Lappeenranta University of Technology

Department of Information Technology

Master's Thesis

Transparent Network Interfaces to Mobile Peer-to-Peer Environment

The subject has been approved by the council of the Department of Information Technology on January 21, 2004

Supervisors: Professor D.Sc. (Tech.) Jari Porras and D.Sc. (Tech.) Jouni Ikonen

Instructor: Professor D.Sc. (Tech.) Jari Porras

Lappeenranta April 23, 2004

Sami Saalasti

Korpraalinkuja 3, 314

FIN – 53810 Lappeenranta

Finland

+358 50 347 4215

sami.saalasti@lut.fi

ABSTRACT

Lappeenranta University of Technology
Department of Information Technology
Sami Saalasti

Transparent Network Interfaces to Mobile Peer-to-Peer Environment

Master's Thesis
2004
68 pages, 34 figures, 11 tables and 1 appendix
Supervisors: Professor D.Sc. (Tech.) Jari Porras, D.Sc. (Tech.) Jouni Ikonen
Keywords: wireless mobility, peer-to-peer, software assisted handover

The fact that most of new Personal Data Assistant (PDA) devices and smartphones have the ability to communicate via different wireless technologies has made several new applications possible. While traditional network model is based on the idea of static hosts, mobile devices can create decentralized, self-organizing ad-hoc networks and act as peers in the network. This kind of adapting network is suitable for mobile devices which can freely join and leave the networks. Because several different wireless communication technologies are involved, flexible changing of the networking technology must be handled in order to enable seamless communication between these networks.

This thesis presents a transparent network interface to mobile Peer-to-Peer environment which is named as Virtual PeerHood. Different wireless technologies and aspects of providing a seamless connectivity between these technologies are explored. The result is a middleware platform for mobile Peer-to-Peer environment, capable of handling several networking technologies.

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Tietotekniikan osasto
Sami Saalasti

Läpinäkyvien verkkorajapintojen hyödyntäminen langattomassa vertaisverkkoympäristössä

Diplomityö
2004
68 sivua, 34 kuvaa, 11 taulukkoa ja 1 liite
Tarkastajat: Professori TkT. Jari Porras, TkT. Jouni Ikonen
Hakusanat: langattomuus, vertaisverkot, ohjelmistollinen verkon vaihto

Nykyiset kämmenmikrot ja älypuhelimet sisältävät useimmiten jonkinlaisen mahdollisuuden kommunikointiin käyttäen erilaisia radiotekniikoita. Tämä kehitys on johtanut siihen, että kokonaan uudenlaisten sovellusten kehitys ja käyttäminen on mahdollistunut. Verrattuna perinteisiin langallisiin verkkoihin liikkuvat päätelaitteet voivat luoda hajautettuja, itsejärjestäytyviä tilapäisverkkoja (ad-hoc) joissa päätelaitteet toimivat verkon päätepisteinä. Tällainen mukautuva verkko soveltuu hyvin liikkuville päätelaitteille jotka voivat liikkua vapaasti verkon alueella. Koska käytössä on tällä hetkellä useita erilaisia langattomia verkkotekniikoita, täytyy niiden välillä järjestää toimiva ja joustava verkon vaihto jotta saavutettaisiin saumaton yhteys erilaisten verkkojen välillä.

Tässä työssä esitellään langattomaan vertaisverkkoympäristöön kehitetty rajapinta joka tunnetaan myös nimellä Virtual PeerHood. Työssä tarkastellaan erilaisia langattomia verkkotekniikoita sekä näihin liittyviä näkökohtia. Tiedon siirtoon käytetyn teknologian saumaton vaihto käyttäjän liikkuessa erilaisten verkkojen alueella on myös työn yhtenä aiheena. Tuloksena on saatu langattomaan vertaisverkkoympäristöön tarkoitettu usean verkkoteknologian hallitseva rajapinta.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

API	Application Programming Interface
ACL	Asynchronous ConnectionLess Bluetooth Connection
BSS	Basic Service Set
CSMA / CA	Carrier Sense Multiple Access / Collision Avoidance
ETSI	European Telecommunication Standards Institute
DHCP	Dynamic Host Configuration Protocol
ETSI	European Telecommunications Standard Institute
GGSN	General Packet Radio System Gateway Support Node
GPRS	General Packet Radio System
GSM	Global Standard for Mobile Communication
GTP	General Packet Radio System Tunnelling Protocol
HCI	Host Controller Interface
HLR	Home Location Register
IP	Internet Protocol
IEEE	Institute of Electrical and Electronics Engineering
ISM	Industrial, Scientifical and Medical
L2CAP	Logical Link Control and Adaptation Protocol
LAN	Local Area Network
LLC	Logical Link Control
MAC	Medium Access Control
NAT	Network Address Translation
PAN	Personal Area Network
PDA	Personal Digital Assistant
PDP	Packet Data Protocol

PTD	Personal Trusted Device
PTM	Point-to-Multipoint
PTP	Point-to-Point
PTP-CLNS	Point-to-Point Connectionless Network Service
PTP-CONS	Point-to-Point Connection Oriented Network Service
QoS	Quality of Service
RFC	Request for Comments
SDP	Service Discovery Protocol
SDAP	Service Discovery Application Profile
SGSN	System GPRS Support Node
SNDCP	SubNetwork Dependent Convergence Protocol
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
UDP	User Datagram Protocol
WAN	Wide Area Network
WLAN	Wireless Local Area Network

1 INTRODUCTION

Currently, Personal Data Assistant (PDA) -devices and smartphones are more common than ever and those are becoming a part of our everyday-life. Computing resources have increased constantly and devices have become smaller and more usable. Simultaneously, the fast development of wireless technologies has provided methods for more flexible communication between mobile devices, because current mobile devices tend to include at least one of these wireless technologies. Every technology has its limitations; question is how to use these different technologies together to make them stronger supporting each other.

Despite the current boom of Peer-to-Peer networks, concept of Peer-to-Peer is not new at all. Peer-to-Peer networks have been successfully applied from the dawn of the Internet when network was formed of distributed workstations communicating directly with each other. Military has used this technology also for ages, to provide communication links for the changing environment. However, latest file-sharing applications like Napster /1/ have made the concept of Peer-to-Peer more popular than ever. For our case, Peer-to-Peer networks provide a suitable environment for wireless mobile devices forming amorphous, constantly changing networks. Peer-to-Peer can adapt these network changes because there are no static hosts, just devices making connections with each other. What makes the situation more interesting, is the fact that current mobile devices are powerful enough to provide and use different services. This leads to environment where even the smallest PDA-devices devices can communicate directly with each other, using several different wireless technologies.

The problem with different mobile devices is that there hasn't existed a common interface for providing and using services. To fulfill this need, a Peer-to-Peer middleware platform, *PeerHood* has been implemented in Lappeenranta University of Technology, Laboratory of Communications Engineering. PeerHood provides all the functionality needed by an application developer to develop applications for mobile devices operating on Peer-to-Peer network using Bluetooth technology. This

functionality includes creating and advertising own services, finding other devices and provided services and of course, the possibility to create connections between found devices. The goal for this thesis was to extend the current concept of PeerHood to support also other wireless networking technologies and provide *Seamless Connectivity* between these technologies. As a result, a concept of *Virtual PeerHood* is defined and implemented. Seamless Connectivity defines a technology for changing the active networking technology automatically if the established connection weakens or breaks. Seamless Connectivity tries to always provide the best possible connection for the user while maintaining connectivity. To visualize Seamless Connectivity, a typical use-case is illustrated in Figure 1.

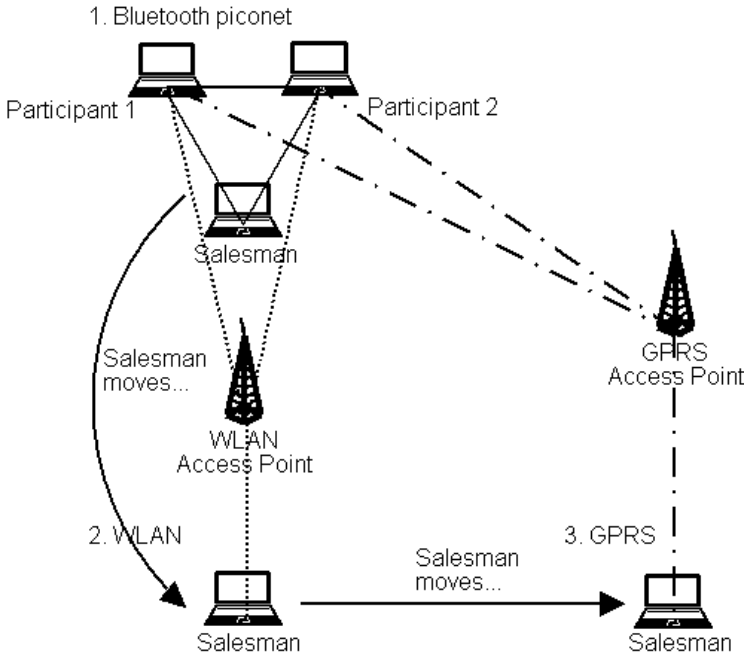

Figure 1.Example of Seamless Connectivity

In example case (Figure 1) a salesman uses Virtual PeerHood enabled Whiteboard application in business meeting to present his ideas for other co-workers. Because all the participants are sitting in a same room, Bluetooth is selected as a communication technology. Bluetooth can be applied only when devices are near each other, for example located in one room area. Before the meeting is over, the salesman has to leave because of other duties. When he walks away from the meeting room, Virtual PeerHood

notices that the Bluetooth connection is weakening and starts to use Wireless Local Area Network (WLAN) technology instead of Bluetooth. Because WLAN covers the whole apartment the salesman can still participate the meeting while moving away. When the salesman reaches the main entrance, Virtual PeerHood notices that the WLAN connection is weakening and starts to use General Packet Radio Service (GPRS) technology instead of WLAN. While traveling away in a bus, the salesman can still maintain connectivity and continue participating the meeting. Even thought, the connection has been changed two times, the salesman doesn't know anything about it and the whole operation has been performed automatically without need of user interaction. This is exactly the functionality what Virtual PeerHood should provide.

This thesis starts with an introduction to the Peer-to-Peer paradigm, continues with a presentation of different wireless technologies and wireless mobility. Finally a short review of Virtual PeerHood implementation is presented.

2 PEER-TO-PEER NETWORK ENVIRONMENT

Unlike the traditional client-server –architecture, Peer-to-Peer –based networks are *decentralized* systems with decreased dependency of a controlling server. The aim is to provide a network where devices can make direct connections with each other. Peer-to-Peer networks are a suitable environment for mobile devices because of the ability to adapt to changes as mobile devices may join or leave the network as they wish.

2.1 Models of Peer-to-Peer networks

Peer-to-Peer models can be divided into the following categories /2/

- Pure Peer-to-Peer
- Peer-to-Peer with a simple discovery server
- Peer-to-Peer with discovery and lookup servers
- Peer-to-Peer with discovery, lookup, and content server

Pure Peer-to-Peer

Pure Peer-to-Peer model works without relying on any central server. Peers act as a both server and client and thus form up the whole network. Peers can implement their own policy and make direct connections with each other. A Pure Peer-to-Peer model provides an advantage of creating effectively Peer-to-Peer connections without any overhead caused by handling of connections by external server. A Problem with this model can be the process of finding other peers. Example of pure Peer-to-Peer applications is Gnutella /3/. Model is illustrated in Figure 2.

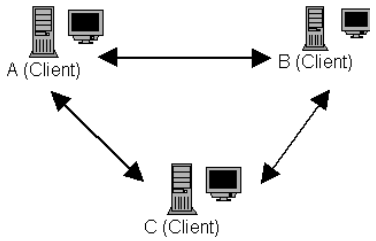

Figure 2. Pure Peer-to-Peer model

Peer-to-Peer with Simple Discovery Server

Peer-to-Peer network with Simple Discovery Server –model utilizes a centralized server for searching other peers. This server provides a list of available peers. Any interaction is made directly between peers. Thus, Peer-to-Peer with Simple Discovery Server is basically a Pure Peer-to-Peer network with centralized peer information database. Several Instant Messaging applications for example ICQ /4/ utilize Peer-to-Peer network with Simple Discovery Server –model to locate other users. The model is illustrated in Figure 3

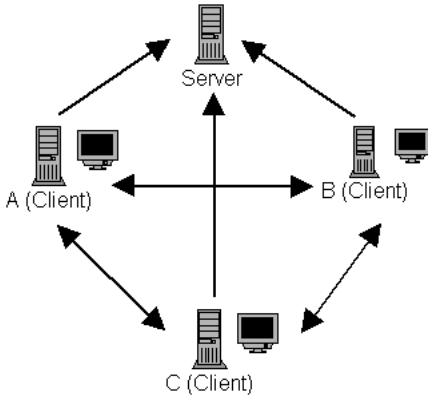

Figure 3. Peer-to-Peer with a simple discovery server

Peer-to-Peer with a Discovery and Lookup Server

Peer-to-Peer with a Discovery and Lookup Server –model defines a network which contains a centralized server for discovering peers and services which are provided by these peers. This model is enhanced model of Peer-to-Peer with Simple Discovery Server. Napster /1/ is based on Peer-to-Peer with a Discovery and Lookup Server –model providing indexing server of users and provided content. Model is illustrated in Figure 4.

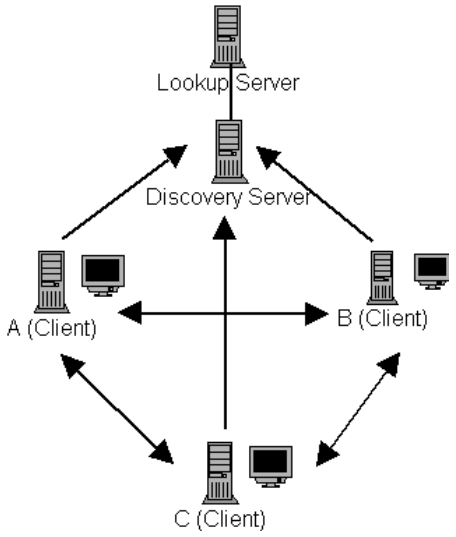

Figure 4. Peer-to-Peer with a Discovery and Lookup Server

Peer-to-Peer with a Discovery, Lookup, and Content Server

Peer-to-Peer with a Discovery, Lookup, and Content Server defines a model which brings Pure Peer-to-Peer model closer to traditional Client-Server –model. All the information of provided services and content is centralized from peers to the server. Peers do not make connections with each other but communicate only with the server which processes requests. Clear disadvantage of this model is the centralized server. If this server goes down, no service can be reached. To give an example, Seti@Home /5/ is based on Peer-to-Peer with a Discovery, Lookup, and Content Server –model. The model is illustrated in Figure 5.

