper layers flexibility and efficiency in protocol interleaving. [18]

4.4 Location Service

Location Service works effectively and **location information is fetched in less than a second in every device**. Different types of networks however are problematic for location estimations. For example Nokia 6630 works in the 3G network and fetches the location information based on the 3G network structure (if there is a network available). So even if the devices were side by side, the location estimation could misinform on the basis of different types of networks. **2G and 3G location information cannot be compared with each other**.

5 Future Development and Aspects

This chapter proposes improvements and considers future scenarios for Mobile Peer-to-Peer communications. There are plenty of possibilities how to enhance wireless communications and improve the utilization of Mobile Peer-to-Peer middlewares like BlueCheese. In sections 5.1 and 5.2 extra features for BlueCheese are proposed, as against in sections 5.3 and 5.4 alternative ways for improving BlueCheese is discussed.

5.1 Combining Peer-to-Peer and Client-Server Architectures

One potential improvement in the future could be to exploit the benefits of both Peer-to-Peer and Client-Server architectures by using them simultaneously. The possibility to choose the architecture in BlueCheese could provide less lack of communication and better availability.

The battery exhaustion is also a big problem in BlueCheese. By using client mode the device can enlarge the standby time hugely because then the device search is not performed. When using the server mode, the device constantly acts as a master for Bluetooth connections searching other devices and trying to establish data links.

The usage of Client-Server architecture could bring opportunities to application development, too. Server side applications could be positioned everywhere. For example stores could use a server side application to offer special offers or the city could offer better information services. The benefit of stationary information distributors is that there is no battery problem and the communication could be done with client or Peer-to-Peer mode in the receiver side. Obviously the infrastructure is needed and the mobility of devices has an important role because stationary distributors work only as hot spots.

5.2 Additional Database Web Server

In addition to using BlueCheese with both Peer-to-Peer and Client-Server architectures, an additional database web server could bring a supplemental benefit. The centralized server stores device and application information, which is available to

all devices regardless of their locations. Naturally the device needs to be connected to the Internet using for example General Packet Radio System (GPRS) or Universal Mobile Telephone System (UMTS) techniques. The additional database web server is represented in figure 5.1.

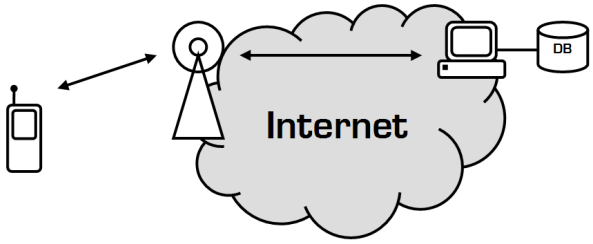


Figure 5.1: Additional web server with database.

Let's review a couple of scenarios concerning an additional database web server.

Scenario 1: *Added value for Gasoline Price Comparison System*
It is possible to ask the server where is the nearest gas station and what is the price. The server gives the information based on the location that the questioner has.

Scenario 2: *Added value for BlueCheese*
The server stores the device information too, so it is possible to ask the server if there are nearby devices. If not, the BlueCheese can be turned off for saving battery power.

Despite of two added values mentioned above, an additional database web server has disadvantages too. Mobile devices need the infrastructure for accessing the Internet and obviously the data transmissions over the Internet cost. Also the devices' locations change all the time, so the maintenance of location information is troublesome and inaccurate.

5.3 New Bluetooth Specifications

BlueCheese has restrictions that inhibit pure Peer-to-Peer communication. One of the most significant complications is that most of the smart phones provide only one Bluetooth action at a time. The limited bandwidth is the reason for supporting only one Bluetooth connection.

The Bluetooth specification 2.0 expands the bandwidth so that the data rate improves about three times. The progression is made by the Enhanced Data Rate (EDR)

technique, which provides the data rate above 2 Mbit/s. That might open up a possibility for devices to support multiple simultaneous Bluetooth connections. [5]

Another important advancement in the Bluetooth specification 2.0 is the power consumption. The specification lowers the power consumption to half which provides better standby times for the devices. In addition, devices with the Bluetooth 2.0 version are compatible with the devices having an older version of Bluetooth. [5]

Besides Enhanced Data Rate, the other improvements are designed by Bluetooth Special Interest Group. In 2005 the focus was to enhance security, power consumption and Quality of Service (QoS). These improvements advance the support of multiple concurrent data connections and also the piconet size is supposed to raise up to 256 devices. In 2006 the aim is to standardize Multicast technique, which could be a remarkable reform concerning Mobile Peer-to-Peer architecture.[5]

New Bluetooth versions give many additional advantages. Several connections and protocols can be used at the same time. The behaviour of BlueCheese could change dramatically because of the simultaneous Bluetooth operations like for example packet and stream data transmission modes or concurrent device search and connection. Consequently the utilization would improve and the data could disseminate more efficiently in Mobile Peer-to-Peer networks.

5.4 Other Wireless Standards

There are several wireless standards besides Bluetooth. This section describes the most prevalent architectures including the following standards: ZigBee, Wireless Local Area Networks (WLANs) and Ultra Wide Band (UWB).

