ESKO KOKKONEN

ENABLING PEER-TO-PEER APPLICATION DISTRIBUTION IN MOBILE TERMINALS

Master's Thesis
11th November 2004

Supervisor: Professor Antti Ylä-Jääski
Instructor: Mikko Lönnfors, M.Sc.

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering
Telecommunications Software and Multimedia Laboratory

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Author:	Esko Kokkonen
Title:	Enabling Peer-to-peer Application Distribution in Mobile Terminals
Date:	11th November 2004
Pages:	66
Department:	Department of Computer Science and Engineering
Professorship:	T-110, Telecommunications Software
Supervisor:	Professor Antti Ylä-Jääski
Instructor:	Mikko Lönnfors, M.Sc.

Current mobile phones allow third-party software to be installed on them. Mobile application distribution practices have been traditionally server-based solutions. This thesis represents an alternative way of distributing applications using the mobile networks, a peer-to-peer solution. Peer-to-peer solutions are not yet fully developed in the application distribution domain, but they may offer an intuitive way of distributing peer-to-peer applications in the future.

Third generation mobile networks will allow applications to communicate with each other in peer-to-peer fashion. To use this feature, peer-to-peer applications need to be installed in the mobile devices. The applications itself can be distributed using peer-to-peer connections. A demonstration environment of this peer-to-peer application distribution solution was created to show that it is technically possible to create such a system.

The demonstration application distribution system was build using SIP as the signaling protocol. The network consists of a GPRS network and a SIP server. The mobile terminals needed changes at the software level in order to enable the distribution process. An API was designed to enable Java 2 ME applications to use this distribution mechanism. The peer-to-peer distribution mechanism proved to be an working alternative to the server-based ones, but it still needs some work on charging, APIs and security issues before it can be commercially exploited.

Keywords:	peer-to-peer, application distribution, superdistribution, SIP, GPRS, UMTS
Language:	English

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Esko Kokkonen
Työn nimi:	Vertaisyhteyksiä käyttävä sovellustenjakelu mobiileissa päätelaitteissa
Päivämäärä:	11. marraskuuta 2004
Sivuja:	66
Osasto:	Tietotekniikan osasto
Professuuri:	T-110, Tietoliikenneohjelmistot
Työn valvoja:	Professori Antti Ylä-Jääski
Työn ohjaaja:	DI Mikko Lönnfors

Nykyisiin matkapuhelimiin on mahdollista asentaa kolmanten osapuolten tekemiä sovelluksia. Perinteisesti matkapuhelinsovellusten jakelussa on käytetty palvelinpohjaisia ratkaisuja. Tässä opinnäytetyössä esitetään vaihtoehtoinen, vertaisliikenteeseen pohjautuva ratkaisu, joka käyttää hyväkseen matkapuhelinverkkoa. Vertaisliikenteeseen perustuvat ratkaisut eivät ole vielä kehittyneet sovellusten jakeluun sopivaksi, mutta saattavat tulevaisuudessa tarjota siihen intuitiivisen tavan.

Kolmannen sukupolven matkapuhelinverkoissa sovellusohjelmat voivat kommunikoida keskenään käyttäen vertaisyhteyksiä. Sovellusohjelmien tulee olla asennettuina matkapuhelinlaitteissa, jotta vertaisyhteyksiä voitaisiin käyttää. Sovellusohjemien jakelu voidaan toteuttaa käyttäen vertaisyhteyksiä. Tämän vertaisyhteyksiä käyttävän sovellustenjakelumenetelmän tueksi rakennettiin kokeiluympäristö, joka osoittaa että kyseinen järjestelmä on teknisesti mahdollista toteuttaa.

Rakennettu kokeiluympäristö käyttää SIP-yhteiskäytäntöä. Verkkoympäristö koostuu GPRS-verkosta ja SIP-palvelimesta. Myös matkapuhelinpäätelaitteeseen ohjelmistoihin pitää tehdä muutoksia jakelun mahdollistamiseksi. Ohjelmointirajapinta suunniteltiin Java 2 ME -ympäristöön, jotta sovellukset voisivat käyttää tätä menetelmää. Tämä vertaisliikenteeseen perustuva menetelmä osoittautui toimivaksi vaihtoehdoksi palvelinpohjaisille järjestelmille. Laskutukseen, rajapintoihin ja turvallisuuteen liittyviä ongelmia on syytä tarkastella lähemmin ennenkuin menetelmää voidaan kaupallisesti hyödyntää.

Avainsanat:	sovellusten jakelu, SIP, GPRS, UMTS
Kieli:	englanti

Acknowledgements

This work was done at the Networking Technologies Laboratory of Nokia Research Center. I would like to thank my supervisor Antti Ylä-Jääski and my instructor Mikko Lönnfors for their valuable feedback and advice.

I also would like to thank my managers here at NRC, Jaakko Teinilä and Jari Selin, for making the creation of this thesis possible. Acknowledgements also go to my colleagues Jari Kinnunen, Kari Kostiainen, Pekka Kuismanen, Olli Rantapuska and all others who assisted me making and finishing this thesis.

Last, but not least, thanks to all my friends and family for your support.

Helsinki
11th November 2004

Esko Kokkonen

Table of Contents

Abbreviations

3GPP	Third Generation Partnership Project
AC	Authentication Center
API	Application Programming Interface
ARIB	Association of Radio Industries and Businesses
ATIS	Alliance for Telecommunications Industry Solutions
CCSA	China Communications Standards Association
CDC	Connected Devices Configuration
CLDC	Connected Limited Devices Configuration
CS-CN	Circuit-Switched Core Network
CSCF	Call Session Control Function
CSD	Circuit-Switched Data
EDGE	Enhanced Data rates for GSM Evolution
ETSI	European Telecommunications Standards Institute
FTP	File Transfer Protocol
GGSN	Gateway GPRS Support Node
GMSC	Gateway Mobile Switching Center
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
HLR	Home Location Register
HSCSD	High Speed Circuit-Switched Data
HSS	Home Subscriber Server
HTTP	Hypertext Transfer Protocol

I-CSCF	Interrogating Call Session Control Function
IETF	Internet Engineering Task Force
IMS	IP Multimedia Subsystem
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ISUP	ISDN User Part
J2ME	Java 2, Micro Edition
JAD	Java Application Descriptor
JAR	Java Archive
JCP	Java Community Process
JVM	Java Virtual Machine
KVM	Kilobyte Virtual Machine
LAN	Local Area Network
MGCF	Media Gateway Control Function
MGW	Media Gateway
MIDP	Mobile Information Device Profile
MIME	Multipurpose Internet Mail Extensions
MMS	Multimedia Messaging Service
MSC	Mobile Switching Center
OTA	Over the Air
P-CSCF	Proxying Call Session Control Function
PCM	Pulse Code Modulation
PDA	Personal Digital Assistant
PDN	Packet Data Network
PDP	Packet Data Protocol
PS-CN	Packet-Switched Core Network
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RTSP	Real Time Streaming Protocol
S-CSCF	Serving Call Session Control Function

SDP	Session Description Protocol
SGSN	Serving GPRS Support Node
SIM	Subscriber Identity Module
SIP	Session Initiation Protocol
SMS	Short Message Service
T-SGW	Transport Signaling Gateway
TCP	Transmission Control Protocol
TSG	Transport Signaling Gateway
TTA	Telecommunications Technology Association of Korea
TTC	The Telecommunication Technology Committee
UA	User Agent
UDP	User Datagram Protocol
UML	Unified Modeling Language
UMTS	Universal Mobile Telecommunications System
UTRAN	UMTS Terrestrial Radio Access Network
VLR	Visitor Location Register
VoIP	Voice over IP
W-CDMA	Wideband Code Division Multiple Access
WAP	Wireless Application Protocol
WLAN	Wireless Local Area Network
XHTML	eXtensible Hypertext Markup Language

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Mobile communication has become very common, and it is now a part of our daily lives. People use their mobile phones every day, making voice calls and sending short messages to each other. Some of us even play games or browse the web using mobile phones. More and more mobile services are launched and advertised in addition to the widely used voice calls and short messaging services.

The services that current cellular networks offer are improving rapidly. The usage of short messaging service has grown greatly after its introduction to the consumers. Operator logos and ringing tones are good examples of providing additional services besides voice calls. New innovative services are launched frequently, covering all areas from entertainment to business services. Games, news services, dating services and stock broker services are only examples of the variety available. The evolution of mobile services do not seem to stop here, as many new ways of using mobile phones are constantly emerging.

It is interesting to note that mobile phones have become more than phones, and some models could be easily presented as portable computers with a capability to make phone calls. Current mobile phones even allow third-party software to be installed and run on them. This creates better possibilities to use the phones for services that are not related to the traditional voice communications at all. Clearly, the mobile phones are taking a different role in the mobile communication business

than they used to have.

Interestingly, most of the currently available mobile services rely on the short messaging service, which is quite a limited way of transferring information. The short messaging service is usually used in person-to-person communication, which is peer-to-peer communication from the users' point of view. But still, as this service is very widely used, it has been practical to use it for different purposes too.

The additional services that use short messaging service do not interact directly with other phones, but rather with a host or a server. These kind of services are also called server-based applications. Apart from person-to-person short messaging, peer-to-peer applications have not been introduced to mass markets yet. But, it may well be that the next killer application could be found among the new innovative peer-to-peer applications.

Here peer-to-peer is defined as a concept in which the direct communication between peers is established by connecting directly to the other peer, and it is of no importance which one of the two peers have actually started the connection. Traditional voice calls are an example of peer-to-peer communications. However, it is important to note that peer-to-peer applications are not exactly the same thing as file sharing applications found in the Internet, even though many of those file sharing applications use also peer-to-peer connections.

Peer-to-peer software has grown popularity in the Internet during last years. For example, instant messaging applications are quite popular these days, in both personal and business use. Peer-to-peer solutions are clearly a growing market with a lot of opportunities, and these solutions could be well applied to the mobile environment too.

When using peer-to-peer applications, all users need to have those applications in question installed on their devices. In other words, the applications need to be distributed to the terminals before the peer-to-peer connectivity can be enabled. This distribution can be done in many ways, like using server-based solutions. This thesis represents a new, interesting way of distributing mobile applications, a peer-to-peer distribution solution.

While distributing applications or content in the Internet has lately changed towards peer-to-peer solutions, the server-based solutions still play very important role. In

the mobile environment, the application distribution is server-based, and currently there are no commercial peer-to-peer application distribution solutions available. The technical limitations of current mobile networks are one reason for this, among with the fact that data traffic is quite expensive in these networks.

At the time of writing this thesis, the application distribution practices still follow the traditional server-based models. Other than voice connections or short messages, the mobile peer-to-peer traffic is practically non-existent. Therefore it is tempting to harness the growth of the peer-to-peer networks to the mobile environment. The features of currently deployed mobile networks limit this opportunity, so it is worthwhile studying how to make full use of peer-to-peer software in the future mobile communication systems.

Peer-to-peer applications designed for the mobile devices could be similar to their Internet point of comparison, like instant messaging or gaming applications. Unlike in the Internet, charging is one of the basis of mobile networks, and it is possible to support charging also in mobile peer-to-peer traffic. This is a challenge to the operators; how to properly bill peer-to-peer traffic and still make it attractive for the subscribers to use?

The distribution of mobile applications has been server-based for many reasons. The applications are usually optimized for specific phone models, and so there must be different versions for different types of phones. Of course, the operators are eager to charge the distribution acts, and that can be implemented in the server-based distribution mechanism.

Application distribution in peer-to-peer fashion would be practical in many ways. Free applications could be launched without the need for advertising, as they would spread from one user to another one. If the applications used peer-to-peer connections when communicating with each other, also the distribution process could be attached to the peer-to-peer communication initiation. Besides, it is more attractive for the users when the distribution can be done without charge.

It is likely that mobile applications will interact more with each other in the future. Instant messaging and mobile gaming are promising areas which could benefit from the possibility of using peer-to-peer connections. With these type of peer-to-peer applications, there should also be a way to distribute these applications

conveniently and efficiently.

1.2 Scope of the Thesis

Application distribution is an important factor in enabling peer-to-peer software to become widely used. User's favorite application must be easily shareable to his/her friends in order to maximize the application distribution. One possible alternative to make this possible is to use peer-to-peer connections in the application distribution process.

Peer-to-peer application distribution (superdistribution) in mobile environment is analyzed and explained, and one possible solution is studied in detail in this thesis. Also the descriptions of the software functionality needed for application downloading and installation in mobile terminals are included in the scope of this thesis.

1.3 Goals

The main goal is to design and describe the implementation of peer-to-peer application distribution mechanism on the Nokia 6600 mobile phone in a 2.5G mobile network.

The description of the implementation includes information about underlying technologies (such as mobile networks and SIP) that form the basis for the application distribution environment. The architecture and the design of the software implementation are described in detail.

1.4 Organization of the Thesis

In Chapter 2, the problem area and technology background are presented and explained. This chapter gives an overview to the mobile network environment and mobile platforms, as well as to the current application distribution practices and session initiation protocol.

Chapter 3 covers a couple of sample usage scenarios and the requirements based

on those scenarios.

The implementation is explained in Chapter 4. The implementation done is one possible solution to the peer-to-peer application distribution problem.

The evaluation of the implementation and the rationale behind the choices made are covered in Chapter 5.

The conclusions and future work are discussed in Chapter 6.