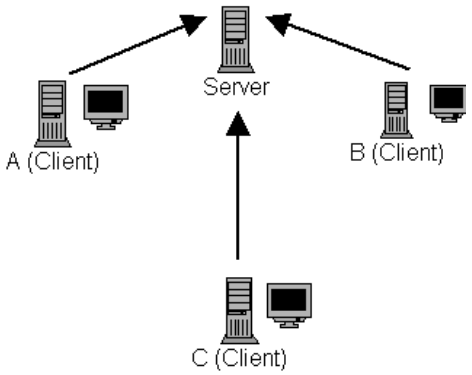

Figure 5. Peer-to-Peer with discovery, lookup, and content server

2.2 Peer-to-Peer Middleware

Peer-to-Peer middleware is software that sits between the local operating system of the peer and the application interfaces that the Peer-to-Peer application can call directly. Peer-to-Peer applications can still exist without any underlying middleware. However using middleware is more preferable: a common platform offers the application developer a possibility to concentrate on the application and not to low-level systems. It also offers to use existing libraries and software. While some common programming language is used, software can be ported to any architecture that supports the used middleware. It also offers easier maintenance. From user's point of view, common middleware offers familiar interface to Peer-to-Peer applications while different applications appearance may change.

Challenges concerning a Peer-to-Peer environment

Before starting to implement a Peer-to-Peer middleware, there are few challenges concerning the network environment which should be taken into account /6/. The challenges are:

- *Lack of trust* - Resources must be granted between peers who are unknown to each other
- *Hardware heterogeneity* - Devices participating to a Peer-to-Peer network may differ a lot. Devices may range from handheld devices to supercomputer.
- *Software heterogeneity* - Operating systems of connected devices may differ a lot. Peer-to-Peer networks are incorporated with peers running on any available operating systems.
- *Network heterogeneity* - Devices may use different networking technologies to connect to a Peer-to-Peer network and thus are expected to have different bandwidth and latency
- *Scalability* - Peer-to-Peer networks can be very different when considering the size of the network. The number of peers varies from few to millions. Napster file-sharing application is a good example of network which has grown to a humongous size.
- *Intermittency* - Systems and configurations are not fixed and static but

dynamically changing. Nodes may come and go as they wish.

- *Location* - Resources such as files, storage or executing programs may not move around during session.
- *Autonomy* - Peers share resources and collaborate while maintaining autonomy and being a final authority over their local system.
- *Local policies* - Peers can have different local policies depending on network and organization.
- *Distance* - Peers are expected to be geographically separated.

Peer-to-Peer middleware services for the application layer

Peer-to-Peer middleware creates a layer between the application and physical interfaces. This middleware layer should provide certain basic functionality for handling demands of Peer-to-Peer networking /6/. These functionalities are then utilized by the application which uses the Peer-to-Peer middleware. Some of these middleware services are illustrated in Figure 6.

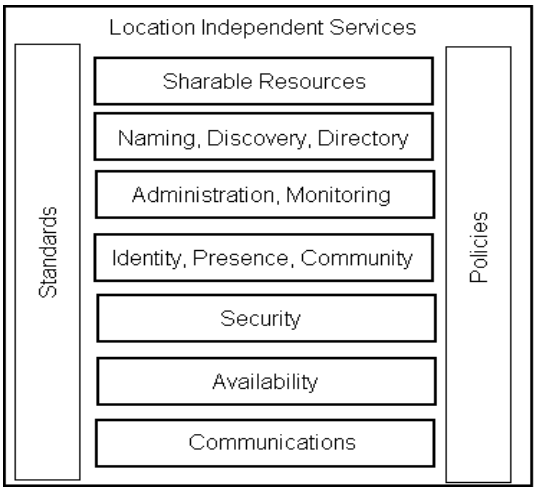

Figure 6. Peer-to-Peer middleware layers

The middleware services provided for Peer-to-Peer applications are:

- *Communications* - Middleware should provide service for crossing firewalls and systems utilizing Network Address Translation (NAT).
- *Availability* - Peers can leave and join freely in Peer-to-Peer network. Thus, middleware has to deal with situations when service is not available or it can't be reached. When this happens, a peer can for example search a new peer providing same service, queue message or just throw a notification for application that service couldn't be reached.
- *Security* - Security issues are emphasized in Peer-to-Peer networks. When communication takes place directly between peers there is always the concern of malicious peer which might try to make an attack against other peers. Because peers might not be known to each other beforehand, lack of trust is also present. The user and the application must manage access control between peer computers.
- *Identity, presence community* - Identity varies a lot depending of a type of Peer-to-Peer network. Identity of peer can be for example a human-readable name or then just simply a unique numeric identifier. Identity of peer can be utilized for example by a security layer performing authentication. Identities can be also used to form certain groups or communities among peers. These groups may apply their own policies and grant or permit special privileges to group members. These privileges which the middleware should handle could be for example permission to access shared resources.
- *Administration and monitoring* - Peers may want to inform how they are using resources and how much resources are available for others to use. Vice versa, other peers may want to have privacy and do not want to share this information. For example for Peer-to-Peer computing administrators can utilize this data to balance resource reservation. Middleware should provide this information if it is needed.
- *Naming, discovery and directory* - Precise and unique naming of users and resources is very important when considering scalability of Peer-to-Peer network. Importance of unique naming emphasizes when number of peers

increase. To be able to provide flexible discovery of resources, an external discovery service could be used. To prevent making same discoveries in vain, a directory service can buffer and store the results of a discovery. This directory could act as a repository of information about peers in the network and their resources.

- *Sharable resources* - A Peer-to-Peer network can provide a great variety of resources. Management of these resources can be provided as a centralized service which provides access to a service provided by a certain peer of Peer-to-Peer network. Management tasks vary greatly depending on the type of the shared resource. For example, when concerning a processor time as a resource, management service could provide scheduling between peers which are using the service simultaneously.
- *Location independent services* - The service should look the same regardless of the platform where it is used. For example, the service should provide same functionality on PDA –devices and desktop computers. Interaction between user and service shouldn't differ depending on location of service.

3 WIRELESS TECHNOLOGIES

Using wireless technologies for communication is an essential part of creating a flexible user experience. Different new wireless technologies are becoming available on many different platforms but only Infrared /7/, Bluetooth /8/ and IEEE 802.11 /9/ standards have succeeded to become general in the area of short-range radio technologies. Infrared however, is not suitable for our case because infrared devices must maintain the line of sight to communicate. If broader coverage area is needed, then GPRS provides another solution for communication. While all these technologies can provide Transmission Control Protocol / Internet Protocol (TCP/IP) connection to the user, they represent completely different approaches. Bluetooth creates a Personal Area Network (PAN) which covers only area of few meters. WLAN creates a Local Area Network (LAN) which can be easily extended to cover area of ten meters to few kilometers, depending on the number and location of WLAN access points. GPRS –service, operating on GSM network provides a Wide Area Network (WAN) which can have a coverage area of hundreds of kilometers, depending on a number and location of GPRS access points. Relative coverage of different network types is illustrated in Figure 7.

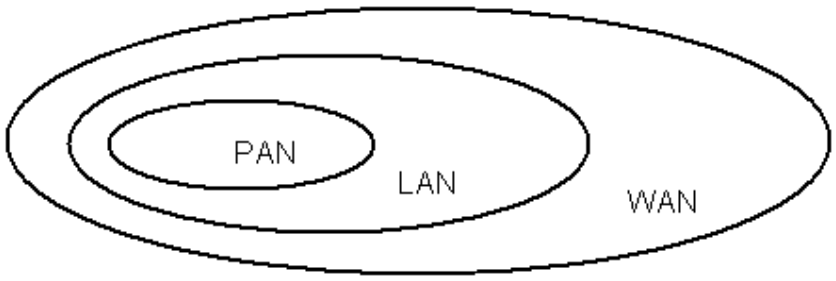

Figure 7. Relative network coverage

Wireless devices operate in a certain frequency band using certain bandwidth. Bandwidth is one factor among others which determines the data transfer speed. To prevent using overlapping frequencies, the use of a radio spectrum is controlled by authorities. However, Industrial, Scientific, and Medical (ISM) radio frequency-band is granted for free use for different purposes. For example, microwave oven, Bluetooth and

WLAN devices operate on the same 2.4GHz ISM band. Available radio frequency-bands are strictly controlled and the upper limit of speed in a wireless network is limited to available bandwidth. Because wireless connections suffer dramatically from interferences caused by environment, every transferred packet has to be validated carefully. This process is one reason which makes the wireless network slower comparing to traditional fixed network. Security is also a concern because on a wireless network, everyone within range of the network can hear the signal and sniff the traffic. In addition, wireless network is very hard to limit into a particular area which makes it easier to hackers infiltrate the network.

The reason why these specific three wireless technologies were selected for this work can be justified when considering the possibility of using three completely different technologies together: Every technology is designed to cover only its dedicated environment, there is no such technology which suits for every environment. Knowing the limitations and benefits of these very different technologies, it is possible to create an environment where a user can always get the best possible connection depending on his needs and priorities.

3.1 Bluetooth

Bluetooth is a relatively new wireless technology which offers short range ad-hoc connectivity for several different electrical devices. Originally Bluetooth was designed only to be technology for replacing wires.

3.1.1 Introduction to Bluetooth

Bluetooth offers a cheap and widely adaptable technology, which can be applied in devices varying from small probes to desktop computers. Traditionally, mostly of mobile devices are used in a way that user always initiates interaction. For example, a user has to manually update address books on cellular phone, PDA and on laptop. Bluetooth will enable wireless communication between all these devices providing communication with no user intervention, or at least interaction requires less attention

from user. Using Bluetooth, devices can for example automatically synchronize information between each other. It is estimated that a Bluetooth chip will eventually cost about 5$ /10/ which will lead it to mainstream applications. However, Bluetooth has its limitations, which must be taken into account when designing Bluetooth –enabled hardware or software. Because Bluetooth wireless connections are established using radio links, interference can impact significantly on a product's operation. The wireless link can weaken so that communication becomes slow or then the link can break completely. Even if the link quality is sufficient, Bluetooth can't offer very wide bandwidth comparing to wired solutions. Slow connection time is also a problem, which will effectively limit targeted products – Bluetooth can't be applied on most critical systems. However, Bluetooth enables different power saving modes, which will make it suitable for low-power mobile devices.

3.1.2 Bluetooth Network Topologies

When several Bluetooth devices are near each other a Bluetooth PAN is formed. This PAN defines a network where Bluetooth devices can communicate with each other with no need of user intervention. Network has a master device which routes all the data in the network and other devices are slaves to this master. Communication can be point-to-point or point-to-multipoint, depending on a size of network: if there is more than one slave, a piconet is formed and all the communication has to be made through master device. Master can be connected up to seven slaves at the same time. Typical Bluetooth piconet is illustrated in Figure 8.

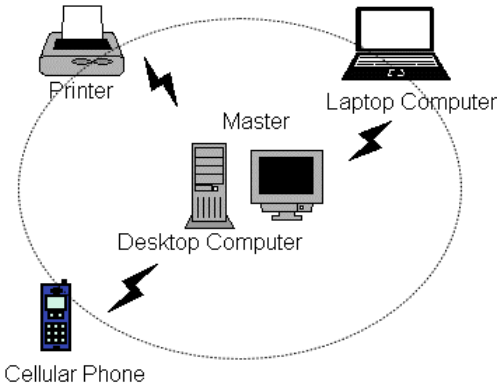

Figure 8. Piconet

When several piconets are located in the same area, a Bluetooth Scatternet can be formed. Scatternet defines a network where one of the slaves of piconet is also a master of second piconet. By forming scatternets the Bluetooth network can be expanded because all the peers of a scatternet can communicate with each other. Typical Bluetooth scatternet is illustrated in Figure 9.

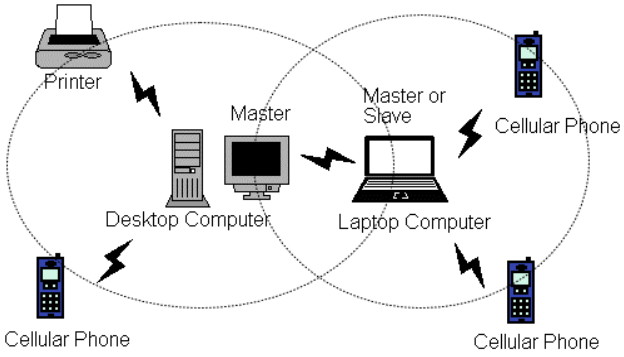

Figure 9. Scatternet

While the Bluetooth specification version 1.1 defines the scatternet, current devices do not support this feature or have problems using it. Most of these problems are related to Bluetooth role-switching feature. Role-switching makes it possible for devices to change roles between master and slave. This feature is needed when scatternet is formed. In fact, even creating a piconet of more than two devices is problematic or impossible for some devices.

3.1.3 Bluetooth Connection Times

Bluetooth connection times are formed of inquiring and connection sequences. If the target device is not known, inquiry has to be performed which takes about 10 seconds. Before any communication can take a place, Bluetooth transmitter and receiver have to be synchronized. This synchronization is implemented using frequency hopping. Frequency hopping is a sequence where devices change rapidly used frequency i.e. ‘devices hop from one frequency to other’. When synchronization takes a place, devices use internal timers to synchronize the hopping frequency with each other. This

connection phase can take up to 10 seconds /10/. When considering these delays, the lengthy connection times of Bluetooth create limitations for applying Bluetooth on different purposes.

When Bluetooth connection is needed, devices must be in *page and page scan* modes. Device initiating the connection starts paging mode and waits for response from page scanning device. Before paging can be started, paging device must know the page scanning device's ID. This ID can be calculated from the page scanning device's 48-bit Bluetooth device address. This device address can be obtained from an inquiry scan, user input or then it can be pre-programmed at manufacturer. If devices are "known devices" with each other (i.e. device addresses are known), inquiry can be avoided and paging can be started immediately.

Situation when device is trying to connect a device which is running constant inquiries is problematic. Because other device is not in page scan mode, connection can't be established. This might be the situation when several devices are trying to find each other and make connections.

3.1.4 Device Discovery

When Bluetooth device discovery is initiated, scanning device must be in *inquiry* mode and devices, which are willing to be discovered in *inquiry scan* mode. Inquiring device transmits rapidly a series of short packets using different frequencies. This switching between frequencies is done 3200 times per second which is the twice of a rate comparing to normal connection. On the contrary, inquiry scanning device changes frequency very slowly, once every 1.28 seconds /10/. Using this kind of searching protocol, inquiring device has very good changes to meet other devices on the same channel. Also, using frequency hopping as fast as this, inquiring device can cover wide area of frequencies and thus, find as many devices as possible. When a device, which is running inquiry scan, receives an inquiry data packet, it waits for a short random time and then if it receives a second inquiry packet, it transmits a response. This response is not sent immediately but using also random delay. These random delays prevent a

situation when all the devices send response simultaneously causing a pulse of radiation in the ISM radio band.

