Virtual networks in the cellular domain

Master of Science thesis

by

Gustav Söderström

performed at

Telia Research AB
Nynäshamn, Sweden 2003-02-10

Department of Microelectronics and
Information Technology (IMIT)
Royal Institute of Technology (KTH)
Kista, SWEDEN

Abstract

Data connectivity between cellular devices can be achieved in different ways. It is possible to enable full IP connectivity in the cellular networks. However this connectivity is combined with a lot of issues such as security problems and the IPv4 address space being depleted. As a result of this many operators use Network Address Translation in their packet data networks, preventing users in different networks from being able to contact each other. Even if a transition to IPv6 takes place and the potential problem of address space is solved, it is not likely that operators will leave their packet data networks open to the Internet.

An alternative to solving the problem on the IP level is to use overlay networks. In an overlay network applications on the cellular devices identify themselves at the application level rather than on the IP level. While full IP connectivity always gives the most efficient routing, an overlay network can offer services that are difficult to implement on the IP level. This can enable an application to ***span Network Address Translating entities*** without having to share the entire device. They can also provide private dynamic virtual networks and groups for users that trust each other. These ***private networks*** can use permissions and ***group casting*** functions, without the problems associated with traditional IP multicast.

The relatively limited bandwidths of the GSM and UMTS networks allow for application level routing of continuous data streams **if** the overlay network is ***distributed*** enough and mapped to the physical network in an efficient way. One of the advantages of using overlay networks is that although standard IP networks may be able to offer similar services in the future, overlay networks can be implemented in the existing IPv4 networks today at comparatively low costs. This may create the incentive needed in order for future larger investments to be justified.

A distributed overlay network not only allows for ***real-time services*** such as instant messaging, which is already possible with a centralized server solution, but it also allows for ***higher bandwidth services*** such as video conferencing, Voice over IP, etc. that are not possible on a large scale with a centralized relaying server.

An overlay network could be implemented by any third party ***without the support of an operator***. This suggests that free networks may be created for what could be called ***reversed file sharing***, i.e. networks where users upload files to each other rather than download as in most existing file sharing networks. These could become direct ***competitors to SMS, MMS*** and other operator-owned services.

The thesis investigates the mentioned possibilities and potential threats. Along with this an implementation of an overlay network for cellular devices is created that is totally independent of the operator's network.

Sammanfattning

Datakonnektivitet mellan cellulära terminaler kan uppnås på olika sätt. Ett sätt är att utrusta de cellulära näten med full IP konnektivitet. Detta medför dock en del problem såsom säkerhetsfrågor och problem med att antalet IPv4 addresser kanske inte kan täcka framtidens behov. På grund av detta använder många operatörer såkallad nätverksadressöversättning i sina paketdatanät vilket hindrar användare i olika paketdatanät från att kunna kontakta varandra. Även om en framtida övergång till IPv6 löser problemen med för liten adressrymd så är det inte troligt att operatörerna kommer att lämna sina paketdatanät öppna mot resten av Internet.

Ett alternativ till att lösa problemet på IP-nivån är att istället använda overlaynätverk. I ett sådant nätverk identifierar applikationer sig själva på applikationsnivån istället för på IP-nivån. Medans ren IP-konnektivitet innebär effektivast möjliga routing av data så erbjuder ett overlaynätverk möjlighet till tjänster som är svåra att implementera på IP-nivå. Bland annat kan applikationsnät som ***traverserar nätverksadressöversatta nätverk*** skapas utan att en mobil terminal behöver exponeras helt och hållet mot Internet. Dessa overlaynätverk kan också skapas dynamiskt och tillfälligt vilket ger användare möjlighet att skapa ***privata nätverk*** och grupper med med enheter de litar på, endast dessa får då tillgång till terminalen. Overlaynätverken kan också erbjuda ***multicast funktionalitet*** inom grupperna utan de problem som hör ihop med traditionell IP-multicast.

De relativt begränsade bandbredderna i GSM och UMTS nätverken tillåter routing av data på applikationsnivån **om** overlaynätverket är tillräckligt väl ***distribuerat*** och effektivt mappat mot det underliggande nätverket. En av fördelarna med att använda overlaynätverk är att även om den eftersökta funktionaliteten kanske kan implementeras på IP-nivå i framtiden med hjälp av ny teknik så kan overlaynätverk implementeras i nuvarande IPv4-nätverk till relativt låga kostnader då de endast består av mjukvara som körs på existerande hårdvara.

Ett distribuerat overlaynätverk erbjuder inte bara ***realtidstjänster*** såsom instant messaging vilket redan är möjligt och fungerar bra med en central serverlösning. Det distribuerade nätverket kan dessutom hantera routing av ***högre bandbredder*** mellan terminaler, såsom videokonferenser, Voice over IP etc. som inte är möjligt i stor skala med en centraliserad lösning.

Overlaynätverk kan implementeras av en tredjepart ***utan operatörers samarbete***. Detta kan innebära att gratisnätverk skapas för vad som skulle kunna kallas ***omvänd fildelning***, dvs. nätverk där användare laddar upp information till varandra snarare än laddar ner vilket är fallet i de flesta existerande fildelningsnätverk. Dessa nätverk skulle kunna bli direkta ***konkurrenter till SMS, MMS*** och andra operatörsägda tjänster.

Examensarbetet undersöker de nämnda möjligheterna och potentiella hoten i dessa nätverk. Utöver detta skapas även en implementation av ett overlaynätverk som är helt oberoende av operatörens nätverk.

Table of contents

Table of figures

1. Introduction

1.1 Problem statement

The goal of this project is to examine the possibilities of utilizing peer-to-peer (P2P) and overlay network concepts in mobile networks. Because the thesis was initiated by Telia, a Swedish fixed and cellular network operator, focus will be put on investigating what the p2p and overlay network concepts could achieve in cellular networks such as GSM and UMTS. In certain cases the combination of cellular and adhoc [1] networks such as GSM or UMTS combined with Bluetooth or IEEE 802.11 WLAN will also be considered.

With the increasing availability of GPRS the infrastructure is in place for services such as casual web browsing, e-mail polling, chatting, and other services that remain online over a long period of time, but doesn't necessarily communicate all the time. These services were not really possible with circuit switched solutions where the user pays per time unit for connectivity and connection set up times are slow. The services that have been offered so far for GPRS have mainly been WAP browsing and email. The devices themselves have not been able to do much more than interpreting a web (or WAP) page. This is now changing with more and more devices now able to run Java and even fullscale C++ programs.

While this offers a user the possibility to download and play games and run standalone applications, applications that are able to network with similar applications on other devices and participate in a large network would add another dimension to mobile device usage. This sort of networking requires functionality that is not available today. Most operator's GPRS/GSM network implementations does not allow for direct data communication between devices on the IP level since they use a pool of private (non unique) IP addresses. This makes it impossible for a device to contact a device that is in another subnet without the help of an intermediate server.

Furthermore there is no support for a device wishing to send data to several other devices in multicast fashion. This has to be done using multiple unicast streams which is not desirable in a network with limited and expensive bandwidth. Additionally, it is currently not possible for a group of devices to form a group or private network, similar to a local area network where they can communicate and share information freely since they trust each other.

Allowing mobile devices to directly communicate doesn't automatically mean an opportunity to charge for new services in cellular networks. While it could increase data traffic revenues, it becomes impossible for an operator to differentiate services unless he owns the services himself. This could become a potential threat to operator controlled services such as SMS and MMS. This is especially significant since they have high profit margins.

Ultimately, it could also become a threat to traditionally circuit switched services such as speech. Anyone on the Internet could then act as a virtual operator and connect customers of different physical operators.

The thesis analyses potential crucial market factors, possibilities and threats, as well as possible strategies for an operator in a future mobile P2P and overlay network scenario.

Along with evaluating different techniques to implement overlay networks, determine areas of use, and identifying what part an operator might have in these kinds of networks, an example implementation of a cellular overlay network infrastructure along with services was created using SUN Microsystems's Project JXTA [2] protocol suite.

JXTA is an acronym for juxtaposed which means side by side. It is a set of protocols aimed at simplifying P2P programming by offering standardized building blocks for creating overlay networks. The protocols offer services such as finding other peers in a network, opening communication channels to other peers, and searching for information in a network made up of peers.

1.2 Limitations

There are several different ways of implementing the overlay networks that are proposed in this thesis. Network nodes can be placed at several different locations in the physical network. Schemes are suggested where overlay network entities are placed inside the operator's GPRS network. However the author did not have access to these networks and hence the demonstration implementation utilizes a network built to be completely independent of the GPRS network. All network support nodes are placed outside the GPRS networks on the public Internet. This approach could be used by **anyone** to implement such overlay networks.

1.3 Demarcations

The focus of the thesis on cellular networks and to some extent combined cellular and adhoc networks is due to the fact that pure adhoc networks have already been rather thoroughly investigated. It is also more difficult to implement billing in these networks because devices communicate without infrastructure. Hence adhoc networks are more interesting from a device manufacturer's point of view than from an operator's. Adhoc networks will mainly be considered where they might provide extended functionality or value-added services to the cellular networks and hence be of interest to an operator.

Furthermore, no load tests or traffic simulations were performed in this thesis. The implementation written for the thesis is only used as a proof of concept and not for simulation or testing. Suggestions about dimensioning of large scale networks are discussed in the context of other reports or investigations of currently available technology. They are only intended to provide ball park figures.

1.4 Why does this problem require a master thesis?

The area of networking between mobile devices as well as between mobile and stationary devices is still rather new. While IEEE 802.11 WLAN has become very popular for more powerful laptops, there is still little data communication taking place between devices in cellular networks (such as GSM) and in non-cellular networks (such as Bluetooth and IEEE 802.11 WLAN in adhoc mode). Although SMS and MMS are forms of data communication, it is not a means of communicating arbitrary data in real-time, but rather a way to send **specific** content via a storage system, thus it is more similar to e-mailing rather than real-time communication. While it is possible to send arbitrary data in an e-mail message or via SMS (for example Nokia's proprietary picture format) it is not very practical for machine to machine communication, since e-mail adds a lot of overhead when small amounts of data are sent frequently and SMS has a small data payload and is furthermore sent on the GSM paging channel rather than a user data channel. Neither of them provide anything that would be perceived as a usable real-time connection (and they were not intended to do so).

The desired networking functionality could be implemented in several ways, one of course being to build in hardware support for these functions by using unique IP addresses for every device and putting multicast support into the routers and gateways of the network. There are no technical limitations preventing operators from enabling multicast, but few operators have done so. IPv6 [3], the next generation of the IP protocol (the current is IPv4 [4]) would solve the problem of the IPv4 address space exhaustion. Users could then have a static personal IPv6 address with which they could be reached.

However there are other issues that are not solved even by using IPv6. One of the reasons why private addresses are used today is in order that these devices not be reachable from the Internet. The possibility of Denial Of Service (DOS) attacks [63] and attacks intended to make a user pay for unsolicited data, along with devices themselves possibly being hacked means that a typical user would have to have a high level of skill and knowledge in order to install and properly configure a firewall (but would still not stop a DOS attack).

Using a standard firewall to prevent attacks a home user can protect his/her local network from accepting unsolicited data. In the cellular case, the user does not own the local GPRS network, but only owns one of the devices inside it. Thus it is up to the operator to provide this functionality. It should be noted that this would not protect users from an attack from **within** the local GPRS network, only from outside attacks.

The solution that this thesis investigates is instead to abstract the problem to the application layer and use P2P and overlay network concepts. An operator does not own the Internet and cannot stop users from participating in it or sending data on it. However an operator could create an overlay network over which he has full control of the participants. Group and multicast functionality could then be implemented in this "trusted" network.

Mobile adhoc networks, such as Bluetooth or IEEE 802.11 WLAN in adhoc mode, could also gain more advanced networking functionality using overlay networks. This is not necessarily the same functionality that is wanted for wide area cellular networks. In adhoc networks bandwidth is usually free and sometimes plentiful. Thus in these networks, every device needs to be capable of routing and searching for resources. P2P technology could be used to achieve advanced networking functionality by letting an application provide fundamental routing and resource finding as well as advanced routing protocol functionality (exchanging routing table information, calculating route metrics etc.), not necessitating the construction of new hardware.

1.5 Why should mobile peer-to-peer networking be considered right now?

GPRS has been around for a while so the question of why peer to peer networking in cellular networks should be particularly interesting right now as compared to a year ago is justified. The reason for this is the maturing of two separately developed areas at the same time. Namely the ability to send arbitrary data in an operator's network using GPRS and the ability to install/run arbitrary software on a manufacturer's device.

Most mobile device manufacturers today support add-on installation of Java programs and several even support installation of C++ software (Nokia 7650, Nokia 3650, Nokia Communicator, and Ericsson P800) in the actual device OS giving a third party developer full control of the device. This means that while earlier a WAP browser application had to be shipped in hardware and thus developed by/with a device manufacturer, a similar application today could be developed by a very small third party on a small scale. There is no need for the application to be supported by the manufacturer since it can be installed after purchase of the device and there is no need for the application to have support from the operator if it uses GPRS to communicate with the Internet or other IP devices.

This development could also be seen as a result of the awaited merge of mobile phones and PDAs actually taking place. Since PDAs are already very powerful and capable of installing software just as any full sized PC it is not very surprising that the most advanced phones being released on the market now are increasingly similar to a PC with a constant Internet connection.

However, the potential alone does not automatically mean that there is a sudden need to communicate lots of data between mobile devices. Looking at a standard PDA or mobile phone there is not a lot of information that one would possibly want to share with others, phonebooks, calendars and other personal data are synced between devices and possibly against a central corporate server, but these are not large amounts of information.

In the stationary network, a lot of the information that is shared is games, music, and applications. While a PDA or advanced mobile phone can hold this kind of information in just the same way as a stationary PC, it is not very likely that a user would want to pay for someone else downloading this kind of information from his/her device (a user has to pay for both sending and receiving data in current GPRS networks). Calculating the size of a standard mp3 song, even tomorrows UMTS networks are much to slow to make that sort of data exchange interesting. This will likely not take place until wide coverage of high bandwidth networks such as WLAN hotspots is available in most places where users are.

Rather, this kind of information that is also available in the stationary network would probably be downloaded in a traditional client-server fashion from the stationary network when dealing with smaller java games, ringtones, icons etc, or via IR, Bluetooth, WLAN or cable from a PC when dealing with larger files.

However there is information that is suitable for sending in cellular GSM and UMTS networks. The introduction of built in cameras in many new mobile devices provides information such as pictures and video clips that are only available where the user is and are possibly only interesting instantaneously or at least for a limited period of time.

Written or spoken conversations could also be performed in the overlay network, allowing a conversation to be continuous and casual and allowing for large group conversations without reserving the amount of network resources that a circuit switched group conversation would.

This rather recent maturing of several different technological factors is the reason why P2P networking in cellular networks now has a lot more potential today as opposed to a year ago.

2. Background

2.1 General Packet Radio Service

General packet radio service[5] is the architecture used in both the existing GSM networks as well as tomorrows planned UMTS networks to enable IP data connectivity.

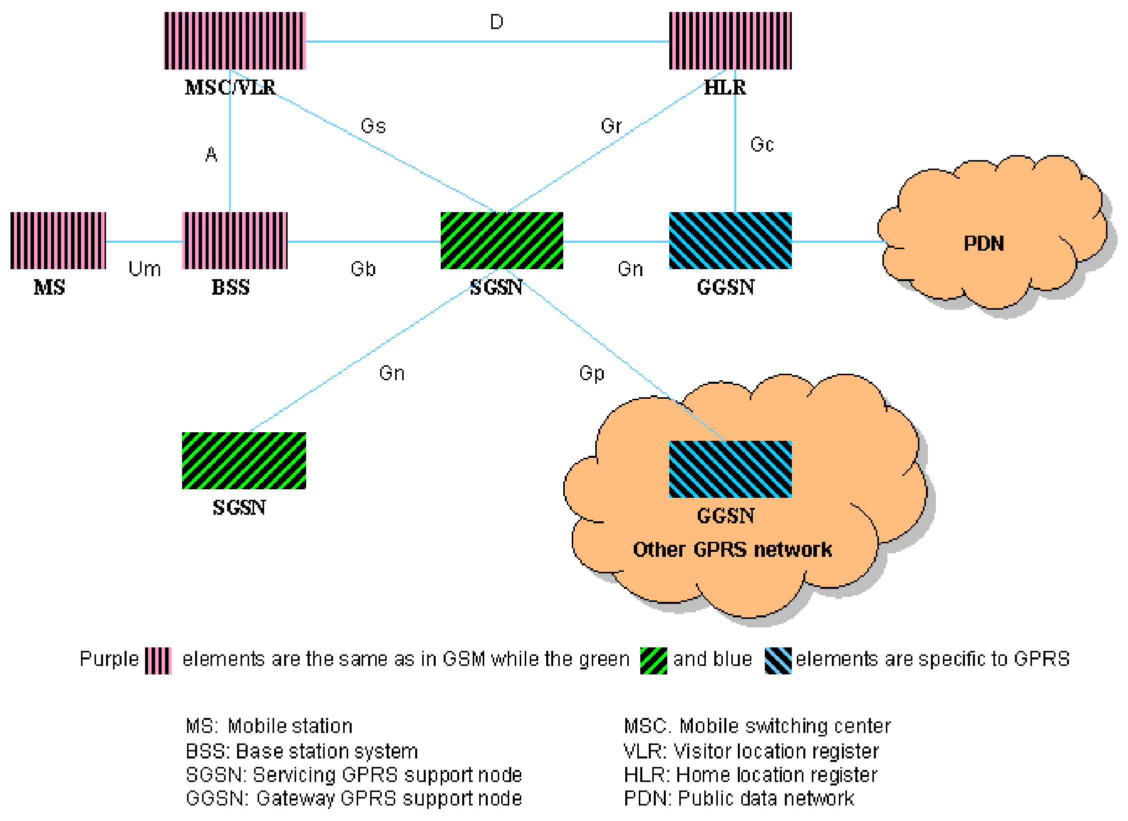

D
MSC/VLR
HLR
Gs
Gr
Gc
A
PDN
MS
Um
BSS
Gb
SGSN
Gn
GGSN
Gn
Gp
SGSN
GGSN
Other GPRS network
Purple ▥ elements are the same as in GSM while the green ▨ and blue ▨ elements are specific to GPRS
MS: Mobile station
BSS: Base station system
SGSN: Servicing GPRS support node
GGSN: Gateway GPRS support node
MSC: Mobile switching center
VLR: Visitor location register
HLR: Home location register
PDN: Public data network

Figure 1. The GPRS network nodes

The GPRS network is not a separate network but rather an add-on to the existing circuit switched network. This means that GPRS "co-operates" with the existing GSM network. GPRS data is sent inside the timeslots of a regular GSM frame using special coding schemes (CS1 – CS4). The GSM network is both time and frequency multiplexed. In a cell there are several frequency channels, a device listens to one of them. Each frequency channel is divided into eight timeslots, which is called a frame. A GSM device making a circuit switched phone call uses one of these eight timeslots. In GPRS however, a single device can listen to data in several timeslots. Theoretically a device could allocate all eight timeslots of a frequency channel. This is usually not allowed by the operator. Currently most operators allow a device to allocate a maximum of four timeslots in either direction.