Chapter 2

Technical Background and Problem Area

2.1 Peer-to-peer Model

Client-server model is the traditional networking model, where one host is a server to which clients have to contact to use services it provides. It is typical that the client does not have the same functionality as the server, therefore clients can be lighter and easier to implement. The servers typically are heavier, more robust and often also much more complicated than clients.

In peer-to-peer model, each party has somewhat equal means or rights to initialize and receive communication requests. This implies also that every node or peer has both server and client capabilities, and both peers are capable of receiving and handling requests and sending requests.

The term servent is sometimes used to describe peers. This servent is an artificial word, a combination of words server and client. When a peer has the capability to act as a server and a client, it can be called a servent [19]. Other peer-to-peer related terminology is explained in Table 2.2.

In the Internet, the term peer-to-peer is commonly used as a replacement for file sharing software. In fact, file sharing software is only a subset of peer-to-peer software, and all file sharing software are not necessarily peer-to-peer. Therefore

it is important to be careful when using these terms.

Peer-to-peer may be defined differently by different authors, but in this thesis, the definition of peer-to-peer follows the definition of Schollmeier [19]:

> "A distributed network architecture may be called a Peer-to-Peer (P-to-P, P2P,.) network, if the participants share a part of their own hardware resources (processing power, storage capacity, network link capacity, printers,.). These shared resources are necessary to provide the Service and content offered by the network (e.g. file sharing or shared workspaces for collaboration). They are accessible by other peers directly, without passing intermediary entities. The participants of such a network are thus resource (Service and content) providers as well as resource (Service and content) requestors (Servent-concept)."

Of course, the network can be a combination of the peer-to-peer model and the client-server model. These kind of networks are called hybrid networks. In these hybrid networks, there are separate servers among servents.

Term	**Explanation**
Peer	A node that is connected to network.
Peer-to-peer	Every peer has a capability to act as a server or a client when connecting to any other peer in the network.
Server	Server offers services to its clients.
Client	Client uses servers services.
Servent	A Peer with a combination of both client and server functionality.

Table 2.2: Peer-to-peer terms

Application distribution is the process of transferring application programs to peers, so that the peer or peers involved can install the application. In peer-to-peer application distribution, applications are distributed directly from peer to another peer, so that there is no application server in between.

In this thesis, the applications programs are mobile applications (such as games) and the peers are mobile devices operating in a mobile network.

2.2 Cellular Networks

Nowadays, cellular networks offer many services beyond voice calls and short messages. Multimedia messaging and web browsing are good examples of what can be done with current technology. The evolution does not seem to stop here, and in future, we will see many more services that are enabled by more sophisticated cellular networks.

Traditional telephone networks and majority of current cellular networks are based on circuit-switched technology. For example, Public Switched Telephone Network (PSTN) and Global System for Mobile communications (GSM) are circuit-switched networks. This means that the call session is set up so that all needed resources are reserved, thus making the session look like a circuit where data can be directly transmitted back and worth until the connection is explicitly closed.

Packet-switched telephone networks have a different approach. Like in the Internet, data is transferred in packets and their delivery is not guaranteed, and the actual route that packets take can vary. The network does not provide a circuit or a connection, it simply transfers packets. The responsibility to ensure the correctness of the data is left to the terminals.

Internet Protocol (IP) (see 2.2.1) is the protocol that is generally used in packet-switched networks, like the Internet. Also third generation networks will utilize IP networks in their packet-switched domains, therefore it is good to know how IP networks work.

Before we go into actual cellular networks, we introduce the basic concepts of the IP networks.

2.2.1 Internet Protocol Networks

IP networks are networks that use IP protocol as the network protocol. These kind of networks form the basis of the Internet, and are also used in many other net-

works, like packet-switched cellular networks. It is also possible to join new networks to existing Internet, and using this feature, connect packet-switched cellular networks to the Internet.

Every node in an IP network has a globally unique IP address, which identifies the node. The data is transferred between these nodes by IP datagrams, which are routed to their destination addresses by other network elements, routers and gateways. The delivering of the packets to their destination is not guaranteed, rather the datagrams are transferred by best effort policy.

The IP protocol stack consists of five protocol layers, each of which play a different part in network connectivity. Here are the layers and their functions explained:

- **Physical layer:** This is the layer at which the physical data (bits) are transferred. The media can be for example an optical fiber or a cable.
- **Link layer:** The link layer delivers IP packets to the physical layer. Ethernet is an example of the link layer.
- **IP layer:** The IP layer routes packets to their destinations. It is not guaranteed that the packets will actually reach their destinations. This layer is basically responsible for the inter-connectivity of the networks that form the Internet.
- **Transport layer:** The transport layer offers higher level protocols to the protocol stack. Key protocols of this layer are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). TCP offers connection-oriented, reliable byte stream to applications. UDP is unreliable and connectionless protocol, and it basically offers a mapping of IP datagrams to applications.
- **Application layer:** Application layer operates on top of transport layer protocols, such as TCP or UDP. Examples of application layer protocols are Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP) and Session Initialization Protocol (SIP).

The network layers are ordered like shown in Figure 2.1. In the figure, upper layer always uses only the services provided by lower layers. For example, the application layer uses the transport layer to communicate with other peers in the

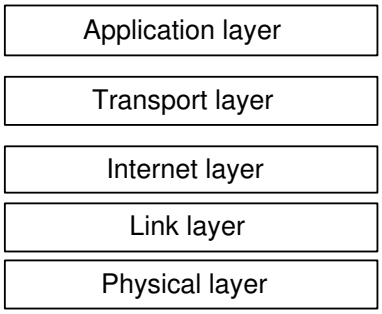

Figure 2.1: An example of IP protocol stack

network. So, layering provides an user-level end-to-end connection that hides the details of the network.

One fundamental idea in the design of the IP networks is that the users or application programmers do not need to know the details of the network architecture in order to use the network [4]. Figure 2.2 shows that two nodes can be connected to each other even if they are not in the same physical network. In the figure, Nodes 1 and 2 can connect each other. The actual data traffic might be routed directly between Net B and Net C, or it might go trough Net A. The actual route might even change without the nodes noticing it.

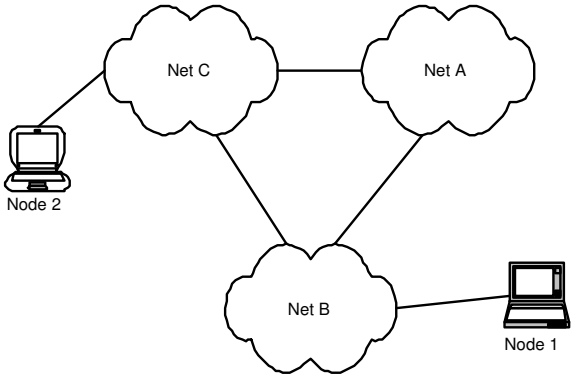

Figure 2.2: Connections in an IP network

2.2.2 GSM

Global System for Mobile telecommunications (GSM) is a second generation (2G) digital cellular network. The basic service of the GSM is telephony, offering international roaming and secured air interface. Other services offered by GSM are short messaging, fax and data services. From these additional services, short messaging has been the most successful.

Short message service (SMS) is a service that allows GSM users to send short messages up to 160 alphanumeric characters to an another subscriber, and includes a possibility to acknowledge the reception of the message. This service is widely used as a person-to-person communication service, and has proven to be useful at implementing other services. Mail notification services, information services and ringing tone delivery are examples of services that are enabled by SMS. In general, many of the current mobile services still rely on SMS service. This is mainly because SMS is part of the widespread GSM system and is supported by most mobile devices available.

These features have made GSM very popular in many countries and continents, and there were over one billion subscribers all over the world at the beginning of year 2004 [8]. The GSM subscribers can use their phones all over the world in more than 170 countries. The success does not seem to stop here, and the GSM keeps on growing as the technology advances.

Wireless application protocol (WAP) [23] is a protocol for delivering and presenting information on mobile devices. It provides a standardized way of providing Internet content to mobile phones. WAP can operate on top of Circuit Switched Data (CSD), General Packet Radio Service (GPRS) (See 2.2.3) or other bearers. The latest version of the protocol is WAP 2.0.

Multimedia Messaging Service (MMS) enables subscribers to send pictures, text, audio and video messages to each other. One simple possibility to use MMS is to take a photo and send it to a friend. The service providers too can use MMS to deliver information to subscribers. MMS has the potential to enable new services that have not yet existed in mobile networks. As an example of MMS service, subscriber could request a map of a specific area to be send to his/her mobile phone.

2.2.3 GPRS

General Packet Radio Service (GPRS) is a service that allows packet based traffic to be sent and received on existing GSM network. It offers higher speed data transfer compared to CSD or High-Speed Circuit Switched Data (HSCSD) (See table 2.3). GPRS is also called 2.5G, mainly because it offers some of those services that would be available in 3G, but it is still based clearly on 2G architecture.

GPRS is an "always-on" connection, as data can be send and received without having to initialize the session separately. Many new services are enabled by GPRS, including web browsing and email services. GPRS is useful in transferring content, and it can be also used to transfer MMS messages.

Enhanced Data Rates for Global Evolution (EDGE) technology enables even more higher speed data rates in GPRS network compared to standard GPRS.

Table 2.3 represents available bearer options in mobile networks.

Technology	**Theoretical bandwidth**	**Typical data rate**
Circuit Switched Data (CSD)	14,4 kbps	9,6 kbps
High-Speed Circuit Switched Data (HSCSD)	43,2 kbps	28,8 kbps
General Packet Radio Service (GPRS)	171 kbps	50 kbps
Enhanced Data rate for GSM Evolution (EDGE)	384 kbps	115 kbps
Wideband Code Division Multiple Access (WCDMA)	2048 kbps	144 kbps

Table 2.3: Mobile network bearers [16]

Using GPRS in a GSM network improves the potential ways of using the network. In the packet-switched GPRS network, it is possible to receive and send packets to/from terminals. Other services, including peer-to-peer networking services, can be implemented on top of this network.

One advantage of GPRS is that it allows users to contact external packet networks from their terminals. In practice, the mostly used external network is the Internet. This possibility increases the number of services that can be accessed by mobile

phones.

2.2.4 UMTS

Universal Mobile Telephony System (UMTS) is emerging technology that provides packed-based transmission of data. Theoretical data rates are significantly higher than in 2.5G networks, which creates better base for developing more sophisticated services.

The Third Generation Partnership Project (3GPP) is leading the development of the 3G mobile network specifications. 3GPP is an collaboration agreement between many organizational partners. At the time of writing, the parters are ARIB (Association of Radio Industries and Businesses), CCSA (China Communications Standards Association), ETSI (European Telecommunications Standards Institute), ATIS (Alliance for Telecommunications Industry Solutions), TTA (Telecommunications Technology Association of Korea) and TTC (The Telecommunication Technology Committee). [1]

> "The original scope of 3GPP was to produce globally applicable Technical Specifications and Technical Reports for a 3rd Generation Mobile System based on evolved GSM core networks and the radio access technologies that they support (i.e., Universal Terrestrial Radio Access (UTRA) both Frequency Division Duplex (FDD) and Time Division Duplex (TDD) modes). The scope was subsequently amended to include the maintenance and development of the Global System for Mobile communication (GSM) Technical Specifications and Technical Reports including evolved radio access technologies (e.g. General Packet Radio Service (GPRS) and Enhanced Data rates for GSM Evolution (EDGE))." [1]

UTMS, like 2.5G GSM network, has a packet-switched core network in addition to circuit-switched network. The improved data rates in UMTS using for example WCDMA (See table 2.3) allows using larger contents in the services. As a consequence, it is possible to transfer data, or even mobile applications, swiftly across the network.

In UMTS Release 5, IP Multimedia Subsystem (IMS) is the core of the network. Its handles call/session set-up and control and roaming, and utilizes IPv6. The signaling protocol used in call set-up and control is SIP (See section 2.5).

2.3 Mobile Device Platforms

The operating system of the mobile phone plays crucial part what comes to the phone features. The ability to install third-party applications to the device is particularly important in our application distribution scenarios. Here we go through the most important platforms and their aspects concerning this thesis.

Symbian is a major player in mobile operating systems business. It is used by mobile phone manufacturers including Nokia, Sony-Ericsson and many others. There are also other mobile operating systems and platforms, such as Palm OS and PocketPC that are widely used in mobile devices. In the scope of this thesis, Symbian is the most relevant of these platforms.

2.3.1 Symbian

Symbian OS is 32-bit multitasking operating system, designed specifically for wireless environments and constraints of the mobile phones (like limited memory).

Symbian based open platform allows third-party software to be installed and run on top of the operating system. A modified version C++ programming language is used in developing software for Symbian OS, and it provides a set of Application Programming Interfaces (APIs) for developers. This openness increases the number of available applications in contrast to closed operating systems.

Developers can use the provided communication and networking features using APIs. In version 7.0s, both TCP and UDP are supported, and on top of them, there is also support for HTTP, WAP, FTP and other application protocols [6]. These features are very useful in developing networking applications.

Series 60 is an example of smartphone software platform optimized for Symbian OS [14]. It has common user interface components and development tools for application developers. The screen size and number of functional buttons are ex-

amples of features defined in the Series 60 platform.

2.3.2 Java

It is estimated that almost every mobile phone will be running Java by 2006 [5]. This makes Java enabled phones very attractive when creating and developing applications to mobile phones. That is why in Symbian OS, Java runtime environment is supported in addition to native Symbian applications.