5.4.1 ZigBee

Most of the wireless standards like Bluetooth or Wireless Local Area Networks (WLANs) are settled to high data rates for use of e.g. voice and video. So far there has not been a standard for applications which do not need high bandwidth, but which do need low latency and low power consumption. ZigBee is designed for industrial sensors and control devices, which are instances of those kind of applications. [36, 11]

ZigBee's origin can be dated to 1998 when Motorola started to devise low power mesh networking. ZigBee consists of IEEE 802.15.4 specification, which defines the RF capability of a system operating in three license-free Industrial Scientific

Medicine (ISM) bands – at 2.4 GHz globally, 915 MHz in North America and 868 Mhz in Europe. The IEEE 802.15.4 standard was based on Motorola's proposal and was ratified in May 2003 by the Institute of Electrical and Electronics Engineers (IEEE) and endorsed shortly afterwards by the ZigBee Alliance, which was formed in mid-2002 in co-ordination with Phillips, Motorola, Invensys, Honeywell and Mitsubishi. The IEEE defines only the Physical (PHY) and Medium Access Control (MAC) layers in its standard. ZigBee Alliance is developing a specification covering the network/link, security and application layers of the standard, which was ratified in December 2004. IEEE 802.15.4 / ZigBee stack is represented in figure 5.2. [4, 11, 26]

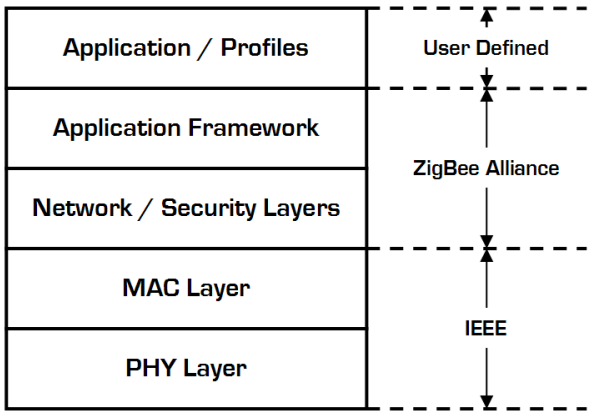


Figure 5.2: IEEE 802.15.4 / ZigBee stack.

Nowadays the ZigBee Alliance has over 70 member companies. Founder members of ZigBee Alliance, Samsung, Ember and Freescale have taken on the promoter status. They think that the market size of ZigBee could be valued at some hundreds of millions of dollars by 2007. Also the ZigBee Alliance is forecasting that there could be 50 to 150 ZigBee devices per household in a few years. [11, 26]

ZigBee uses three frequency bands, so the standard is able to operate globally. The specification for each band differs slightly. At 2.4 GHz there are 16 channels providing the maximum data rate of 250 kbps. In lower frequency bands there are 10 channels with 40 kbps for 915 Mhz and in 868 Mhz only one channel with the maximum data rate of 20 kbps. The multiplexing is done with Direct Sequence Spread Spectrum (DSSS) in all frequency bands, but the modulation techniques are different. 2.4 GHz uses Offset Quadrature Phase Shift Keying (OQPSK) whereas lower frequency bands are based on Binary Phase Shift Keying (BPSK). [26]

ZigBee supports three network topologies – star, mesh and cluster tree (see figure 5.3). The star network is very common and simple whereas the mesh network

has a capability of routing packets through various paths. The cluster tree (also called as hybrid) network is deployed for networks requiring complexity and it is a combination of star and mesh networks. [26, 36]

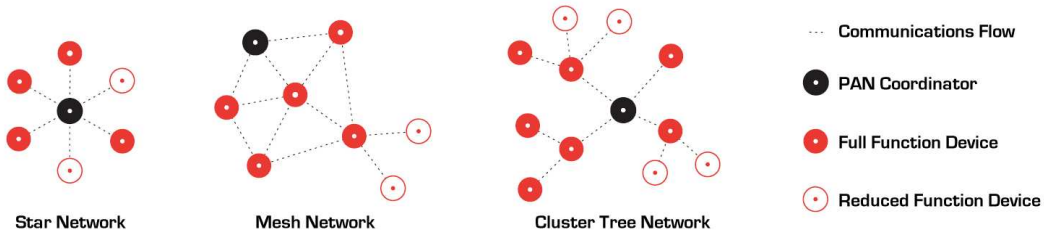


Figure 5.3: Network topologies of ZigBee.

The IEEE standard defines two types of devices: Full Function Device (FFD) and Reduced Function Device (RFD). FFD can function in any topology and is capable of being a Personal Area Network (PAN) coordinator or a router. RFD is limited to the star topology as a network-edge device and communicates only to a PAN coordinator or a router. [36]

ZigBee network requires one FFD as a PAN coordinator. This device sets up a network, transmits network beacons, manages network nodes, stores network node information and routes messages between paired nodes. Network may be extended through the use of ZigBee routers. [36]

ZigBee networks can use beacon or non-beacon environments. ZigBee uses the Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) technique for channel access avoiding unnecessary clashes in a non-beacon environment. Beacon mode is a mechanism for controlling power consumption in extended networks such as cluster tree or mesh. It enables all the network devices to know when to communicate with each other. The beacon wakes up the devices, which listen for their address and go back to sleep if they do not receive it. [36]

Data is transferred as packets with a maximum size of 128 bytes containing 104 bytes payload. The standard supports 64-bit IEEE addresses as well as 16-bit short addresses. The 64-bit addresses identify devices and the short addresses are used when the network is set up. The network supports over 65000 devices, which is much compared to Bluetooth's piconet with 8 devices. ZigBee also has an optional superframe structure with a time synchronizing method. This method allows Guaranteed Time Slot (GTS) for each device for the high-priority communication. [26, 36]

The key factor of ZigBee is low power consumption. ZigBee places reliance on a central mains-powered coordinator for yielding minimal power consumption to the

nodes. Advances in low-power design have enabled battery life to be typically measured in years whereas the Bluetooth power consumption in general is measured in days. In transmit/receive mode the ZigBee drains approximately twice less than Bluetooth.