Bluetooth device discovery can be very problematic in environment where many devices are constantly searching other devices using inquiry. When device enters into inquiry state, it cannot be seen by other devices. If devices are running inquiries on same constant interval, devices are invisible to each other. Other problem is the length of inquiry. If inquiry length is very short and there are several surrounding devices, inquiring device can't find all the devices. If the inquiry length is too long, other devices have problems of finding the inquiring device because it is still running its own inquiry. To prevent these situations, devices running constant inquiries should randomise the delay of inquiry interval. The inquiry length is problematic because the number of surrounding devices varies greatly depending on environment. Therefore the inquiry length must be always suited to meet the environment where device is currently operating.

3.1.5 Service Discovery

Nature of ad-hoc networks are strongly based on the idea of *zero-configuration* networks where network configures and administers itself. Because these networks don't contain any centralized infrastructure and devices can move in and out simultaneously, capability of finding and using services is essential part of Bluetooth protocol.

Bluetooth Service Discovery Protocol (SDP) provides a method for directly discovering remote entities, i.e. services from other Bluetooth device. A service can provide information, execute an action or access a resource. SDP consists of Service Discovery data structures, the Service Discovery Protocol, and the Service Discovery Application Profile (SDAP). SDAP is responsible for determining how SDP is used by applications. For a device to be able to announce services, it must provide an SDP server. This server contains information about provided services and how it can be accessed using service attributes. When Bluetooth device discovery founds a new device, it can ask about service info using SDP. Device can ask for a particular service record or it can browse

available service records.

3.1.6 TCP/IP and Bluetooth

Because Bluetooth devices vary widely from headsets to printers, the protocol stack has many software layers. Host Control Interface (HCI) divides the stack to lower and middle layers. Lower layers are mainly similar in providing over-air transmissions. Middle layers hide the complexity of lower layers and provides multiplexing from many different data formats and protocols into packets, which can be then transmitted over air. Upper layers provide specific protocol implementations like TCP/IP and SDP. While it is possible to use TCP/IP over Bluetooth, this work used socket interface directly over Logical Link Control and Adaptation Protocol (L2CAP) –layer. Because TCP/IP is quite heavy protocol to use over Bluetooth, L2CAP –sockets provided lighter solution to create socket connections between Bluetooth devices. Simplified Bluetooth protocol stack is illustrated in Figure 10.

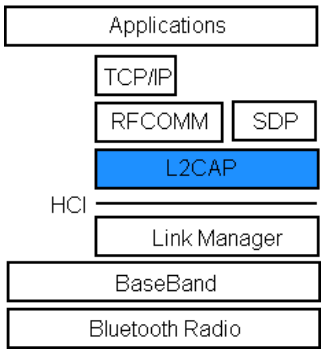

Figure 10. Simplified Bluetooth protocol stack

3.2 Wireless LAN

Wireless LAN provides a technology to build new and extend existing TCP/IP –networks using wireless medium as transport layer. While Bluetooth was mainly targeted to be a low-range cable replacement technology, WLAN can be considered as local area networking technology. WLAN doesn't provide any device or service discovery methods.

3.2.1 Possibilities of WLAN

WLAN offers many new possibilities comparing to fixed networks. Mobility allows using same services as in fixed network freely without being chained into particular fixed location. Mobility provides wireless freedom where users can move and still access to centralized data in motion. For example, a user can have wireless Internet access using PDA or laptop while sitting in a park. Wireless network can be deployed easier, faster and it can more be suitable for areas which are difficult to wire for traditional LANs for example older buildings and historic preservations.

WLAN is flexible. Flexibility means that wireless networks allow users to quickly form small group networks for meeting without pulling or connecting cables. This can be very useful for example on conference situations. Flexibility offers hot-spots for hotels, airports, train stations, libraries and cafes. These hot-spots can provide services like Internet connection, ticket automaton and so on. In some cases, costs can be reduced by using wireless technology for example forming a bridge between two buildings without digging or throwing cables.

3.2.2 WLAN Architecture

WLAN networks can be infrastructure or ad-hoc networks. Both network types can be built using a basic service set (BSS) which is the basic building block of the WLAN network architecture. BSS has a basic service area which is defined by the propagation of wireless medium and when a WLAN device is inside basic service area, it can communicate with the other devices on the area.

Ad-hoc networks can be as simple as two WLAN devices communicating straight with each other. Ad-hoc WLAN networks are typically established for short time purposes for some particular use, for example in conferences or meetings. Basic service area is amorphous because nodes can move in and out, expanding and shrinking network area /11/. Ad-hoc network is illustrated in Figure 11 which presents three mobile devices creating the WLAN ad-hoc network.

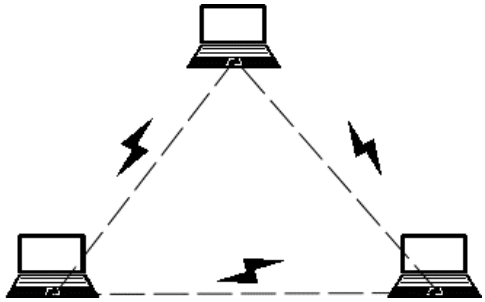

Figure 11. Ad-hoc network

In infrastructure architecture, networks are formed using access points. Differing from ad-hoc connections, the data frame is first sent to an access point which then forwards it to the receiver station. Because the station must be inside of an access point coverage area in order to send or receive frames, basic service area can be defined more accurately than in ad-hoc network. Although indirect communication between stations requires more transmission capacity than direct ad-hoc connections, infrastructure network has two major advantages. If a station is in access point coverage area, it can communicate with every other station regardless of distance or location of these stations (assuming that stations are inside BSS). When station is attempting to save power, access point notices this and starts buffering data for this station. Thus, battery powered devices can power down transceiver and only receive buffered data when it is needed. Before station can communicate within BSS, it must associate with an access point. Stations initiate association process and access point can grant or deny station to join the network. Currently, a station can be associated only to one access point at the time /11/. Infrastructure network is illustrated in Figure 12 which presents three mobile devices using WLAN network through WLAN access point.

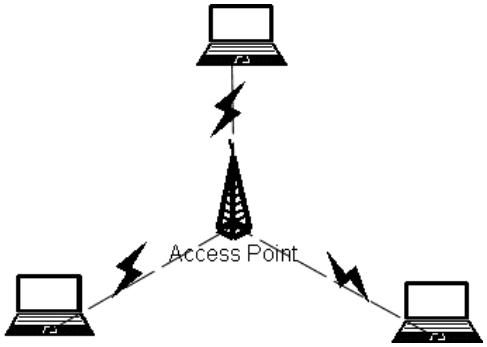

Figure 12. Infrastructure network

3.2.3 Network Operations

To be able to use a wireless network it has to be first found and identified. This process is called *scanning*. Scanning can be performed by using *active or passive scanning* modes. Passive scanning can be done without transmitting anything: station moves to each channel on the channel list and waits for beacon frames. Beacon frames information about access point and station can use this information to join provided wireless network. Because passive scanning, a station doesn't need to send anything, scanning is not very battery consuming. Active scanning requires more interaction: on each channel a station sends probe request frame to solicit responses from a network with a requested name. This is the difference between passive and active scanning: active scanning tries to find specific network but passive scanning just listens available ones.

When a network has been found, the station needs to associate with an access point before data transmit can occur. After this, the station can authenticate with access point. While on fixed networks authentication is provided by physical access, wireless networks must implement authentication methods to gain access control. The 802.11 - standard /9/ specifies two major authentication models: open-system authentication and shared-key authentication. Open-system authentication provides possibility for everyone to associate with access point. Shared-key authentication uses Wired Equivalent Privacy (WEP) to provide access control and network security. When authentication succeeds, the station sends association request and waits for granting of request. If request is granted, access point starts to process frames from mobile station and normal data transfer can be started.

3.2.4 TCP/IP and WLAN

Physical layer takes care of data transfer over air. Carrier Sense Multiple Access / Collision Avoidance (CSMA / CA) is used to avoid packet collisions. CSMA/CA listens for the transfer medium if there are other devices sending data. If the medium is reserved, transmission is delayed. After this delay, CSMA/CA checks the medium again

and sends the data if there are no other devices sending. The IEEE 802.3 standard /12/ Logical Link Control (LLC) protocol provides data encapsulation from lower Medium Access Control (MAC) layers. This work utilized TCP/IP over WLAN using standard socket API. Simplified structure of the WLAN protocol stack is illustrated in Figure 13 which presents the protocol layers.

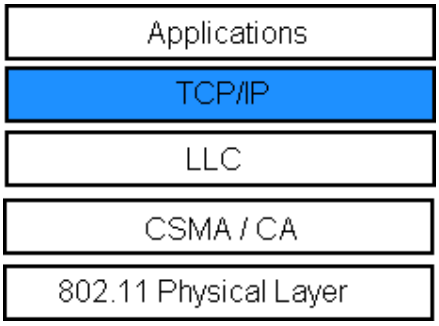

Figure 13. Simplified WLAN protocol stack

3.3 General Packet Radio System

The General Packet Radio System (GPRS) defined by the European Telecommunication Standards Institute (ETSI) is a service which provides packet based data transfer on radio network for Global System for Mobile Communications (GSM) and Time-Division Multiple Access (TDMA) users. GPRS doesn't provide any device or service discovery methods.

3.3.1 Introduction to GPRS

GPRS uses the GSM network and tries to provide high-quality packed based services using as much as possible the already constructed GSM networks and protocols. GPRS makes it possible to use several protocols but mainly Internet Protocol (IP) is used. Benefits of packet based data transfer are clear: GPRS saves network resources because it allocates bandwidth only when data is actually transferred. GPRS requires additional support from GSM devices and also new components and updates to GSM network. In addition, because of increased transfer capacity, GPRS provides also new possibilities

for building services to mobile and handheld devices. GPRS plays an important role on evolution of GSM to 3G.

One GPRS subscriber can have several Packet Data Protocol (PDP) – connections. New PDP -connection can be activated to establish IP-connection and to obtain an IP-address. GSM network takes care of transformation between IP and GSM networking. From outside, this connection looks like a regular subnet and thus can't be seen as mobile device.

GPRS defines radio channels which can be allocated flexibly. From one to eight radio-interface timeslots can be allocated per TDM frame. Active users share timeslots and up and downlinks are allocated separately. Resources between speech and data can be shared dynamically /13/.

3.3.2 GPRS Connections

GPRS provides Point-to-Point (PTP) or Point-to-Multipoint (PTM) connection methods which can be classified into smaller subsections by connection characteristics /13/. Characteristics are:

- Point-to-Point defines a connection method which can be used for sending frames between two endpoints of network.
    - PTP Connectionless Network Service (PTP-CLNS) defines connectionless Point-to-Point service which transfers one or more packets separately from sender to receiver.
    - PTP Connection Oriented Network Service (PTP-CONS) defines connection oriented Point-to-Point service which transfers multiple packets from sender to receiver through logical connection.
- Point-to-Multipoint defines a connection method which can be used for sending frames for a specified group.
    - PTP Multicast defines a service which sends message to sender specified group. Recipients can belong to a specific group or then a message can be sent to all the devices on area.

- PTP Group Call defines service which sends messages to all the devices which belong to user specified group on a specific area
- IP Multicast is part of TCP/IP –protocol family and provides method for sending messages to a specific group defined by IP addressing.

3.3.3 GPRS network components

To enable GPRS on existing GSM networks two new components have been introduced: System GPRS Support Node (SGSN) and Gateway GPRS Support Node (GGSN). SGSN provides authentication, mobility management and routing. SGSN makes queries to the Home Location Register (HLR) to obtain GPRS user profile data, detects new mobile devices in a service area and keeps track of movement on network area. GGSN acts as gateway node between GPRS network and external networks. GGSN terminates incoming GPRS Tunnelling Protocol (GTP) tunnels and encapsulates user packets over Internet Protocol / User Datagram Protocol (IP/UDP) transport paths providing also set of control messages to set up and modify tunnels /13/. Simplified GPRS –architecture is illustrated in Figure 14 where mobile station connects to BSS and is ready to use GPRS –service.

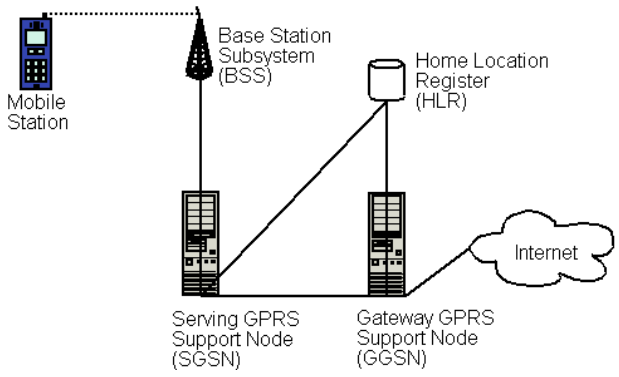

Figure 14. Simplified GPRS architecture

3.3.4 TCP/IP and GPRS

GPRS protocol stack is very complicated /14/. Because Virtual PeerHoods scope was mainly to use the TCP/IP socket interface over GPRS –service, following example of IP-message transfer over GPRS is extremely simplified.

When mobile station sends an IP-message to network, it is first transferred to SubNetwork Dependent Convergence Protocol (SNDCP) layer. SNDCP transfers segment to LLC layer, which encapsulates the segment into a frame. MAC layer chops the LLC-message into smaller parts which are then transferred over GSM network into Base Station Subsystem (BSS). When BSS receives the message, it relays message to SGSN. SGSN has many tasks, for example keeping track of the individual mobile stations location and performing security functions and access control. SGSN decapsulates the received LLC message and then encapsulates it using TCP/IP. SGSN then forwards the message to GGSN which is connected to packet based network. Finally, GGSN sends IP-message to its destination. Message flow from Mobile Station to GGSN is illustrated in Figure 15 which presents the simplified version of GPRS protocol stack.