The Java 2 Platform, Micro Edition (J2ME) is designed for embedded devices, including mobile phones, PDAs, and TV set-top boxes, and it is currently deployed in millions of devices.

The J2ME architecture consists of configurations, profiles and optional packages. Figure 2.3 illustrates the relations between these concepts. J2ME platform, like other Java platforms, is a set of standard Java APIs [22]. The applications running on specific Java platform must use the APIs provided by that particular platform.

Virtual machine and minimal set of libraries compose configurations. In J2ME, there are two configurations, the Connected Limited Devices Configuration (CLDC) and the Connected Devices Configuration (CDC). The virtual machine in CLDC is called a Kilobyte Virtual Machine, KVM in short.

Profiles are higher level APIs, and combined with configurations, they provide the complete runtime environment. The profiles available in J2ME are Mobile Information Device Profile (MIDP), Foundation Profile, Personal Profile and Personal Basis Profile.

There may exist optional packages on top of configurations and profiles. Optional packages typically offer APIs for new and emerging technologies. These packages may include support for example for wireless messaging or Bluetooth.

A number of applications have already been written for J2ME. Many mobile games available today are implemented for MIPD compatible devices. Still, the games are usually only single player games. In personal computer world, multiplayer games have been very popular, and there is no reason why mobile multiplayer games could not be also successful.

Currently, in MIDP2.0 environment, it is possible to initialize HTTP or TCP con-

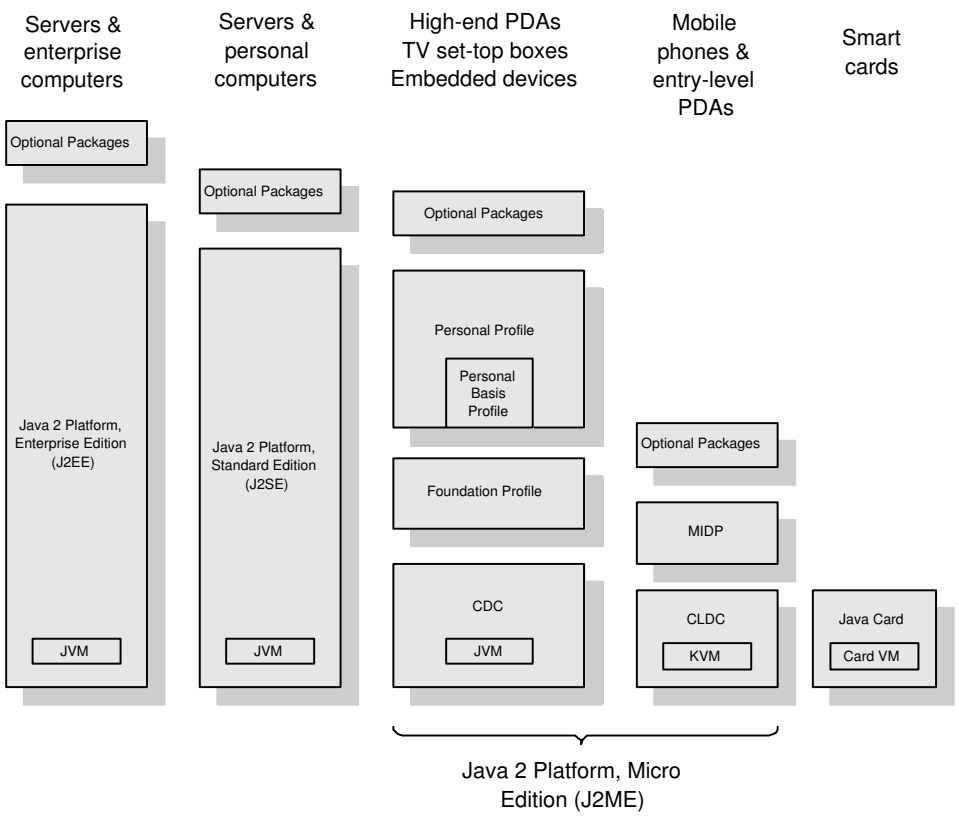

Figure 2.3: Java Platform [22]

nections and send SMS messages and UDP datagrams. With these, application developers have the tools to create multiplayer applications that use mobile networks.

Mobile Java also offers a security model that protects the mobile device from malicious applications. So-called sandbox model ensures that the device cannot access any native interfaces, only Java APIs can be used. Some platform operations, such as making a HTTP connection or making a telephone call, are possible, but it is the duty of the platform to make sure that the user has allowed such behavior. [12]

2.4 Current Application Distribution Practices

Many mobile terminals allow third-party applications to be installed and run on them. For distributing these applications, there are several methods that can be used. One alternative is to use the existing GSM network to transfer the applications to the terminals. All the distribution mechanisms are not equal, and the main features of current methods are explained in this section.

It is common practice to advertise mobile games and applications in medias, such as the web, newspapers and magazines. In the advertisements, the phone number where to send a specific SMS message is visible. The mobile user can then order the application by sending the SMS, and the delivery is done by method described below.

The application can be delivered using Over the Air (OTA) technology. After ordering, the server sends an SMS that indicates where the application can be downloaded. The downloading typically done by using GPRS connection. If the phone is intelligent enough, the only thing the user has to do is open the received SMS message and the downloading and installation begins automatically.

Before there was a way to use WAP or GPRS to download applications, data could be sent using chained SMS messages. This method is not very convenient because of limited amount of data and the delays in the SMS delivery. However, it can be used to transfer small amounts of data, for example ringing tones, but is not really usable for larger applications.

The WAP browsing and Internet connectivity make it possible to initialize application distribution process with a browser. This suits distribution purposes well, if there is no need to charge each download. Usually, many application providers demand revenue from their applications, so SMS initialized application distribution suits them and the operators better.

Some applications are available on MMC (MultiMediaCard) cards. For example, Nokia N-Gage mobile gaming device allows users to buy games on MMC cards. The distribution of these cards is similar to traditional PC- or console game distribution, the games can be bought from shops and stores. However, this is quite slow compared to mechanisms that use network technologies in the distribution process.

Lastly, there are ways to distribute the applications using short range proximity technologies, like Bluetooth or infrared. These methods are practical if the applications are available on a personal computer or a laptop with a Bluetooth or infrared support, and the mobile device supports these technologies. The user can install mobile applications downloaded from the web directly to the mobile device. The downside is that there is no evident way to implement billing with these technologies, if the applications can be downloaded freely from the Internet.

In general, there are ways to distribute the applications so that the charging can be used. These methods of application distribution suit the application providers and the operators better that the methods where there is no possibility to use charging. The users are likely to prefer methods that are more cheaper of more convenient, so from their point of view the free distribution mechanisms are rational choices.

2.5 Session Initialization Protocol

SIP [18] is an application level protocol for initializing and managing multimedia sessions. It can operate on top any transport level protocol, for example UDP or TCP. This protocol will be one of the signaling protocols in 3G networks, and it is publicly developed by IETF.

Generally, SIP can be used to establish and tear-down various kinds of sessions, including Voice over IP (VoIP) sessions and multimedia game sessions. It can also be used to implement presence and instant messaging applications and services.

SIP is text-based protocol, like HTTP, and it has same kind of request/response transaction model. The requests are answered with responses, and each response has a numeric code that has a specific meaning. In SIP, one request can result in multiple responses. Both requests and responses can contain a body, and the bodies are handled according their type.

The actual parameters of the session are transferred in the bodies of SIP requests. Session Description Protocol (SDP) [9] is generally used to describe the session parameters, but other session description protocols could also be used. The next generation of SDP protocol is currently under development by IETF.

The SDP does not limit the usage to only voice connections, rather it can be used to

open various kinds of multimedia sessions, including video conferences and game sessions. The SDP messages include such data as application type, protocols, port numbers and codecs.

SIP can be used to initialize multiplayer mobile gaming sessions. The parties involved must have the game installed in their SIP capable mobile terminals. Typically, only the session set-up and termination is done with SIP, and the traffic between applications uses other methods of connectivity, like TCP or UDP connections. The session connectivity methods session are defined in the SDP bodies.

SIP can be used in a network that has a SIP architecture, which consists of several entities. The key entities are explained in short in the following list:

- User Agent (UA) is an entity that is capable of creating and sending SIP requests, and can generate and send responses to incoming requests. UA can be thought as a piece of software running on a computer, mobile terminal or logical entity.
- Proxy is a SIP server that is capable of forwarding requests. Stateless proxy simply forwards all incoming requests, whereas stateful proxy maintains the state of the call.
- Registrar is a server that accepts registration requests, and stores the registration information for location services.
- Location service knows the locations of the callees. Proxies or redirect servers can use this information and route incoming requests accordingly.
- Redirect server is a server that directs the client to contact alternative URIs.

SIP users are identified with SIP Unified Resource Indicators (URIs). These URIs typically contain a username and a hostname, in the similar form to email addresses. This helps to locate the end system where to contact when initiating a session.

SIP also provides knowledge about the user availability. In practice, user can decide whether he/she wants to be available for specific type of requests, like voice calls or instant messages. This is done with REGISTER requests. These requests

inform the registrar server about the availability of the SIP entity involved. This is very useful in mobile gaming too, since user can announce when he/she is available for gaming sessions.

The characteristics of SIP according to Wisely et al. [24] are:

- **Simplicity:** SIP is designed to be a lightweight protocol. SIP has only a couple of compulsory headers and request types. This enables SIP to be used in resource-limited devices also.
- **Generic session description:** In SIP, the signaling part is separated from the actual session description, allowing SIP to be used in various types of sessions.
- **Modularity and extensibility:** SIP is an extensible protocol. Extended protocol requests can still be handled like plain SIP protocol. For example, new extensions have been defined to be used in an IMS network.
- **Programmability:** The programmability allows the protocol to be intelligent what comes to relaying and altering messages. SIP servers can re-direct or copy messages when needed. Plain-text protocol can be handled using scripts or using some other high-level languages.
- **Integration with other IP component technologies:** Other IP protocols have influenced in the design of SIP. SIP complements other IP protocols like Real Time Streaming Protocol (RTSP).
- **Scalability and robustness:** Scalability is achieved by making SIP servers stateless. Stateful proxies exist, but are typically designed for advanced services.

These characteristics and features make SIP potential for mobile devices and networks also. In addition, the various SIP extensions allow SIP to be used for application distribution purposes, as we can see in Chapter 4.

Table 2.4 shows an example of typical SIP INVITE request message with a SDP body. The INVITE method is used in initiating sessions, and typically results in call flow sequence presented in Figure 2.4 in a case of successful session setup.

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: ...

v=0
o=alice 2890844526 2890842807 IN IP4 10.47.16.5
s=Session
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Table 2.4: Example of a SIP INVITE request

The figure shows a typical case of two proxies, Proxy 1 being in Alice's domain and Proxy 2 in Bob's domain.

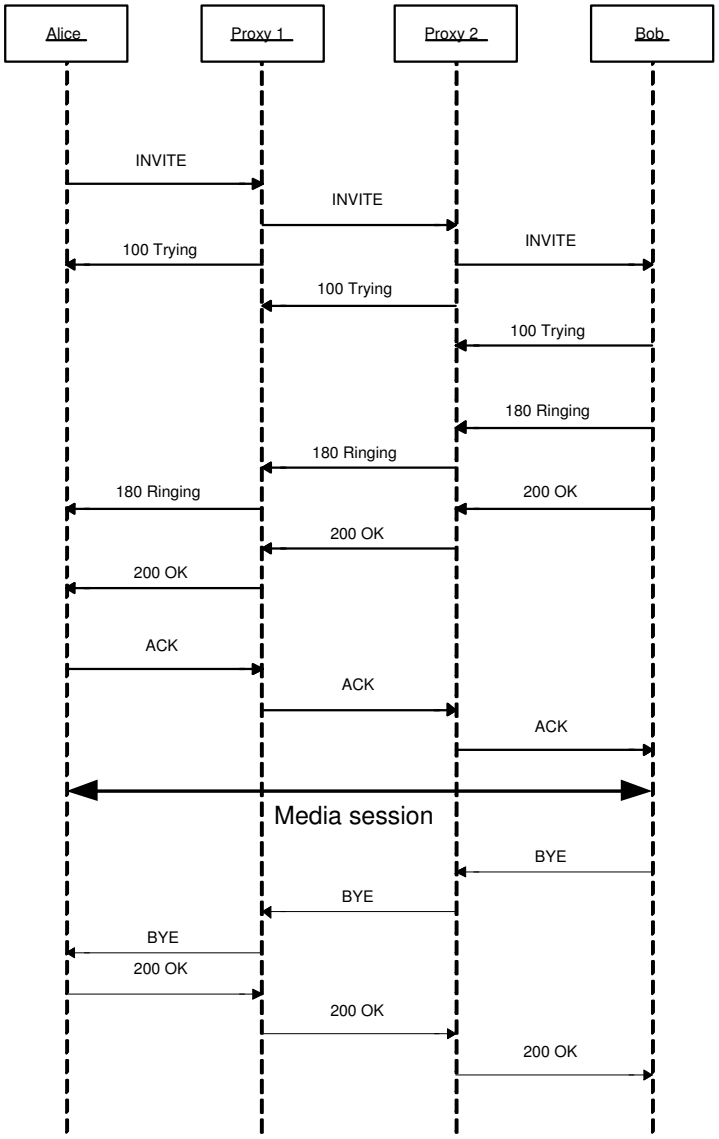

Figure 2.4: Basic SIP signaling

Chapter 3

Requirements

This chapter presents the requirements for the implementation of peer-to-peer application distribution system. The requirements are mostly based on usage scenarios, which help to understand the problem at hand. The requirements itself form the basis for building the demonstration environment.