Low latency is another important feature of ZigBee. CSMA-CA and beaconing bring in high throughput and low latency for devices. ZigBee devices can quickly attach, exchange information, detach, and then go to deep sleep to achieve a very long battery life. Bluetooth devices require about 100 times more energy for this operation. The latencies of ZigBee and Bluetooth are compared in table 5.1, which shows a great difference in non-active slave operations. [11, 26, 36]

	ZigBee	**Bluetooth**
New slave enumeration	30ms	20s
Sleeping slave changing to active	15ms	3s
Active slave channel access	15ms	2ms

Table 5.1: Comparison of latency with relation to ZigBee and Bluetooth.

According to table 5.1, ZigBee is a swift network builder compared to Bluetooth. Bluetooth requires about 20 seconds for an inquiry which is not sufficient for efficient Mobile Peer-to-Peer communications. The same operation can be done in less than a second with ZigBee. Although ZigBee has a quadruple lower bandwidth, it is a more suitable technique than Bluetooth for middlewares like BlueCheese because of its very low power consumption and extremely low latency. The problem is the lack of support in contemporary devices.

5.4.2 Wireless Local Area Networks (WLANs)

Wireless Local Area Networks (WLANs) are rapidly becoming a popular communication infrastructure. The growth of laptops and personal mobility products have strengthened the demand of WLAN infrastructure. Business users expect that they can access the private intranet of their own company by using other WLAN networks. [6]

The IEEE 802.11 standard has been referred to Wi-Fi (for wireless fidelity), which is in fact a trademark certifying device interoperability relative to a set of tests defined by Wi-Fi Alliance. Successful products are awarded Wi-Fi sticker providing interoperability between other Wi-Fi approved products.

The IEEE 802.11 standard family defines the physical and the MAC layer for

wireless communications within a short range. The series of IEEE 802.11 standards is illustrated in table 5.2. [12]

Standard	Description	Status
IEEE 802.11	WLAN; up to 2 Mb/s; 2.4 GHz	Approved 1997
IEEE 802.11a	WLAN; up to 54 Mb/s; 5 GHz	Approved 1999
IEEE 802.11b	WLAN; up to 11 Mb/s; 2.4 GHz	Approved 1999
IEEE 802.11g	WLAN; up to 54 Mb/s; 2.4 GHz	Approved 2003
IEEE 802.11e	New coordination functions for QoS	Task group development
IEEE 802.11f	IAAP (Inter-AP Protocol)	Approved 2003
IEEE 802.11h	Use of the 5 GHz band in Europe	Approved 2003
IEEE 802.11i	New encryption standards	Approved 2004
IEEE 802.11n	MIMO physical layer	Task group development

Table 5.2: IEEE 802.11 standards family. [12]

The IEEE 802.11 standards differ at the physical layer, but the family shares the same MAC layer. The MAC layer is responsible for overseeing that the devices take it in turn to access a shared transmission medium. Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA) and an optional Request to Send / Clear to Send (RTS/CTS) is used for sharing and accessing the radio channel. The MAC layer also takes care of packet transmission errors and makes sure that a lost or broken packet is retransmitted. This is simply achieved by returning positive acknowledgement (ACK) for every transmitted data packet. If the ACK does not come through, either the packet or ACK itself is lost, which necessitates resending the previous data packet. [6]

IEEE 802.11b is specified in 2.4 GHz band and it was ratified in 1999. The modulation is Complementary Code Keying (CCK) that is based on the original Direct-Sequence Spread Spectrum (DSSS) modulation of the IEEE 802.11 physical layer. IEEE 802.11b is usually used in the infrastructure topology, wherein an Access Point (AP) is providing communications to one or more clients located in the coverage area. Typical indoor range is tens of metres whereas outdoors it is over a hundred metres. IEEE 802.11b products can operate at 11, 5.5, 2 and 1 Mbit/s depending on the signal strength. When the signal weakens, the lower data rates are used with less complex and more redundant methods of encoding the data, which reduce the corruption and interference problems. [17, 20, 12]

In 2003, the IEEE approved the 802.11g standard. The modulation technique is Orthogonal Frequency Division Multiplexing (OFDM) that provides signal rates of

6, 9, 12, 18, 24, 36, 48 and 54 Mbit/s in the 2.4 GHz ISM band. The IEEE 802.11g devices are backward compatible with IEEE 802.11b devices, but the presence of an 802.11b participant significantly reduces the speed of an 802.11g network. The maximum range of IEEE 802.11g is a little bit greater than in IEEE 802.11b, but the coverage area of full bandwidth is much smaller. [17, 12]

5.4.3 Ultra Wideband (UWB)

In the past Ultra Wideband (UWB) was used for radars, sensing, military communications and niche applications. A large-scale research for using UWB for data communications started up in February 2002 when the Federal Communications Commission (FCC) unleashed the UWB spectrum. [35, 23]

Ultra Wideband (UWB) is a new very high bit rate radio technology under standardization in IEEE 802.15.3a. It operates in 3.1 - 10.6 GHz band and can offer even 500 times greater data rate than current Bluetooth. In addition, the spatial capacity (the ratio between the aggregated data transfer speed and the transmission area used) of UWB is enormous compared to other wireless standards (see figure 5.4). The UWB rulings issued by the FCC limit the transmit power to -41.25 dBm/Mhz, which relegates UWB for short-range high data rate (HDR) or very low data rate (LDR) in long-range. UWB is capable of hundreds of Mb/s in the distances less than 4 metres and above 100 Mb/s for ranges of 10-20 metres. [23, 27]