Circuit switched traffic is always prioritised over the packet based and if the resources that a GPRS customer has allocated are needed for a circuit switched call he will lose them.

The functional nodes that are added to the existing GSM network in order to enable packet based traffic are the green and blue nodes in the above picture.

When a device (often abbreviated from the mobile station as "MS") wishes to activate packet based traffic it connects to an SGSN and requests to establish a packet data protocol context (PDP context). The SGSN can then interrogate the VLR and HLR of the GSM system for subscriber details in order to find out if the subscriber should have access to the network. If so, the SGSN establishes a PDP context with the device and contacts the GGSN that was requested by the device. The GGSN is a router and is also the gateway to the Internet. The

GGSN hands out IP addresses to devices using DHCP and may also perform DNS functions. The SGSN now establishes an IP tunnel to the GGSN. The MS now has packet data connectivity to the Internet. An MS can move between cells and still be connected to the SGSN. A user can also roam to another SGSN without losing connectivity. Packets that arrive at the old SGSN are forwarded to the new SGSN until the new SGSN has established a context with the GGSN.

A packet sent from the device travels over the air interface and to the SGSN using various GPRS specific protocols. From the SGSN the packet is tunnelled using IP to the GGSN where it is unpacked and sent out on the Internet.

As described above, the GPRS scheme in itself provides full IP connectivity. However, since the GPRS network is a network where users have to pay for incoming traffic along with the fact that devices in the GPRS network are unprotected from intrusion has resulted in most operators introducing an extra node in the infrastructure that is not GPRS specific, namely a Network Address Translating (NAT) function [46] between the GGSN and the public data network.

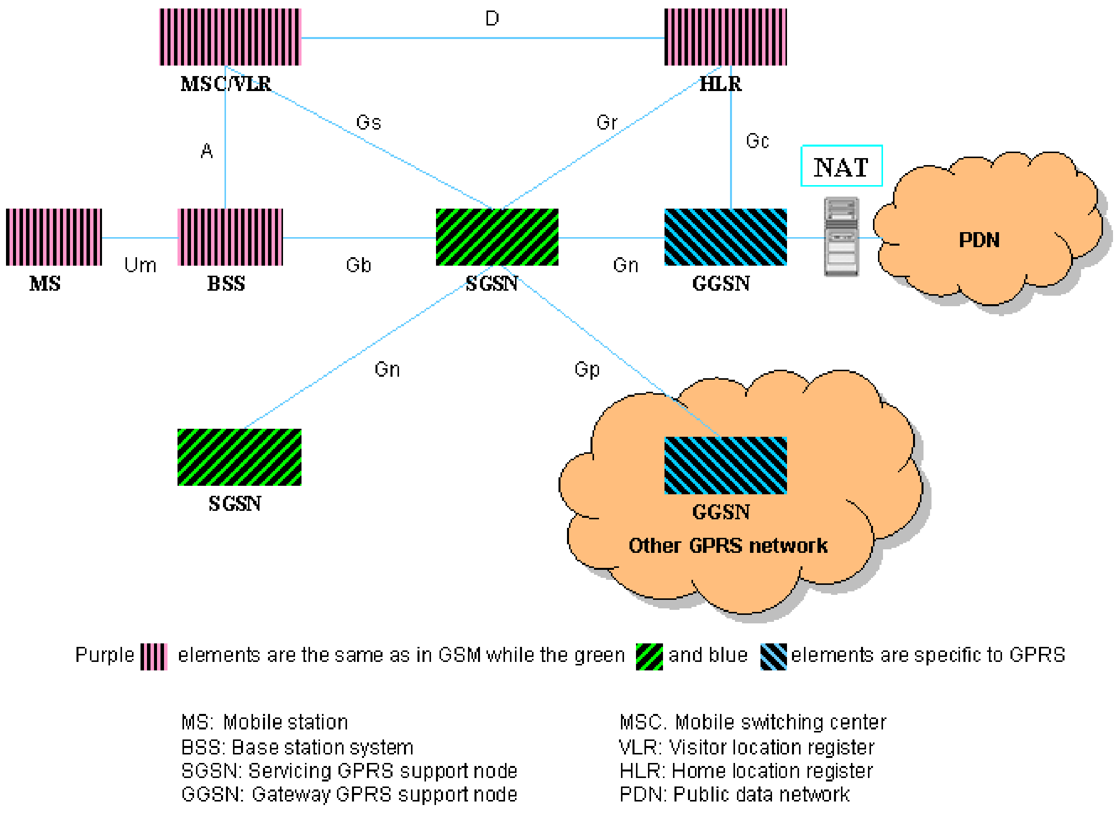

D
MSC/VLR
HLR
Gs
Gr
A
Gc
NAT
PDN
MS
Um
BSS
Gb
SGSN
Gn
GGSN
Gn
Gp
SGSN
GGSN
Other GPRS network
Purple ▥ elements are the same as in GSM while the green ▨ and blue ▧ elements are specific to GPRS
MS: Mobile station
BSS: Base station system
SGSN: Servicing GPRS support node
GGSN: Gateway GPRS support node
MSC. Mobile switching center
VLR: Visitor location register
HLR: Home location register
PDN: Public data network

Figure 2. The NAT node in the GPRS networks

This NAT function prohibits IP sessions being initiated from the Internet to the operators network and is the major reason why end-to-end data connectivity is not possible on the IP level today. Apart from offering protection, NAT can also be used to save IP address space since private addresses are used inside the network.
The different GPRS subnets of an operator or several operators could be viewed as separate subnets or Local Area Networks (LANs) connected over the Internet.

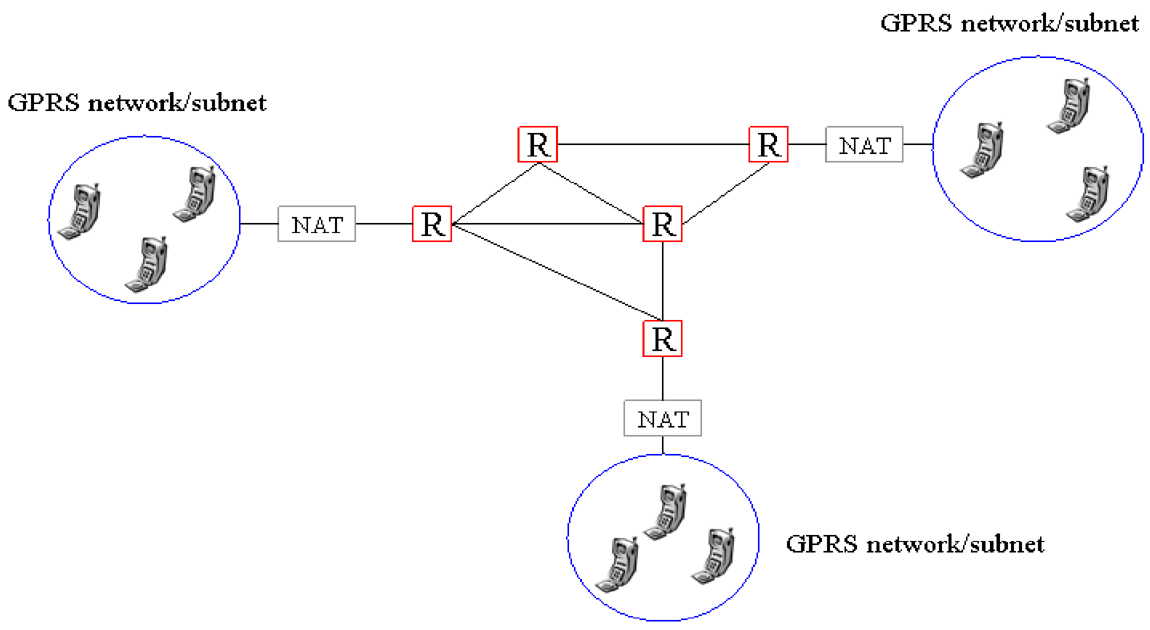


Figure 3. GPRS networks can be viewed as just another subnet of the Internet (R = Router)

With IPv6 offering sufficient address space, there are solutions where the NAT function does not exist (see section 5.1.5). This does not mean that the network is left open for intrusion and unsolicited data, instead security functions are implemented in the GGSN. What the security policies of these GGSNs will look like is not clear at the moment. Even though a device in the IMS scheme (section 5.1.5) could be able to authorise an IP session in the GGSN before receiving data from the Internet, it may still not be allowed to by the operator since this could compete with regular telephony. However, as will be shown later there is no effective way to prevent such services via a third party.

2.2 JXTA

JXTA is a specification defining a set of protocols that any application can use in order to discover and communicate with other applications in a P2P fashion. There are several implementations of the JXTA Application Programming Interfaces (APIs) available to programmers to use this protocol suite, the first and most well documented being for Java2SE [47]. The APIs are also available for C++, Java2ME, Python, Perl, Smalltalk, TINI, and other languages. The aim is that writing P2P applications will become much easier using the APIs of JXTA than starting from scratch.

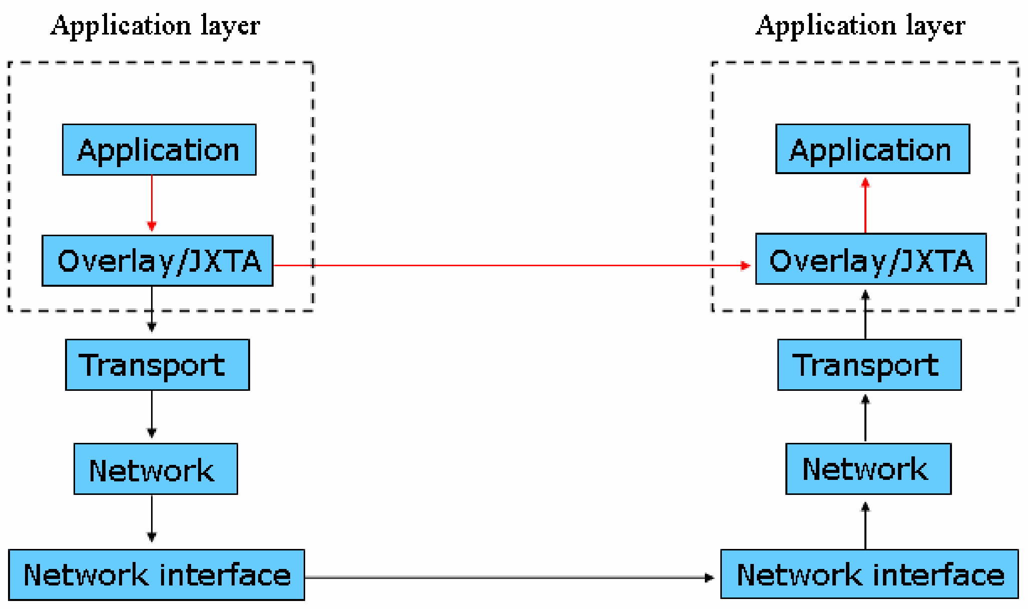


Figure 4. The JXTA APIs divide the application layer of the TCP/IP stack in two parts

The big difference between JXTA and for example ICQ [10] and MSN [9] networks, is that while ICQ and MSN build an overlay (or virtual) network for their particular service, they are **not** open to other services, JXTA provides a general overlay network that any application can use to send arbitrary data by simply implementing the JXTA platform and protocols. Additionally ICQ and MSN utilizes a centralized server (most likely implemented as a server park with load balancing), acting as an application level router (ALR) for messages, JXTA instead has an infrastructure built upon several ALRs (called rendezvous) that are intended to be distributed geographically.

The JXTA platform defines the following seven protocols:

• Peer Discovery Protocol (PDP) - Resource search
• Peer Resolver Protocol (PRP) - Generic query service
• Peer Information Protocol (PIP) - Monitoring
• Peer Membership Protocol (PMP) - Security
• Pipe Binding Protocol (PBP) - Addressable messaging
• Rendezvous Protocol (RVP) - Propagate messaging
• Peer Endpoint Protocol (PEP) - Routing

There are three components of a JXTA network that need to be understood in order to build effective applications, they are called peers, relays, and rendezvous. A single host could be one, two or all three of these components at the same time. Peers are simply the endpoint of the peer network, they are able to discover and communicate with other peers. Relays are used because of the extensive use of firewalls and gateways implementing NAT/NAPT technology today. A peer inside a NAT/NAPT network can keep in contact with the JXTA world network by connecting to and using HTTP to poll a relay server outside the NAT/NAPT network. The relay receives messages meant for the peers inside the NAT/NAPT network and waits for these peers to contact the relay which then forwards the buffered messages. The rendezvous are used to aggregate information about peers and the services that they offer so that information can be found without the amount of flooding that occurs for example in Gnutella where there are no aggregating functions (see further discussion about Gnutella and other overlay networks in section 2.4).

There are two additional components that are necessary in an overlay or P2P network just as in any other network, these are router services and name services. Since a JXTA rendezvous routes data on the application level as well as answers requests about other peers presence it could be said to contain both the router and nameserver functions of the JXTA network.

The routing protocol of the Rendezvous differs a lot from that of an IP router. Since JXTA addresses are not location dependant but instead randomly assigned, messages cannot be routed based on a subset of the address, as is the case with IP addresses. Instead a JXTA rendezvous keeps track of peers that are connected to it and routes requests directly between them. If a request arrives about a peer that the particular rendezvous does not know, it will send the request on to all other rendezvous that it knows of in a flooding manner. In order for the number of requests not to overload the network or circulate the network forever if no peer is found, a Time To Live (TTL) variable is used. This variable is decreased by every rendezvous that a message passes by, when the TTL value reaches zero, the message is not relayed farther. This TTL variable is set to seven in the current JXTA implementation. The particular value used is based on lessons learned from other P2P networks such as Gnutella.

Gnutella implemented the equivalent of the JXTA rendezvous in every client (called a fully distributed overlay network), resulting in a very large network of routers. A TTL of seven in this network would not get the user very far (i.e. find a lot of other users). However If the TTL was increased, so much traffic was generated from requests that the network basically stopped functioning [6].

JXTA has learned from this and instead the rendezvous aggregates information about several peers at certain points in the network, thus a TTL of seven in JXTA will yield a large response because hopefully every rendezvous knows about a number of peers. This helps to keep the traffic load at a more reasonable level as compared to Gnutella.

However, there is always the risk of a flooding based system becoming “to popular for it’s own good” meaning that if a single JXTA network becomes very popular, it would have to be expanded in order to handle the traffic load so that data could use more paths and be more distributed in the physical network. But as the JXTA network expands, more hops are needed if a user wants to have a good chance of being able to reach anybody in such a network, resulting in the amount of traffic increasing as the average number of other rendezvous that a rendezvous knows about minus one (where the request was received from), raised to the number of steps that the TTL is specified to.

$$\left(\overline{NumberOfRdvConnections} - 1\right)^{TTL}$$

Therefore we must consider what type of traffic is expected in such a network and how many peers are expected to connect to a single rendezvous (hence, how many rendezvous will be needed) in order to decide if JXTA is a suitable solution to a problem. Calculations on how large the network will become enables a developer to estimate if a TTL of seven is sufficient to make it likely that all peers can find each other. In some applications, such as networks made for information sharing and searching, this may not be crucial at all while for services such as messaging (as the application made for this thesis) this may be a crucial factor.

Apart from a JXTA rendezvous having the responsibility of a router and nameserver, it also does a lot more than these services. It answers requests about the peers that it is responsible for, and stores advertisements about the services that they offer. This could be any type of service, thus applications using the JXTA network can find each other by publishing and searching for a particular advertisement. An example of this would be a chat service, where when a user wants to create a new topic, he/she publishes a chat group advertisement to the JXTA network, another chat application could then ask the network for all advertisements of type chat group and choose which advertisement/group to use/join.

When the user chooses to join a certain group, he uses the Pipe Binding Protocol to open a communication channel (called a JXTA pipe) either to a certain peer or to a certain group of peers (called unicast or propagate pipes), the Rendezvous then makes sure that a message sent to this group is replicated and sent to each peer that has opened a pipe to the same service. This functionality can also be found in the network layer routers that use IP multicast. In IP multicast; a multicast group is joined and the network routers then make sure that any message sent to the group reaches everyone. The difference between the application level JXTA "multicast" (called propagate) and the network layer IP multicast is that while IP multicast demands that an entire host be available to the network on the IP level, the application layer network only demands that the particular application is available to the network.

In other words, if a totally distributed (i.e. without any central server) chat service would be implemented, the network could be said to be the server and propagate messages aimed at several peers in a chat group.