3.1 Example Usage Scenarios

3.1.1 User Initiated Application Distribution

Consider two mobile phone users, Bob and Alice, to be average mobile phone users who happen to like playing computer games (and chess). Bob is very enthusiastic about playing chess, so he has found a good two player chess application program for his mobile device at a web site. He downloads the application to his personal mobile terminal by sending a SMS message to the number given at the web page and downloads the program using WAP browser.

After that, Bob makes a phone call to Alice and tells her about his new chess application. He also wants to try the application at once, so he tells Alice that he can distribute the application using mobile network.

The call is ended, and Bob opens up the chess application. He selects “Distribute application” from the chess application menu and sets Alice to be the receiver. Alice is expecting the distribution request to pop up on her terminal screen, and

when she sees the request, she then accepts it.

Application is then distributed to Alice's terminal. After the distribution process is over, Bob starts the chess playing session from the chess application menu with Alice. When Alice accepts the invitation, the game begins.

3.1.2 Automatic Application Distribution

The initial state is same as in the previous scenario. Bob has got a brand new chess application program which he would like to try with Alice. He selects the Alice's name from his contact list and selects "Play chess" option from his mobile phone, assuming that Alice already has it. But Alice does not know anything about the chess application Bob uses, so her terminal rejects the chess session automatically, and Alice sees no activity on her terminal.

The application in Bob's terminal receives the negative response and automatically asks Bob if he wants to distribute the chess software to Alice, because she does not have it. As Bob is very interested in playing the game, he accepts the distribution with a single press of a button.

Alice's terminal rings, so she looks at it and sees the following message: "Bob <bob@x.com> wants to distribute application 'Chess' to you, accept?". Alice accepts the distribution, and the installation process starts. When the application is installed, the chess session is automatically initiated again, and this time Alice is able to accept the invitation. The game session then begins.

3.2 Requirements

The requirements are divided to following categories, representing different views to the system.

- **Terminal requirements.** These requirements affect directly to the functionality of the mobile terminal, and to the functionality of the software running on the terminal.
- **Network requirements.** All requirements that are related to network com-

ponents or their characteristics belong to this category.

- **User requirements.** User requirements are the functionality that the terminal user wants. This category also includes usability requirements. Also the characteristics of the application programming interface offered to application developers belongs to this category.
- **Operator requirements.** The requirements that cellular network operator has regarding application distribution in its network.

The following requirements should be used as guidelines in application distribution system. In the list below, the requirements are followed by rationale of the requirement. Shorter descriptions of the requirements are listed in Table 3.2.

Terminal requirements:

Requirement 1: Mobile terminal must support the ability to install third-party applications. Otherwise the terminal user will not be able to run the application.

Requirement 2: Installing third-party software to the terminal must be safe. The application must function like it is expected to, without compromising the users privacy or terminal security.

Requirement 3: Installed third-party software must be able to use network services. This includes the ability to act as a server and a client. The platform must not restrict the functionality (for example, network access) of the installed software.

Requirement 4: The distribution process must handle different software versions correctly. Terminal must know which version of the application, if any, is already installed on the device. Upgrading a software version must be possible.

Network requirements:

Requirement 5: The network must support direct connections between peers. The network operator could block direct connections, thus disabling peer-to-peer application distribution. If only some type of direct connections are enabled (maybe restricted by protocol type or port number), the implementation can use those as "carrier media" for the actual content, if possible. The connection must be bidirectional, so that either of the peers can act as a server and a client when needed.

In any case, establishing and managing direct peer-to-peer connections must be possible.

User requirements:

Requirement 6: Terminal user must be able to decide whether the offered application is installed or not. User interaction should be minimal, simple yes/no -dialog is better than complex one.

Requirement 7: Originating user must be able to decide when the application distribution is possible. For example, if the application distribution is started automatically after session initiation fails, the user still has to accept or reject the distribution process explicitly. In short, the applications can not be distributed without users acceptance.

Requirement 8: The application distribution feature should not complicate the application development process. If the application developer needs to use a specific distribution API when developing software, the API must be lightweight and simple to use.

Operator requirements:

Requirement 9: From the operator side, the ability to control and charge the application distribution is important. There must be way to log the distribution act to some storage for these purposes.

#	Requirement type	Requirement	Description
1	Terminal	Application installation	User has to be able to install third-party software on the terminal.
2	Terminal	Security	Distribution of the applications must be secure.
3	Terminal	Networking support	The third-party software in the terminal must be able to establish connections to network.
4	Terminal	Version control	Terminal needs to be aware what version of the software is already installed.
5	Network	Session initiation	There must be a way to establish and manage session between end users.
6	User	Installation acceptance	User can accept or reject incoming distribution request.
7	User	Distribution acceptance	Originating user decides when the distribution is possible.
8	User	Simple APIs	Developing applications supporting distribution must not be too complicated.
9	Operator	Charging and logging	Operator must be able to log application distribution transactions and charge for the application transfer.

Table 3.2: Requirements

Chapter 4

Implementation

This chapter describes the implementation of peer-to-peer application distribution and gives detailed information about the underlying architecture. We will examine both GSM and UMTS architectures to be able to understand how peer-to-peer connections in general are possible in present and future mobile networks.

4.1 GSM Architecture

The mobile terminals use mobile networks, like GSM network, to communicate with other mobile terminals and network servers. The 2.5 GSM network architecture offers services like voice connections, sending and receiving of SMS messages and GPRS data connections.

GSM network consists of two broad parts: Radio network and core network. Simplified picture of the GSM network architecture is presented in Figure 4.1. The radio network, or Base Station Subsystem (BSS), controls the link to the the mobile station (cell phone).

The core network, or network subsystem, performs the switching of the calls between mobile phones. It also switches calls to other networks, like PSTN. The management of the mobile services (like authentication) is on the responsibility of the core network too.

The functionalities of most important core network components are explained in

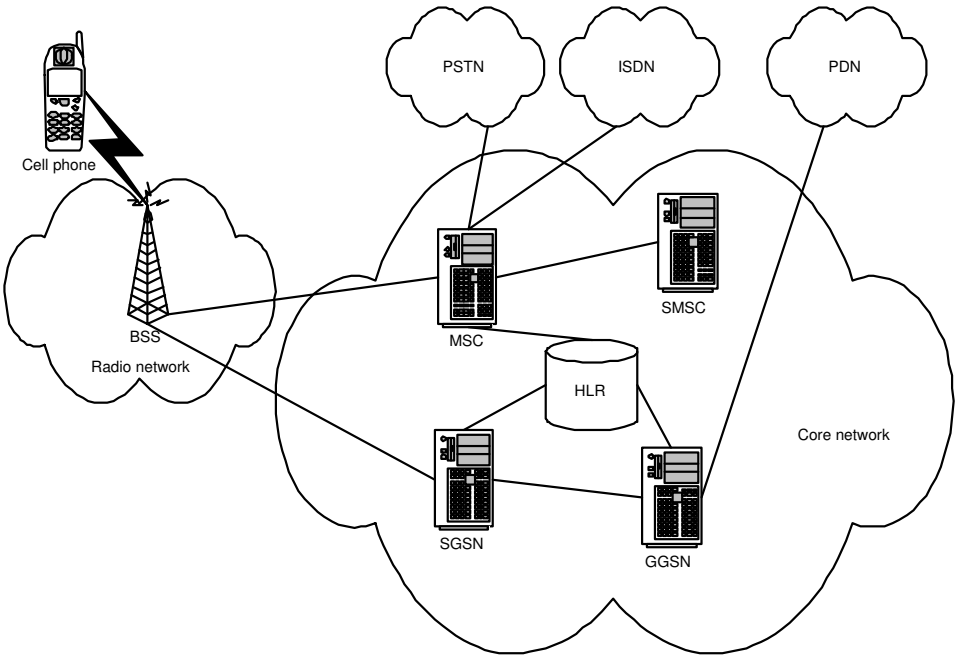

Figure 4.1: GSM architecture

the following list: [3], [21]

- Mobile Switching Center (MSC) is the central component of the whole network subsystem. The main responsibility is to switch calls, but it also includes all the functionality needed to handle mobile subscribers, including registration, authentication, handovers and call routing.
- Home Location Register (HLR) contains information of each subscriber registered in the network, and their current locations.
- Short Message Service Center (SMSC) handles the storing and the delivery of SMS messages between mobile phones.
- Serving GPRS Support Node (SGSN) does the delivery of data packets from and to the mobile stations, and stores subscription information and location information.
- Gateway GPRS Support Node (SGSN) is a component that acts as an interface between the GPRS network and external packet networks. It also stores

subscription and routing information for subscribers that have at least one active Packet Data Protocol (PDP) context.

GPRS packet data traffic goes trough the GPRS Support Nodes. PDP context is the connection for tunneling packets between the mobile terminal and Packet Data Network (PDN). In short, the mobile terminal creates a PDP context in order to access PDN networks like the Internet.

4.2 UMTS Architecture

In this section, we will present two versions of the UMTS architecture, Release '99 and Release 5 (R5). Both are specified in 3GPP, and R5 is the newer of these releases.

4.2.1 UMTS Release '99

UMTS Release '99 network consists of three components: UMTS Terrestrial Radio Access Network (UTRAN), Circuit-Switched Core Network (CS-CN) and Packet-Switched Core Network (PS-CN). The CS-CN can be seen as an evolution from GSM core networks and the PS-CN as an evolution from GPRS core networks. In addition, there is W-CDMA air interface between user equipment and the radio access network. The core network is independent from the access network, so in theory, the access network can be UTRAN or Wireless LAN.

The CS-CN is responsible for supporting connectivity to Public Switched Telephony Network (PSTN) and Integrated Services Digital Network (ISDN), thus it provides traditional telephony services. It has also support for SMS and circuit-switched data services.

Packet-based connectivity is done by the PS-CN. It handles the connectivity to packet-based networks like the Internet. Virtual Private Network (VPN) and SMS are examples of services supported by PS-CN [15].

Figure 4.2 represents the architecture of UMTS Release '99. Here are brief explanations of the most important components in UMTS core network, mostly based on Wisely et al. [24]. These components form the base of UMTS Release '99.

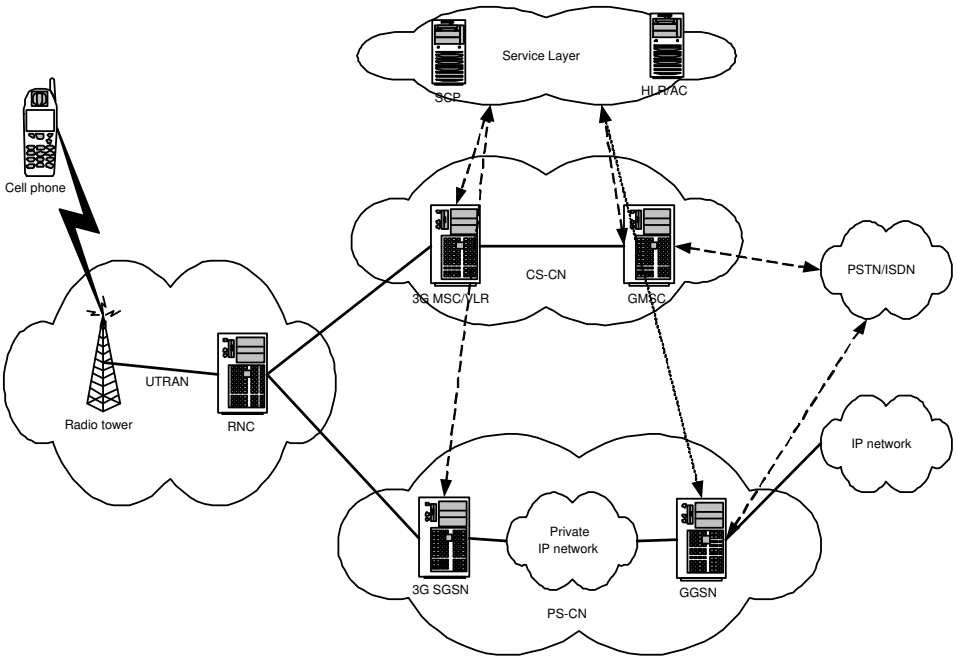

Figure 4.2: UMTS Release '99 Architecture [15]

- Mobile Switching Center (MSC) is responsible for circuit set-up and mobility management in the circuit-switched domain.
- Gateway Mobile Switching Center (GMSC) is a gateway for incoming and outgoing connections to external networks in the circuit-switched domain.
- Home Location Register (HLR) is a database consisting of user data, such as subscription information and available service data.
- Visitor Location Register (VLR) contains such selected user data from HLR that is necessary for call control and billing in the geographical area controlled by VLR.
- Authentication Center (AC) supports authentication functions for both circuit-switched and packet-switched domains.
- Serving GPRS Support Node (SGSN) is responsible for session management and producing charging information. It routes packets to Radio Network

Controllers (RNCs) in UTRAN. It takes care of security and authorization also.