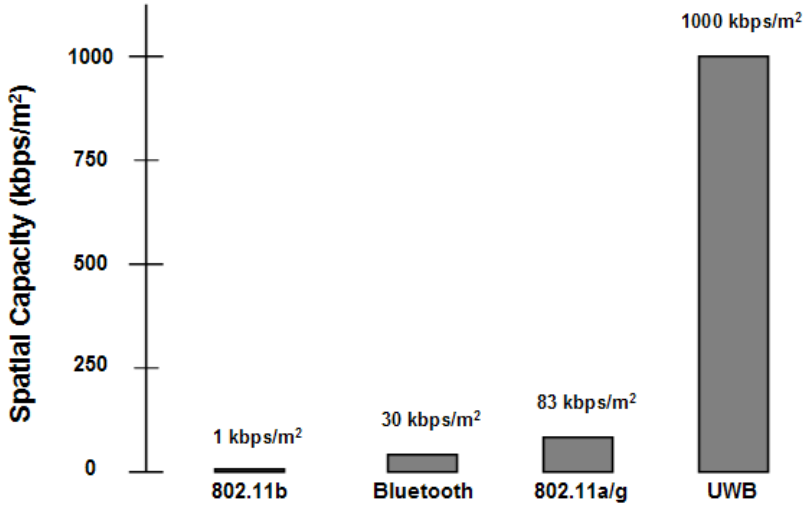


Figure 5.4: Spatial capacity of wireless standards. [23]

The major potential of UWB is its ability to switch between short-range HDR and wide-range LDR. The trade-off is facilitated by the physical layer signal structure. A

UWB transmission is based on pulses, where multiple pulses are combined to carry 1-bit information. Basically the data rate can be changed by simply increasing the number of pulses to carry 1-bit. More pulses per bit leads to a lower data rate and a greater transmission distance. [23]

Although with strict power restrictions, UWB holds great potential for ad hoc and peer-to-peer communications. The figure 5.5 shows envisaged applications that are well suited for UWB. UWB is assumed to conquer as a new wireless world as a result of a comprehensive integration of existing and future wireless systems. Integration consists of wide area networks (WANs), wireless local area networks (WLANs), wireless personal area and body area networks (WPANs and WBANs) as well as ad hoc and home area networks that link devices such as personal computers, home entertainment and mobile devices. [23, 27]

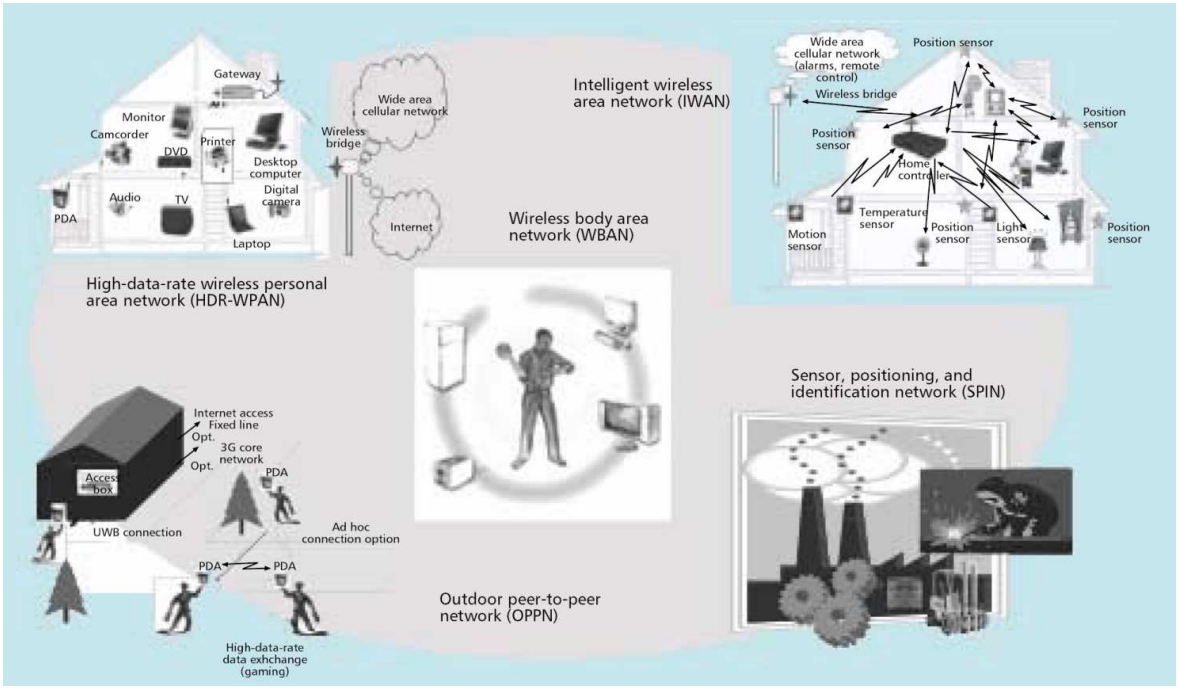


Figure 5.5: Envisaged UWB applications.

Currently, there are two competing UWB specifications. The following is a brief overview of these two specifications: [17]

- *OFDM-UWB*
  For widening the data over the spectrum, the WiMedia Alliance and Multi-Band OFDM Alliance (MBOA) special interest groups use orthogonal frequency division multiplexing (OFDM) that is used also in WLANs 802.11g and 802.11a. MBOA comprises over 170 member companies, including for example Texas

Instruments, Intel, Samsung, Nokia, Philips, etc. The OFDM technique is based on dividing the spectrum into 500 MHz subbands and using fast frequency hopping.