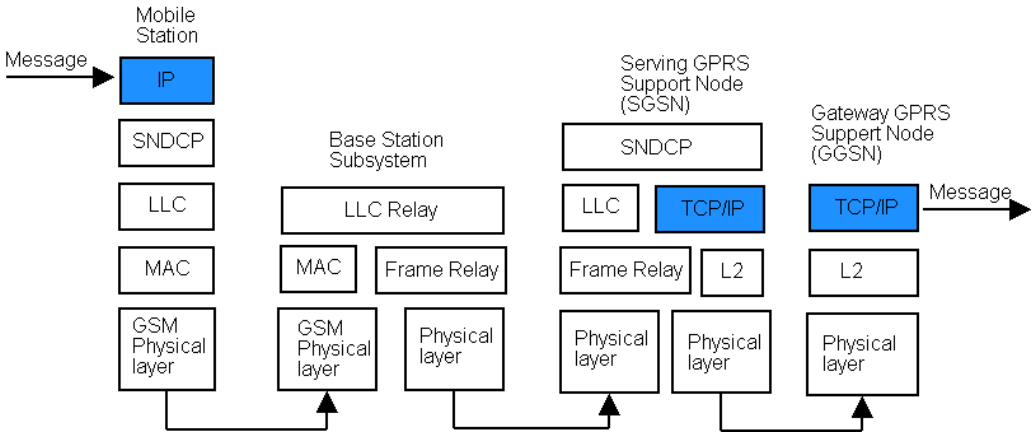

Figure 15. Simplified GPRS protocol stack

4 WIRELESS MOBILITY

While mobile devices and wireless technologies have evolved, our perceptions of connectivity have changed dramatically. Mobile phones for example have provided completely new possibilities for our ways of working and now the same evolution continues on area of mobile computing. PDA –devices, laptops and smartphones equipped with different wireless technologies are becoming more common providing possibilities for to carry literally Internet in our pocket. Currently, most popular wireless technologies used by these devices are Bluetooth, Wireless LAN and GPRS. When considering environment with several available wireless technologies and where user can move from one network to other, seamless change from one network to other creates new possibilities to provide and use different wireless services. Using these technologies together, user can eventually have Seamless Connectivity which provides always the best available connection. Seamless Connectivity is discussed further in this chapter.

The process where device changes from one networking technology to other is called handover. Handover can be also performed without changing networking technology. For example, WLAN –device can make handover from one WLAN access point to another. However, on this thesis, handover means always a handover between different networking technologies. Handover is usually performed on situations when device is moving away from current network area and connection is weakening. This also results as a decreasing link quality level. The link quality level is a relative measurement unit which describes the quality of current wireless connection. Handover is discussed further in Chapter 5. Typical situation when handover is initiated is illustrated in Figure 16.

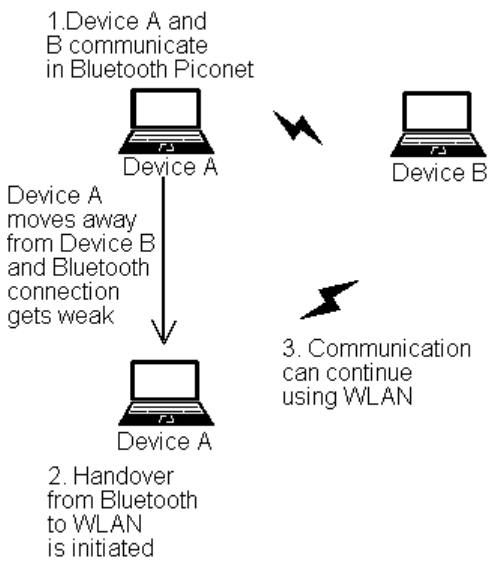

Figure 16. Handover from Bluetooth to WLAN

In example situation (Figure 16) devices A and B communicate in Bluetooth piconet. The Device A starts to move away from The Device B and established Bluetooth connection gets weak. When the Bluetooth connection is about to break, the Device A performs handover from Bluetooth to WLAN. A new Connection to the Device B is then established using WLAN and communication between devices can continue.

4.1 Challenges and possibilities

Mobility provides flexibility to use services on very different environments while moving. While the same service can be reached using several different wireless technologies user can select the most suitable technology for different situations. For example Bluetooth provides low power consumption /15/ while GPRS and WLAN networks cover wider area and provide flexible moving between access points. While the advantages of Seamless Connectivity are very clear, many challenges have to be considered when implementing a system capable of Seamless Connectivity. In addition, wireless data transfer itself causes problems concerning different types of interferences.

As said, providing Seamless Connectivity between different wireless technologies can be a very demanding task. To prevent a link break, one must always know the quality of

established link. If the link quality seems to get too low, a handover should be initiated immediately before the connection breaks. The system should also be continuously aware of its environment and available alternative wireless technologies. If the link breaks or device changes wireless technology, connection is broken temporarily. On this situation, a device can't transmit or receive anything. To prevent losing information during this period, communicating peers need to be informed that connection is temporarily broken. When devices get this information, transmission is on hold until another announcement of link validity has been received. Besides this, data must be buffered all the time during the transmission. If connection breaks up suddenly, device can still process the buffered data and then inform that link is broken.

To be able to provide a transparent handover between different wireless technologies, at least following requirements must be fulfilled:

- **Measure constantly wireless link quality**
  When connection has been established, used link must be monitored constantly to obtain information about link quality. If link quality seems to be weakening, system must be able to react quickly. Worst case would be the situation when link gets broken.
- **Search and identify other available technologies**
  The system must constantly perform inquiries of other available technologies. It is very important to have this information continuously available because when handover needs to be performed, there is no time for inquiries if the user needs seamless handover.
- **Buffer transmitted data**
  The system should buffer data before transferring it to network and vice versa before transferring it back to upper levels of protocol layers. Even if the handover can be performed successfully, there is always a period of time when data can't be transmitted and buffering is needed to not to lose any transmitted data.
- **Provide signaling between sender and receiver to inform handover**
  If buffering is not used or available buffering resources are light, signaling can

be used to prevent data loss on handover and link break situations. When the link quality seems to go low, system could inform upper levels of protocol not to send any data and also signal the other peer to interrupt the transmission. When handover has been completed or link quality is good, system signals that data transmission can continue again.

- **Provide transparent handover for upper levels of protocol**
  While the actual physical connection can change, upper protocol layers should not directly know about this. The system should provide a virtual connection to application layers. Virtual connection is always valid and can be used for sending and receiving data. This virtual connection is then mapped to actual physical connections. If this virtual connection is not provided, application layers should always be aware of situations when connection has been changed and previously used connection is not valid anymore.

4.2 Current state of implementations

Currently, there are not so many systems which provide seamless mobility between different wireless networks. However, Mobile IP is one of these, perhaps the most known to public. This implementation is described briefly on next section.

Mobile IP

Mobile IP provides methods for mobile devices to move seamlessly in wireless networks. In traditional IP –network, a device is always given a new IP –address when it changes a network and thus it doesn't allow transparent moving from one network to another while still using the same IP –address. Mobile IP /16/ solves this problem using two IP –addresses: a fixed *home address* and a *care-of address* which changes at each new point of attachment. The home address is defined by a home network where also a *home agent* operates. Because this home address is static, it seems like mobile device is always ready to receive data. The care-of-address can be obtained from a *foreign agent* which operates on a foreign network. A mobile node can locate foreign agents by agent solicitation messages which are sent constantly. When a mobile device moves to a new network, it registers itself with a new care-of-address using a local foreign agent and

notifies a home agent of this new registration.

If the mobile device is not attached to home network all the data which is sent to a home address is forwarded to a mobile device by a home agent. When a mobile device sends data, packets are first sent to a home agent which then forwards packets to correct destination. To forward packets, a home agent needs to modify packets header information.

Mobile IP has three tasks to care of: discovering the care-of address, registering the care-of address and tunneling packets to the care-of address. Home and Foreign agents send agent advertisement messages constantly on specific interval. A mobile node can also send solicitation for example if it needs a care-of-address. This solicitation message is broadcasted to the network and any home or foreign agent which receives it, sends a reply back. An example use-case is illustrated in Figure 17 which presents the sending of a message using Mobile IP.

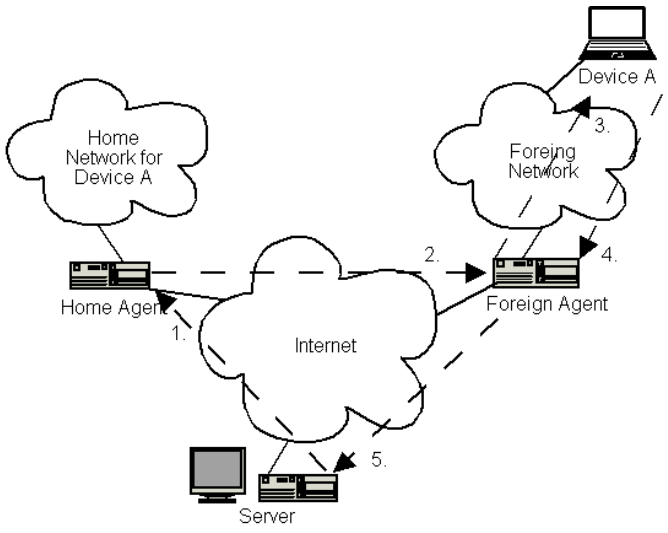

Figure 17. Mobile IP use-case

In Mobile IP use-case (Figure 17) server sends an IP –datagram destined to the Device A using the Device A's home address (1). At the home network, the home agent encapsulates the message using the Device A's care-of-address as a destination address in packet header. The home agent then retransmits the package using Internet (2). A foreign agent strips outer encapsulation and forwards the message to the Device A using foreign network (3). The Device A receives the message and sends a reply back to the Server using its fixed address (4). Finally, the foreign agent forwards the message to the Server using Internet (5).

5 IMPLEMENTATION OF SEAMLESS CONNECTIVITY

Implementation part of the work consisted of creation of plugins for WLAN and GPRS –technologies, the GPRS Gateway for routing GPRS –connections and upgrading PeerHood –middleware to utilize features defined by Virtual PeerHood concept. Development environment was Linux operating system.

Bluetooth in Linux

Currently there exist a few Bluetooth protocol implementations for the Linux. Most popular however is BlueZ /17/ which is currently also part of the Linux kernel 2.4 and 2.6 series. Because BlueZ is licensed under the GNU Public license (GPL) it allows good possibilities for developers to explore BlueZ source-code and develop Bluetooth -software. Other Bluetooth protocol stack implementations are Axis /18/ and Affix /19/. This work utilized 3COM 3CRWB6096 Bluetooth PCMCIA interfaces which supported Bluetooth 1.1 standard /8/.

WLAN in Linux

The Linux-wlan Project /20/ aim is to develop a complete, standards based, wireless LAN system using the GNU/Linux operating system based on the IEEE 802.11 standard /9/. Linux-wlan makes it possible to use several manufacturers' 802.11b wireless network interfaces on Linux. There exist also other open source drivers. For example, for Orinoco cards there are driver implementations like Orinoco_cs, Wavelan and Wavelan2 /21/. This work utilized D-Link DWL-650 WLAN PCMCIA interfaces which were IEEE 802.11b standard compatible.

GPRS in Linux

GPRS interface was Nokia D211 PCMCIA card which was used through Nokia's GPRS Linux driver /22/. Nokia doesn't provide full sources for this driver, but on the scope of this work, it was enough just to use GPRS -technology as a standard network interface providing TCP/IP connections through standard socket API.

5.1 Definition of PeerHood

PeerHood concept /23/ defines a system which can be utilized by several different kinds of mobile devices. Common naming for all these mobile devices is Personal Trusted Device (PTD). In future, PTD will provide many purposes of use in our every-day life. For example, PTD can be used to buy tickets, to chat and send e-mails and so on. Common thing for all these services is the fact that services can be provided using Peer-to-Peer networking. PeerHood provides a middleware which can be used to form a Peer-to-Peer network between PTDs.

Because the wireless network is never static but constantly changing, network devices should be always aware of network environment. This includes the finding of devices and services. If devices want to provide own services, it should be also possible to advertise own services to other. Finally, devices should be able to make connections between each other and in case of interest, monitor the device which is currently used. PeerHood provides all these functionalities as a middleware platform providing an API for software developers.

5.2 Definition of Virtual PeerHood

Virtual PeerHood implements a middleware for providing, locating and using services on different types of wireless networks. Virtual PeerHood extends the concept of PeerHood which was based on the idea of creating peer-to-peer connections between mobile devices. While PeerHood utilized Bluetooth as a communication technology, Virtual PeerHood provides also the use of WLAN and GPRS to establish connections between PeerHood devices. Virtual PeerHood implements also the Seamless Connectivity between these technologies. While Bluetooth and WLAN connections can be made directly peer-to-peer, GPRS connections must be made through the Virtual PeerHood GRPS Gateway. This arrangement is made because GPRS users can seldom have a public IP –address and thus Peer-to-Peer connections are not possible. The Virtual PeerHood GPRS Gateway routes all connections between GPRS –users and provide also device database which includes information about connected GPRS –users. As Figure 18 illustrates, Virtual PeerHood provides methods for communication

between different mobile devices using Bluetooth, WLAN and GPRS.

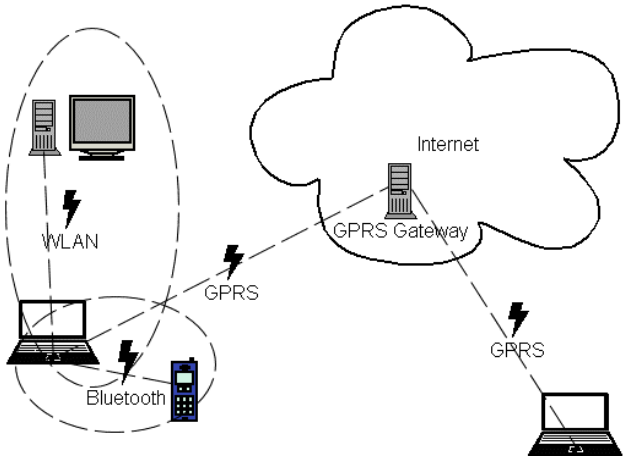

Figure 18. Virtual PeerHood architecture

5.2.1 System components

While the previous Figure 18 illustrated the physical architecture of the Virtual PeerHood, this chapter presents software components used in Virtual PeerHood implementation. Virtual PeerHood consists of a daemon and a library. The daemon is one independent process which takes care of locating other devices. The daemon implements device discovery using network specific plugins though the plugin-interface. The library interface provides all the Virtual PeerHood functionality to the application. The applications can use Virtual PeerHood –services through the library –interface. Communication between the library and the daemon is established using local sockets through socket –interface. System software components are illustrated in Figure 19.