- Gateway GPRS Support Node (GGSN) is basically a gateway or router, and its function is to forward traffic to other mobile networks or to the Internet. It also contains a firewall that restricts unwanted traffic, and can produce charging information too.

As can be seen above, the Release '99 GPRS components are similar to 2.5G GSM network components.

4.2.2 UMTS Release 5

Release 5 version of the UMTS network is presented next. R5 architecture defines a new subsystem to the PS-CN known as the IMS. IMS is the core of the packet-switched domain of the R5 UMTS. R5 proposes to offer packet switched services and traditional telephony services in this single converged network, IMS.

R5 will allow voice calls to be established using SIP as the signaling protocol. SIP messages are routed by Call Session Control Function (CSCF) network elements. Figure 4.3 shows the IMS architecture, and how the different network components are connected together.

There are many CSCFs in the IMS. All of them can be thought as different types of SIP servers, and each of them have specific responsibility area that they take care of. The list below explains the functionalities of the IMS CSCFs. Or course, the IMS has components that are not SIP servers, as can be seen from the list.

- Proxy - CSCF (P-CSCF) is the first contact point to the visited IMS network for the mobile equipment. P-CSCF acts as a SIP proxy in SIP architecture. This component has two main functions: it handles Quality of Service (QoS) and provides local control for emergency services.

- Interrogating - CSCF (I-CSCF) is an optional node in IMS network. It basically finds and contacts home network from visited network. It can also handle load balancing and billing operations.

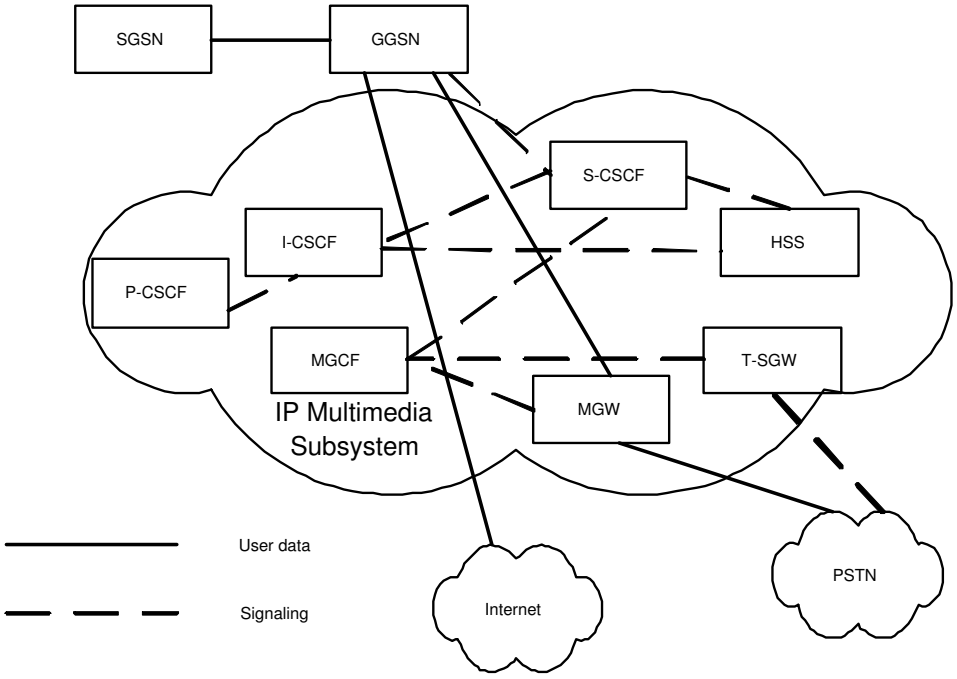

Figure 4.3: IP Multimedia Subsystem [15]

- Serving - CSCF (S-CSCF) performs the session management in the IMS. Usually the S-CSCF of the home network is responsible session control, but this duty can be transferred to P-CSCF in the visited network if needed.
- Home Subscriber Server (HSS) is an evolution from Rel '99 HLR. It acts as a centralized subscriber database, and provides subscription information to P-CSCF and I-CSCF.
- Media Gateway (MGW) does conversions between medias formats, for example from PCM (Pulse Code Modulation) to IP based format.
- Media Gateway Control Function (MGCF) controls Media Gateways. This includes converting different signaling protocols, for example from SIP to ISUP (ISDN User Part).
- Transport Signaling Gateway (T-SGW) is a component that provides compatibility with PSTN signaling.

It is intuitive to use packet-switched domain to transfer packet-based data. This thesis focuses on the packet-switched domain, where both data transfer and signaling can be done.

4.3 Peer-to-peer Connections in Mobile Networks

GPRS allows users to make outbound data connections in cellular networks. Using this feature, mobile users can download software application from network servers and install the downloaded applications. Inbound connections are more troublesome. For example, the billing mechanism, where mobile subscriber pays for sent and received traffic, is problematic if the user cannot control the inbound traffic, or is not aware of it.

Direct peer-to-peer application distribution requires a network that allows connections to be created directly to other terminals. This is also a requirement for peer-to-peer applications to function. There should also be a possibility to establish sessions between networks of different operators, to allow more subscribers to create peer-to-peer connections.

Direct peer-to-peer application distribution is most likely to be used with multiplayer applications that utilize peer-to-peer connections themselves. For example, a chess game could use a peer-to-peer connection to exchange game information. Therefore it might be possible to use same signaling channel to initialize application downloading and installation processes.

In current 2.5G mobile networks, peer-to-peer connectivity is quite limited. Voice connections or SMS messages are peer-to-peer connections, but they are not suitable for transferring large amounts of data. In GPRS, direct connections to mobile terminals are usually blocked for security reasons. The operators do not want mobile terminals to be vulnerable to denial of service attacks, for example. Also the billing problem mentioned above is a major factor.

Many operators use Network Address Translation (NAT) [7]. NAT enables a LAN to use larger address space for internal traffic than for external traffic. The addresses used in the LAN are different from the ones used in external networks. IP address depletion and scaling problems in routing are the main reasons for using

NAT, and it has gained popularity when the Internet has expanded. Technically, the address translation is done by a NAT box. Unfortunately, using NAT makes it impossible to access arbitrary node in a local network from external network. So, if a network operator uses NAT to get more address space for the mobile terminals, the terminals cannot access each other if they are not in the same network.

The demonstration environment used in this thesis provides a 2.5G based network with SIP support and a capability to set up peer-to-peer connections. The IMS R5 has SIP as signaling protocol, but there is not any IMS network available for demonstrating purposes at the time of writing, therefore an 2.5G network with a SIP proxy is used. The NAT problem is not addressed, in the demonstration will work only when both mobile terminals use the same network.

The demonstration network described above offers packet-switched connectivity between terminals, and in that sense it functions like UMTS network. This relies on assumption that the GPRS is very similar to UMTS packet-switched core network in terms of functionality. The demonstration network is actually a proprietary test network. The usage requires an access point specific to the operator, and proper access rights.

Figure 4.4 shows what paths the signaling and media traffic use in R5 network. As we can see, the signaling path differs from the media traffic path. The signaling goes trough CSCFs (SIP servers), and the traffic goes directly from GGSN to the destination network. The destination network can be another R5 network, a legacy GSM network or any IP network.

R5 networks use IPv6 in their networks. In IPv6, the address space is not a problem, so it is likely that operators will not use NAT in their R5 UMTS networks. But still, operators may choose to restrict the traffic by firewalls, and so restrict also peer-to-peer traffic.

4.4 Using SIP REFER Method in Application Distribution

In previous sections we saw how underlying mobile networks offer capabilities to route traffic between terminals. Next, we investigate more closely the actual

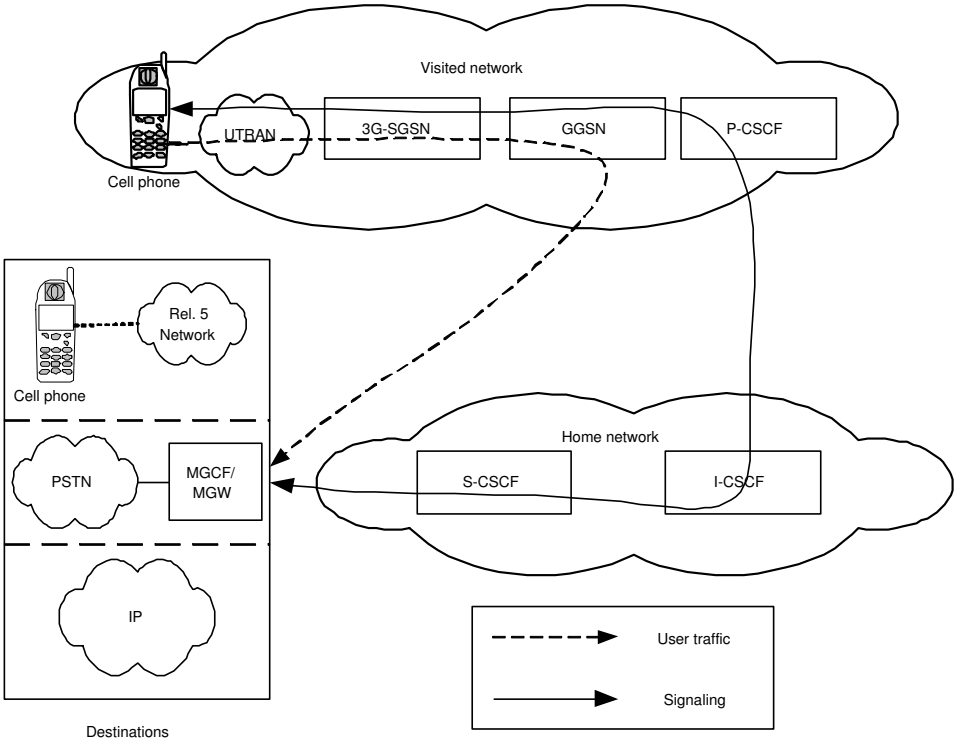

Figure 4.4: Signaling and traffic paths in IMS network [15]

protocols involved in the signaling and content transfer of the distribution process.

4.4.1 The SIP REFER Method

One way to implement application distribution is to use SIP REFER requests in the signaling process. The REFER extension is defined in RFC 3515 [20]. Like its name hints, its purpose is to refer resources to the terminating party. Based on the success of the refer operation, the referred party sends notifications based on the outcome of the refer activity. This can be used in several applications. For example, call transfer can be implemented using the refer mechanism.

REFER is also used in IMS R5 SIP extensions. In general, SIP signaling and the extensions used in 3G networks is defined by 3GPP Technical Specification Group. [2]

The REFER method contains contact information which the receiving party uses

to contact the referred third party. The contact information is transferred using single “Refer-To” header field in the REFER request. This header contains an URL, which can be for example another SIP address or an HTTP address. The receiver of the REFER request interprets this header, and takes necessary actions. The receiver has the possibility to accept or reject the REFER request. A correct response is send according to the decision.

The REFER request may also have a body. The receiver can then process the content according to its content type. By default, there is no body in the REFER request.

If the “Refer-To” URL is accepted by the terminating party, a 202 Accepted response must be send, and the terminating party must try to access the referred URL and report the status of the referring process to the originating party. Also a NOTIFY request is send immediately after the accepted response has been send.

A successful REFER request establishes a subscription to the refer event. This means that the referrer subscribes to notification events from the referred party, and after subscription the referred party sends the notifications according to the subscription. The subscription events are specified in RFC 3265 [17].

When user or terminal software rejects the REFER request, the subscription is not established, and the referred resource is not accessed. The request is responded by a response with an error code.

The notifications are done with NOTIFY requests. In the REFER case, the NOTIFY requests have a body of type “message/sipfrag”. This assumes that the referred resource is a SIP resource, but as the RFC 3515 defines, the resource can be an HTTP URL as well.

A SIP request with a body of type “message/sipfrag” contains a SIP or an HTTP message. At least the first line of the SIP or HTTP message must be included in the body. This information is interpreted in the originating side to find out what is going on on the terminating side.

4.4.2 Initiating Application Distribution with SIP REFER

Figure 4.5 represents the typical SIP message flow used in the the application distribution signaling. Here, User A is the party that wants User B to have a particular application, which User A has already installed. Therefore, User A is the originating party, and User B is the terminating party.

1. User A sends a REFER request to User B. The request contains a Refer-To -header field which has the HTTP resource location of the application to be distributed. The resource is in this case the distributable application package which is located at the terminal of User A. Table 4.1 shows the REFER request in detail.

   The body of proprietary content type "application/install" is used so that the receiving end can identify the REFER request, and determine that it is used in application distribution.

```
REFER sip:mpeer2@10.128.0.12 SIP/2.0
From: "Multipeer 1" <sip:mpeer1@10.128.0.12>;tag=143055456
To: sip:mpeer2@10.128.0.12
Contact: sip:mpeer1@10.128.0.169:5060
Call-ID: 1742844642@10.128.0.169
CSeq: 1 REFER
Max-Forwards: 70
Via: SIP/2.0/UDP 10.128.0.169:5666;branch=z9hG4bK1742844642
Refer-To: http://10.128.0.169/BattleShips.jad
Content-Length: 18
Content-Type: application/install

BattleShips v. 1.0
```

Table 4.1: Sample REFER request

2. User B sends 202 Accepted response to A, indicating that the refer request is going to be handled. This is a preliminary acceptance, the user is not asked to accept the distribution at this point. This is because we want to answer

the REFER request as soon as possible, and asking the user would delay it unnecessarily.