- *DS-UWB*
  Freescale Semiconductor (subsidiary of Motorola) defines another UWB approach, which uses the direct sequence (DS) technology for spreading the signal. DS-UWB uses a combination of a single-carrier spread-spectrum design and wide coherent bandwidth. This approach provides low-fading, optimal interference characteristics, inherent frequency diversity and precision ranging capabilities. DS-UWB transmits data by pulses of energy generated at very high rates.

Unfortunately, in early 2006 an industry working group announced that it would disband because two competing factions could not agree on a single UWB standard. Therefore consumers will once again face a technology war until one faction gains a clear lead.

Wireless Universal Serial Bus (WUSB)

Wireless USB (WUSB) is a new wireless extension to Universal Serial Bus (USB) intended to combine the speed and security of wired technology with the ease-of-use of wireless technology. WUSB is based on the UWB radio efforts by MultiBand OFDM Alliance and WiMedia Alliance.

Wireless USB is backward compatible with wired USB. WUSB uses a star topology, which is shown in figure 5.6. The host initiates and commands data transmissions by allocating time slots and data bandwidth among the devices connected to it. These relationships are referred to as clusters. A Wireless USB host can connect up to 127 devices.

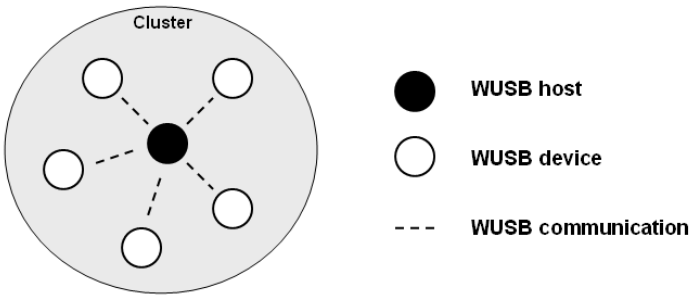


Figure 5.6: Wireless USB topology.

The topology also supports multiple clusters in the same area. By exchanging the data between clusters or devices, the second-level connection (i.e., a network) between two hosts can be used (see figure 5.7). WUSB also supports so-called dual-role devices, which can act as either hosts or clients. For example, a digital camera could act as a client when connected to a computer, and as a host when transferring pictures directly to a printer.

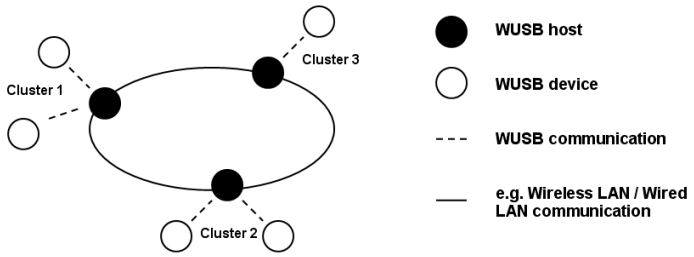


Figure 5.7: Combined Wireless USB clusters.

WUSB offers bandwidths of 480 Mbit/s at three meters and 110 Mbit/s at 10 meters. The maximum bandwidth is sufficient for example for managing multiple High-Definition television (HDTV) streams while still having the capacity to support other hi-speed data streams. As the UWB evolves and future process technologies take shape, the bandwidth could exceed 1 Gbps. The power consumption of WUSB is approximately 300 mW, but the target is to decrease it to 100 mW. It is estimated that WUSB will be available in smartphones in 2008. [31]

5.4.4 Comparison and Summary of Wireless Standards

Figure 5.8 represents the spectrum of the most common wireless standards in terms of two main functional characteristics – wireless radio range and data transmission rate. Although there are several wireless standards, the depiction attests that the standards are intended to differentiate from each other. Table 5.3 clarifies the characteristics of the most common wireless standards. [4]