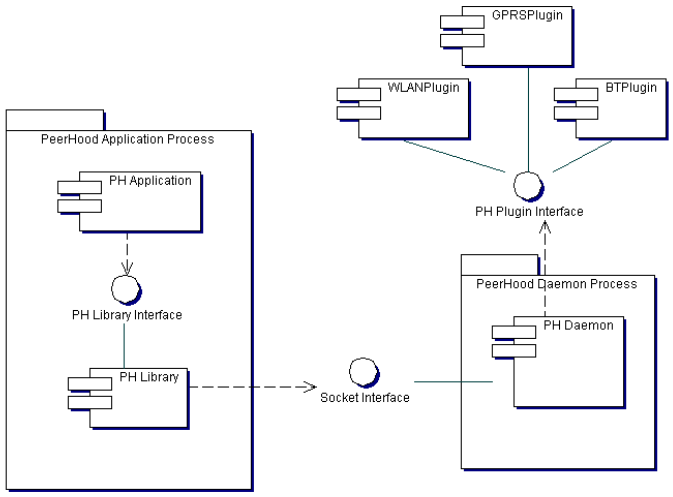

Figure 19. Components of Virtual PeerHood

5.2.2 Component structure

Because the daemon and the library are independent components, Virtual PeerHood class structure is also divided into two parts. Important part of whole Virtual PeerHood structure is the abstraction between networking specific classes and interfaces. Virtual PeerHood allows developers to add new plugins which can be used by the daemon. To be able to provide this architecture, the whole technology-specific functionality must be contained in one class which implements specific interface. This allows the core components to remain the same while adding new functionality using plugins. Because Virtual PeerHood extends PeerHood concept, classes are mainly the same. However, to notify differences from following class diagrams, new Virtual PeerHood classes are highlighted using gray color. Daemon class diagram is illustrated in Figure 20.

CDaemon is the main class which uses different plugins and connection –objects. CDaemon uses the network specific plugins through MAbstractPlugin interface to search other mobile devices and services and to advertise own device and provided services. CDeviceStorage acts as a repository for found devices and services. CDeviceStorage follows a Singleton design pattern which makes sure that there is always only one instance of it present. Daemon uses network specific connection objects through MAbstractConnection interface. Connection objects are needed by plugins to

exchange service and device information between devices.

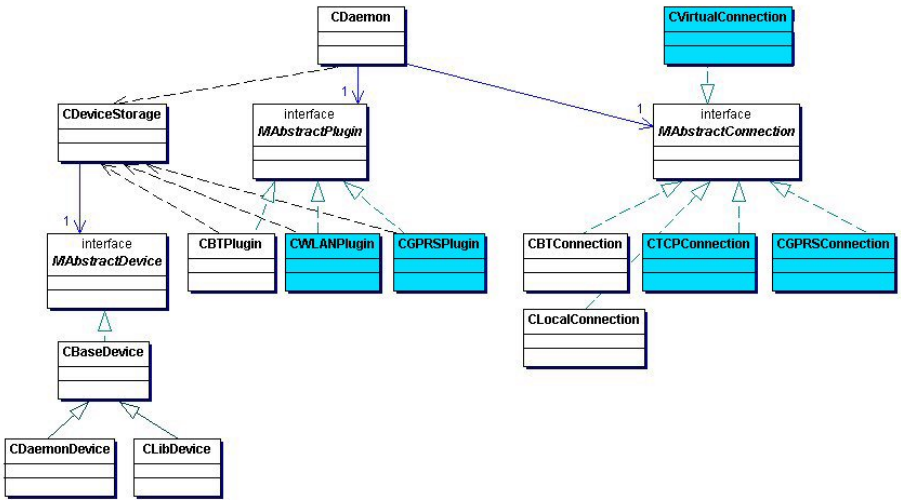

Figure 20. Daemon class diagram

The library, which is illustrated in Figure 21, provides an interface to applications through MPeerHood –interface. MPeerHood doesn't include actual implementation but it is implemented in CPeerHoodImpl –class. Abstraction is used to hide network specific connection objects from core components using MAbstractConnection and MAbstractCreator –interfaces. The library creates instances of connection objects using creator classes through Factory class. This design follows Abstract Factory design pattern. CEngine takes care of handling existing connections and passes new connections to applications using CBasicCallback. CLogger provides logging interface for all the classes. All the previous classes are part of the original PeerHood. Classes MAbstractMonitor, CVirtualConnection and CThreadInfo are new Virtual PeerHood classes which enables Virtual PeerHood to provide roaming. MAbstractMonitor provides an interface for technology specific signal level monitoring. CVirtualConnection provides virtual "handle" for the application. This virtual connection can create actual network specific connections hidden from application. The application sees all the time the same virtual connection but the actual connection which virtual connection points to, may be changed at any time. CThreadInfo is simply a container class for passing information between virtual connection and a thread which takes care of providing Seamless Connectivity between network technologies. This thread is called RoamingThread. If this Seamless Connectivity option is enabled, the

RoamingThread is created to monitor established connections, search services which are currently in use from alternative locations and to provide handover from one networking technology to other. The RoamingThread is described more carefully on section 5.7.

.

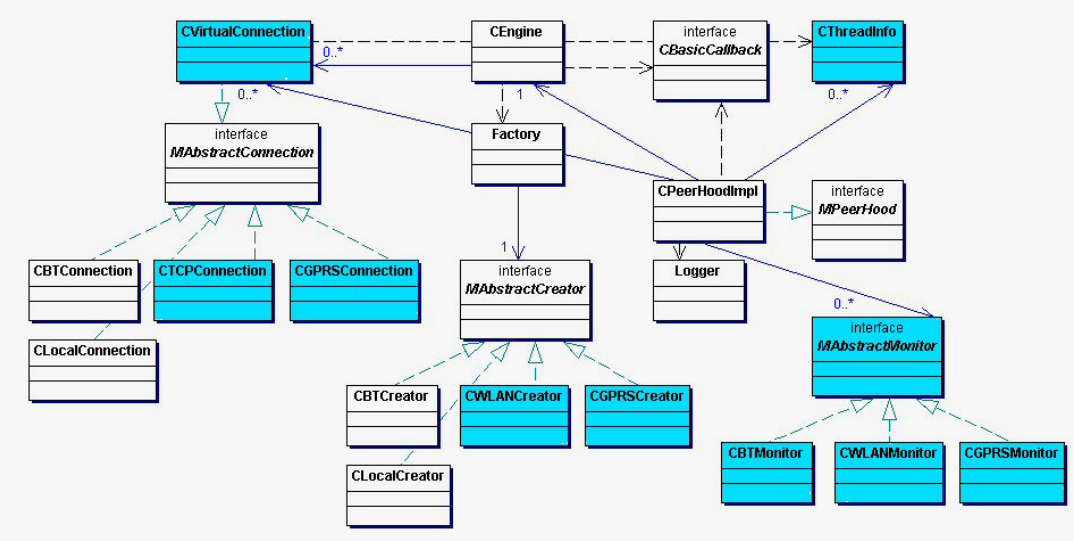

Figure 21. Library class diagram

5.2.3 Plugin connections

Plugins implement service and device inquiries using different type of connections. There are three types of connections between plugins: Signaling Connection, Data Signaling Connection and Data Connection. Signaling Connection is used for finding devices and services, Data Signaling Connection is used for negoating a new Data Connection and Data Connection is the connection which transfers the data coming from application layer to another device's application layer. Actual implementation of service and device discovery protocols are discussed further, this chapter describes only connections which are established between different plugins.

Bluetooth –plugin

Connections established by Bluetooth –plugin are illustrated in Figure 22. The Device A performs device inquiry using Bluetooth inquiry (1). When the Device A founds the Device B, it checks if the Device B is PeerHood –enabled using Bluetooth SDP (2). If SDP concludes that the Device B is PeerHood –enabled, the Device A can check

services from the Device B using Signaling Connection (3). When the device and service are found, application can send data to remote device using Data Connection (4). Connection types are described in Table 1.

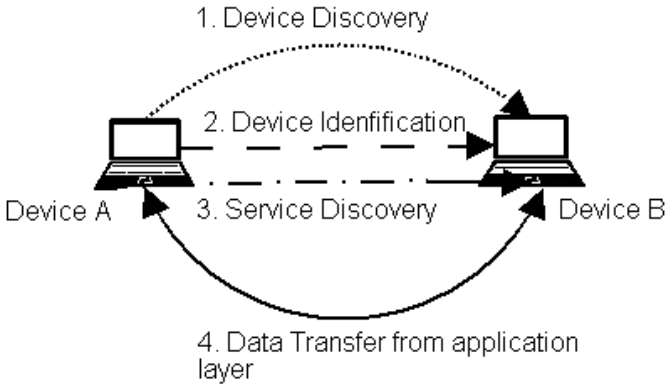

Figure 22. Bluetooth -plugin connections

Table 1. Description of WLAN -plugin connections

Connection	Type	Life span	Description
1	Bluetooth inquiry	Valid only during Bluetooth inquiry	This connection is used for device discovery.
2	Asynchronous ConnectionLess (ACL) Based SDP Connection	Valid only during SDP inquiry	This connection is used by SDP to identify PeerHood compatible devices.
3	L2CAP Socket Connection	Valid only during service discovery	Signaling Connection. This connection is used for service discovery.
4	L2CAP Socket Connection	Application decides the lifetime	Data Connection. This connection is used to transfer data from application layer to other device.

WLAN -plugin

Connections established by WLAN –plugin are illustrated in Figure 23. The Device A performs device discovery using connectionless Signaling Connection (1). If the Device A founds the Device B, it can perform service discovery establishing new Signaling Connection to Device B (2). When the device and service are found, application can send data to remote device using Data Connection (3). Connection types are described in Table 2.

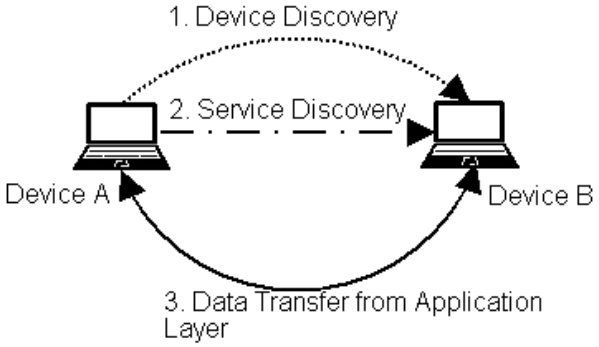

Figure 23. WLAN -plugin connections

Table 2. Description of WLAN -plugin connections

Connection	Type	Life span	Description
1	UDP Socket Connection	Connectionless. Valid during sending and receiving messages.	Signaling Connection. This connection is used for device discovery.
2	TCP Socket Connection	Valid only during service discovery	Signaling Connection. This connection is used for service discovery.
3	TCP Socket Connection	Application decides the lifetime	Data Connection. This connection is used to transfer data from application layer to other device.

GPRS –plugin

Connections established by GPRS –plugin are illustrated in Figure 24. The Device A performs device and service discoveries using previously established Signaling Connection (1). When the device and service are found, application can negotiate a new Data Connection using Data Signaling Connection (2). When the Data Connection has been established, application can send data to remote device through the GPRS Gateway (3). Connection types are described in Table 3.

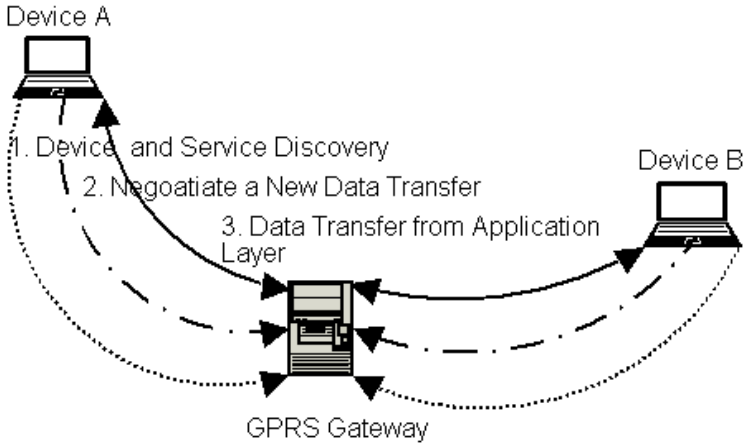

Figure 24. GPRS -plugin connections

Table 3. Description of GPRS -plugin connections

Connection	Type	Life span	Description
1	TCP Socket Connection	Valid during plugins lifetime.	Signalling connection. Used for to discover devices and services. Used also for checking connection validity.
2	TCP Socket Connection	Valid during plugins lifetime.	Data Signalling connection. Used for negotiating new data connections between devices.
3	TCP Socket Connection	Application decides the lifetime	Data Connection. This connection is used to transfer data from application layer to other device.

5.3 Device discovery

Device discovery must be transparent but an effective operation which provides information about surrounding PeerHood –devices. Device discovery should be periodically executed in a way that provides up-to-date information but shouldn't consume too much system resources or battery. While Bluetooth can utilize its service discovery protocol, WLAN or GPRS need to implement some kind of method for discovering devices using TCP/IP –layer.

Device Identification

To be able to distinguish devices from each other, the devices must contain some unique information. Using just network interfaces MAC –address, might be acceptable but if the device contains more than one interface, there must be also some additional information which connects interfaces to devices. Virtual PeerHood allows user to name devices freely. To prevent duplicate names, devices get also unique checksum number which can be used to distinguish devices if they have the same name. This checksum is currently the same as the Process ID (PID) of the daemon process.

5.3.1 Bluetooth –plugin

Bluetooth –plugin establishes four connections: Bluetooth inquiry connection, Bluetooth SDP connection, Signaling Connection and Data Connection. Bluetooth inquiry connection is used for finding devices, SDP connection is used for checking the devices PeerHood capability, Signaling Connection is used for service discoveries and Data Connection is established for transferring data from application layer to another device's application layer.

Bluetooth –plugin performs device discovery using Bluetooth inquiry. It first inquires all surrounding Bluetooth devices and then starts a SDP –query. Bluetooth –plugin performs SDP –query for every found device and tries to find the PeerHood tag. If this tag is found, the device is then marked as PeerHood capable. Bluetooth inquiry process is illustrated in Figure 25.