JXTA is built around the principle of groups. Everything is done in the context of a group and in order to use a certain service that a group offers, a peer must join that group. When a JXTA platform is launched, it automatically attempts to join a group called the world peer group. This is first done by multicasting on the local network to find out if there are any peers or rendezvous present that are part of the world peer group. This multicast enables devices that are not connected to the Internet to discover each other if they are on the same subnet, this is crucial if the system is going to be usable in an adhoc environment. If no other peers are present in the local network, the platform continues by trying to contact known rendezvous in a predefined list. By default this is a list of rendezvous and relays that SUN offers to the public. Proprietary rendezvous could be put in the default list instead.

A central protocol to JXTA is the Peer Membership Protocol. This allows a user to create secure groups. When a user creates a group, he/she can add a digital signature/certificate to the group advertisement. The peer group membership service implemented by the JXTA rendezvous then makes sure that any other peer that wishes to join the group needs to present a matching signature/certificate. This however means that the group is secure if, and only if, the network of rendezvous can be trusted.

Another quite important feature of JXTA is the pipes previously mentioned that are used to communicate between peers. Pipes are an abstraction of Java sockets enabling a user to set up a unicast or propagate communication channel to one or many peers without having to pay attention to things such as if one of the receivers is actually behind a firewall and hence only polling a relay using HTTP while other peers may be using TCP; the pipe protocol takes care of the conversions and message replications automatically.

The Java2ME binding of JXTA is called JXME [7]. It is intended for less capable (small footprint) devices such as mobile phones. It does not implement a complete JXTA platform since many small devices do not have the capacity needed to implement a full peer. For example, after being online for some time, a full JXTA peer may require several Mb of storage space for advertisements while the JXME implementation does not save advertisements locally. Another reason is that a device in a mobile network such as GPRS a user would generally not like to receive all the traffic that is associated with being a full peer.

Because of this, a Java2ME peer does not connect directly to a JXTA rendezvous as "normal" peers do. Instead it connects to a special HTTP relay. The Java2ME peer polls the relay using HTTP (which is the only communication protocol supported in Java2ME under the MIDP 1.0 standard [48] currently used). The relay in turn is actually a full peer "representing" the mobile device in the stationary network. So the program running on the mobile device may be regarded as client software even though it behaves as a peer thanks to the relay in the stationary network.

The Java2ME peer polls the HTTP relay, by always *initiating* contact we solve the problem of a cellular GPRS device often being inside a NAT network and thus not being reachable from the Internet.
The different components that can make up a JXTA network are shown below.

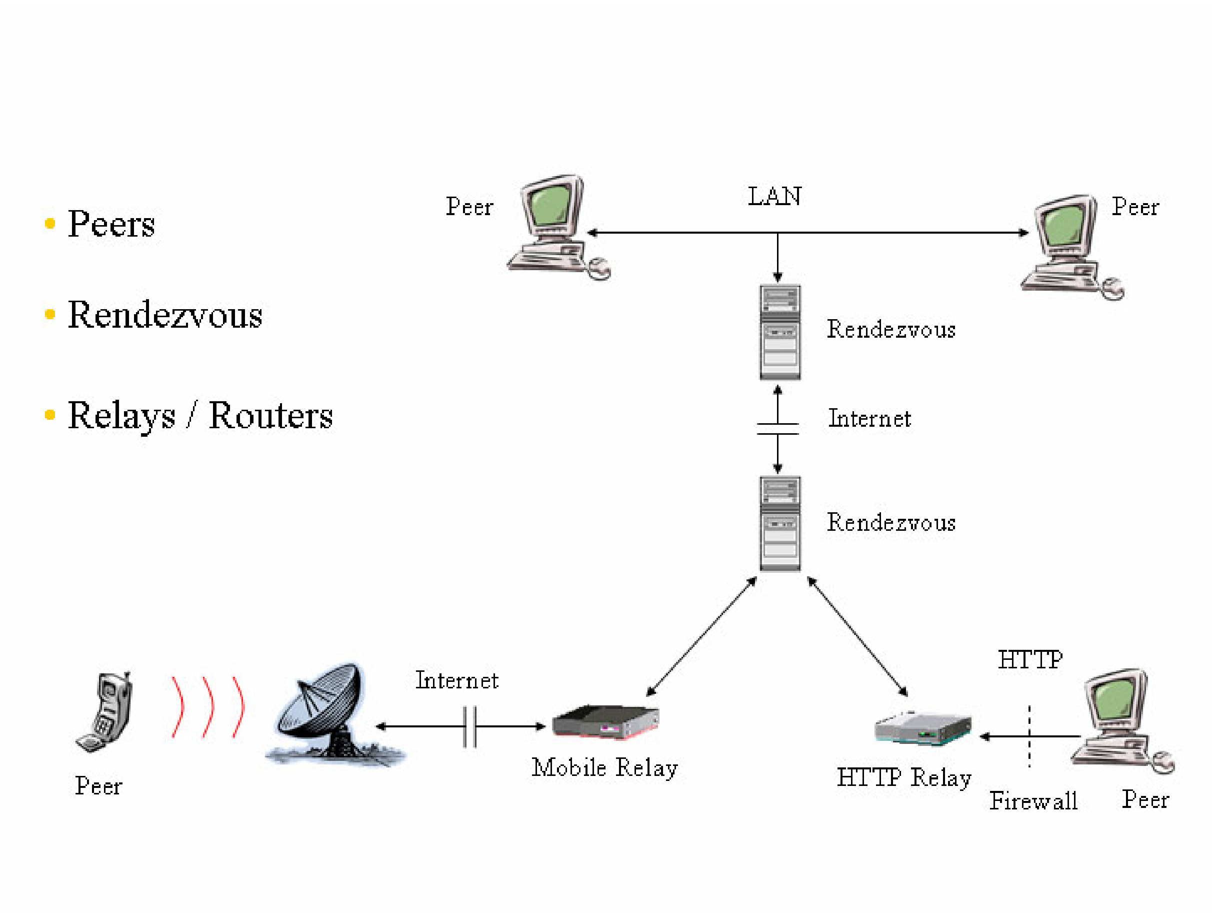


Figure 5. The different elements of the JXTA network

2.3 Traditional networking

The traditional approach to networking has long been that several users connect to a central server in order to access information. This has proven to be a good solution for several reasons. Desktop computers have traditionally had quite limited computational power and storage capacity while in comparison servers have been much more powerful (and therefore more expensive), thus the idea of the desktop computer simply being a terminal used to access a more powerful resource has prevailed. However, this David and Goliath relationship, that spawned the client – server technology, has not been true for quite a while. The desktop computers of today are many times equal to the servers they access in both computational power and storage capacity, i.e. they are peers (equals). This means that while traditionally desktop computers did not provide any services to the network, but simply accessed information that was aggregated on the servers, today desktops connected to the network contain huge amounts of information, often surpassing the amount of data stored at the servers, and in fact most information today is found at the leaves of the network.

2.4 Peer to peer and overlay networking

2.4.1 Peer to peer communication

The idea of one computer directly accessing resources on another is called P2P communication. Although there have been applications around that may be considered P2P for a long time, one of the first and probably the most famous that hit a wide audience was Napster [8]. Although some argue that this was not a true P2P product, because a central server was still used, it was definitely the program that started information flowing from the desktop towards the Internet (and other desktops).

It can be quite difficult to separate what is P2P and what is client - server, often a combination of the two is the best solution to a problem. There are no widely accepted definitions of what P2P is and this report does not intend to deal with this problem in any depth.

A description that probably covers most of the applications that are called P2P today would be applications where the end user device (at the network leaves) provides resources to the network, rather than simply demanding. This could be in the form of information or hardware capacity. Today there are several products using P2P technology, many of which are for file sharing and unfortunately P2P has been connected with software piracy in the popular press. However, P2P is not a technology for file sharing alone although this is one of it's applications. It is a technology where one computer directly accesses *any kind of resource* on another computer, be it the file system, processor, printer, or simply the person owning the computer as is the case of chat services such as MSN Messenger [9] and ICQ [10]. Examples of applications using resources other than the file system are SETI@home [11] (Search for Extra Terrestrial Intelligence at home), which borrows a desktops processor capacity when idle to search for patterns and signs of extra terrestrial intelligence in blocks of data downloaded from radio telescopes. Folding@home [12] which is built on the same idea as SETI is an application that instead uses the capacity to calculate possible ways of folding proteins in order to gain knowledge about diseases such as Alzheimer's disease.

The idea of sharing resources between computers is not new, most offices have some kind of local area network (LAN) where users can move files between computers and print a document on a printer connected to another computer. This could very well be considered P2P technology on a local scale. P2P technology on a larger scale can provide the possibility to create private networks or groups on top of the IP network (i.e. on the application level) letting users at different locations of the Internet share information and resources between each other as if they were inside a local area network.

While there are several applications that implement overlay networks, there are also several different schemes that are used. There are centralized overlay networks such as MSN Messenger and ICQ. A central server means that the system can keep track of all connected users and what IP address they currently use. On the other hand, a central server will receive search queries and messages from all peers in the network, and will therefore be heavily loaded. Napster is another example of another centralized overlay network, Napster was also known to suffer from heavy traffic loads.

Gnutella is a file sharing application that creates a fully distributed overlay network where the search and routing functions are implemented in each peer, there are no central points in such a network, users are connected directly to each other and thus each peer must implement all network functionality. This is often called a truly decentralized or truly distributed overlay network. Since there is no central place in such a network, flooding has to be used to send messages to find other peers and search for information. In a large network the amount of flooding request and messages can become a problem.

Today, a combination of the two former systems are often used, these are called semi-distributed systems. In these systems there are several nodes acting as a distributed version of the central server used in for example Napster. These nodes route information between each other and peers on the application level and will be referred to as application level routers (ALRs). Flooding is still used to find peers and information, but only in the network of ALRs.

Common for all the different approaches is that they implement some sort of overlay network on top of the existing IP network, whether data is routed by end nodes or by special ALR nodes in the network.

2.4.2 Building yet another network on top of IP

In order to understand the point of building yet another (and probably more inefficient) network on top of an already existing network, it is necessary to understand some of the hardware that the Internet is built on.

The IP network is in fact also an overlay, or virtual network built on top of several different physical networks. Looking at the hardware level of the IP network, there are different network technologies such as for example IEEE 802.3 Ethernet. Devices in these networks have link layer (MAC) addresses that enable them to send data and communicate with each other. Since an international organization is responsible for every Ethernet hardware address being unique, large networks can be built without the risk of addresses colliding (not entirely true since there are "pirate" non-standardized IEEE 802.3 Ethernet cards available). Switches can forward IEEE 802.3 Ethernet packets and multi-hop networks can be built if the switches implemented bridging using the hardware addresses. However, since the IEEE 802.3 Ethernet addresses were not assigned to specific physical locations, frames cannot be routed in a certain direction just by looking at part of the hardware address. Instead every bridge in the world would have to have a complete list of the location of all computers in the world. Alternatively flooding of the network would have to be used to make sure that the bridge nearest to the receiver gets a copy of the message.

Although it may seem that it would have been smart to hand out the IEEE 802.3 Ethernet addresses geographically so that they could be used for global routing, it would mean that a computer with a specific Ethernet card would be locked to a specific geographical area. There is also the problem that IEEE 802.3 Ethernet is not the only network technology used, although it is becoming increasingly popular even for core network transport.

Instead, the IP layer was conceived. Unlike the hardware interface address, an IP address is virtual and is only temporarily mapped to a specific hardware address. Since the IP addresses handed out are geographically specific, the IP addresses are suitable for global routing. The temporary mapping enables changing the network interface card (NIC) and thus the hardware address of a server without actually changing the server's IP address. It also means that an interface can be moved from one part of the world to another and be remapped to a locally valid IP address in that part of the world.

In a more abstract sense, the IP network can be said to free a host from a particular hardware address since the host identifies himself to others on the IP level (although not true for IPv6 where the hardware address is intended make up part of the IP address). However, even with IPv6, the IP layer still allows for several IP addresses to be assigned to a single hardware address.

Creating yet another network on top of the IP network means that instead of a host identifying himself to others as a host, separate applications identify themselves with a unique id that is temporarily mapped to the IP address of the host where the application happens to reside at that moment. Just as in the case of several IP addresses being able to be mapped to a single hardware address, several application level network addresses can be mapped to a single IP address. So as IP addressing added a level of address multiplexing on top of the hardware addresses (though perhaps seldom used) application level addressing adds yet another level of multiplexing.

In analogy with the previous discussion, the overlay or application level network can be said to free the application from a particular IP address and under some circumstances (depending on how the overlay address scheme is implemented), the geographical subnet that the IP address belongs to.

So a user can bring his/her application with him/her, from one subnet to another, and it will still be the same application to others on the Internet (in the overlay network). While it is perhaps not very usual for users to actually physically carry an application around, services such as MSN Messenger and ICQ that are present on many computers could be said to be generic applications that are created with the password that the user provides, in fact if a new MSN Messenger client is started on a computer while another that was instantiated with the same password is running on another computer somewhere else in the world, the latter will actually exit, which is in line with the concept of the application actually having moved since it cannot be instantiated in several places at the same time. If the MSN Messenger service is considered to be a generic application or stub, the user could be said to carry the final (although very small) part of the application in his/her head.

While this is great for backpackers, trotting the globe visiting Internet cafés as well as people who want to be the same chat user at work as at home, there are also occasions where applications actually physically change their IP address. Many ISPs as well as most cellular network operators hand out dynamic IP addresses to computers and mobile devices, resulting in the applications on the device actually changing IP address now and then, this is also the case if a cellular device moves into another subnet or roams into a subnet of another operator.

However, there is a large problem with overlay networks that implement this sort of random addressing scheme that is not bound to particular subnets or areas. This is the fact that routing functions in such a network either has to know the current location, i.e. IP address mapping of every application (as is the case with the centrally routed MSN and ICQ networks) or alternatively in the case of a distributed network (such as for example JXTA) use flooding to ask other nodes if they know the mapping of a user that they don't know themselves. This is the same problem that was discussed earlier regarding the use of Ethernet addresses for global routing since these are not bound to a location either. The fact that flooding has to be used makes very large scale distributed overlay networks (i.e. consisting of a large numbers of routing nodes) using this address scheme difficult to implement because the request traffic load of a flooding system does not grow linearly with the number of routing nodes, but instead grows exponentially. Existing large scale distributed overlay networks such as the MMS network in fact uses location bound addresses since the telephone number that is used to address a message is usually bound to a certain area, making it possible for the routing functions to route based only on the country and operator prefix of the user. Demands of number portability on operators means that telephone numbers may not be usable as location bound addresses very much longer.

This in effect means that overlay networks using address schemes that do not bind an address to a certain area are mainly only useful for lower bit rate applications where data from many users can be relayed/routed more or less centrally or at least in a small network where it is either possible to keep track of all current application to IP address mappings in each routing device or flooding can be used without too much traffic being generated. This is the case with the MSN Messenger and ICQ messaging services where only small sporadic messages are to be routed.

If the IP address of the device is globally routable (i.e. the device is not inside a NAT/NAPT network), an overlay network is not the only solution to the mobility problem, instead, the hostname to IP address mapping can be changed to map to a new IP address, when the address is changed. There are several free dynamic DNS services today that offer this kind of service [49].

An idea that is very similar to the dynamic DNS idea is the Session Initiation Protocol (SIP) where a global SIP username is dynamically mapped to one or more current IP addresses. Since a SIP username contains the identification of your SIP service provider, another user knows who to ask in order to get the current IP mapping of the username, just as a user knows which DNS domain to ask for a certain user's hostname to IP mapping by looking at the hostname address (see section 5.1.3).

Apart from making limited mobility possible, the technique of using overlay networks is especially interesting for devices inside NAT/NAPT networks. Even though the entities inside the network cannot identify themselves on the IP level to others outside the network, an application could identify itself on the application level. Since a device can contact the Internet, an application could contact a server, present a unique id and wait to receive data through this already established connection. Others could then send the user data by connecting to the same server and using the device's unique application level id.

This is what happens when a computer inside a NAT or NAPT network logs on to the ICQ or MSN chat services. Even though the computer does not have a globally routable IP address, messages can still be sent to it (via the chat server) using the unique application level id as an identifier

An application level network in the stationary network designed to relay continuous high bandwidth streams of data rather than only sporadic text messages would have to consist of a very large number of high end ALRs that are geographically distributed in a smart way, closely coupled to the physical network in to avoid certain routers becoming congested. This sort of world wide investment is obviously not a cheap nor easy task (although possible).

For exactly these reasons, services such as MSN Messenger and ICQ currently only use the overlay network for messaging and technical traffic, voice conversation, video conversation, and file transfers are not done in the overlay network, but sent directly between computers on the IP level and hence only the messaging function works if both users are inside NAT/NAPT networks. The random addressing scheme discussed above also means that if the services were to be very well distributed the problems due to flooding would increase.

Note that there are messaging services that do allow for limited data transfers inside the overlay network. Yahoo Messenger [14] allows a user to send files of up to 1.5 Mb in size to users who are not online. This means that two users that are inside two different NAT/NAPT networks can communicate files of this size (bit rates during such transfers have not been measured).

2.5 What has been done so far?

Many products exist today that use P2P technology in different forms and much research has been done in the field. Different forms of P2P networks have been presented over the last couple of years and they have been evaluated in numerous theses and projects. While Napster was the first P2P application to make it big, Gnutella is often viewed as the first implementation of a truly distributed P2P network.

2.5.1 Stationary networks

Messenger services such as MSN Messenger, Yahoo Messenger, and ICQ , as well as file sharing services like Napster have been around for quite some time (although Napster is currently not active). These services all use the centralized overlay network approach such as the one shown below.