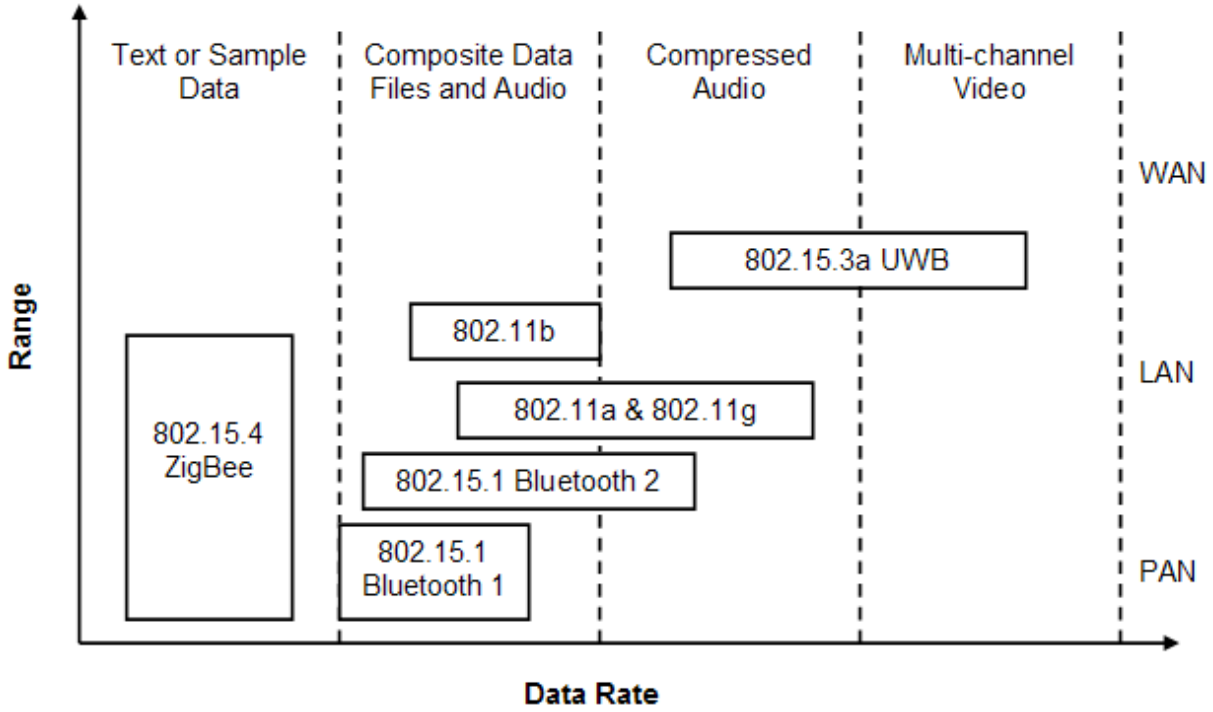


Figure 5.8: Wireless landscape with the most prevalent standards. [4]

	ZigBee	Bluetooth	WLAN 802.11b	WLAN 802.11g	UWB
Standard	IEEE 802.15.4	IEEE 802.15.1 (V1.1)	IEEE 802.11b	IEEE 802.11g	IEEE 802.15.3a
Max. data rate	40-250kb/s	1Mb/s	11Mb/s	54Mb/s	110-200Mb/s
Max. distance	30m	10m	100m	100m	10m
Frequency	868MHz, 915MHz, 2.4GHz	2.4GHz	2.4GHz	2.4GHz	3.1-10.6GHz
Channel bandwidth	0.3MHz, 0.6MHz, 2MHz	1MHz	25MHz	25MHz	0.5-7.5GHz
Number of RF Channels	1, 10, 16	79	3	3	1-15
Modulation	BPSK, OQPSK	GFSK	11MBaud QPSK (CCK coding)	OFDM64 + CCK (legacy)	BPSK, QPSK
Spreading	DS-SS	DS-FH	CCK	OFDM	Multiband
Power consumption	< BT	BT (40-100mW)	$\sim$ 4BT	$\sim$ 4BT	$\sim$ 2-3BT
Cost	$\sim$ 0.5BT	BT ($\sim$ 5$)	$\sim$ 4BT	$\sim$ 4BT	$\sim$ 1-2BT

Acronyms used: BT = reference to Bluetooth

Table 5.3: Summary of characteristics of leading wireless standards.

6 Conclusion

BlueCheese is a mobile peer-to-peer middleware, which provides interfaces and functionalities for mobile peer-to-peer applications. From an application programmer's point of view, BlueCheese is a server, which means that the applications have to establish a session for obtaining services like data transmissions over Bluetooth and location service.

The main component in case of measuring and analysing BlueCheese is Bluetooth. Mobile peer-to-peer communication requires an advantageous combination of power consumption, availability, data transmission range and data transmission rate.

Although Bluetooth has a low power consumption, the device search consumes too much power. If the device discovery is performed all the time, a full battery is exhausted within less than half of a day in all tested devices (Nokia N-Gage, Nokia 6600 and Nokia 6630). Because there are no low-powered low-level detectors for finding other Bluetooth devices, the device search has to be done manually at some intervals. If the interval is entended, the data spreading will be less efficient since there will be fewer connections whereas if the interval is decreased, the power consumption grows. Finding the optimal interval between device searches is very hard since it cannot be known when there is a considerable number of other Bluetooth devices around and especially devices with BlueCheese. There is a possibility of exhausting the battery without having discovered any device.

Availability signifies for the latency time of establishing the connection. The Bluetooth device discovery lasts approximately 10-20 seconds depending on how many devices are around. This longish discovery time becomes critical when devices are actively moving during the discovery. For example, if one device is moving past a second device, the time required to perform the discovery may in fact exceed the time during which the two devices are in range of one another. This could cause inability of communication between devices even if the device discovery runs without delays non-stop. The short range of Bluetooth with a lengthy discovery makes Bluetooth an unsatisfactory solution in a Mobile Peer-to-Peer environment.

The device discovery in BlueCheese is performed by completing the procedure all the way. To enhance the BlueCheese device and service discovery, a condensed inquiry might be used as well, like for example halving the inquiry time. Then the

probability of finding devices decreases, but the availability increases. Also a specific connection handshake protocol could be useful, because there are two connections in the connection establishment procedure: service search (SDP) and the actual data connection (L2CAP). Thus we might improve the time consuming connection establishment by creating only one connection instead of two.

Symbian OS development is also a problem in the construction of BlueCheese. For example, if two devices are within the achievable data transmission range, the discovery fails if both devices discover concurrently because of the restriction of one Bluetooth activity at a time. Therefore it is suitable to prevent one device from discovering, if the other device is already discovering. This brings along the problem of implementing that due to the dissimilarities of Symbian OS versions. In all Symbian OS versions it is possible. However a solution that works in versions 6.1 and 7.0, is not compatible with Symbian OS version 8.0 even if it is more recent. That is the reason for the new Bluetooth Application Program Interface (BAPI), which defines a whole new entity for using Bluetooth functionalities. Bluetooth API entails several benefits but the API cannot be utilized due to the fact that it is not supported in older versions. Therefore it is necessary to build a different version of the software for different Symbian OS or utilize conditional compilation.

Bluetooth as a transmission technique is not suitable for Mobile Peer-to-Peer communications on account of the reasons mentioned above. Bluetooth is very suited for Client-Server based communication like for example multiplayer gaming over Bluetooth or a stable point-to-point connection such as the wireless headset providing hands-free audio. The previous chapter presented some point of views regarding supportive actions or alternative prospects of enhancing BlueCheese or any other similar Mobile Peer-to-Peer middlewares.