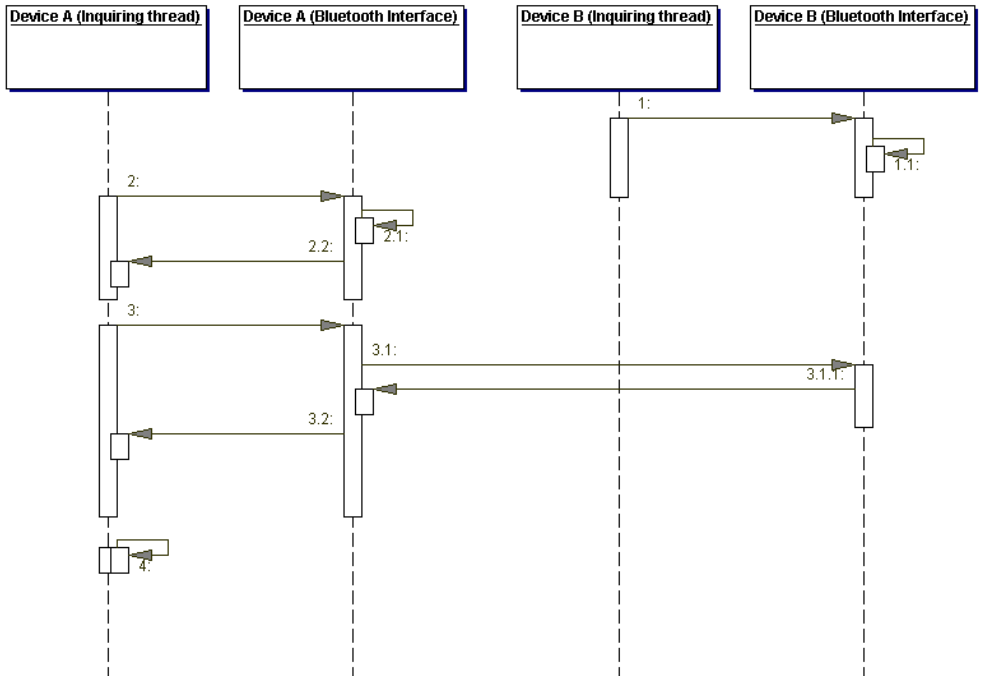

Figure 25. Bluetooth device discovery

In illustrated Bluetooth device discovery case (Figure 25) the Device B inserts PeerHood –tag into its SDP –database. The Device A starts Bluetooth inquiry and founds the Device B. The Device A checks if the Device B is PeerHood compatible using Bluetooth SDP. The Device A finds out that the Device B has PeerHood –tag in it's SDP so the Device B must be PeerHood –enabled device. The Device A adds the Device B to it's device database. Messages involved in this process are described briefly in Table 4.

Table 4. Description of Bluetooth device discovery

Message	Description
1	The Device B Inquiring thread utilizes Bluetooth Interface to insert PeerHood –tag into SDP -database
1.1	The Device B Bluetooth Interface inserts PeerHood –tag into SDP –database
2	Device A utilizes Bluetooth Interface to perform Bluetooth inquiry
2.1	Device A Bluetooth Interface performs Bluetooth Inquiry
2.2	Device A Bluetooth Interface returns a list of found Bluetooth devices (the Device B was found) to Inquiring thread

3	Device A utilizes Bluetooth Interface to perform SDP –query for PeerHood devices
3.1	Device A Bluetooth Interface performs SDP query for device B
3.1.1	The Device B Bluetooth Interface responds to SDP –query by sending its SDP service record information
3.2	Device A Bluetooth Interface returns information about the Device B SDP service record
4	Device A Inquiring thread parses service record and finds the PeerHood –tag. The Device B is then added to Device A device database.

5.3.2 WLAN –plugin

WLAN –plugin establishes three different connections: two Signaling Connections and one Data Connection. Signaling Connections are used for service and device discoveries and Data Connection is established for transferring data from application layer to another device's application layer.

WLAN –plugin performs device discovery using similar approach as Dynamic Host Configuration Protocol (DHCP) for service discovery /24/. First, device broadcasts query to every device located on the same subnet. When other PeerHood –device receives this query, it sends a reply back to the original sender. This Signaling Connection is implemented using UDP –packets.

UDP is suitable for this purpose because it provides possibility to send data via connectionless connections. In addition, connectionless connections are more suitable for amorphous wireless environment where devices can come and go. Thus, it would be waste of resources to always create a TCP connection, send one short message and then close the connection. WLAN inquiry process is illustrated in Figure 26.

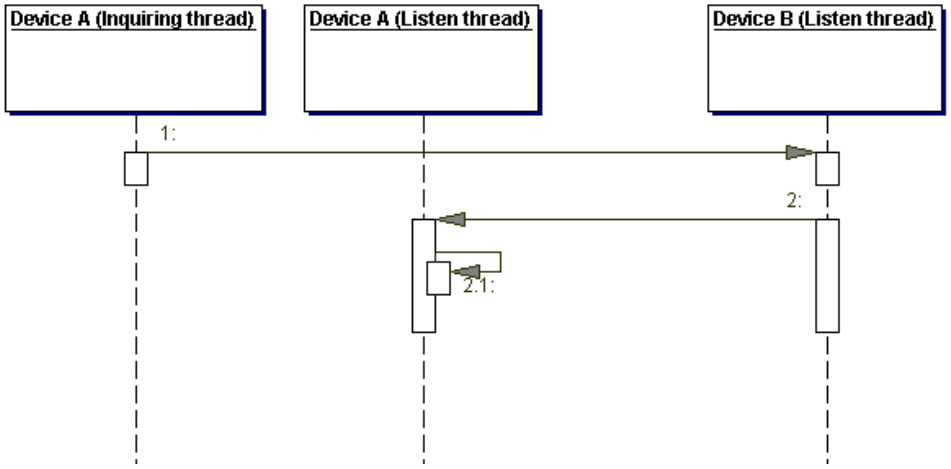

Figure 26. WLAN device discovery

In illustrated WLAN device discovery case (Figure 26) the Device A broadcasts UDP messages using well-known port. The Device B listens this port and sends a unicast UDP -message back to the Device A. The Device A receives the message and decides that the Device B must be PeerHood –enabled device. The Device A adds the Device B to it's device database. Messages involved in this process are described briefly in Table 5.

Table 5. Description of WLAN device discovery

Message	Description
1	Device A Inquiring thread broadcasts UDP -message using well known port. the Device B is listening to this broadcast address and port
2	The Device B Listen thread receives message and sends a reply back to device A Listen thread.
2.1	The Device A adds the Device B to its device database.

5.3.3 GPRS –plugin

While Bluetooth and WLAN –plugins can establish direct peer-to-peer connections, GPRS –plugin must route connections through the GPRS Gateway. GPRS –plugin establishes three different connections to the GPRS Gateway: Signalling Connection,

Data Signalling Connection and Data Connection.

The Signalling Connection is a channel for devices to perform inquiries of other devices and services. The Signalling Connection is also used for registering and deregistering of devices and monitoring validity of previously established connections. The Data Signalling Connection is used for negotiating new Data Connections between GPRS –enabled PeerHood –devices. Data Connections are responsible for transferring data from application layer to another device's application layer.

The GPRS Gateway takes care of registering and deregistering PeerHood devices, maintaining device database, checking connection validities and routing signal and data connections between PeerHood devices. Before a device discovery can take a place, the device must register itself into the GPRS Gateway.

Device Registration

When GPRS –enabled PeerHood device connects to the GPRS Gateway for first time, it tries to obtain ID -number from the GPRS Gateway using GETID -message. When the GPRS Gateway receives this, it replies with REPID message containing the granted ID number. The GPRS Gateway then adds information about the connection and the device ID to its database. The registration process is illustrated in Figure 27.

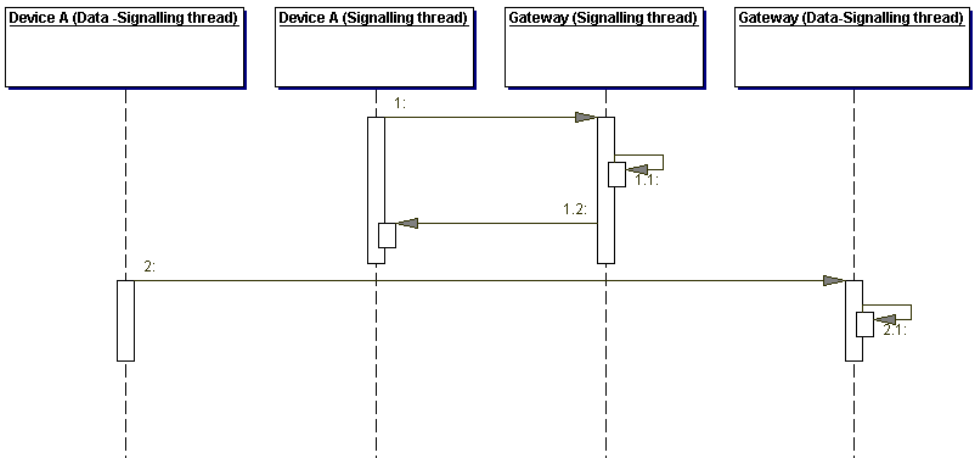

Figure 27. GPRS device registration

In illustrated GPRS device registration case (Figure 27) the Device A creates a new TCP -connection to the GPRS Gateway and requests an ID number. This connection is known as Signaling Connection. The GPRS Gateway grants a new ID number, sends it to the Device A and adds device to it's device database. When the Device A receives the ID, it establishes a new TCP –connection to the GPRS Gateway. This connection is known as Data Signaling Connection. The GPRS Gateway associates these two connections with the ID –number and registration is completed. Messages involved in this process are described briefly in Table 6.

Table 6. Description of GPRS device registration

Message	Description
1	The Device A Signaling -thread creates a new TCP Signaling Connection to the GPRS Gateway and sends GETID –message
1.1	The GPRS Gateway Signaling –thread creates unique ID –number and associates the number with accepted Signaling Connection. This information is then added to the device database.
1.2	The GPRS Gateway sends REPID –message using established Signaling Connection. This message contains unique ID –number to identify GPRS –devices.
2	When a valid ID –number has been received, the Device A Data – Signaling –thread creates a new TCP Data Signaling Connection to the GPRS Gateway and send ADVERT –message containing the previously received ID –number.
2.1	The GPRS Gateway Data Signaling thread associates accepted Data Signaling Connection with received ID –number and updates the device database

Device Deregistration

If the GPRS Gateway notices a timeout or Signaling Connection or Data Signaling Connection breaks, it removes the lost device entry from the device database

Device Discovery

GPRS –plugin implements a device discovery utilizing the GPRS Gateway device database. The GPRS Gateway keeps a track and updates device database of GPRS –enabled PeerHood devices. The GPRS Gateway grants continuously a time period to every registered device. The GPRS Gateway goes through the device list from its device database and allows a time period for every device at a time. Device can use this period of time for device and service discoveries. During this time, other devices are listening for incoming service requests and ready to respond. After device has completed inquiry, it informs the GPRS Gateway which then grants next device to perform an inquiry. If the device doesn't complete inquiries in a certain time, the GPRS Gateway closes Signaling and Data Signaling Connections for this device and erases it from device database.

Devices can be discovered using GETDEV -message. When the GPRS Gateway receives this message, it checks the validity of the connected devices and replies with REPDEV –message containing the number of connected devices. Then devices are sent, one by one using DEVICE –message containing device ID –number. The connection validity check is illustrated and described later. The device discovery process is illustrated in Figure 28.

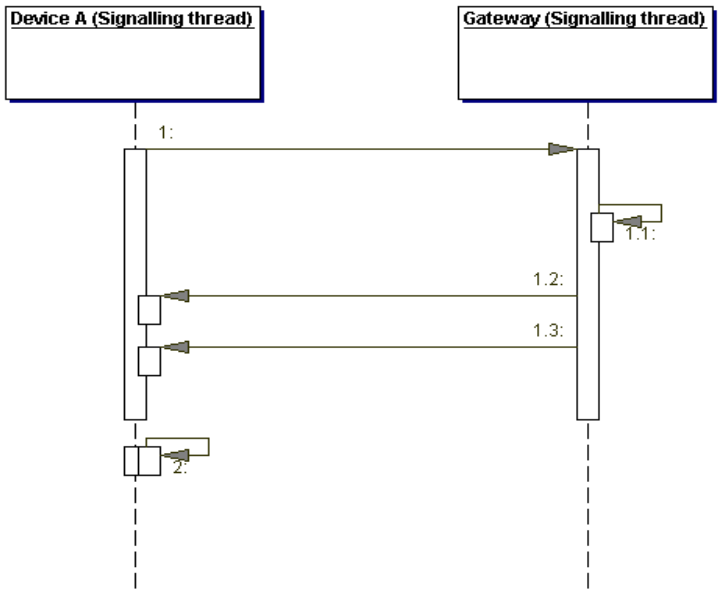

Figure 28. GPRS device discovery

In illustrated GPRS device discovery case (Figure 28) the Device A performs device discovery using previously established Signaling Connection. This discovery is implemented by querying from the GPRS Gateway the number of connected devices. When the GPRS Gateway receives this query –message, it checks the number of connected devices from it's device database. Then the GPRS Gateway sends the number of devices and device ID –numbers to the Device A. The Device A then adds received devices to it's device database. Messages involved in this process are described briefly in Table 7.

Table 7. Description of GPRS device discovery

Message	Description
1	The Device A Signaling thread inquiries devices using GETDEV –message.
1.1	The GPRS Gateway Signaling thread checks its database for registered devices.
1.2	The GPRS Gateway Signaling thread sends REPDEV –message containing the number of registered devices.
1.3	The GPRS Gateway Signaling thread sends all the devices using DEVICE –message containing the device ID –number.
2	The Device A Signaling thread adds devices to its database

Connection validity checking

The GPRS Gateway can check the validity of connection using PROBE –message. When the GPRS Gateway wants to make sure that the connection and the device is still alive, it sends PROBE –message and waits for certain amount of time. If REPPROBE –message is received the remote device and the connection are ok, otherwise the connection is considered as broken and the device must be erased from the device database. The connection validity check is performed using Signaling Connection. The GPRS –connection validity checking is illustrated in Figure 29.

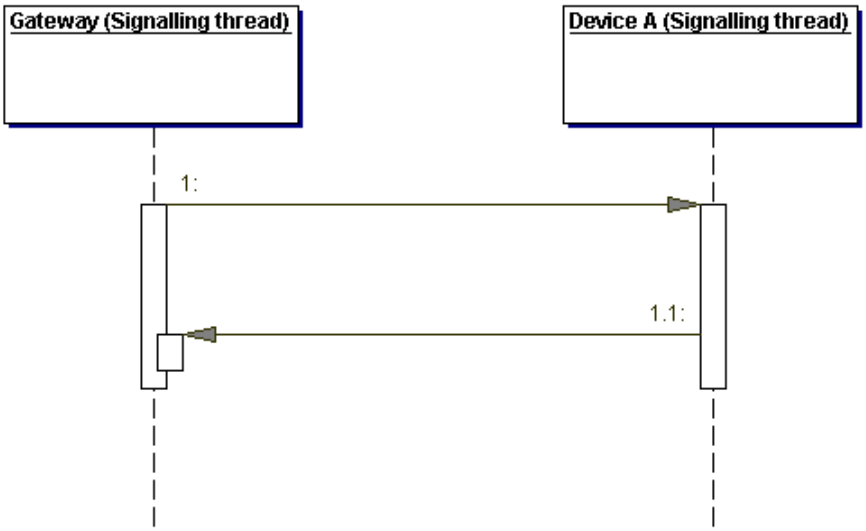

Figure 29. GPRS -connection validity check

In illustrated GPRS –connection validity check case (Figure 29) the GPRS Gateway performs a connection validity check. Connection validity check makes sure that every connected device is alive, and connections are valid. This check is simple process where the GPRS Gateway sends a probe –message to connected device and waits for a response. If response is not valid or the GPRS Gateway receives a timeout, device is verified to be disconnected. All disconnected devices are then erased from the GPRS Gateway's device database. If a connection and a device are ok, connected device sends a reply message to the GPRS Gateway. Messages involved in this process are described briefly in Table 8.