3. User B sends a NOTIFY request to A. The request contains information in the body of the request. The information here is of type "message/sipfrag" and contains "100 Trying" message. This means that User B is trying to use the resource that was indicated by the initial REFER request. Table 4.2 shows the NOTIFY request in detail.

```
NOTIFY sip:mpeer1@10.128.0.12 SIP/2.0
From: "Multipeer 2" <sip:mpeer2@10.128.0.12>;tag=449288394
To: "Multipeer 1" <sip:mpeer1@10.128.0.12>;tag=143055456
Contact: sip:mpeer2@10.128.0.170:5060
Call-ID: 174284642@10.128.0.169
CSeq: 1 NOTIFY
Max-Forwards: 70
Via: SIP/2.0/UDP 10.128.0.170:5060;branch=z9hG4bK4785324626
Event: refer
Subscription-State: active
Content-Length: 20
Content-Type: message/sipfrag

SIP/2.0 100 Trying
```

Table 4.2: Sample NOTIFY request

4. User A responds to the NOTIFY request with a 200 OK response.

5. User B requests the application from A with HTTP, as soon as User B has accepted the distribution process to begin. The user B still has the option to reject or cancel the distribution at this point.

6. The actual content containing the application is transferred to User B's terminal. The content is delivered in the HTTP response. When the transfer is complete, the terminal installs the downloaded software package.

7. After the content is transferred and application is installed, User B sends again a NOTIFY request, this time the request indicates that the application

has been installed in B's terminal. The content of the request is "200 OK" of type "message/sipfrag".

8. Again, User A responds to the NOTIFY request with 200 OK response.

After these steps, we see that the application was distributed successfully from User A's terminal to User B's terminal, and User A knows that User B has installed the application.

If, for some reason, the referred resource cannot be accessed, the terminating side can send a terminating NOTIFY with proper error code in the body. For example, "503 Service Unavailable" can be used if the reference failed, or "603 Declined" if the user declined or canceled the install process. In both cases, the distribution fails, and the process is terminated.

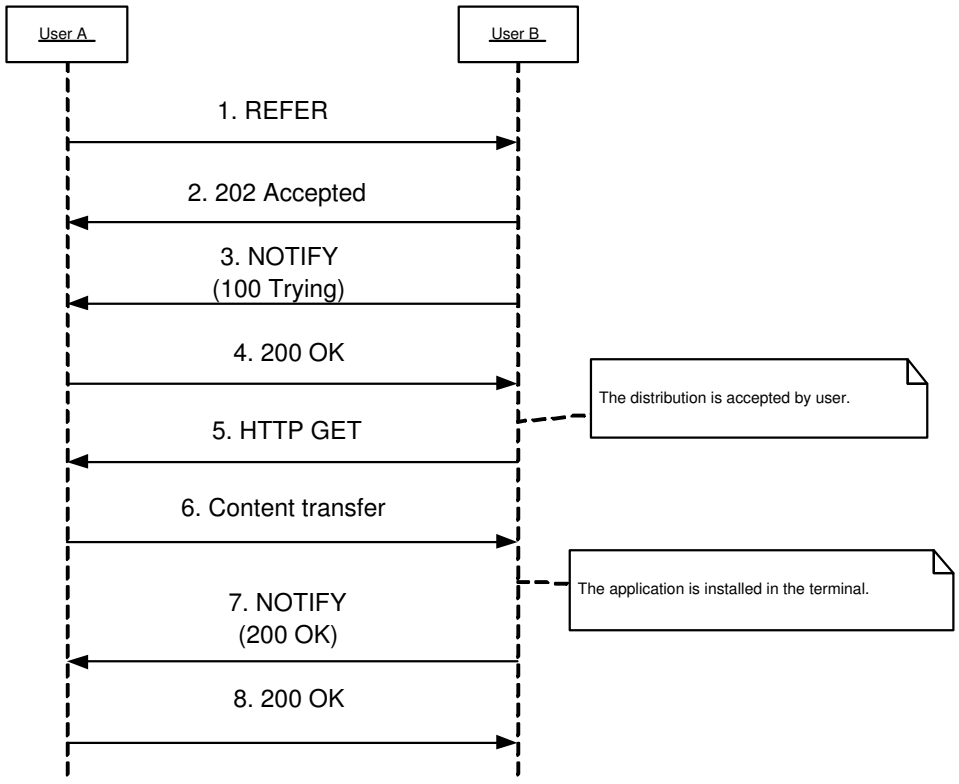

Figure 4.5: The message flow in application distribution scenario

Next we examine the software functionality of both sides. In peer-to-peer fashion, every mobile terminal must be able act as originating or terminating side. But, when it comes to one individual distribution act, the sides can be named as originating and terminating side.

Originating side

The originating side of the distribution process is responsible for sending the initial REFER request. It also parses incoming NOTIFY requests and holds the state of the distribution process.

In our case, the application to be distributed is a Java application that runs on top of MIPD2.0. The application uses specific Application Distribution API (later Distribution API, see section 4.7) that is build on top of Java SIP API. The Java SIP API is an implementation of JSR180 (Java Specification Request 180).

In theory, the Java application uses the distribution API provided by the platform to enable the application distribution. But in our implementation, the actual implementation of Distribution API is included in the Java application.

The originating side acts as a HTTP server in the process, and therefore it has software running that has the required functionality to process HTTP requests. SmallServ [13] is a free symbian HTTP server, designed to run on Nokia Communicator. The source code of the SmallServ is freely available. In this implementation, a modified version of the SmallServ acts as HTTP server. The modifications were necessary to enable SmallServ to be run on Nokia 6600 mobile phone.

A J2ME application consists of a jar-file and a jad-file. The jad-file is Java Application Descriptor, and it describes the contents of the jar-file, and may include application specific definitions needed to run the application. When distributing Java applications, HTTP server needs to have access to both of these files in order to provide the application to other parties.

Figure 4.6 shows the architecture on Java side. HTTP server is not in the picture, as it is merely a standalone software application.

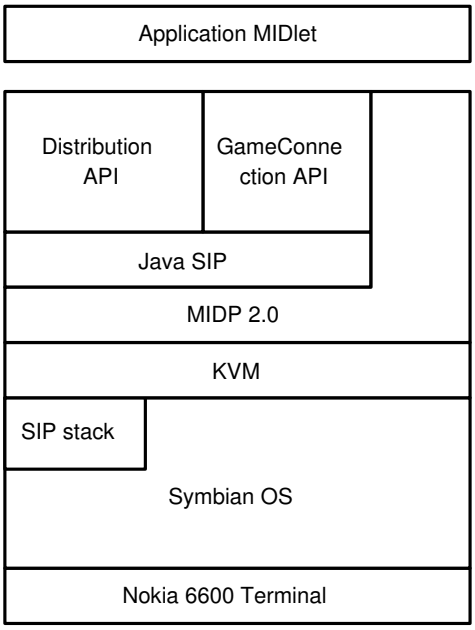

Figure 4.6: Originating side software architecture

Terminating side

In the terminating side, we have a application that receives the REFER requests and sends the NOTIFY messages to the originating side. This application is a native Symbian application called the Installer component. The SIP communication with the originating side is done with the native Symbian SIP stack. Figure 4.7 shows the architecture of the terminating side.

Terminating side has to download and install the software package. In our implementation, the application descriptor file (jad) is downloaded first by the Installer-component. The descriptor contains the URL of the jar-file. The jar-file is downloaded from the address in the URL. In practice, the URL contains the address of the originating mobile terminal and a path to the file. This means that the jad-file must be modified locally in order to ensure that the URL is valid.

Of course, it is up to the terminating party to install the downloaded application. The Installer component installs the application and is responsible for informing the status of the installation to the originating side also. The actual installation

process is explained in section 4.6.

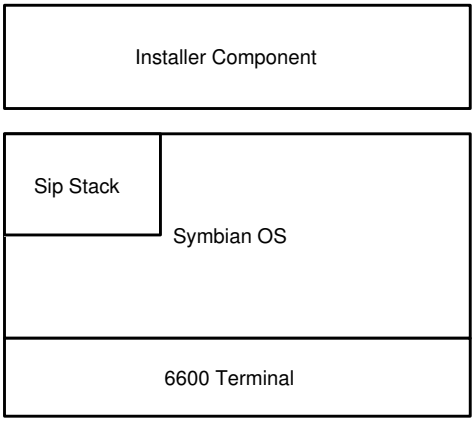

Figure 4.7: Terminating side software architecture

4.5 SIP Message Dispatching

It is important to know how the mobile terminal handles incoming SIP requests. As there may be several SIP applications installed on the terminal, there must be a way to forward incoming requests and responses for correct applications. The message dispatching is the process where it is decided which software component, if any, handles the incoming message. Usually, the SIP stack is responsible for message dispatching in the terminals.

The SIP stack has a registry of all installed applications which support SIP. The SIP stack can then send correct messages for correct applications. The registry is updated when new application that supports SIP is installed, or when such an application is removed. The SIP stack used in this implementation updates the database based on XML-file that describes what types of SIP messages it can handle.

Figure 4.8 shows how the incoming SIP messages are dispatched. The message dispatching is done according to the Content-Type of the incoming message, or m-line of the incoming SDP body. Content-Type is one of the headers of the SIP message, whereas m-line is a header of the SDP body.

In the unsuccessful INVITE case, the SIP stack itself responds with “488 Not ac-

ceptable here" response, which indicates that the terminal is not capable of handling the media. This can be interpreted so that the terminal does not have the correct application installed.

In the successful REFER case, the REFER request is dispatched to the Installer component. Installer the sends NOTIFY requests and receives responses for them trough the SIP stack.

The successful INVITE represents the case where the session is actually established. The correct application for the INVITE is now found in the SIP stack registry. The game data is UDP-based, so it does not go trough the SIP stack.

The applications do not necessarily need to be running to be able to receive SIP messages. The SIP stack can start the needed application when it receives the correct message. This feature is called the push mechanism, and it is very practical in resource-limited devices, as unnecessary applications do not need to be running continuously in the background.

4.6 Application Installation

The underlying mobile device platform on Nokia 6600 phone is Symbian 7.0 with Series 60. In Series 60, it is possible to let the platform decide how specific files are handled. This helps to invoke the native symbian application installer functionality needed to actually install the downloaded files, so that the application can be run like any other application on the particular mobile device.

J2ME applications are usually in two parts: in a jad-file and a jar-file, where jad is the file that has description of the jar-file. Native Symbian applications are packed in one sis-file. The native application installation functionality can process both Java and Symbian applications.

When the application is installed, it needs to be registered to the SIP stack. The native install process is responsible for doing that. Registering SIP capable Java applications is not supported in the standard Symbian OS, so a special version of the OS is used to enable this. In addition, the used native Symbian SIP stack is an add-on to the OS, not a part of the OS itself.

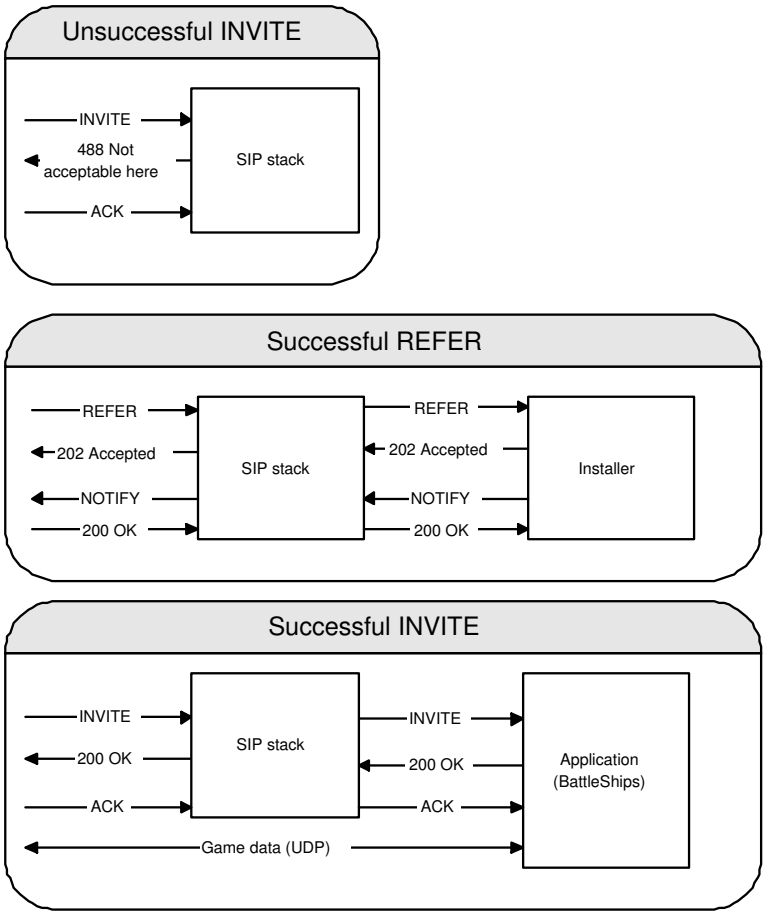

Figure 4.8: SIP message dispatching

After the application has been registered to the SIP stack, it can receive SIP messages from the stack. Also, the application can be automatically started when it receives a SIP request of a certain type. In our case, when the distribution is complete and the application is installed, the SIP stack automatically launches BattleShips application when it receives a SIP INVITE message of type “application/battleships”.