A BlueCheese Application Programming Interface

Application Programming Interface (API) of BlueCheese, containing all the classes and functions, is described in this section. `BcInterface.h` is needed for including BlueCheese interface in the project for providing classes, functions and enumerations. `BlueCheese.dll` also has to be included in the project by adding it to the mmp -file.

Class `RBcSession`

This class is used for communicating with BlueCheese: registering, unregistering, closing connection, comparing locations, getting location and sending data.

```
RBcSession()
```
▷ Is the constructor of the class.

```
void ApplicationRegisterL(TBufC<16> aStringId, TUint
        aMeetTimeout, MBcEventHandler* aHandler)
```
▷ Is used to register an application to BlueCheese.

```
void ApplicationUnregister()
```
▷ Is used when an application wants to unregister from BlueCheese.

```
void CloseConnection()
```
▷ Is used to close the connection when the application has exchanged data with the connected device.

```
void CompareLocationsL(CLocationTable& aLocationTable)
```
▷ Is used to request locations comparison.

```
void GetLocation()
```
▷ Is used to get the current location.

```
void Send(TDesC8& aData, TBcSendingMode aSendingMode)
```
▷ Is used to send the data to the connected device.

Class `MBcEventHandler`

This class is used for handling the events from BlueCheese.

`MBcEventHandler()`
▷ Is the constructor of the class.

`void ReceiveEvent(TEvent* aEvent)`
▷ Is used to receive a pointer to TEvent object. This abstract function has to be overridden in the derived class.

Class `TEvent`

This class describes the events of BlueCheese.

`TEvent()`
▷ Is the constructor of the class.

`TEvent(TBcEventType aType, TAny* aData)`
▷ Is the constructor of the class for creating a new event containing the type and the data defined in the parameters.

`TAny* Data()`
▷ Is used to get a pointer to the data.

`void ExternalizeL(RWriteStream& aStream)`
▷ Is used to write TEvent object to the stream.

`void InternalizeL(RReadStream& aStream)`
▷ Is used to read TEvent object from the stream.

`TBcEventType Type()`
▷ Is used to get the type of the event (Types are: `EBcApplicationRegistered`, `EBcConnectionClosed`, `EBcConnectionOpened`, `EBcCurrentLocation`, `EBcErrorRegister`, `EBcErrorSend`, `EBcLocationComparison`, `EBcReceiveData`, `EBcUnknown`).

Class `CLocationTable`

This class is used to store location information and comparison results.

`CLocationTable()`
▷ Is the constructor of the class.

`C̃LocationTable()`
▷ Is the destructor of the class.

`void AddLocation(TLocation* aLocation)`
▷ Is used to add TLocation pointer to CLocationTable.

`TInt Count()`
▷ Is used to count how many TLocation objects are in CLocationTable.

`void ExternalizeL(RWriteStream& aStream)`
▷ Is used to write CLocationTable object to the stream.

`void InternalizeL(RReadStream& aStream)`
▷ Is used to read CLocationTable object from the stream.

`TLocation& operator[](TInt aIndex)`
▷ Is used to get a reference to one item of CLocationTable.

Class `TLocation`

This class is used to store location information.

`TLocation()`
▷ Is the constructor of the class.

`void TLocation(TUint aNetworkId, TUint aLocalAreaId, TUint aCellId, TDistance aDistance = EUnknown)`
▷ Is the constructor of the class. It is used by the application or the location service to create an instance.

`void ExternalizeL(RWriteStream& aStream)`
▷ Is used to write TLocation object to the stream.

`TUint GetCellId() const`
▷ Is used to get the cell id of TLocation object.

`TUint GetDistance() const`
▷ Is used to get the distance (Values are: `EFarAway, EQuiteClose, EVeryClose, EUnknown`) of TLocation object.

`TUint GetLocalAreaId() const`
▷ Is used to get the local area id of TLocation object.

`TUint GetNetworkId() const`
▷ Is used to get the network id of TLocation object.

`void InternalizeL(RReadStream& aStream)`
▷ Is used to read TLocation object from the stream.

`void SetDistance(TDistance aDistance)`
▷ Is used to set the distance to TLocation.

`void SetLocation(TUint aNetworkId, TUint aLocalAreaId, TUint aCellId)`
▷ Is used to set the location information to TLocation.

References

[1] O. Alanen, K. Haukimäki, T. Juonoja and P. Rönkkö, *MoPeDi Project - Software Design*, Jyväskylän yliopisto, 2004.

[2] O. Alanen, K. Haukimäki, T. Juonoja and P. Rönkkö, *MoPeDi Project - Specification*, Jyväskylän yliopisto, 2004.

[3] J. Al-Jaroodi, N. Mohamed and H. Jiang, *Distributed Systems Middleware Architecture from a Software Engineering Perspective*, Information Reuse and Integration (IRI) - IEEE International Conference, 2003.

[4] N. Baker, *ZigBee and Bluetooth strengths and weaknesses for industrial applications*, IEEE Computing & Control Engineering, Volume 16 (April-May), Issue 2, 2005.

[5] Bluetooth SIG, *The Official Bluetooth Website*, available in www-format <URL: `http://www.bluetooth.com`>, 2005.

[6] L. Burness, D. Higgins, A. Sago and P. Thorpe, *Wireless LANs – Present and Future*, BT Technology Journal - Volume 21, 2003.

[7] Cheese Factory, *Cheese Factory*, available in www-format <URL: `http://tisu.it.jyu.fi/cheesefactory`>, 2006.