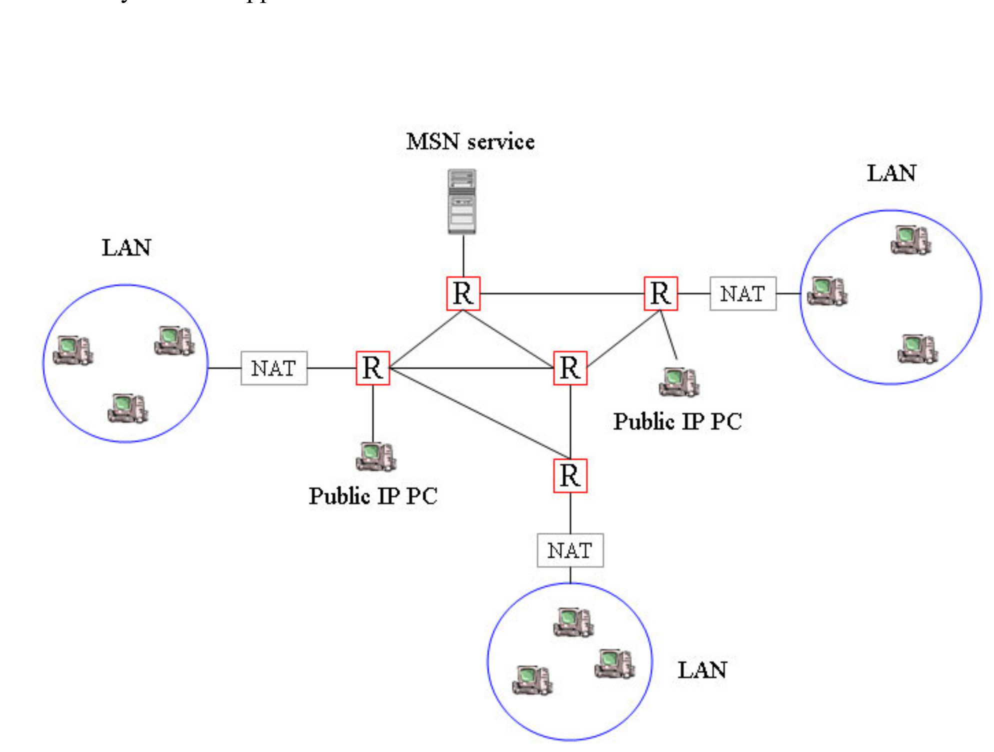


Figure 6. A network with different entities

The server is in fact likely implemented as a server park with load balancing, but the servers are still situated more or less in a single geographical position.

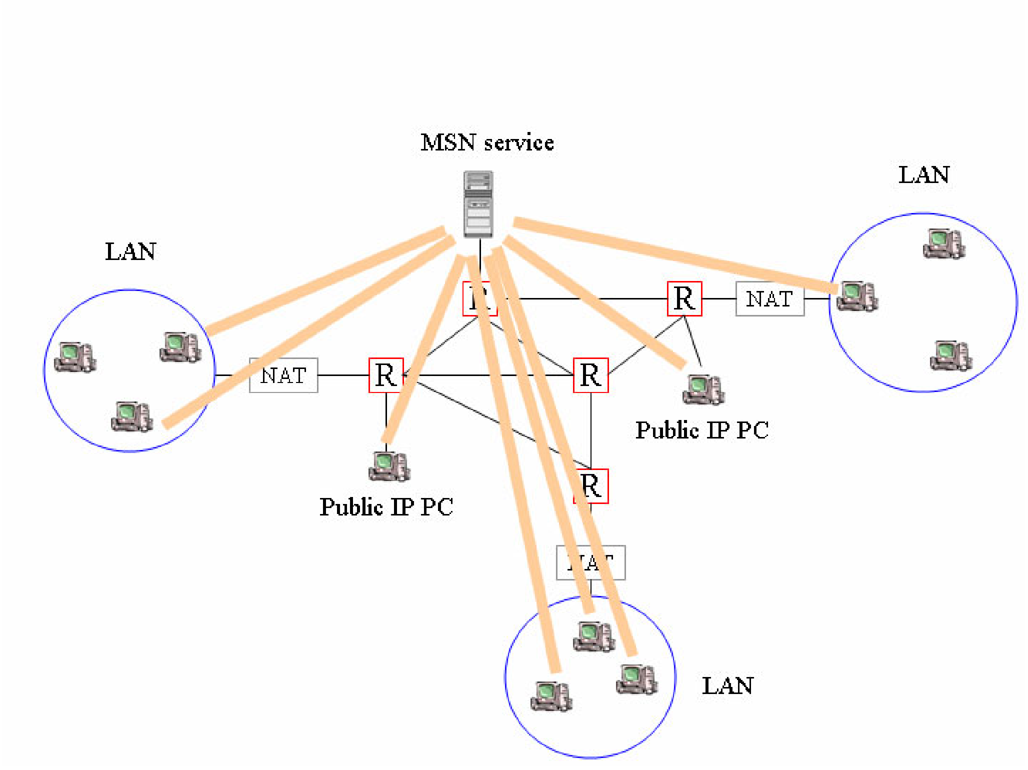


Figure 7. A centralized overlay network is implemented

All applications connect to the same place; in this way, it is easy to search for other peers and information since the server can have a list of all connected peers and the information they might share. It is also easy to enforce security since a central server can have a list of peers that are allowed to connect, create or join groups, etc.

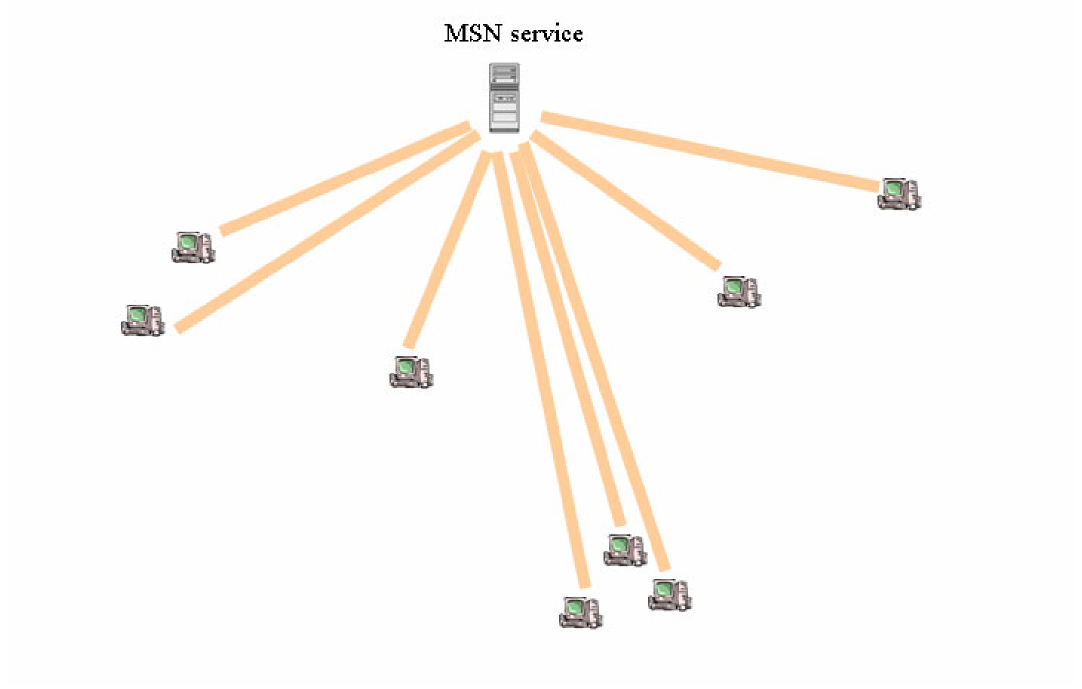


Figure 8. The overlay network is transparent to the users if data is routed inside it

The network does not have to use any form of flooding and it puts very little strain on the peers that are part of it. However the central server may easily become a bottleneck if a lot of requests or messages are sent to it. Because there is only a single (or very few) routing point(s) in these networks they do not handle large bandwidths very well. Hence these overlay networks are usually only used for small bandwidth messaging or search requests, actual file transfers are instead performed directly between the peers on the IP level (thus some of the peers in the above picture cannot exchange files). Recently there are messaging applications such as Yahoo that do allow for relaying and caching of small (1.5 Mb) files inside the overlay network if a user is not online (or is behind a NAT or firewall).

Gnutella was formed to have no central point and implements what is called a fully distributed overlay network..

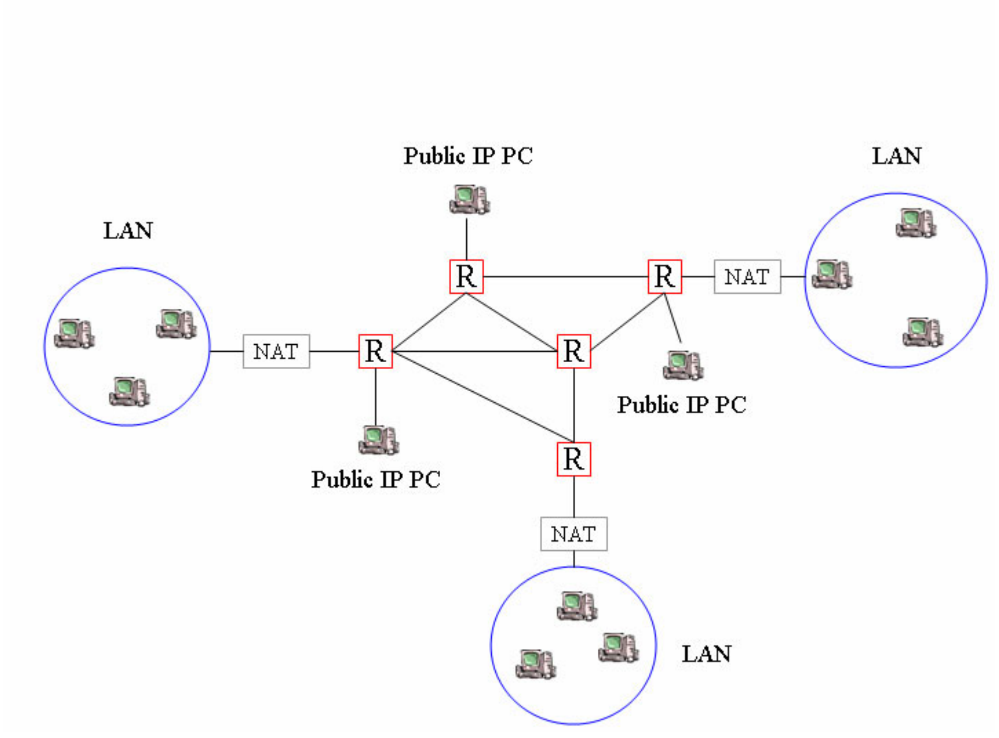


Figure 9. A network with different entities

Instead of users connecting to a central server, a user connects directly to another user who is hopefully in turn connected and has knowledge of yet more users, and so on, until a large network without any central point is formed.

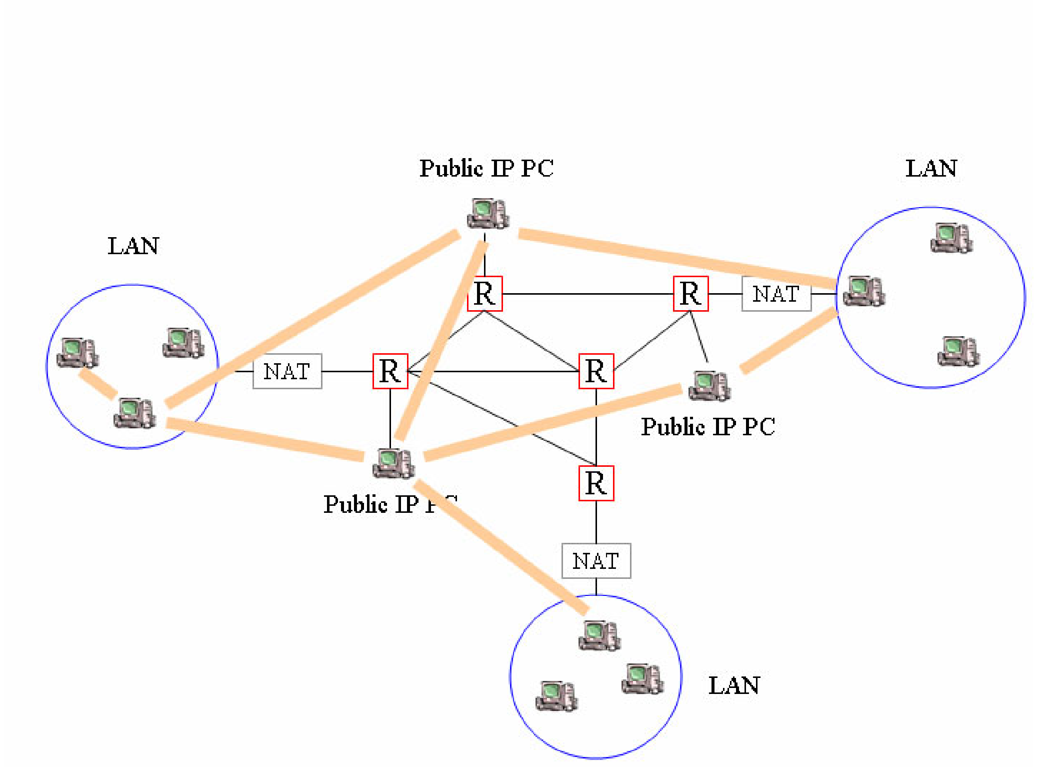


Figure 10. A fully distributed network where all hosts can act as ALRs

To search for information in this kind of network, a user asks the nodes that he is connected to if they have the information, if they don't they in turn relay the request to other nodes that they are connected to, and so on, until the information is either found or is regarded as not available (as stated earlier, in Gnutella the number of steps to relay a request was decided to be seven, in order that the requests should not flood the entire Internet).

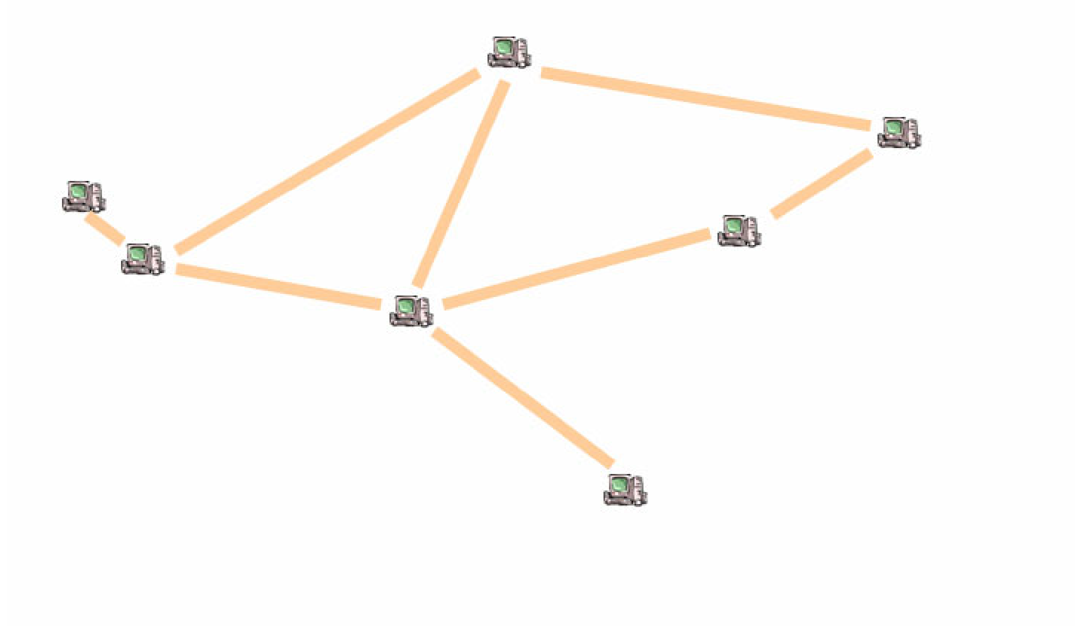


Figure 11. The overlay network is transparent to the users if data is routed inside it

While this makes for a very robust network that is largely independent of certain routes or servers, research showed that as the popularity of the network grew and the number of search requests increased, enormous amounts of traffic was generated [6]. After a while, just the requests that poured into a peer while online saturated the connection of a normal modem user.

Another problem in Gnutella that is present in most P2P networks was that there were no means of control or mechanisms to punish bad behaviour. In a central server system like Napster it is fairly easy to keep an account for every user and not allow access if the user doesn't share or misbehaves. The Gnutella network had to rely on people's good intentions. This resulted in implementations disregarding the 7 step limit in order to increase their chances of finding content, resulting in the network becoming even more flooded. Apart from this, the "Free riding on Gnutella" [15] paper calculated that 70% of the Gnutella users shared no content and 1% of the users provided more than 50% of the responses to a given request. The network thereby lost its robustness and distribution advantage and resembled a client server network where 1% of the nodes were acting as servers providing most of the content and hence were always heavily congested.