Table 8. Description of GPRS -connection validity check

Message	Description
1	The GPRS Gateway Signaling thread sends PROBE –message
1.1	The Device A Signaling thread replies using REPPROBE -message

5.4 Service discovery

After devices have been found, a device must make a survey of provided services. Service information consists of service name, service port and service parameters. All the plugins provide a similar functionality for querying services: Virtual PeerHood device running Bluetooth, WLAN or GPRS -plugin, connects to other Virtual PeerHood device using specific networking technology. Target device listens to a well known port, accepts incoming connection and sends service information.

Bluetooth –plugin

The Bluetooth –plugin establishes a L2CAP –socket connection and connects directly to a device using Bluetooth address and well-known port. Target device accepts the connection, sends the service information and closes the connection.

WLAN –plugin

The WLAN –plugin establishes a TCP Signaling Connection and connects directly to a device using its IP -address and well-known port. The target device accepts the connection, sends the service information and closes the connection.

Bluetooth and WLAN service discovery process is similar and it is illustrated in Figure 30.

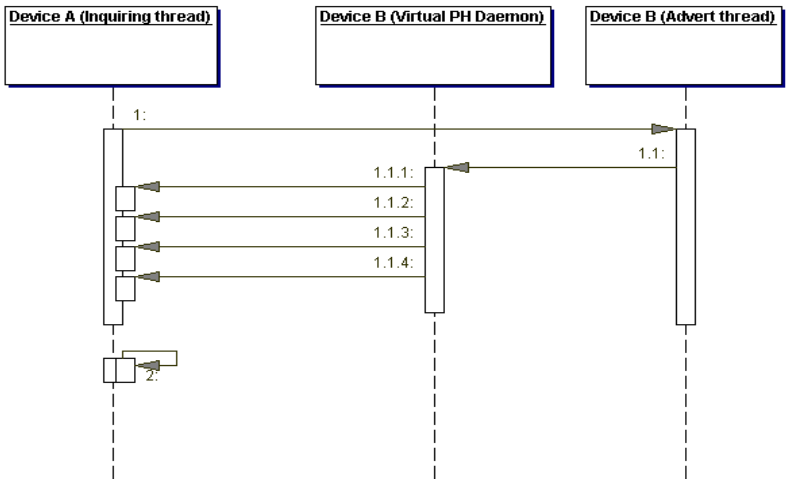

Figure 30. WLAN and Bluetooth service discovery

In illustrated WLAN and Bluetooth service discovery case (Figure 30) the Device A establishes a new TCP Signaling Connection to the Device B using a well-known port. The Device B accepts this new connection and sends information about provided services. The Device A receives services and updates it's device database. Messages involved in this process are described briefly in Table 9.

Table 9. Description of WLAN and Bluetooth service discovery

Message	Description
1	The Device A Inquiry thread connects to Device B using well-known port.
1.1	Device B Advert thread accepts a new connection and initiates sending services using Virtual PeerHood –daemon.
1.1.1	Device B Virtual PeerHood –daemon sends Device B device name.
1.1.2	Device B Virtual PeerHood –daemon sends Device B device checksum.
1.1.3	Device B Virtual PeerHood –daemon sends number of registered services.
1.1.4	Device B Virtual PeerHood –daemon sends all the services.
2	The Device A adds received service info to its device database

GPRS –plugin

GPRS –plugin implements service discovery using previously established TCP Signaling Connection. Services can be discovered using GETSERV –message. GETSERV –message contains the device ID –number which is wanted to be discovered. When the GPRS Gateway receives this message, it checks validity of target device. If connection is valid, it forwards GETSERV –message to target device. Target device then replies with DEVICEINFO -message which contains device's name and checksum. After this, device sends REPSERV –message containing the number of provided services. Services are sent one by one using SERVICE –message. The GPRS Gateway forwards all this information directly back to device which originally started service discovery. GPRS service discovery process is illustrated in Figure 31.

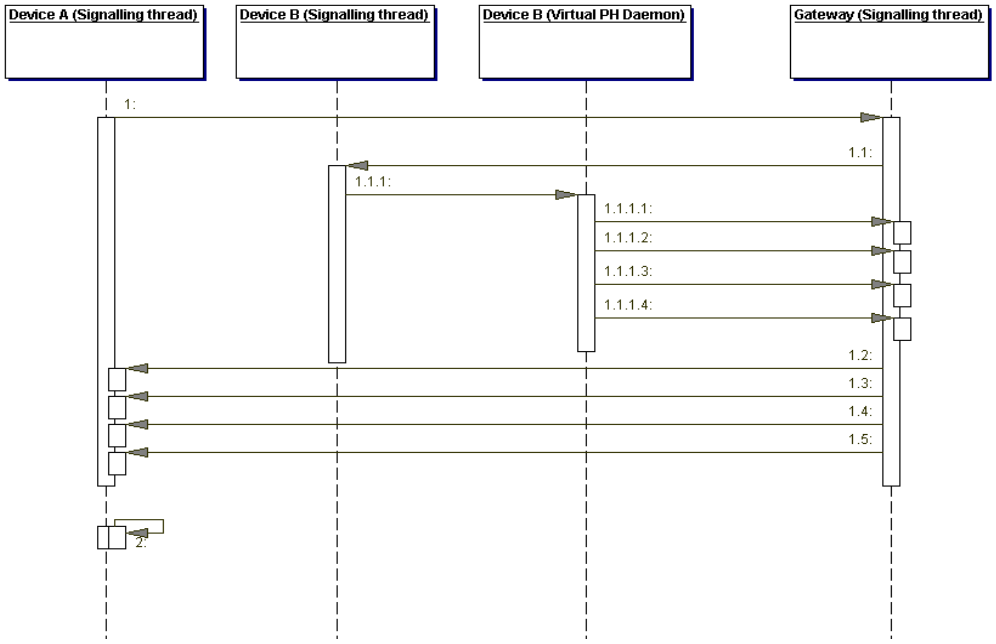

Figure 31. GPRS service discovery

In illustrated GPRS service discovery case (Figure 31) the Device A want's to know what services are provided by the Device B. The Device A starts service discovery sending service request message to the GPRS Gateway using the Signaling Connection. The GPRS Gateway forwards service request message to the Device B using the Signaling Connection. When the Device B receives this service request message, it sends service information to the GPRS Gateway. The GPRS Gateway then forwards this service information to the Device A which updates it's device database. Messages involved in this process are described briefly in Table 10.

Table 10. Description of GPRS service discovery

Message	Description
1	The Device A Signaling thread sends GETSERV –message to the GPRS Gateway using Signaling Connection. Message contains devices ID –number which is wanted to be discovered.
1.1	The GPRS Gateway Signaling thread forwards GETSERV –message

	to the Device Busing Signaling Connection.
1.1.1	Device B initiates sending services using Virtual PeerHood –daemon.
1.1.1.1	Device B Virtual PeerHood –daemon sends Device B name to the GPRS Gateway using Signaling Connection.
1.1.1.2	Device B Virtual PeerHood –daemon sends Device B device checksum to the GPRS Gateway using Signaling Connection.
1.1.1.3	Device B Virtual PeerHood –daemon sends the number of registered services to the GPRS Gateway using Signaling Connection.
1.1.1.4	Device B Virtual PeerHood –daemon sends all the services to the GPRS Gateway using Signaling Connection.
1.2	The GPRS Gateway Signaling thread sends DEVICEINFO –message name to The Device A using Signaling Connection. This message contains Device B name and checksum.
1.3	The GPRS Gateway Signaling thread sends REPSERV –message to The Device A using Signaling Connection. This message contains number of registered services of Device B.
1.4	The GPRS Gateway Signaling thread sends SERVICE –message to The Device A using Signaling Connection. This message contains length of the service data.
1.5	The GPRS Gateway Signaling thread sends the service data of Device B to The Device A using Signaling Connection.
2	The Device A adds received service info to its device database

5.5 Making connections

WLAN and Bluetooth -plugins can create connections directly with other devices running WLAN and Bluetooth -plugins. If device running GPRS –plugin wants to establish a new data connection with other GPRS –device, it first creates a new Client Data -thread which creates a new Data Connection to the GPRS Gateway and sends CONNECT –message containing target devices ID –number. The GPRS Gateway then checks validity of target devices connection and forwards this CONNECT –message

using the Data Signalling connection to target device. When the target device receives a CONNECT –message, it creates a new Server Data -thread which establishes a new Data Connection to the GPRS Gateway and sends a REPCONNECT –message. The GPRS Gateway forwards the REPCONNECT to the original device which initiated the process and creates a new Data thread. This new Data -thread starts to route data between Client and Server Data -threads. Because of this parallel implementation, the GPRS Gateway can easily handle many simultaneous data transfers. Connection process is illustrated in Figure 32.

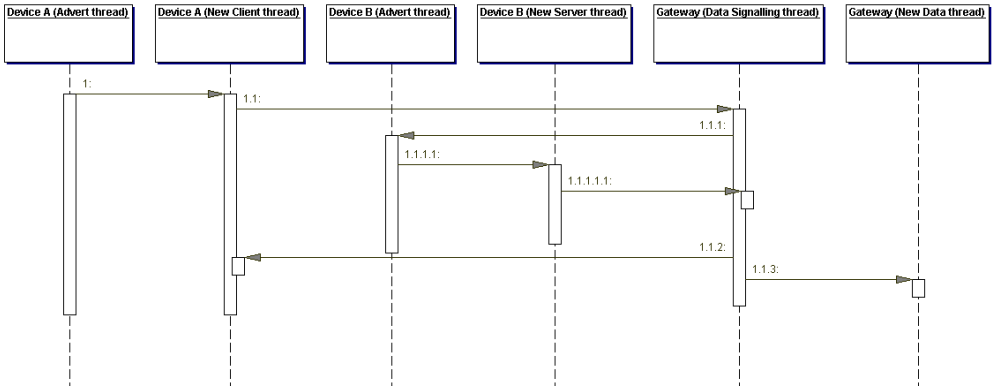

Figure 32. Establishing GPRS Data Connection

In illustrated GPRS Data Connection establish case (Figure 32) the Device A now knows that the Device B is providing a service and the Device A want's to use this service. The Device A creates a new Client Data -thread which establishes a new Data Connection to the GPRS Gateway and sends a connection request message. When the GPRS Gateway receives this connection request message, it forwards the message to the Device B using previously established Data Signaling Connection. When the Device B receives the connection request message, it creates a new Server Data -thread which establishes a new Data Connection to the GPRS Gateway and sends reply to connection request message. The GPRS Gateway then forwards this connection request reply message to the Device B using established Data Connection and creates a new Data -thread. This new Data -thread starts to route data between The Device A and Device B Data Connections. Messages involved in this process are described briefly in Table 11.

Table 11. Description of establishing GPRS data connection

Message	Description
1	The Device A Advert thread creates a new Client Data thread and informs which device is wanted to be connected
1.1	Device Client Data creates a new Data Connection to The GPRS The GPRS Gateway and sends CONNECT –message to The GPRS Gateway Data Signaling thread. This message contains the target devices ID –number
1.1.1	The GPRS Gateway Data Signaling thread forwards CONNECT –message to the Device B Advert thread using Data Signaling Connection.
1.1.1.1	Device B Advert thread creates a new Server Data thread and informs which device wanted to make a connection
1.1.1.1.1	Device B Server Data thread creates a new Data Connection to The GPRS Gateway and sends REPCONNECT –message to The GPRS Gateway Data Signaling thread.
1.1.2	The GPRS Gateway Data Signaling thread forwards REPCONNECT –message to The Device A Client Data thread using existing Data –connection.
1.1.3	The GPRS Gateway Data Signaling thread creates a new Data thread and passes both Data Connections to it. Data thread then starts to route data between these Data Connections.

5.6 Connection Monitoring

When user is on the move and using wireless connection, link quality can vary greatly caused by interference of environment. For in able to create and maintain reliable connection, link quality must meet a certain level. If the link quality seems to go too weak or breaks, application must be able to react quickly for in able to establish a new connection and continue data transfer. Thus, connection must be monitored simultaneously while transferring data. GPRS –plugin doesn't implement monitoring

because at the scope of this work, GPRS –connection is considered to be always on.

Bluetooth Signal Monitoring

Bluetooth link quality determines the quality of the link between two Bluetooth devices and it can be only monitored when physical connection has been established. Link quality can be achieved from ACL –connection using HCI Read Quality –command. This command will return link quality value as from 0 to 255. There can be differences on a scaling link quality depending on a manufacturer of Bluetooth chip. However, the higher the value is, the better the link quality is.

WLAN Signal Monitoring

WLAN link quality determines the quality of the link between mobile device and access point and this can be measured at any time when the device is on access point's area. Link quality can be manually measured by investigating beacon –packets sent by access point. However, Virtual PeerHood implementation uses Wireless Extension /21/ which provides tools for monitoring WLAN statistics and link quality. Wireless Extension provides link strength which determines the signal strength at the receiver and link quality which indicates how good the reception is (for example the percentage of correctly received packets).

5.7 Handover

There are two types of handovers. Most handovers occur between in access points of the same network technology and are termed *horizontal handovers*. Handovers between different access points belonging to different networks (e.g. WLAN to GPRS) are referred to as *vertical handovers*, and pose a significantly greater challenge /25/

Handover can be controlled using software or hardware. For example, Virtual PeerHood middleware takes care of performing software aided vertical handovers between different networking technologies and WLAN access points take care of performing hardware-aided horizontal handover to next WLAN access point.

When link quality gets weak or breaks, Virtual PeerHood tries to continue the data transferring by establishing a new Data Connection to the same device using an alternative wireless technology. While connection is established to a service the RoamingThread constantly searches for the same service using other available wireless technologies. If the same service can be found from the same device using alternative technology, handover can be performed instantly if needed. User can define priorities for searching alternative technologies. For example, because of low power consumption of Bluetooth, it could have the highest priority on a situation when a low battery consuming is important. GPRS could be the priority when the user wants to maintain connectivity while on the moving long distances. Example of handover is illustrated in figure 33. Activity diagram of RoamingThread implementation is illustrated in Appendix 1.