4.7 Distribution API

In the distribution scenario, the application that is distributed must use a specific API to in order to be distributable. As can be seen in Figure 4.6, the application MIDlet is build on top of MIPD 2.0, GameConnection API and Distribution API.

The GameConnection API hides the details of the actual SIP protocol messages, giving the application developer a simpler way to open connections to another SIP-capable terminals and applications.

The Distribution API offers a simple mechanism for application distribution. Figure 4.9 shows how an application can use the API. It hides the sending of the SIP REFER requests and receiving of NOTIFY request behind a simple programming interface. The application has to implement the ReferListener interface, and register itself to the ReferConnection class before it can start the distribution process. All interaction with the user is left to the application programmer.

The core of the API is the enableDistribution method in the ReferConnection class. It takes destination SIP URL, application package URL and a short description of the application as parameters and uses them to initialize the distribution session. Later, the feedback from the progress of the distribution is delivered trough the ReferListener interface.

4.8 Distribution Process from End-User's View

Now, let us look the distribution system from mobile phone user's view. Our example application MIDlet (BattleShips) is build to use the Distribution API, so it can distribute itself.

Table 4.3 shows the needed interaction with the user in successful distribution scenario. This scenario is very similar to the one in section 3.1 which describes automatic application distribution.

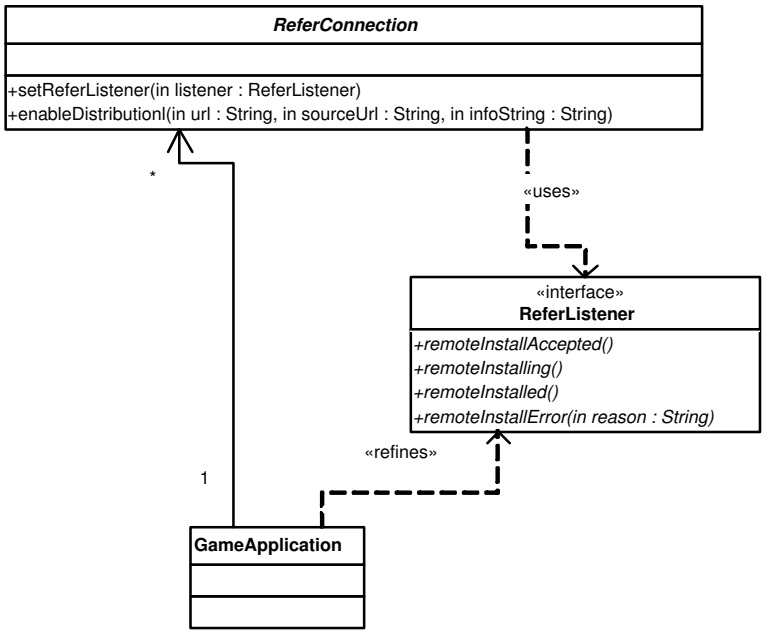

Figure 4.9: UML diagram of Distribution API

Description
1. Originating user starts BattleShips.
2. Originating user tries to start BattleShips session with the terminating user.
3. Session fails, and originating user starts distribution process.
4. Terminating user responds positively when asked to start the installer functionality.
5. Terminating user proceeds with the installation.
6. Terminating user accepts all dialog asked by the application installer.
7. After software is installed, the originating user restarts BattleShips session.
8. Terminating user accepts the launching of BattleShips, and session is started.

Table 4.3: Successful distribution process steps

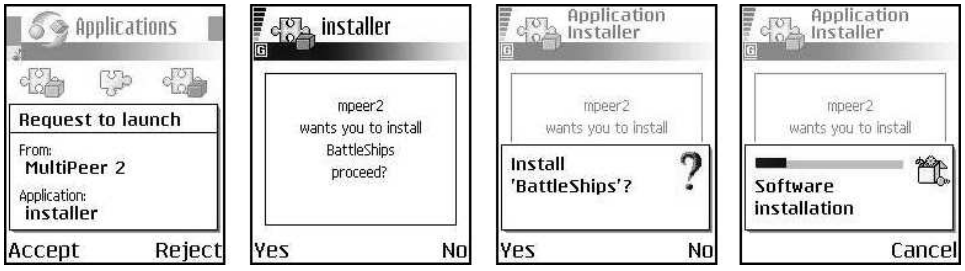

Figure 4.10: Terminating side screenshots

Figure 4.11: Originating side screenshots

4.9 Limitations of the Implementation

This demo implementation of peer-to-peer application distribution is by no means complete to be re-implemented for commercial purposes. It merely demonstrates that it can be done, rather than describes a full solution. Still, conclusions can be made to cover production implementations also.

The demonstration environment is different from real production implementation in many ways. Table 4.4 contains a list of features that differ from commercially exploitable, production implementation. These features are explained in more detail below.

In the demonstration network, the GPRS access point is configured so that the GPRS traffic is send to special test network, which allows direct traffic between terminals. To be usable in real production network, there should be no need to make special configurations to allow peer-to-peer connections.

Current cellular networks do not have SIP servers or CSCFs, so the demonstration

Feature	**Demo Implementation**	**Production Implementation**
Network	Special GSM/GPRS network with IETF SIP proxy	UMTS IMS Network
Terminals	Nokia 6600 terminal with added software and modified OS	Typical commercially available cell phone
Operator	Specific operator	Any operator
Peer-to-peer connectivity	Peer-to-peer traffic possible inside specific network	Peer-to-peer traffic possible between networks of different operators
SIP support	External SIP stack and proprietary Java SIP API implementation	Build in SIP stack and Java SIP API
Connection APIs	General connection APIs implemented in the application	General connection API implementations provided by the operating system.
Distributable applications	Specific Java applications	All applications supporting Distribution APIs
HTTP server	Always on	Communicates with Distribution API implementation
Authentication	None	Yes
Version control	Basic version control	Proper version control

Table 4.4: Differences between demo implementation and production implementation

network has a CSCF which acts as a standard IETF SIP proxy. This limits the operator to a specific one with the CSCF. Real production networks, like 3G UMTS, would have an architecture like IMS to enable SIP negotiations and peer-to-peer traffic.

In the current implementation, the peer-to-peer connections are limited. Both of the mobile terminals have to use the same specific access point to be able to com-

municate with each other directly. For example in IMS, it would be necessary to enable direct traffic from one network to another. However, the policy is operator specific. Even now, some mobile operators allow incoming traffic to be send directly to the terminals. In any case, the direct connections between terminals could be opened dynamically or they could be allowed as default.

The terminals used in the demonstration, Nokia 6600 mobile phones, have modified versions of the Symbian operating system. Modifications were necessary for implementing SIP support on the Java side. The Java SIP implementation is a proprietary extension to the Symbian Java virtual machine, and the modification of the virtual machine needs changes to the OS itself. Real production implementation would have SIP capable terminals, with native Symbian SIP stack and Java SIP API implementation. The application registration mechanism would be built-in in a production implementation. As SIP will be implemented in future networks, the terminals are likely to be 3G terminals.

Push feature means that the operating system can automatically open an application when it receives a request trough the network. The request can be for example a SMS message or, in our case, a SIP request. Symbian SIP stack natively supports push for native SIP applications, but the support in Java side must be changed in the OS. So, the terminals in the demonstration environment are special in this way too, but in production implementation, the terminals would support push feature natively for both Symbian and Java.

The implementation of the Distribution API is now integrated with the implementation of GameConnection API. In addition, both APIs are implemented in the application MIDlet. In production implementation, it is likely that the Distribution API would be part of generic Connection API. The Connection API would hide the SIP implementation details from the application developers. Of course, the Connection API would be supported by the operating system. It is also important to note that there must be different version of the API for different platforms, for example for Symbian and Java.

The problem with APIs in general is that they need to be standardized. For example, to create a new API for Java, a new Java Community Process (JCP) must be started for the standardization. Unfortunately, this may be a very long process, and

it is still not evident that the standardized API will become common.

In ideal case, it would be possible to distribute all available applications in peer-to-peer fashion. Our implementation currently limits the distribution from one specific demo application that uses the Distribution API. To be able to distribute applications in production networks, the distributable applications must use the defined Distribution API.

Because HTTP server is a separate component in our demonstration environment, it is hard to control when the application is distributable and when it is not. It would be necessary to bind the HTTP server with the Distribution API and the Installer component in order to ensure that the application is downloaded only when user has allowed the distribution.

For security reasons, the user must be able to trust that the user recommending the application is who he/she claims to be. So, in reality, there is a need for authentication. The demonstration system assumes that the "From"-field in incoming REFER request is valid. In commercial implementation, this may not be enough. The From field can be modified by the sender, so there is a possibility to impersonate as someone else.

Security currently relies on the fact that Symbian OS application installer warns users if the application to be installed is not signed. In addition, J2ME has a security model that provides security for the installed Java applications, ensuring that the applications do not do anything dangerous (like open connections) without user's acceptance.

Currently, the version control is the basic version control of Symbian OS. This has some limitations: It is not possible to check what version of the application, if any, is installed already on the phone, when the REFER request arrives. However, trying to initiate a session succeeds if there is a compatible version available, so this can be used to check if a compatible version is found. Production implementation would require a better way of version control. In reality, some kind of a version check must be done in SIP stack level, to enable different versions of applications to communicate.

4.10 Summary of Work Done in This Thesis

Lastly, table 4.5 sums up the work done during the writing of this thesis.

Item	**Work done**
Installer component	Installer component was coded in Symbian
Distribution API	Simple Distribution API was defined in Java and implemented using Java SIP API
Application	Sample application, BattleShips, was modified to use Distribution API
SmallServ	Ported SmallServ GUI to work on Series 60 phone

Table 4.5: Summary of work done in this thesis

Chapter 5

Evaluation

In previous chapter, the implementation of the distribution system was explained. In this chapter, we will analyze the implementation more thoroughly, and provide rationale behind the choices made. First, there is general discussion and second, the implementation is evaluated based on the requirements presented in Chapter 3.

5.1 Peer-to-peer vs. Server-Based Approach

The main differences between peer-to-peer and server-based distribution models were introduced in section 2.1. Now let us look more thoroughly the properties of both approaches.

Server-based distribution model has many characteristics that are beneficial in application distribution cases:

- Application distribution can be controlled easily. So the operator or other application distributer can control the distribution process. This includes starting and stopping the distribution process and providing updates for the customers.
- Distribution can be measured, and statistics can be made based on these measurements. This can be beneficial for marketing purposes, for both the operator and the application developers.

- Server could provide any (needed) version of the software. Different versions for different phones can be provided, if necessary, and there is also the possibility to provide updates for the applications.
- Charging can be implemented in the servers by making charging records. The servers providing applications can allow the downloading when charging records have been made.
- No extra peer-to-peer connections need to be opened. In a network that does not even allow peer-to-peer connections, application distribution is still possible.
- Only official applications can be downloaded, so security improves. This of course requires that there exists some kind of a trust relationship between the user and the provider. But once the provider has been authenticated, the user can be sure that the application is an official version.
- The server is capable of distributing many applications any time, bandwidth is not limited in the server like it is limited in an access network.
- Mobile networks have servers already, so additional servers fit in.

On the other hand, these properties are against using servers in distribution systems:

- Application servers require resources. Providing applications will be more costly for the developers, as there will be a need for renting or buying application servers.
- All applications need to have a server in the network. Of course, one server can provide different applications.
- Process needs a fixed network address for every application, because the every application needs to know the address of the server it.
- It can be hard to decide what is the lifetime of the application, and when the support can be revoked. When the distribution support stops, the users need to be aware of it.

Peer-to-peer model has also some properties that are useful in application distribution:

- Distribution is only dependent on the terminals, no servers or gateways are required. All the functionality needed is implemented in the terminals.
- It is intuitive to recommend an application to another user and to "give" the application right away. This is especially valuable when the applications are multiuser applications and when the connectivity to other users is essential part of these applications.
- Peer-to-peer connections are opened anyway if the distributed application is a multiuser peer-to-peer application. There may be a possibility to use this opened channel in distribution purposes too.
- Anonymity (no logging records are generated) can be considered to be a good thing among the users. This offers more privacy for the users.
- Applications can be distributed even when the application server is down.

Of course, there are some problems in peer-to-peer distribution also:

- The version control is more difficult to handle. The user may not be aware where he/she can get the latest version of the application.
- Charging of the distribution is more difficult than in the server-based model. When using fixed data transfer charging model, the operator can charge from the transferred content. Unfortunately, in such charging model, the application developer does not have an easy way to profit from the distribution.
- The network must support direct connections in some level. The connections can be dynamically opened, if they are not open by default. Opening ports can be a security risk in mobile networks, as it may be security risk in the Internet too.
- In some network configurations security (user identities) may be hard to verify, so authorization is needed.

- There is no default place to get the application, if nobody who has the application is known.
- The terminals will have to have more software, and therefore reserve already limited resources unnecessarily.

So both peer-to-peer and server-based distribution mechanism have their pros and cons. One solution is the combination of these two, a possibility to use either peer-to-peer or server-based mechanism, using the best features of both alternatives. This solution is not perfect either. This raises questions about on what kind of circumstances is the selection made between the two possible mechanism.

From the technical point of view, these distribution described above differ clearly. From the user's point of view, these mechanism may seem to be quite similar. It is possible to use peer-to-peer mechanisms in the signaling of the distribution process, and then download the content from a separate server. So, the user may not know where the content is actually coming from.