Another fully distributed P2P network that was created with the notion of total freedom of speech rather than file sharing is Freenet [16]. Freenet resembles Gnutella, but with the difference that while the response to a request in Gnutella contains the name of the computer that responds so that the information could be retrieved directly on the IP level (i.e. not inside the actual overlay network, which is only used for requests), this was not the case in Freenet. A request in Freenet only contains the name of the last peer to relay the message, when a peer receives a request, it puts its own address as the sender and relays the request to another peer while saving a message ID for that particular request, so there is no way for the next peer to know if the request actually originated from the peer he received it from or was only relayed by him. The response to a request is sent back to the peer that the request was received from, that peer in turn use the request ID to relay the response back to the peer that it originally received the request from and so on until the response has backtracked all the way and finally reaches the peer that originally requested the information. Unlike Gnutella the requested data is also sent in the same way (i.e. inside the overlay network). Hence there is no way of knowing who you are actually downloading from or who is downloading from you. It is obvious that this scheme puts even more strain on the individual peers as they now have to cache data as well as process each response. However, Freenet unlike Gnutella does enable two users behind different NATs networks to share information if there is at least one intermediate user outside the networks

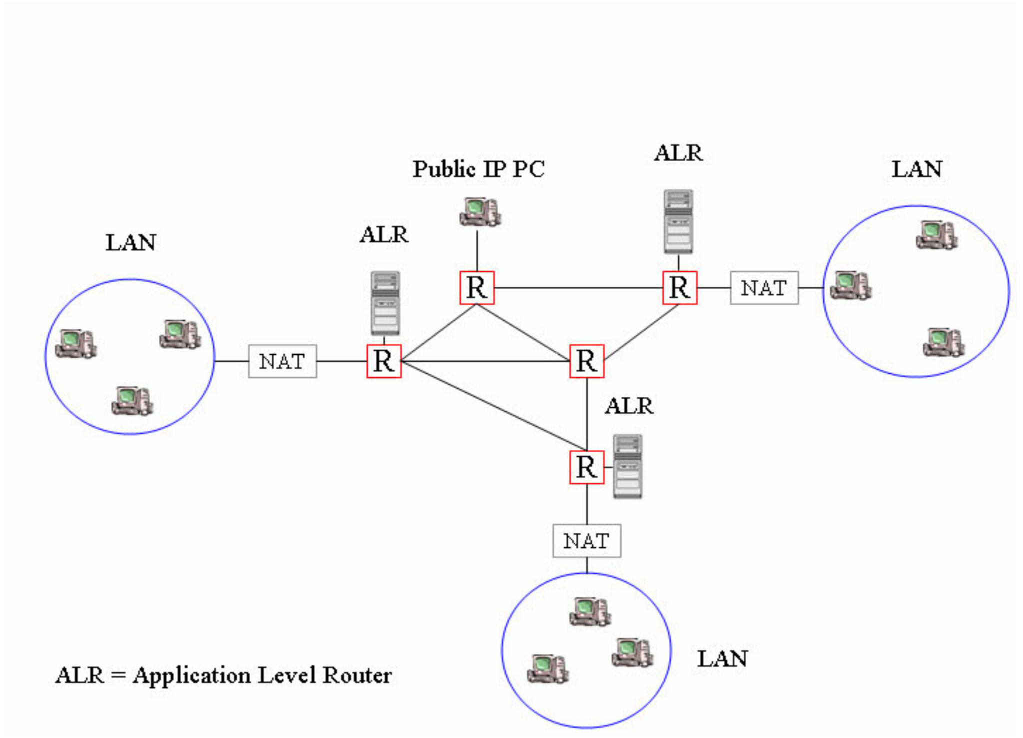


Figure 12. A network equipped with application level routers (ALRs)

Learning from Gnutella and Freenet, other systems have evolved that use a combination of centralized and distributed systems. These so-called semi-distributed overlay networks use special ALRs, i.e. computers running special routing and group software, scattered across the Internet. These ALRs are sometimes called hubs (as in the Direct Connect [13] file sharing network) or Rendezvous (as in JXTA). They aggregate information about a local group of peers and what information and services they offer.

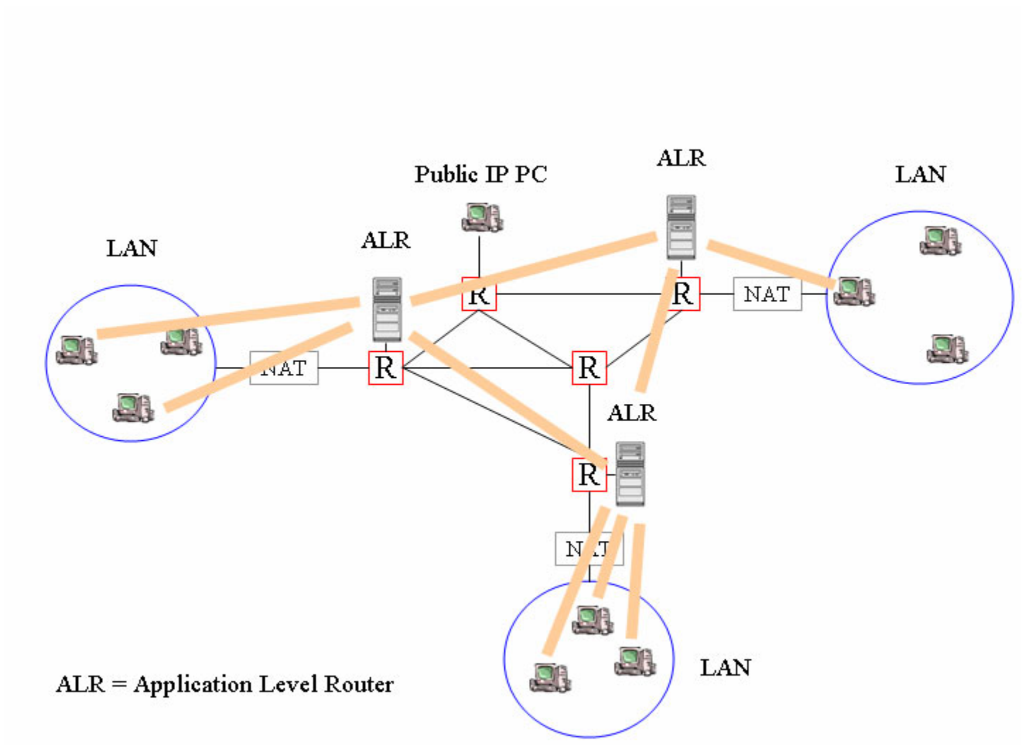


Figure 13. Users connect to the ALRs in order to get overlay network functionality

A peer that wants to find information sends a request to the ALR that it is connected to, if this ALR has no knowledge of another peer that has the requested information, it sends the request to another ALR that has knowledge about other peers and so on. Hence a request is only flooded within the network of ALRs rather than between the actual peers as in the fully distributed case. The ALRs can also potentially propagate messages from a peer to a group of other peers and thus provide "multicast" functionality on the application level.

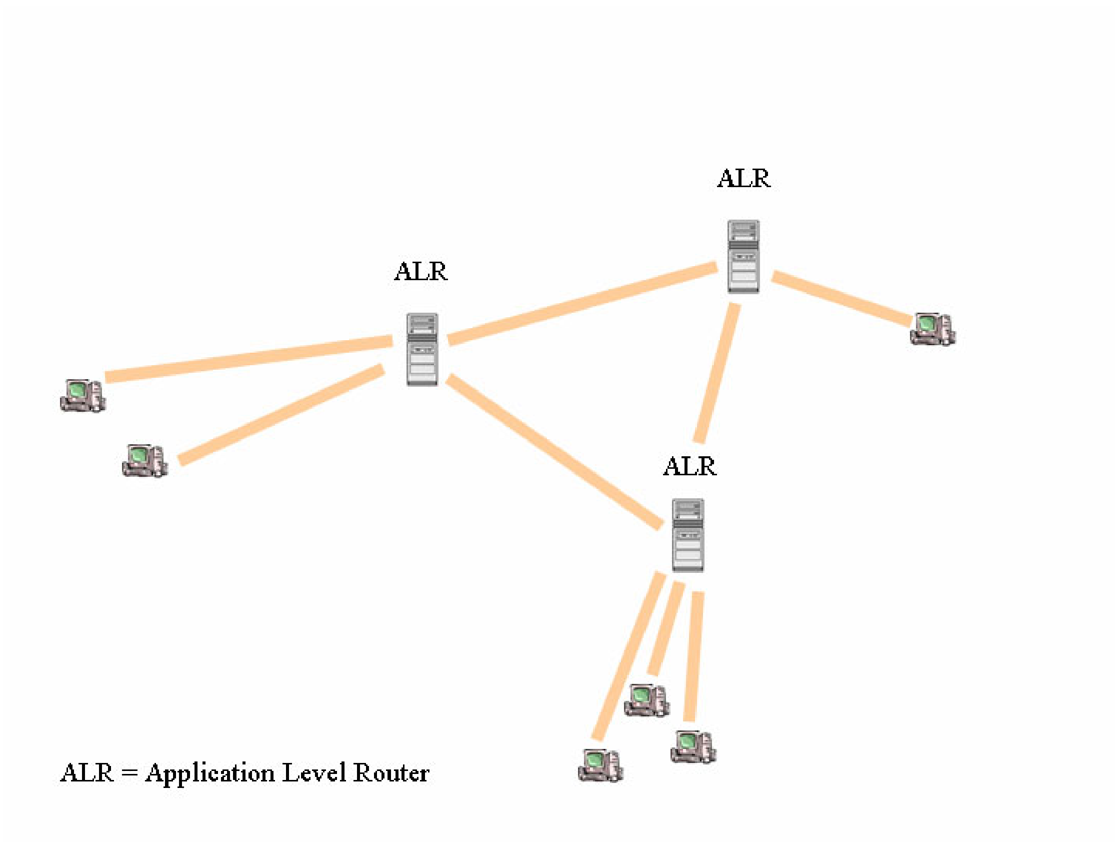


Figure 14. The ALRs implement most of the functionality offer a reasonably distributed network

In this way, the strain on the peers in a fully distributed system (where peers have to answer each and every request) is reduced while the problem associated with a central server solution being overloaded is also avoided. Much of the robustness of the fully distributed P2P networks is kept as there is still not a single point of failure in such a semi-distributed network.
Applications that use the P2P concept for CPU sharing (sometimes called Grid computing) such as SETI@home and FOLDING@home have already been discussed. While it may seem strange that CPU sharing has not become more popular in the corporate world with most of a company's computers being idle all night long (and available for data processing), it is quite difficult to find tasks that are easy to divide and distribute. A lot of computations simply cannot be done in parallel and are either very simple or very complicated requiring a lot of time. SETI is a good example of a task where demanding and independent calculations need to be done on many fairly small data blocks.

Abacast [17] and Blue Falcon [18] are two examples of bandwidth sharing applications used for streaming data to large numbers of users. They both use the concept of the listening application relaying the stream it listens to.

The first listener actually connects to the server that is providing the stream. The origin server that provides the stream keeps track of who is listening to or watching the stream and when another user asks the server to send the stream, the server might tell the application to instead pick up the stream from one of the users that is already connected. In this way the stream spreads through the network in an upside down tree fashion resembling IP multicast, but at the application level. It means that a user or company with very limited resources in the form of hardware and/or bandwidth could potentially serve a stream to a huge number of users. Buffering is used to solve the problem of a node in the tree switching off his/her computer, during the time the buffered stream is being played, an alternative node is found with the help of the server. As is the case of the adhoc networks, the quality of the stream could (at least theoretically) improve as more users attach. For example, two dial-in modem users with poor bandwidth may experience low quality if they are the only ones trying to connect to a user who is streaming from a modem connection himself.

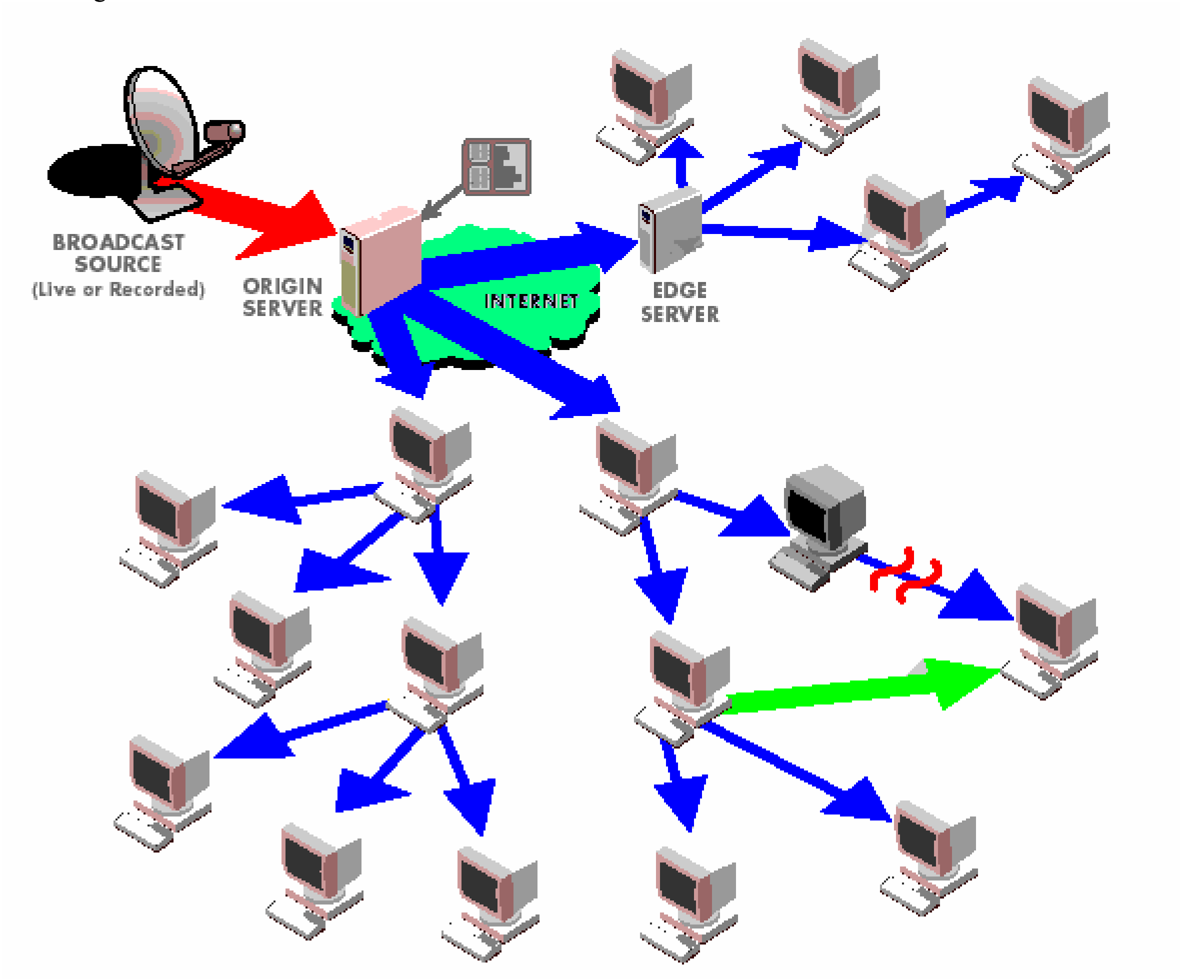


Figure 15. The Blue Falcon solution to peer-to-peer aided broadcasting

The modem connection may only be able to serve one user with reasonable quality, and since that user does not have enough bandwidth to spare so that he could share the stream, the next user connecting from a modem might experience difficulties. However, if the next user to connect is connecting via a high-bandwidth interface, he could take over the original stream and serve several low-bandwidth users and the quality would improve because the bottleneck was removed. The topology of the tree can hence be changed dynamically.

2.5.2 Adhoc networks

Common to most adhoc networks is that a device should be able to advertise it's existence and the services that it can offer to others (e.g. if it's a printer it should advertise that it can print documents and so on). In order to be able to build larger networks the devices should also be able to relay messages to other devices. Some technologies support discovery of other devices in hardware (such as Bluetooth) while others do not.
The Konark project [19] is a Service Discovery and Delivery Protocol design for Adhoc Networks. This protocol is designed to be used as a middleware to enhance the capabilities of currently available low level adhoc network technologies in general. There are currently implementations for WinCE 3.0 Compaq iPAQ PDAs running the Jeode java virtual machine [50]. There are many other projects dealing with the problems of discovery and routing in adhoc networks using different schemes. For example, Location Aided Routing (LAR) [20], where the devices keep track of each other using GPS, have been explored.

Siemens has a project looking into building nomadic adhoc networks using the JXTA platform on their Simpad devices [21]. The JXTA platform offers the overlay network services in this project.

2.5.3 Cellular networks

P2P or overlay networking in cellular networks is not as deeply explored, at least not as defined P2P projects. There are services that could very well be considered P2P. For example, Nextel, RIM, and Motorola (using Motorola's iDEN technology [64]) cooperate to offer packet based voice service between Nextel\Blackberry [22] messaging devices as well as certain specialized proprietary Nextel mobile phones using Nextel's direct connect service [23] (that offers packet based voice service directly between two or several devices in the same Nextel service area giving the possibility of group communication channels competing with legacy radio systems for group communication). Apart from the voice data being able to be locally routed, a packet based voice group is only virtual and does not lock the kind of network resources that a regular circuit switched call/groupcall would. Nextel offers flatrate billing for the direct connect service and is marketing it as a digital walkie talkie service, this can be a bit misleading since it gives the impression that the devices communicate directly with each other even where there is no base station infrastructure as most walkie talkies can, however this is not the case since the radios are not actually communicating directly with each other but are still using the base stations to send and receive data. Of course, a regular phone call might also be seen as an example of P2P communication and data can certainly be sent over a circuit switched connection. However, the resource allocation of a circuit switched channel in the network makes it practically impossible for large groups of devices to be always on line; which is what is wanted for services such as a push-to-talk "radio channel" and instant messaging. In the packet case, general packet switched data can be sporadically sent to an online device, so any application on a device could communicate and collaborate with a corresponding application on one or several other devices in a P2P fashion. The Blackberry messaging devices have provided services such as emailing for some time using the MOBITEX [24] network and recently also the GPRS network. The problem of a device in a network such as GPRS not being able to be contacted directly by another entity from outside of the network is solved by the Blackberry devices contacting and keeping in touch with a Blackberry Enterprise Server situated in the fixed network (maybe in the corporate head quarters) which can then forward messages and email via this mobile initiated connection. In the case of the Nextel direct connect service the problem does not exist since the service is only offered to users in the same service area (i.e. they do not have to be reached from outside the network), anyone inside the Nextel network can address any other device also inside the network directly using the private addresses (which are valid and routable inside the network).