[8] G. Coulouris J. Dollimore and T. Kindberg, *Distributed Systems: Concepts and Design, 3rd edition*, Addison-Wesley, 2001.

[9] L. Dao-Hui, L. Gang and G. Bao-Xin, *The Radio Networking of Bluetooth*, 3rd International Conference on Microwave and Millimeter Wave Technology Proceedings on 17-19 August, 2002.

[10] W. Emmerich, *Software Engineering and Middleware: A Roadmap*, Proceedings of the Conference on The Future of Software Engineering, 2000.

[11] C. Evans-Bughe, *Bzzzz zzz [ZigBee wireless standard]*, IEEE Review, Volume 49, Issue 3, 2003.

[12] E. Ferro and F. Potortì, *Bluetooth and Wi-Fi Wireless Protocols: A Survey and a Comparison*, IEEE Wireless Communications, Volume 12, Issue 1, 2005.

[13] X. Guang-Tao, L. Ming-Lu, D. Qian-Ni and Y. Jin-Yuan, *Stable group model in mobile peer-to-peer media streaming system*, Mobile Ad-hoc and Sensor Systems IEEE International Conference on 25-27 October, 2004.

[14] J.C. Haartsen, *Bluetooth: A new radio interface providing ubiquitous connectivity*, 51st Vehicular Technology Conference Proceedings (VTC-Spring Tokyo) - IEEE Volume 1, 2000.

[15] K. Haukimäki, *Mobiilien vertaisverkkojen väliohjelmistot*, Bachelor's Thesis, University of Jyväskylä, available in www-format <URL: `http://tisu.it.jyu.fi/cheesefactory/documents/MobiilitVertaisverkkoValiohjelmistotKandiTutkielmaHaukimaki.pdf`>, 2004.

[16] P.S. Henry and H. Luo, *WiFi: What's Next?*, IEEE Communications Magazine - Volume 40, 2002.

[17] Institute of Electrical and Electronics Engineers, *Welcome to the IEEE*, available in www-format <URL: `http://www.ieee.org`>, 2005.

[18] M.J. Jipping, *Symbian OS, Communications Programming*, John Wiley & Sons, Ltd, 2002.

[19] P. Jäppinen, *Wireless Services Engineering - Personal Area Networking*, Lecture Notes, available in www-format <URL: `http://www2.lut.fi/~pjappine/Lectures/WSE/PAN.pdf`>, Lappeenranta University of Technology, 2005.

[20] A. Kotanen, M. Hännikäinen, H. Leppäkoski and T.D. Hämäläinen, *Positioning with IEEE 802.11b Wireless LAN*, IEEE, 2003.

[21] J. Mitchell and A.J. Sánchez-Ruíz, *An Architectural Pattern for Adaptable Middleware Infrastructure*, Information Reuse and Integration (IRI) - IEEE International Conference on 27-29 October, 2003.

[22] Nokia, *Nokia - Nokia Sensor*, available in www-format <URL: `http://europe.nokia.com/nokia/0,1522,,00.html?orig=/sensor`>, 2006.

[23] I. Oppermann, *The Role of UWB in 4G*, Wireless Personal Communications: An International Journal, Volume 29, Issue 1-2 (April), 2004.

[24] Z. Pei, L. Weidong, W. Jing and W. Youzhen, *Bluetooth – The Fastest Developing Wireless Technology*, Communication Technology Proceedings - Volume 21 - International Conference on 21-25 August, 2000.

[25] P. Persson and Y. Young, *Nokia Sensor: From Research to Product*, available in www-format <URL: `http://www.perpersson.net/Publications/Sensor_DUX2005.pdf`>, Conference on Designing for User eXperience (DUX), San Francisco, 2005.

[26] R. Poole, *What exactly is . . . ZigBee?*, IEEE Communications Engineer, Volume 2, Issue 4 (August-September), 2004.

[27] D. Porcino and W. Hirt, *Ultra-wideband radio technology: potential and challenges ahead*, IEEE Communications Magazine, Volume 41, Issue 7 (July), 2003.

[28] R. Schollmeier, *A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications*, Peer-to-Peer Computing - First International Conference on 27-29 August, 2002.

[29] R. Steinmetz and K. Wehrle (Eds.), *Peer-to-Peer Systems and Applications*, Lecture Notes in Computer Science, Volume 3485, pp. 419-433, 2005.

[30] R. Shorey and B.A. Miller, *The Bluetooth Technology: Merits and Limitations*, Personal Wireless Communications - IEEE International Conference on 17-20 December, 2000.

[31] USB.org, *USB.org*, available in www-format <URL: `http://www.usb.org`>, 2006.

[32] O. Volovikov, *Mobile Encounter Networks Application: A Gasoline Price Comparison System*, Master's Thesis, University of Jyväskylä, available in www-format <URL: `http://tisu.it.jyu.fi/cheesefactory/documents/gpcs_final.pdf`>, 2006.

[33] O. Volovikov, M. Vapa, M. Weber, N. Kotilainen and J. Vuori, *Mobile Peer-to-Peer Encounter Networks and Their Applications*, University of Jyväskylä, 2005.

[34] Wikipedia, *Wikipedia, the Free Encyclopedia*, available in www-format <URL: `http://www.wikipedia.org`>, 2006.

[35] L. Yang and G.B. Giannakis, *Ultra-wideband communications: an idea whose time has come*, IEEE Signal Processing Magazine, Volume 21, Issue 6 (November), 2004.

[36] ZigBee Alliance, *ZigBee Alliance – Wireless Control That Simply Works*, available in www-format <URL: `http://www.zigbee.org`>, 2005.