When the peer-to-peer signaling is used with server-based content transfer, the user may experience the distribution act as a wholly peer-to-peer distribution. However, the it is important for the user to know how much does the application downloading cost.

5.2 Protocol Choices

Let us analyze the use of SIP protocol in the application distribution system. SIP is designed to be used in session set-up and tear down. It has many extensions that serve many different goals. These extensions are used to different purposes, such as instant messaging and presence applications.

3GPP has defined SIP to be one of the signaling protocol in 3G networks. This means that 3G networks will have an infrastructure that is capable of routing and transmitting SIP messages. Therefore it is justified develop and build solutions that use this infrastructure, especially when there is no need to change or improve it.

In our case, the signaling of the application distribution can be done using SIP REFER. As REFER extension is also included in 3G standards, the signaling of

the distribution process has a working infrastructure, which can be utilized.

The SIP protocol has already gained attention along with VoIP. Since SIP does not have restrictions on which type of sessions it is used to manage, it is more probable that SIP will be used also in other purposes than VoIP, although VoIP will still remain as an important application for SIP.

SIP can be used in setting up and managing for example peer-to-peer gaming sessions. The negotiation between peers is done using SIP, but the in game traffic itself is not limited to any particular protocol. The game protocol can be proprietary and the traffic can use TCP, UDP or any other protocol. However, it is necessary to open the data channel in order to allow peer-to- peer communication. This is not a problem, since IMS CSCFs will parse the SDP bodies from the SIP requests, and they can open the needed data channel dynamically.

One advantage of SIP is that is publicly developed and therefore anyone can contribute to creating more extensions to the protocol. Besides, SIP provides a standard way of initiating sessions. Of course, when the protocol is publicly available, anyone can build implementations of their own.

However, SIP can be a bit too complicated for average application developer to be used as such. Consequently, offering a higher level API for the developers is beneficial. Currently, there are only few number of SIP APIs available. Jain SIP [11] defines the SIP API for Java platform. There is also SIP API defined for J2ME [10]. Higher level APIs, like general connection APIs or distribution APIs, would be beneficial for the application developers.

In the implementation, HTTP is used in content transfer. HTTP is very common protocol in the Internet, and with WAP 2.0, mobile environment too is turning to use standard HTTP when downloading web content.

HTTP, like SIP, is a public developed protocol, and has proven to be useful in many contexts. It is text-based and quite a simple protocol. Consequently, HTTP is supported in number of ways. There are many HTTP server and client implementations available to developers, and many programming environments have HTTP APIs, including Symbian OS 7.0 and Java MIDP.

5.3 Distribution API

A specific Distribution API was created in order to enable the application distribution mechanism explained in Chapter 4. The API is very simple, but still, the application developer must support the distribution feature to make the application distributable. Even with a simple APIs, there is always the possibility that the developers decide not to support it. It is even more likely, when the API to be used is not compulsory to enable the basic functionality of the application. Additional work, even small, can make the difference.

One of the biggest problems with APIs is that they need to be standardized to be fully usable. Otherwise, the application distribution solutions would be application or vendor specific. The standardization requires work contribution, and may be a very long lasting process. Our Distribution API is only an example, and in reality, the API could become more complicated in the standardization process. But, even proprietary distribution methods could become popular, if they are well implemented and marketed, and when there is no real need for interoperability between different technologies. Many competitive software platforms are likely to exist in the future too, and most probably they are not going to be fully compatible in the application level, so the distribution mechanisms could differ as well.

Many mobile manufacturers use proprietary platforms, such as Symbian. In order to enable application distribution in as many terminals and applications as possible, several versions of the API should be made. This problem is quite similar to the general application compatibility problem, applications need to have several version for different phones. Also the standardization must be done separately for different platforms.

5.4 Analysis Based on Requirements

The requirements for the implementation of application distribution were explained in Chapter 3. The implementation is evaluated here based on those requirements.

The mobile terminal used in the implementation was Nokia 6600 smartphone. It supports installing third party applications for both Symbian and Java, so it covers

the requirement 1.

Symbian OS warns if the application to be installed is not signed. The security requirement is covered with the fact that the user should not install any software that is not signed. And, installation of Java applications is quite secure because J2ME has a specific sandbox-model for security.

Installed third party software can establish connections trough the network. This may require interaction with the user before the connections are allowed. But in general, connections are possible and requirement 3 is covered.

Version control is provided by the Symbian platform. To check version compatibility with another peer-to-peer application, a session must be tried to set-up. This may be a little inconvenient for the user, but provides a way of checking the version. This functionality fulfills requirement 4.

The network has a SIP proxy that can be used to establish peer-to-peer connections between terminals. SIP provides a standard way of negotiating session parameters and managing and terminating sessions. This fulfills requirement 5.

The Installer component is responsible for interacting with the user when incoming distribution request arrives. The component shows user a dialog where the user can accept or reject the incoming request, and therefore the requirement 6 is covered.

On the originating side, initiating the distribution process can proceed when the user has started a specific HTTP server that serves the application. So the user has to manually start the server before the distribution can start, and has to stop the server when the distribution is complete. This is a little complicated for the user, but at least the user then knows what the terminal is doing at the moment. This covers requirement 7.

A sample application programming interface for the distribution was developed. The API is lightweight and provides a simple way for the developers to enable peer-to-peer distribution. The simplicity can be confirmed from Figure 4.9. So, requirement 8 is covered.

The only operator requirement, the support for charging, could not be implemented in this peer-to-peer distribution system. In theory, it would be possible to record the signaling process of the distribution and use that data in charging, but that would

be very complicated. The SIP REFER extension is used also in other purposes, so SIP servers cannot determine the distribution process in this way. Of course, the SIP servers could search for specific application distribution bodies, but it would require the standardization of these message body types. Therefore, requirement 9 is not properly fulfilled.

Chapter 6

Conclusions

The main goal of this thesis was to design and describe the implementation of peer-to-peer application distribution functioning in a mobile network. The implementation done is not ready to be used as such in an commercial environment, it rather shows that it is technically possible to create a working implementation. Nevertheless, it is possible to make conclusions based on the work done.

The implementation described in this thesis is not too complex. The SIP REFER extension is already defined, and it can be applied to application distribution signaling. The REFER mechanism is quite easy to understand, and it is supported by the 3G signaling infrastructure. The software implementation is a bit troublesome, but not too complicated. The Installer component implementation was quite straightforward on top of the Symbian SIP stack. The originating Java side implementation could be done using Java SIP API, but it required some changes to the operating system itself. A web server can be set-up on a mobile terminal, and there are already free implementations of mobile web servers available in the Internet.

Building a working application distribution system in an commercial mobile network requires resources, researching, standardization and implementation work. The future IMS networks will have the infrastructure needed for peer-to-peer distribution to work, so in that sense there are no evident obstacles. The main difficulty is to equip future mobile terminals with distribution-enabling software. But, even when there is the technical means to implement this distribution mechanism, the operators and software vendors too must decide to support it.

In general, SIP is a good choice to be used for signaling in mobile networks, as it is standardized in the IETF and it is extensible. SIP will be supported by future mobile networks and terminals, and application developers may use it in other than VoIP applications, like multiuser multimedia applications and games. Using it in the signaling of the application distribution process does not require large scale changes to any components, therefore it is fairly logical choice.

Peer-to-peer and server-based solutions both have their pros and cons. In short, peer-to-peer mechanism is more intuitive for the users, but commercial operators and application developers may favor the server-based mechanisms. What can be concluded from the implementation of this thesis, the peer-to-peer solution is a working alternative to traditional server-based distribution systems. It is not necessary to use only the pure peer-to-peer distribution mechanism, and probably only some parts of the distribution process will use peer-to-peer technology. So, the server-based solutions are likely to remain utilized as well.

6.1 Future Work

Application distribution in mobile networks has not yet been thoroughly researched, and there are research topics that would be interesting and beneficial to study further. As mobile industry is beginning to realize that there is a certain need for mobile applications, the ways of developing, implementing and distributing these applications are becoming increasingly important.

The standardization of the APIs related to the distribution process would create a base for application distribution in general. Otherwise, this distribution process would be used only by specific applications or specific vendors, which would limit the popularity of the method. But, even proprietary distribution methods could become popular, if they are well implemented and marketed, and when there is no real need for interoperability between different technologies. Many competitive software platforms are likely to exist in the future too, and most probably they are not going to be fully compatible, so the distribution mechanisms could differ as well.

There may well be other solutions to the application distribution problem, like us-

ing combined peer-to-peer and server-based approaches. These would be worth of examining in more detail, and it would be also beneficial to research the charging methods that could be used with peer-to-peer distribution mechanisms. It may be possible to enhance the charging possibilities by combining server-based distribution with peer-to-peer signaling.

It would also be important to study the security and privacy issues involved in the application distribution solutions, as they were not covered in detail in this thesis. The peer-to-peer distribution mechanism may be unsafe if there is no user authentication and application signing. Lack of these functions would endanger the security of the mobile terminal and the privacy of the subscriber.

One obvious future task would be implementing the application distribution mechanism in 3G mobile terminals and networks. Currently, 3G mobile networks are still on their way, and the research and development process is continuing.

This work demonstrated that peer-to-peer application distribution mechanism could be implemented in future IMS networks. More work is still needed in order to exploit this mechanism commercially, including API development and embedding distribution-enabling software to the mobile terminals. The distribution mechanism is quite intuitive for the users to use and also understandable for the application developers.

Bibliography

[1] 3GPP. About 3gpp. http://www.3gpp.org/About/about.html, April 2004. [referred 1.6.2004].

[2] 3GPP Technical Specification Group Core Network. IP Multimedia Call Control Protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP). Technical Specification 24.229 v.5.8.0, 3GPP, March 2004.

[3] William Alexander. GSM Architecture. http://www.m-indya.com/mwap/gsm/gsmarchitecture.htm, October 2000. [referred 31.8.2004].

[4] Douglas E. Comer. *Internetworking with TCP/IP Vol I, 3rd Ed.* Prentice Hall, 1995.

[5] Martin de Jode. Symbian on Java. http://www.symbian.com/technology/standard-java.html. [referred 14.4.2004].

[6] Kevin Dixon. Symbian OS Version 7.0s Functional Description. http://www.symbian.com/technology/7.0s_functional_description2.1.pdf, June 2003. [referred 1.6.2004].

[7] K. Egevang and P. Francis. The IP Network Address Translator (NAT). RFC 1631, IETF, May 1994.

[8] GSM World. Today's GSM Platform. www.gsmworld.com/technology/gsm.shtml. [referred 12.2.2004].

[9] M. Handley and V. Jacobson. SDP: Session Description Protocol. RFC 2327, IETF, April 1998.

[10] JSR-180 Expert Group. SIP API for Java 2 Micro Edition. http://jcp.org/en/jsr/detail?id=180, September 2003. [referred 22.7.2004].

[11] JSR-32 Expert Group. JAIN SIP API Specification. http://www.jcp.org/en/jsr/detail?id=32, August 2003. [referred 26.8.2004].

[12] Qusay Mahmoud. Wireless Java Security. http://developers.sun.com/techtopics/mobility/midp/articles/security/, January 2002. [referred 8.9.2004].

[13] John McAleely. SmallServ: A simple http server for Symbian OS. http://www.symbian.com/developer/techlib/apps/smallserv.html, June 2003. [referred 16.8.2004].

[14] Nokia. Series 60 Platform Product Overview. http://www.series60.com/pics/pdf/Series_60_Platform_Product_Overview_June04.pdf, June 2004. [referred 16.8.2004].

[15] Narayan Parameshwar and Chris Reece. Advanced SIP Series: SIP and 3GPP. http://www.awardsolutions.com/downloads/Award_Advanced_SIP_3gpp.pdf, 2001. [referred 17.5.2004].

[16] Philips Electronics. 2003 Worldwide Wireless Communication Standards. http://www.semiconductors.philips.com/acrobat/literature/9397/75010429.pdf, November 2002. [referred 8.4.2004].

[17] A. B. Roach. Session Initialization Protocol (SIP)-Specific Event Notification. RFC 3265, IETF, June 2002.

[18] J. Rosenberg, H. Schulzrinne, G. Camarillo, G. Camarillo, J. Peterson, R. Sparks, M. Handley, and E. Schooler. Session initialization protocol. RFC 3261, IETF, June 2002.

[19] R. Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *Proceedings of the First International Conference on Peer-to-Peer Computing (P2P'01)*, 2001.

[20] R. Sparks. Session Initialization Protocol (SIP) Refer Method. RFC 3515, IETF, April 2003.

[21] Special Mobile Group of ETSI. Digital cellular telecommunications system (phase 2+); Network architecture. http://www.3gpp.org/ftp/Specs/archive/03_series/03.02/0302-710.zip, January 2000. [referred 9.9.2004].

[22] Sun Microsystems, Inc. Datasheet Java 2 Platform, Micro Edition. http://java.sun.com/j2me/docs/j2me-ds.pdf. [referred 14.4.2004].

[23] Wireless Application Protocol Forum Ltd. Wireless Application Protocol Technical White Paper. http://www.wapforum.org/what/WAPWhite_Paper1.pdf, January 2002. [referred 1.6.2004].

[24] Dave Wisely, Philip Eardley, and Louise Burness. *IP for 3G, Networking Technologies for Mobile Communications*. Wiley, 2002.