The idea of using a server in the stationary network to enable devices without unique IP addresses to contact each other is also used in the JXTA concept of wireless P2P. In JXTA the solution aims to be a bit more distributed. The special peers called rendezvous behave as a distributed version of the enterprise server, where peers can "register" so that other peers can contact them via the rendezvous. However in JXTA, peers do not actually maintain open connections to the rendezvous so if a gateway or firewall implementing NAT/NAPT is between the rendezvous and the peer (as is usually the case in GPRS), the peer will either have to poll the rendezvous for new messages (HTTP can be used through many firewalls) or if polling is undesirable, a mobile device could keep an open connection (a socket) to a relay server with a valid IP address in the fixed network (outside the NAT/firewall) that in turn can be contacted by the rendezvous and relay messages to the device over the open socket. The idea of a relay could prove useful for other tasks than simply relaying everything it receives. For example it could and is currently being used to filter unwanted traffic that a user in a GPRS network would not like to pay for. It could also be used to enhance the power of a limited mobile device by handling demanding tasks that may require a lot of calculation or bandwidth.

3. Implementation

3.1 Why this particular implementation

3.1.1 Proving a concept

The application written was chosen in order to prove several of the concepts discussed earlier in this report. One of these concepts is device to device data connectivity for different applications using a distributed overlay network. Another is the concept of keeping TCP connections alive in order to be able to be contacted. The concept of being able to group applications virtually is crucial and finally, one of the most important aspects is to prove that all this could be achieved **without** any support from the operator network.

3.1.2 Available hardware and software

The particular implementation was also chosen with regard to the hardware that was available at the time of the project initiation. Since the Symbian OS 6.1[25] based phone Nokia 7650 was just released, this was chosen as the base for the implementation.

This particular phone has several features that made the potential benefit of using peer to peer communication more obvious. The first and foremost was that it uses Symbian OS. This means that applications can be written in C++ directly for Symbian OS, giving a developer full control over the phone hardware as opposed to for example Java2ME that many other phones run. Java2ME does not give access to any hardware specific features of the phone, unless the manufacturer provides specific APIs for this. Since the Nokia 7650 has a built in camera, this hardware control was very much desired as it enabled an application to directly utilize the camera hardware.

The ability to use the camera meant that one of the biggest issues of why there would be any need for peer to peer functionality was solved, i.e. the issue of if there was actually any interesting information to communicate between devices (see section 1.5).

The second benefit of using a Symbian OS based phone was the availability of a standard UNIX socket implementation as opposed to using Java2ME as its MIDP 1.0 specification allows only HTTP requests. Although there are now personal Java implementations available on some camera equipped mobile devices allowing full Java sockets, these were not available at the time the project began, furthermore, the problem of accessing the camera from a Personal Java application would have remained.

Writing a Symbian OS application means that a standard TCP connection can be created and kept open for incoming data, avoiding the limitations of the HTTP application level protocol. Due to the NAT/NAPT implemented in many GPRS networks, this particular behaviour is very important.

3.2 Technical description of the implementation

3.2.1 Architecture

The server side of the overlay network is written in Java and implemented as a JXTA network (see section 2.1) consisting of the core JXTA component, the rendezvous servers. The JXTA concept of relay servers representing the mobile devices with a persistent ID in the JXTA network is used. This could be seen as the equivalent of a home agent in Mobile IP [51]. However, the mobile device HTTP relay server for Java2ME implemented and offered by SUN Microsystems was not used. Instead a specialized relay was written that allows a mobile device to connect using TCP or UDP and keep the connection open for two way communication, rather than using HTTP requests. This saves the device from having to poll the relay for new data and also saves some overhead that the HTTP protocol would have introduced. Another factor making it necessary to write a specialized relay was that the APIs offered by SUN at the time for utilizing this relay were only available for Java2ME. While the devices used as clients in this network are capable of running Java2ME applications, it was necessary to write the client applications in C++ (for Symbian OS) in order to be able to access the available resources on the devices, such as the built in camera and the local file system.

This means that a proprietary protocol is used between the device and the JXTA relay in order for the device to control the relay. Thus, this relay is application specific since the protocol was designed with a particular application in mind.

If an operator decided to build a full scale general overlay network, the protocol used to control the relay would have to be generalized and implemented as an API that could be implemented by third party applications. If it was decided to use JXTA as the basis for such a network, the JXTA APIs should be ported to the Symbian OS. Alternatively, special Java2ME APIs could be written that would allow a user to access the requested device hardware from a Java2ME application. In that case, the existing JXTA Java2ME APIs along with the HTTP relay could be used.

It was not considered within the scope of this thesis to create a full port of the JXTA APIs in order to offer a truly general overlay network. This application was only intended to be used to evaluate the possibilities of the technologies rather than creating a final full scale solution. Thus in this version, the relays are tailored for the specific applications in this version.

Two applications were created to run on the JXTA network. The first application is a chat service where users can chat in a group (currently a static group). They can ask for a list of the users that are currently in the group and messages are sent to everyone currently in the group. Users have a username that is shown along with the received message.

The second application is a messaging application where users can send pictures (JPEG images) in "high" quality (JPEG quality factor 25 is used), video clips (MPEG-4), or stream low quality (JPEG quality factor 10 is used) images from the camera continuously to a group (different from the chat group).

All data is sent to all members of a group, one to one messaging is not implemented since it was the group casting feature that was deemed to be the most interesting.

The applications were developed for the Nokia 7650 phones. A PC client was also developed in order to show that any hosts on the Internet may participate in the overlay network. This PC client was written in Java2SE and utilizes the JXTA HTTP relay (**not** the mobile device HTTP relay that was discussed earlier) in order to connect to the JXTA network if it is situated inside a LAN using NAT or a firewall which prevents an incoming TCP connection.

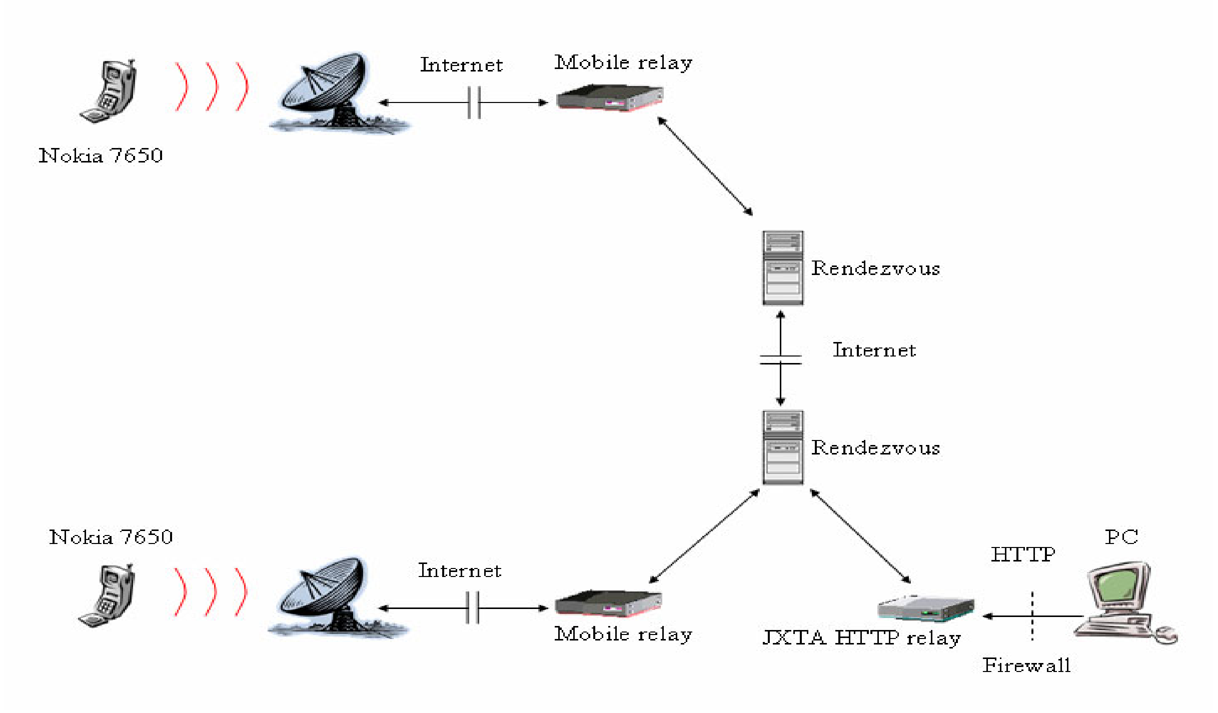


Figure 16. The set up of the implementation network using JXTA

The JXTA HTTP relay in the diagram is already included in the SUN Microsystems implementation of the Rendezvous. If a peer wishes to utilize the HTTP relay function of a Rendezvous, it simply contacts the

Rendezvous on a specific port using HTTP. The mobile relay that was written for this implementation on the other hand is not included in the rendezvous but is a separate program, although it could be run on the same physical machine.

3.2.2 Flow charts

The following flow charts describe the functionality of the relays and respective client applications. They are intended to make understanding of the general design of the implementation possible without the having to spend large amounts of time reading code.

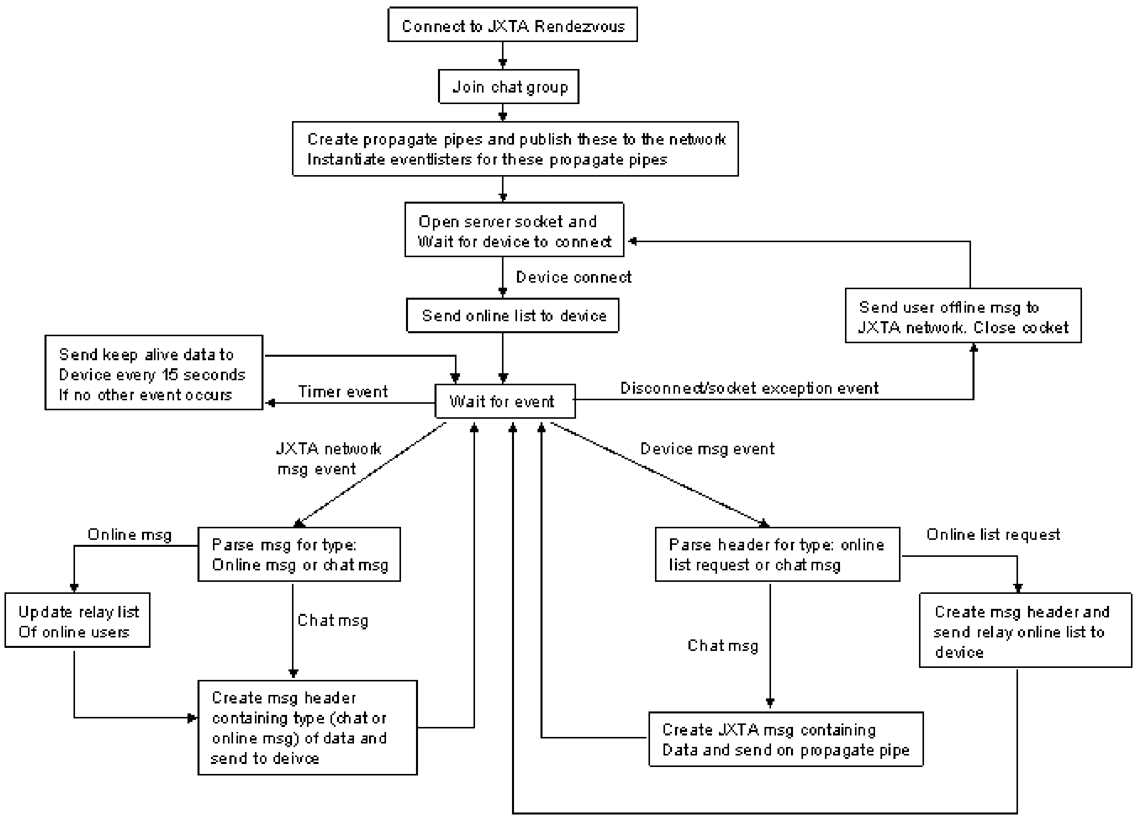


Figure 17. Flow chart describing the states and behaviour of the chat relay in the fixed network

The first flow chart describes the behaviour of the chat relay. A chat application connects to this relay in order to be represented in the JXTA network. Please note that this relay does not implement the picture and movie clip features, they are in a totally separate application in the phone, therefore I chose to use a totally separate relay in the network in order to preserve the analogy of each application implementing a separate JXTA platform and thus having a separate IDs. If both applications would connect to the same relay they would be represented by the same JXTA ID in the JXTA network. In the future, when creating a general purpose relay the best design would likely be a single process that instantiates a thread for a separate JXTA platform for each application that connects. This gives connecting applications transparency as compared to the current implementation where an application has to know which relay to connect to.

The TCP based keep alive function is implemented in the chat relay. The quite short interval of 15 seconds means that the GPRS device will also stay in its ready state where propagation time is low (this is discussed further in section 5.2.3). This was necessary in order for the demonstration to be as smooth as possible, although it will make the device consume more battery power and may negatively affect the network in large scale implementations.

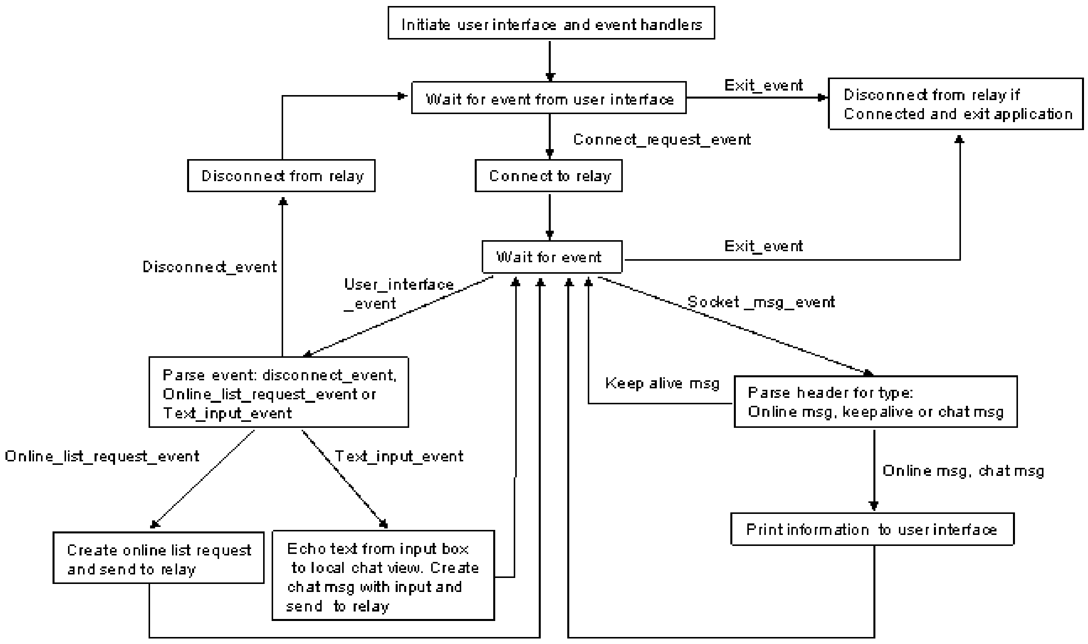

Initiate user interface and event handlers
Wait for event from user interface
Exit_event
Disconnect from relay if Connected and exit application
Connect_request_event
Disconnect from relay
Connect to relay
Wait for event
Exit_event
Disconnect_event
User_interface _event
Socket _msg_event
Parse event: disconnect_event, Online_list_request_event or Text_input_event
Keep alive msg
Parse header for type: Online msg, keepalive or chat msg
Online_list_request_event
Text_input_event
Online msg, chat msg
Create online list request and send to relay
Echo text from input box to local chat view. Create chat msg with input and send to relay
Print information to user interface

Figure 18. Flow chart describing the states and behaviour of the chat client application

The chat client can either send a chat message to the relay or request an online list that the chat relay keeps updated. Apart from receiving the entire online list when requested, the client will also receive information when particular users go on or offline.

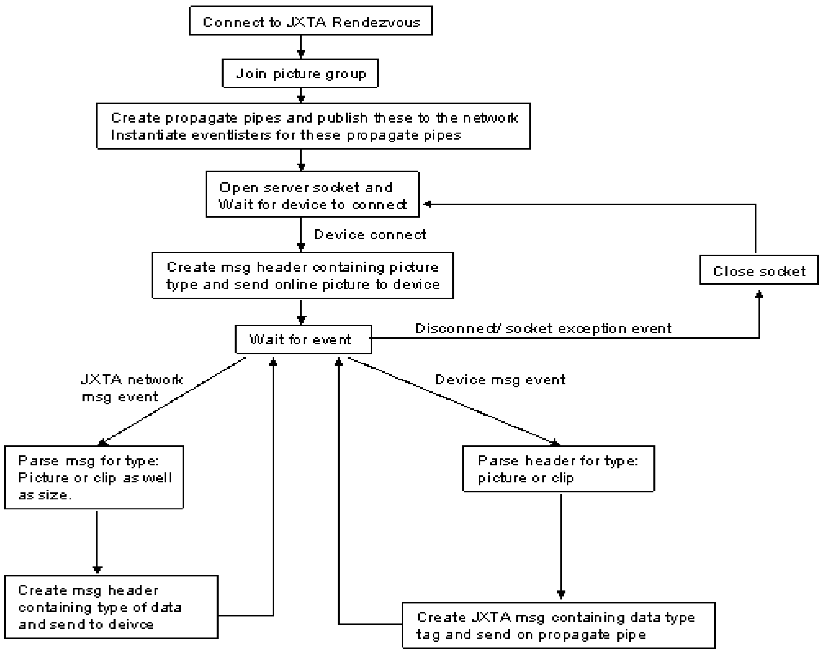

Connect to JXTA Rendezvous
Join picture group
Create propagate pipes and publish these to the network Instantiate eventlisters for these propagate pipes
Open server socket and Wait for device to connect
Device connect
Close socket
Create msg header containing picture type and send online picture to device
Wait for event
Disconnect/ socket exception event
JXTA network msg event
Device msg event
Parse msg for type: Picture or clip as well as size.
Parse header for type: picture or clip
Create msg header containing type of data and send to deivce
Create JXTA msg containing data type tag and send on propagate pipe

Figure 19. Flow chart describing states and behaviour of the picture and video clip relay

The picture relay is somewhat simpler in the design than the chat relay. It does not implement a keep-alive function (it is expected that the chat application is run in the background when demonstrating). Furthermore it does not implement online functionality since it is intended to be used together with the chat application (as an add-on feature).