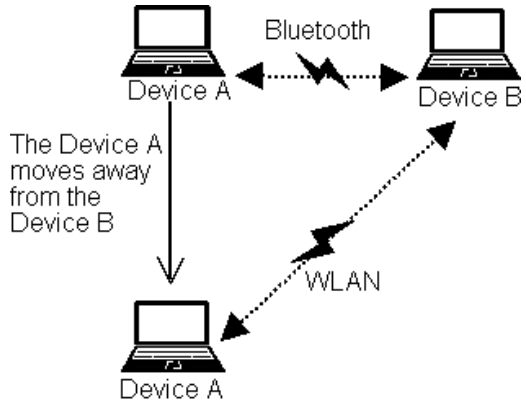

Figure 33. Handover from Bluetooth to WLAN

In illustrated handover process (Figure 33) the Device A has established a new Bluetooth Data Connection with the Device B (1). Virtual PeerHood RoamingThread constantly monitors established connection link quality. RoamingThread also searches constantly the same service which is currently used using alternative networking technologies. It happens to be the case that the Device B has also WLAN –technology available so the RoamingThread notifies that WLAN can be also used. The Device A moves then away from the Device B and the link quality gets weak. The RoamingThread notices that link quality has been weak for a while and performs a handover from Bluetooth to WLAN. The Device A establishes a new Data Connection

to the Device B using WLAN (2). Service can be then used over WLAN (3).

Data Buffering

Current implementation doesn't include any kind of data buffering. Without buffering, Seamless Connectivity is not very useful because data can be lost when networking technology is changed during data transfer. Buffering could be implemented in Virtual PeerHood –library which buffers data after it comes from application layer and needs to be sent.

6 RESULTS

Virtual PeerHood was implemented for Linux operating system. Early versions were developed only for desktop environment running Debian Linux distribution /26/ but Virtual PeerHood was eventually cross-compiled also for Compaq iPAQ PDA –devices, running Familiar Linux distribution /27/. A Simple file-sharing application was implemented to test functionality of different plugins. This application allowed transferring files between devices using Bluetooth, WLAN and GPRS –plugins. A screenshot of implemented application is illustrated in Figure 34

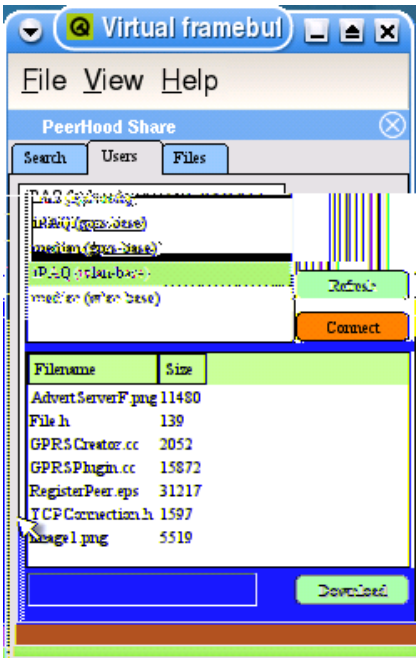

Figure 34. Virtual PeerHood testing application

Seamless connectivity between different networking technologies was implemented as an experimental feature of Virtual PeerHood. Current implementation provides simplified Seamless Connectivity which takes care of changing the networking technology automatically if signal is weakening. However, this implementation doesn't provide buffering for data which is badly needed if the feature is used on any other

situations than testing purposes.

What comes to current performance of Virtual PeerHood, there are quite many parameters which must be optimized to make system more usable. These parameters include for example device inquiry and service discovery intervals. Current Bluetooth version 1.1 -standard has still some problems concerning connection and inquiry times which may be reduced by new standard. However, Bluetooth is still only one networking technology among others which is supported by Virtual PeerHood. If future shows that, Bluetooth becomes a bottleneck of the system, it can be replaced using some other low-range networking technology.

Future work will include more detailed specification of the concept of the virtual neighborhood. This specification should define how the devices should allow other devices to query and use provided services and how the devices should identify each other.

7 CONCLUSIONS

Primary goal for Virtual PeerHood was to prove that wireless mobile devices can utilize several network technologies simultaneously in Peer-to-Peer environment. Now when the implementation of Virtual PeerHood has been done, it can be said that this goal were achieved. Current implementation has been used on both desktop and mobile environment which has showed that Virtual PeerHood is very flexible in many ways: it can be easily transferred between different architectures and it can be extended to provide new functionality using third party plugins.

Virtual PeerHood has also proven that the whole concept of 'network neighborhood' is only a question of how it has been defined. While Bluetooth defines a clear neighborhood area based on its limited signal range, GPRS and WLAN can provide virtual neighborhood where devices can be located all over the globe. However, when concerning Virtual PeerHood and virtual neighborhood on global scale, Quality of Service (QoS) issues become important because of increased communication delays.

Virtual neighborhood provides new possibilities to use services freely without any location restrictions but it also creates problems for example when concerning user identification. In network neighborhood where devices are located physically next to each other, and device users can see each other, access control and device identification are not as great concern than in virtual neighborhood. Thus there are still many things which should be concerned if Virtual PeerHood is applied for any commercial use.

REFERENCES

/1/ The Napster. Peer-to-Peer file-sharing tool [www-pages]. Available at http://www.napster.com/ [Referred 14.4.2004]

/2/ Dreamtech Software. Peer-to-Peer Application Development. Hungry Minds, 2002. ISBN 0-7645-4904-9

/3/ The Gnutella. Protocol Specificiaton Documentation v0.4 [www-pages]. Available at http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf [Referred 14.4.2004]

/4/ The ICQ. Instant messaging tool [www-pages]. Available at http://web.icq.com/ [Referred 20.4.2004]

/5/ Seti@home. Peer-to-Peer distributed computing tool [www-pages]. Available at http://setiathome.ssl.berkeley.edu/ [Referred 14.4.2004]

/6/ David Barkai. Peer-to-Peer Computing. Intel Press, 2001. ISBN 0-9702846-7-5

/7/ The Infrared Data Association. Specifications for Infrared protocols. [www-pages] Availablse at http://www.irda.org/ [Referred 20.4.2004]

/8/ Bluetooth Special Interest Group. Bluetooth Specification Version 1.1 Core. [e-document]. Available at http://www.bluetooth.org [Referred 6.4.2004]

/9/ IEEE 802.11 standard. Specification Documentation [e-document]. Available at http://standards.ieee.org/getieee802/portfolio.html [Referred 15.4.2004]

/10/ David Kammer. Bluetooth Application Developer's Guide: The Short Range Interconnect Solution. Syngress, 2002. ISBN 1-928994-42-3

/11/ Matthew S. Gast. 802.11 Wireless Networks The Definitive Guide. O'Reilly 2002. ISBN 0-596-00183

/12/ IEEE 802.3 standard. Specification Documentation [e-document]. Available at http://standards.ieee.org/getieee802/portfolio.html [Referred 15.4.2004]

/13/ Kaj Granlund. Langaton Tiedonsiirto. Docendo, 2001. ISBN 951-846-091-4

/14/ GPRS Protocol Stack [www-pages]. Available at http://www.palowireless.com/gprs/tutorials.asp [Referred 20.4.2004]

/15/ Power Consumption Estimates for Bluetooth [e-document]. Available at http://www.siliconwave.com/pdf/73_0008.pdf [Referred 15.4.2004]

/16/ Mobile IP. Request for Comments: 2002 [e-document]. Available at http://www.ietf.org/rfc/rfc2002.txt [Referred 21.4.2004]

/17/ Bluez. Official Linux Bluetooth Protocol Stack. [www-pages]. Available at http://bluez.org [Referred 6.4.2004]

/18/ Axis. Bluetooth stack. [www-pages]. Available at http://developer.axis.com/software/bluetooth/ [Referred 6.4.2004]

/19/ Affix. Bluetooth stack [www-pages]. Available at http://affix.sourceforge.net/ [Referred 6.4.2004]

/20/ Linux-wlan. WLAN driver and utility for Linux. [www-pages]. Available at http://www.linux-wlan.com/linux-wlan/ [Referred 6.4.2004]

/21/ Wavelan and Wireless Extensions. MPL/GPL drivers for the Wavelan IEEE/Orinoco cards [www-pages]. Available at

http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Orinoco.html [Referred 6.4.2004]

/22/ Nokia. Linux driver for Nokia D211 GPRS interface [www-pages]. Available at http://www.nokia.com/ [Referred 22.4.2004]

/23/ Porras Jari, Hiirsalmi Petri, Valtaoja Ari. Peer-to-Peer Communication Approach for a Mobile Environment. IEEE Computer Society, 2004. ISBN 0-7695-2056-1

/24/ Dynamic Host Configuration Protocol. Request for Comments: 1531 [e-document]. Available at ftp://ftp.isi.edu/in-notes/rfc1531.txt [Referred 7.4.2004]

/25/ Chakravorty Rajiv, Vidales Pablo, Patanabongpibul Leo. On Inter-network Handover Performance using IPv6 [e-document]. Available at http://www.cl.cam.ac.uk/users/rc277/handovers.pdf [Referred 23.4.2004]

/26/ Debian. Linux distribution [www-pages]. Available at http://www.debian.org/ [Referred 21.4.2004]

/27/ Familiar. Linux distribution [www-pages]. Available at http://familiar.handhelds.org/ [Referred 21.4.2004]

/28/ Nokia. Mobile Internet Technical Architecture, Technolgoies and Standardization. Edita Publishing 2002. ISBN 951-826-668-9

Appendix 1. Activity diagram of the RoamingThread

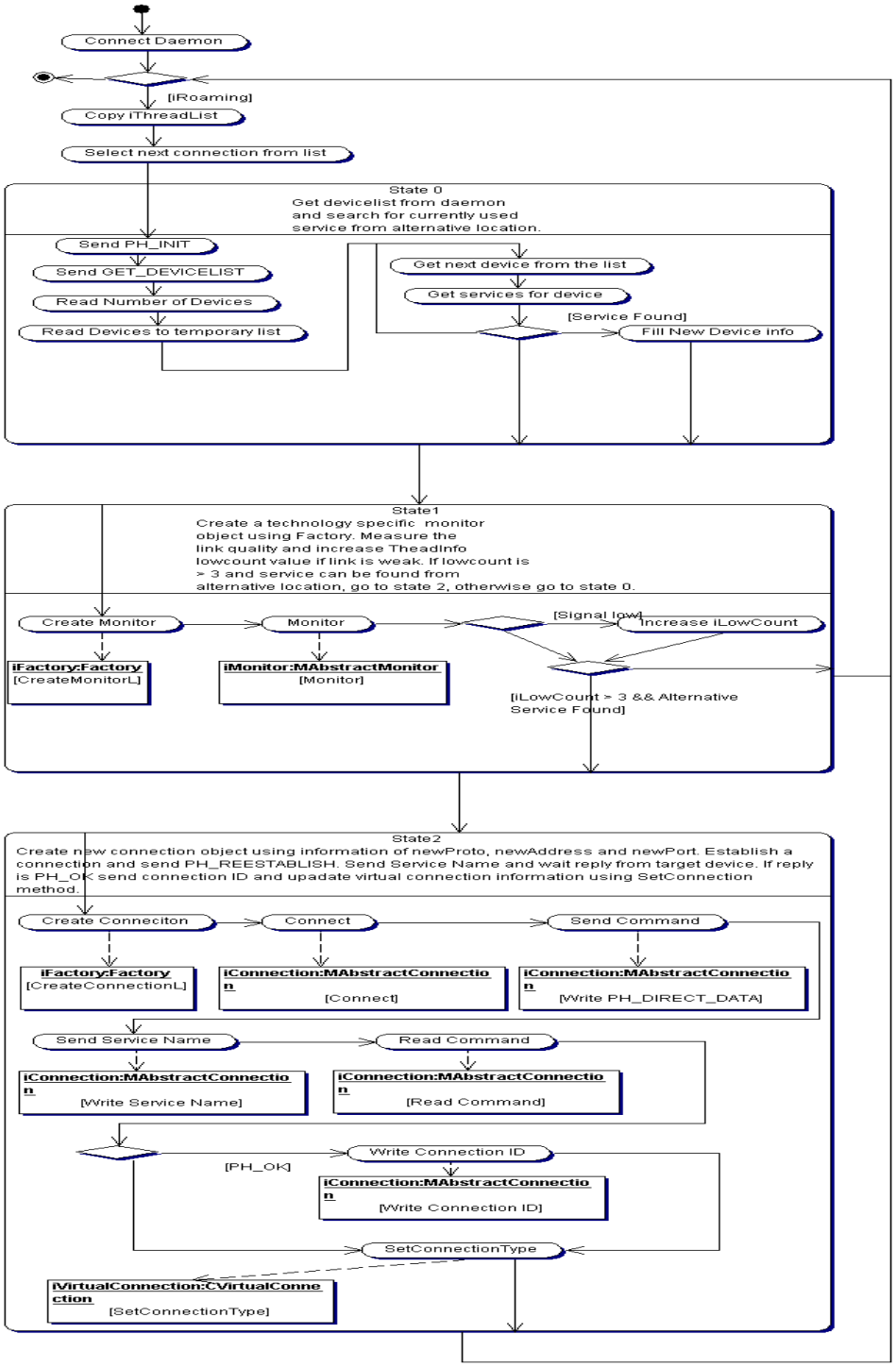

Figure 1. Activity diagram of the RoamingThread

(continues)

(Appendix 1 continues)

Figure 1 shows the implementation of the Virtual PeerHood RoamingThread. The RoamingThread takes care of performing handover between networking technologies. When a Data Connection has been established between two PeerHood –enabled devices and current connection seems to be weak, the RoamingThread performs a handover if there is an alternative networking technology available. This handover simply creates a new Data Connection to target device and closes then the previous Data Connection. All data between devices is then transferred using this new Data Connection.

The RoamingThread consists of three states: state 0 searches the currently used service from remote device using alternative technologies, state 1 monitors the connection link quality and state 2 performs the handover if it is needed.

The RoamingThread has an access to the iThreadList. iThreadList contains information of all established Data Connections. The RoamingThread goes this list through constantly and checks every connection if the handover is needed. The RoamingThread uses a counter for every connection which determines how many times the connection has been weak. If a connection has been weak three times in a row, and there is an alternative technology available, a handover is initiated. If alternative technology is not available, RoamingThread does nothing but waits if the connection breaks.

Virtual PeerHood provides Seamless Connectivity as a feature which can be used by enabling it in a configuration file. If the feature is not enabled, RoamingThread is not created and user must take care of creating a new Data Connection if the old one seems to be weak.