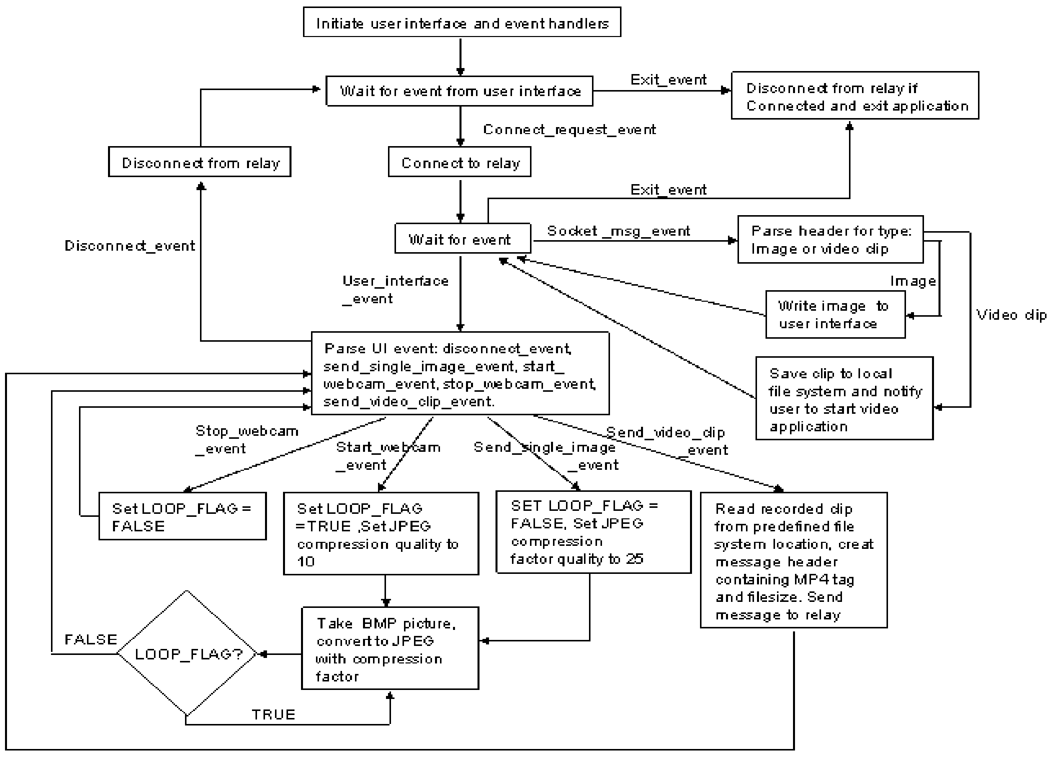

Initiate user interface and event handlers
Wait for event from user interface
Exit_event
Disconnect from relay if Connected and exit application
Connect_request_event
Disconnect from relay
Connect to relay
Exit_event
Disconnect_event
Wait for event
Socket _msg_event
Parse header for type: Image or video clip
Image
User_interface _event
Write image to user interface
Video clip
Parse UI event: disconnect_event, send_single_image_event, start_ webcam_event, stop_webcam_event, send_video_clip_event.
Save clip to local file system and notify user to start video application
Stop_webcam _event
Start_webcam _event
Send_single_image _event
Send_video_clip _event
Set LOOP_FLAG = FALSE
Set LOOP_FLAG =TRUE ,Set JPEG compression quality to 10
SET LOOP_FLAG = FALSE, Set JPEG compression factor quality to 25
Read recorded clip from predefined file system location, creat message header containing MP4 tag and filesize. Send message to relay
FALSE
LOOP_FLAG?
Take BMP picture, convert to JPEG with compression factor
TRUE

Figure 20. Flow chart describing the states and behaviour of the picture and messaging client application

The picture client on the other hand is a bit more advanced since it offers several options to the user. The options are to send a single JPEG image to the group in a high quality mode, or stream images to the group in a low quality mode (generates images of about 1Kbyte giving an estimated frame rate of 1.5 -2 fps depending on GPRS conditions). The user can also record an MPEG-4 videoclip using Hantro [26] recording and playing software (any recording software and format can be used as long as the file is named video.mp4 and is placed in the same catalogue as Hantro uses by default). The user can then send this clip from the application. If the user receives a clip from another user a message pops up informing the user that a clip has been received and that the videoplayer should be started. The received file is also placed in the default catalogue and named video.mp4. Hence this received clip is played by default when the player is started (note that old clips are overwritten unless renamed).

3.2.3 Development tools

Series 60 SDK 1.0 for Symbian OS, Nokia edition was used for writing the Symbian C++ code. The Series 60 SDK is free to download from Forum Nokia [74]. Microsoft Visual Studio 6.0 IDE was also used. Application wizards for Microsoft Visual Studio are included in the Nokia SDK along with several example applications.

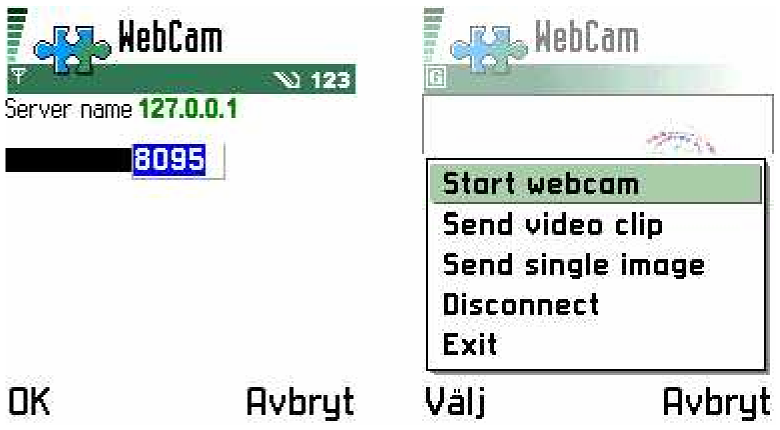
Java2SE 1.3.1 was used to write the mobile relays. GNU Emacs was used as development environment.

The overall experience of coding for series 60 Symbian OS devices is that Symbian C++ is a complex but very powerful language with a steep learning curve. It is very structured and once the coding idioms of the series 60 system are understood it is quite easy to create very well structured code. Most services, such as sockets, camera, file system, etc are implemented as asynchronous servers, the calling function being the client (issuing a request and listening for a response event). This makes for very compact main routines and easy multitasking but also means that care needs to be taken for events that may need to be synchronous.

The 7650 phone just having been released when this thesis was initiated meant that some bugs were encountered, but these were few and could be circumvented by coding around the problems.

3.2.4 Screenshots

Below are screenshots from one of the applications that were developed for the overlay network. The left screenshot shows initial screen that the user is prompted with on start up. The user can modify the IP address and port of the default ALR to one of his choice. The right screenshot shows the user menu once the application has connected to the overlay network.



Server name 127.0.0.1

8095

Figure 21. Screenshots from the cellular clients showing the ALR address and menu choices

4. General analysis

4.1 Possibilities of using overlay networking in mobile networks

4.1.1 Cellular overlay networks

Due to the nature of cellular networks, mobile devices do not communicate directly with each other, but rather via an infrastructure of base stations, etc. Thus, in systems such as GSM there is nothing similar to the Ethernet shared channel and link layer multicast that can provide direct contact between devices (using for example Net BIOS). Such functionality has to be implemented by higher layers, such as in the IP layer or entirely in the application layer in the form of a virtual network. As bandwidths as well as device capacity increases we wish to be able to utilize this (e.g. chatting in groups, looking at pictures from a camera on a phone, video conversations, playing games, etc.). Already, PDAs are quite powerful and could potentially provide a lot of unique location based computing and information to the network and to other mobile users while the device is on the go. However, if the user wants to communicate this information to another user, he or she must either be in the immediate vicinity of the other user in order to use IR, Bluetooth, adhoc mode IEEE 802.11 WLAN, or cables to communicate with the device directly, or the user has to send the information as email or a file to a server on the Internet that the other party can contact.

While it is possible to communicate with the fixed network over a cellular data network such as GPRS, networking directly between two mobile devices is usually difficult since most (but not all) operators provide dynamic private IP addresses using NAT or possibly NAPT in their GPRS networks (see section 2.1), preventing the mobile device from being able to be directly addressed by another device that is in another network. This problem is of course true for any device (mobile or stationary) that is inside a NAT or NAPT network.

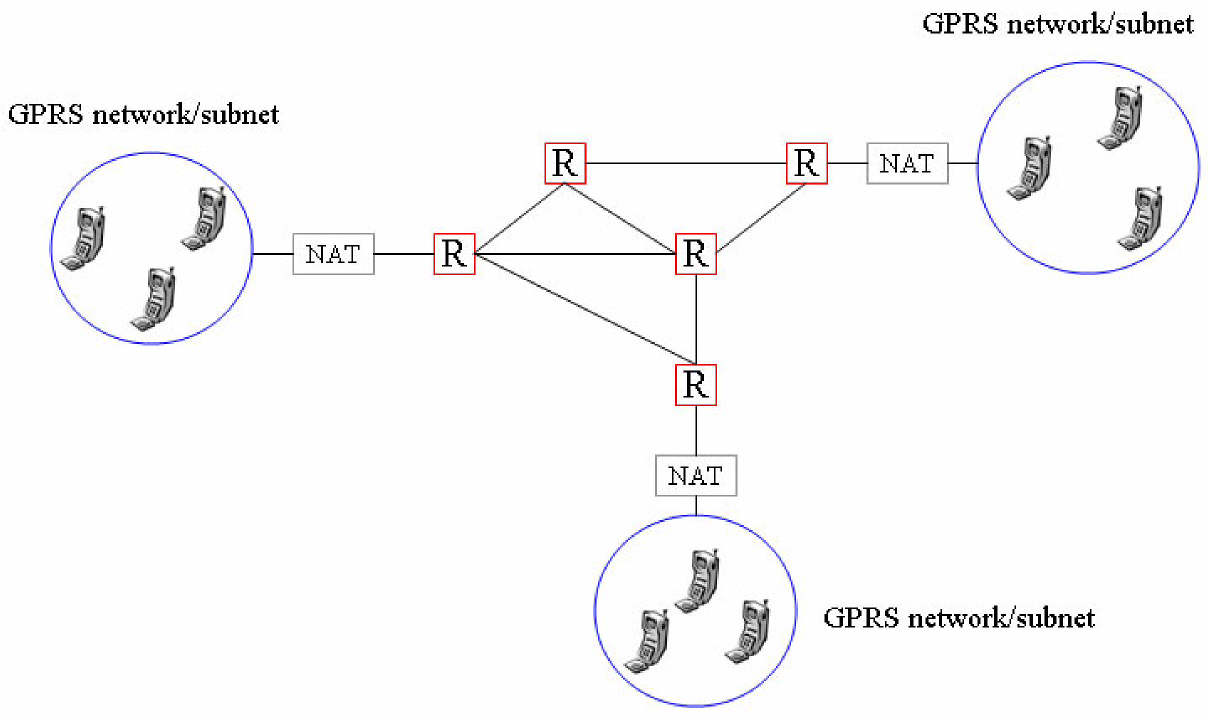


Figure 22. An example of a standard GPRS network environment

There are several ways of solving the issue of direct device-to-device data communication. For example, just as in any LAN, devices in the same GPRS address space, i.e. inside the same NAT or NAPT network can address each other directly, so users of the same operator and in the same physical GPRS address space area could actually communicate directly using their private IP addresses. The physical size of a GPRS address space area differs from one operator to another, but a typical GGSN solution such as the Siemens @vantage CPG-3000 can handle 200 000-400 000 simultaneous attached users (i.e. ready to send or receive).

However, some sort of mapping functionality would have to be implemented that maps a user's temporary private IP address to for example a user name or phone number. This could be done via for example DNS, SIP, or ENUM [52].

While this gives users true P2P connectivity, it is under very harsh constraints. No inter-operator communication is possible, communication possibilities within an operator are location dependant, Thus two users could be close to each other physically, but still in two different GPRS subnets in the above picture (i.e. connected to two different GGSNs), hence unable to communicate. Furthermore there is no support for multicasting, which would be very desirable in group communication applications.

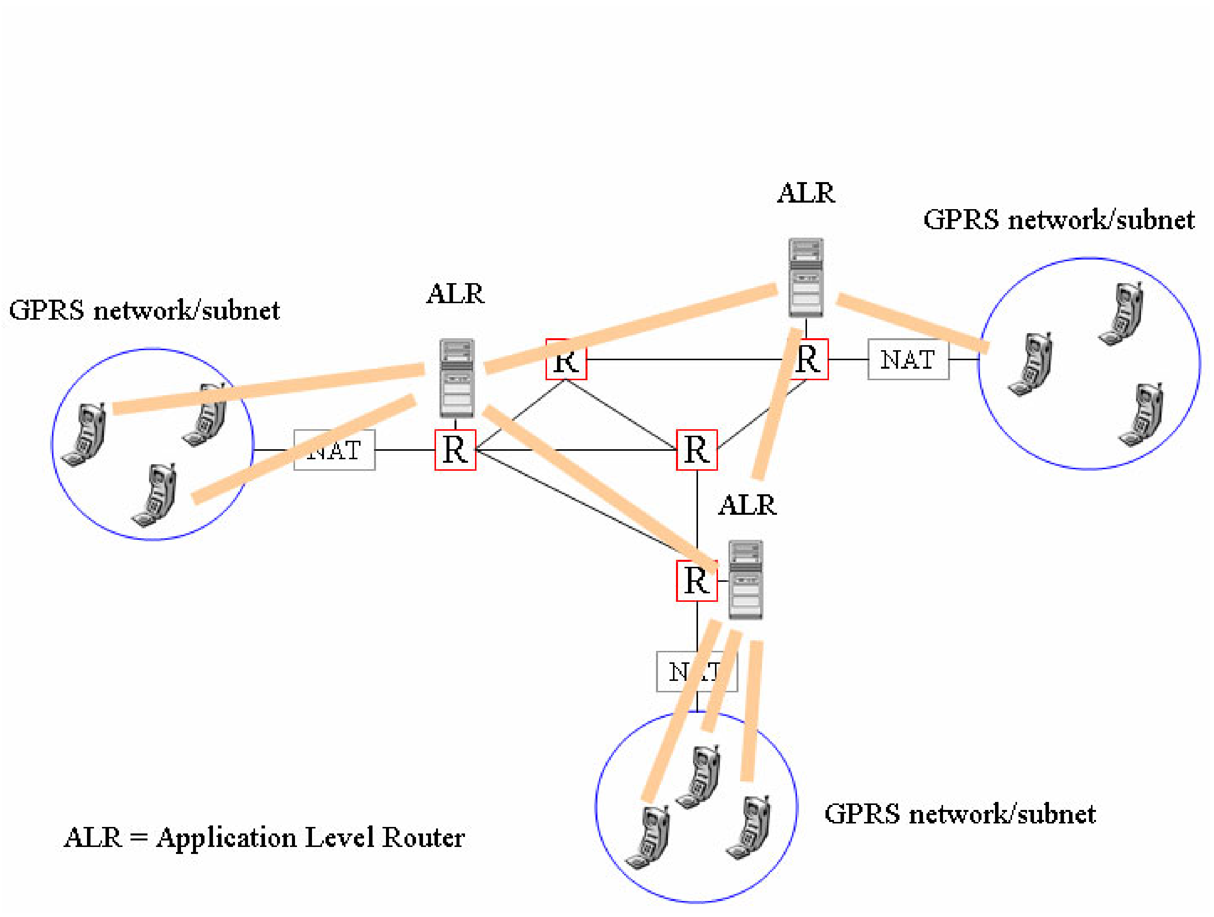


Figure 23. Devices can utilize an overlay network just as devices in the fixed networks can

An overlay network where the phone applications identify each other, using for example their already globally unique phone number could solve these problems. What is interesting is that many of the problems associated with creating such an overlay network in fixed networks are not true for a many cellular networks.

One of these factors is that bandwidths are very small in most cellular networks (not including WLAN hotspots). In GPRS over GSM the maximum bandwidth is around 40Kbit/s and in the planned GPRS over UMTS networks the maximum bandwidth will be around 400 Kbits/s. There are firewalls today, such as the Nokia IP740, claiming to do stateful packet inspection at up to 2 Gbit/s. While this is a hardware implementation, there are software firewall implementations such as Check Point's Open Server Solutions intended to run on a standard Pentium/Athlon machine with 512 MB of memory, claiming a throughput of 1Gbit/s while doing stateful packet inspection, i.e. fully application aware. This suggests that an ALR in an overlay network such as the ones discussed in this thesis might be able to handle a peak usage of about 25000 GPRS over GSM users (approx. 40Kbit/s) or alternatively around 2500 GPRS over UMTS users (approx 400Kbit/s) if only total throughput is regarded. With a standard PC today costing less than 1000$ this would give a price per (peak) user of less than

0.04 $/user in the GSM scenario and 0.4 $/user in the UMTS scenario. Depending on what kind of traffic distribution is expected (i.e. the likeliness of users amounting to certain bandwidths) it is likely that such an ALR can accommodate several more users than the number of possible simultaneous peak users, thus lowering the price per user even more.

As is discussed in section 5.2.3, the latest GGSN implementations do not support throughputs larger than 150-250 Mbit/s and existing implementations in operators networks probably support much less total throughput. This indicates that even if potential ALR performances cannot be directly interpolated from existing firewall implementations such as suggested above, it is likely that ALRs could at least be implemented in a one-to-one ratio with the GGSNs in a GPRS network without creating bottlenecks. Hence, dimensioning of such an overlay network would not be very expensive.

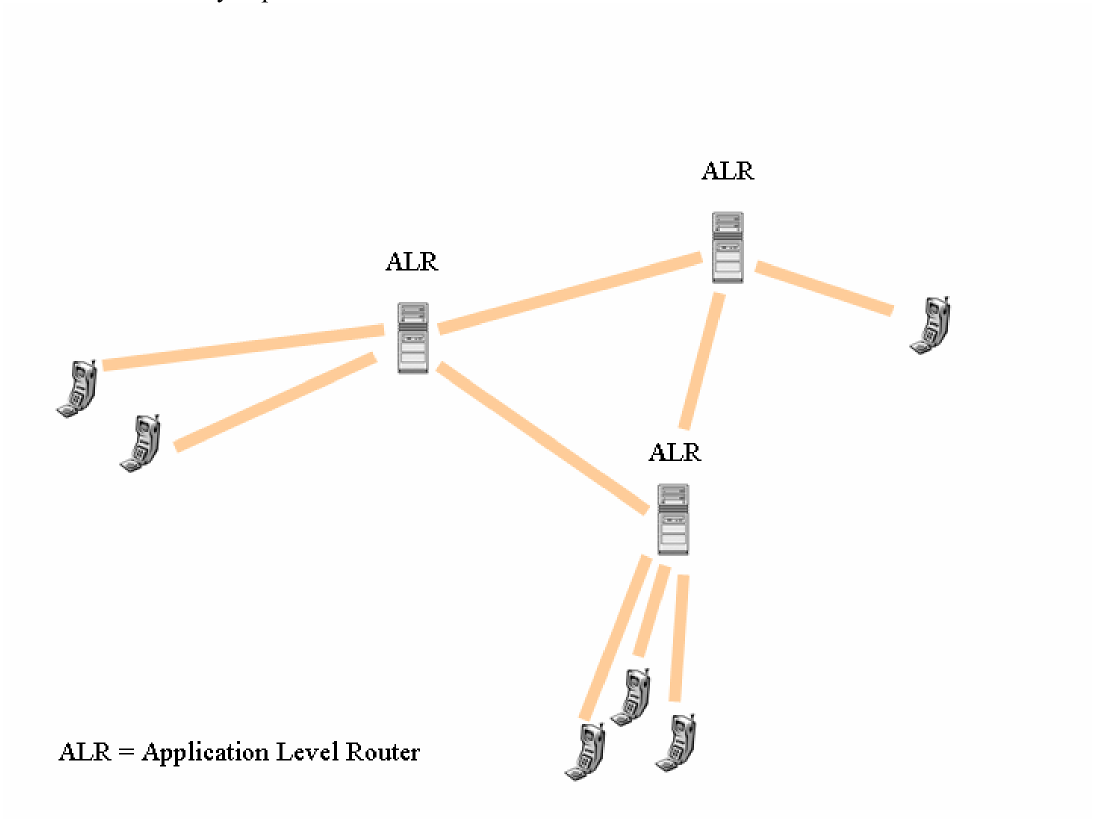


Figure 24. Groups of devices can form arbitrary and temporary private networks / groups

Another factor that is specific for the cellular networks is that since users of a GPRS network can be expected to be situated within the same operator most of the time (except when roaming), addresses based and routed upon operator names (or GGSN names) would result in rather effective data routing, avoiding the problems associated with queering for other users in a flooding scheme. Furthermore, the routers of the network do not have to be stateful in order to remember where a user is situated once he/she is found which might simplify the application. However, stateful routers may be whished for anyway in order to make them more efficient. For example, IPv6 is intended to use something called flow labels which avoid a router having to look up an IP address in the routing table for every packet in a flow. Since the router keeps track of this flow label to IP address mapping for a while it could be argued to be stateful.

Yet another factor that differs from the stationary case is that while no one owns the entire Internet, an operator does own his/her entire cellular network and can fully control the implementation of an overlay network **inside** it. If other operators choose to join the network, it can work with them as well, but there is no need for a simultaneous worldwide implementation or negotiations with other ISPs in order for an operator to offer this service to his/her whole customer base.

Because the operator owns the network all the way from the customer's mobile device to the Internet gateway, the routing function of the overlay network can be placed as close as possible to the customer's device. In the case of GPRS this could be right after the data is converted to IP format in the Serving GPRS Support Node (SGSN). This tight coupling to the physical network means that the overlay network will be very efficient since data will follow the same routes as if it was routed on the IP level. It also means that the operator can guarantee security in his/her overlay network all the way from a device to another of his/her devices or to the Internet gateway if inter-operator communication is taking place. Of course, security could instead be implemented end-to-end by the devices instead, using for example IPsec [53] if the devices are capable of this.

Although there are no existing overlay networks for arbitrary streams of data in cellular networks today, there are in fact very large message based overlay networks functioning similarly to the above described networks. One of these is the MMS network [69]. MMS is not only a protocol (in fact the WAP [65], POP3 [66], SMTP [67], and MIME [68] protocols are used), but also a network of servers situated at different operators. A device can then implement the MMS protocol on top of IP/GPRS to send pictures or text to another user via an email like system. The data is sent to an MMS server who routes it to a telephone or another operator's MMS server using the telephone number as the address. It is obvious that if the MMS servers are viewed as similar to the ALRs proposed above, they in fact form an existing full scale overlay network.

The difference is of course that the MMS network is message based rather than stream based and although it would be possible to insert any data instead of only pictures, text, etc that MMS was intended for (if the content converter node of the MMS network is enabled to handle this), this would mean implementing yet another "protocol" on top of the MMS overlay protocol resulting in even more overhead. In any case, MMS is not suitable for streaming or real time services since it is message based. Try to use it in this way would be similar to attempt streaming and real-time services over an e-mail system.

An interesting feature about the MMS network is that while there are other examples of overlay networks on the Internet (such as the ordinary e-mail server network), the MMS network implements billing as part of the network routing function. One of the big problems of building an overlay network in the fixed network is that the operator(s) would like to charge users for using the network. Because of the standardized ways of handling call records and IDs in the cellular networks, it is possible for a user to be identified and billed at any point in a cellular overlay network. However, this would be very difficult (or impossible) on the Internet as a whole.

4.1.2 Adhoc networks

The above discussion has addressed how P2P techniques could be used to create device to device connectivity on top of a cellular network. The technique is also applicable to other types of mobile networks. For example, even if a GPRS network does not provide transparent end to end data connectivity between devices, it does provide a lot of advanced functionality such as packet routing, name server functionality, and keeping track of device movement, thus making it possible to move *while* communicating. P2P technology can be used to create this kind of advanced networking capabilities in networks where none of this functionality is originally provided, such as in adhoc networks, i.e., mobile networks where devices communicate with each other directly without the support of base station systems.

Consider the case when several mobile devices are close enough to communicate directly using any of the above mentioned methods (Bluetooth, adhoc mode IEEE 802.11 WLAN, etc). Although it is possible to do direct networking between the devices, but more advanced networking is still difficult. Say for example that two PDA's or adhoc enabled cell phones are too far apart to communicate with each other directly, but they are both within range of a third device. There is currently no native support for the two devices to communicate with each other via the third device that they both can reach. In the P2P concept of JXTA every peer can be configured to relay messages to other peers that it knows about (i.e. every peer is also a rendezvous and hence a fully distributed overlay network can be created). So every device could in fact route messages between other devices on the application level. JXTA also provides a standard for publishing discovery advertisements that peers can publish via a multicast if the link layer supports this, enabling peers in the area to automatically discover and keep track of other peers movements.

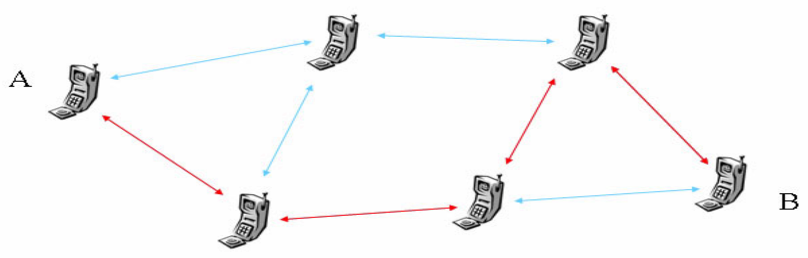


Figure 25. A fully distributed system in ad hoc environments provides advanced networking features

Since the functionality of routing and discovery is not available in the network layer of the infrastructure (since there is no infrastructure), the intelligence of the network can instead be created on the application level, enabling a “simple” network technology to have advanced functionality using the power and resources of the devices that are in the network. Since all devices in such a network would have full networking capabilities (as opposed to for example a desktop in the fixed network that usually doesn’t have full routing capabilities enabled, even if available, this is left to specialized routers), they are all truly equals i.e., they are peers.

As another example of how mobile networks might be given enhanced capabilities using P2P technology, Bluetooth should be considered. Bluetooth has some limitations built into the system that are hard to get around on the network level. It is a master – slave system, meaning that one device must be appointed master and the rest must then be set to work in slave mode. The master then acts as a server for the network, called a piconet, and the other devices communicate using the master. Because of the fact that the Bluetooth protocol only uses 3 bits for device addresses this means that since 3 bits can only be combined into 8 possible addresses and the master needs one for broadcast to all slaves, the master can only communicate with 7 devices (slaves) at a time, limiting the total number of devices in a Bluetooth piconet to 8.

On the other hand, it is possible for a device to be a slave in one piconet and a slave or even a master in another at the same time using time multiplexing, so a device can actually be part of several Bluetooth piconets at the same time. These networks are called scatternets. At present there is no native support for communication between these piconets, but if a peer application was running on the device that has access to several piconets, it could relay messages on the application level from a peer in one network to one in another using the peer id to identify the specific devices instead of the device addresses. The problem of a network having only 8 possible addresses is then solved by the peer application generating a unique id (or using the MAC address), thus making it possible to build larger networks where devices can still be uniquely identified.

This functionality could be used to create so-called nomadic temporary networks in places where there is no base station infrastructure. These kinds of systems can be said to be both robust and fragile. For example, two subnets of a nomadic network could actually depend on a single device that can reach them both, making it very fragile. On the other hand, in a situation where there are a lot of units roughly uniformly distributed, for example the army could equip every soldier with one of these devices, it would be very difficult to destroy the network since a very large number of devices would have to be destroyed before the network stops functioning (although it might still be partitioned). Also there is the nice feature that the network keeps moving along with the soldiers. Using P2P techniques in mobile networks is also interesting when considering the information a mobile device might be able to contribute based on its context. In the stationary network, much of the sharing between peers is static information, usually in the form of files being shared. But in a network of mobile devices each has a different view of the world.

While mobile devices probably won’t be able to contribute huge amounts of static information for a while yet (if someone would even want them to), mobile devices have access to a lot of information that desktops do not because they are mobile and often distributed over large areas. If the devices where equipped with sensors such as thermometers, humidity sensors, pressure sensors, and GPS's they could provide data for services such as very local weather information (micro weather) or traffic models. Devices like mobile phones are already being equipped with sensors such as cameras that could be used to measure light intensity or perhaps a news corporation would be willing to pay a person to let them stream pictures from an accident or unexpected event as it happens in a remote location so everyone could be a potential reporter. Though this might begin to sound a bit farfetched and having little to do with P2P technology, it is still the same concept of devices sharing their resources with someone else, be it a camera or other type of sensor, bandwidth to relay information, a file system or processor capacity.

A mobile device need not necessarily be a mobile phone (even though this is what we usually think of), instead it could be a modern car which contains a huge number of sensors, such as accelerometers and thermometers and also possesses a lot of computational power enabling it to produce advanced assessments of road conditions that it presents to the driver. If these assessments along with the car's position where sent over the mobile network, road conditions of a busy highway could be monitored down to every meter. A cellular network does not necessarily have to be used. Often local information is only interesting to people in that area. Using the adhoc networks discussed above, cars could be equipped with a device that can relay a message for a few hundred meters, and if the car senses for example that the road is icy and the tires are spinning or that the airbags have deployed, it could send out a warning that is relayed to any car in the area (for example 5 hops) thereby only warning those that might be concerned with this information.

Yet another example could be a disaster area where the cellular infrastructure has been lost. If the infrastructure was a CDMA (Code Division Multiple Access) network such as the proposed 3G networks, it could be feasible to simply bomb the area with new base stations. The base stations would then be peers that automatically discover each other via a high power radio interface when they have landed and create a network so that people's existing cellular phones start working again. These base stations could then be moved to more strategic places than where they landed, such as a hilltop. Even if it loses connectivity while moved it will rediscover other base station peers and the network will recover and heal.

4.1.3 Multi-interfaced devices

So far, P2P has been discussed for two different cases, cellular networks and adhoc networks. As discussed above they often have different characteristics and needs. A very interesting development that can be seen in the last few years is that without any particular joint effort or plan, mobile device developers have started to equip their devices with support for both kinds of networking. Many mobile phones today can communicate data via GPRS in the cellular network and via Bluetooth to other devices directly. This means that, in effect, there are already large numbers of cellular to adhoc gateways walking around although they are not being used, other than perhaps when someone occasionally connects their laptop to the GPRS network using their phones Bluetooth connection. Imagine a device such as a mobile phone being a peer in both a cellular (GSM) network and a adhoc (such as Bluetooth or IEEE 802.11 WLAN) network at the same time, thus opening up a realm of very interesting possibilities.

Looking a bit further into the future, all sorts of resource sharing could be imagined, for example the limited bandwidth that the cellular networks provide could be shared in situations where groups of people are gathered and interested in the same information, providing a means for very local (one cell at a minimum) broadcasts as opposed to TV or Digital Video Broadcast (DVB) [70] that allocate resources over large areas and thus requires administration of available frequencies as well as costly broadcast technology that can not be justified by a small number of (from a few hundred to perhaps a few thousand) users. Consider an instant replay channel being broadcasted to the spectators at a sport event. Since everybody wants to see the same information it would (theoretically) be sufficient if one device received the stream from the cellular network and then relayed it out to others in his/her vicinity via Bluetooth or some other adhoc technology (of course the delay would be very high).

The stream could then use almost the entire capacity of the cell since it only needs to cater to one user. Though it is possible to assign more resources to a single device in TDMA (Time Division Multiple Access) networks such as the GSM network (for example by assigning more timeslots or using devices that can listen to several TDMA channels), bandwidths in future networks may more easily be dynamically assigned. In CDMA systems for example, the length of the spreading code could be adjusted depending on the number of users in a cell. Fewer users means that a shorter spreading code is needed to separate the users which results in a higher data rate. In the same manner, in networks such as Digital Audio Broadcast (DAB) [71] and DVB that use FDMA (Frequency Division Multiple Access) the number of sub carriers that are assigned to a device is also dynamically adjustable.

Another application of dual networking could be in emergency situations in crowded places where a lot of people need to be informed of something, but the cellular network lacks capacity. A group of people could share the same channel, if their devices started by asking for the channel over the adhoc interface. Only if no one else was within reach, would it proceed to ask for a separate stream from the cellular network.

There are many other features of adhoc interfaces that could enhance the use of a cellular networks. For example, when devices communicate directly, they could use timing to estimate the distance to each other, three devices could even triangulate their positions relative to each other. If one device asks for a base station triangulation (which will be possible in the future CDMA networks where a device can have contact with several base stations simultaneously) it could then share this information with other devices close to it. These devices could then recalculate their own position, given the distance to the other devices.

Going even a step further, in an environment with many devices and under the (not always true) assumption that most devices move more or less independently of each other, a device with a large enough range could even try to estimate what is actually the physical reference system based on the fact that if all devices move independently, the sum of all their speeds should converge towards zero as the number of devices grow (if the measuring device is standing still, otherwise it should converge to the speed of the measuring device). Having a notion of the physical reference system, the device could then calculate its own movements while only occasionally updating its position by network aided positioning (triangulation).

It is obvious that this will not work very well in many situations, for example on a train where all devices move at the same speed in the same direction, some kind of scheme would have to be used to discover those situations, in any case a sometimes poor update is usually better than none at all since it would only occasionally actually worsen the original estimate even if it doesn't improve it. A nice feature of such a system is that as the number of devices in an area grows, the need to use the network for positioning lessens, thus the system becomes self-regulating. This could help save the network from having to perform the quite resource demanding triangulations constantly in order to follow a device "live" as it moves.

Distance measuring using Bluetooth and 802.11 WLAN has been explored [27] and resolutions of down to 1 meter have been achieved. There are many issues making the services discussed above difficult to implement today. While IEEE 802.11 WLAN provides a full 11Mbps it has no power control (although there is research going on to develop powercontrol for such devices [54]) and is therefore not well suited for devices with very limited power capacity. Bluetooth (currently 720kbps) has power control and the advantage that it advertises itself so that others might find and connect to it. The process of setting up a connection in Bluetooth when another device is detected is unfortunately quite slow (a straightforward transaction may well take 30 seconds from start to finish depending on the profiles used), making it difficult for devices to "roam" between each other even at walking speed, hence making the system less mobile. There are ideas and schemes (Bluesoft Inc. [28]) to overcome the problem of setting up quick connections. Using distance measurement and preparing communication with a device by putting it in a standby mode with a special 8 bit address if it comes within a certain distance, a device can be in standby mode in several piconets and quickly switch to using a new piconet when contact with the current one is lost. Another problem is that the range of the adhoc interfaces present in most devices is still quite short, most mobile phones of today only support the 10 meter Bluetooth specification, there are however specifications for up to 100 meters. While 10 meters may be within the needs of being able to create a useful immediate vicinity network while not consuming too much power, something closer to 30 – 80m would probably be necessary in order to give the device time (as well as a statistical chance) to find and connect to another piconet when roaming without loosing connectivity from the old one. This of course depends on the device density in the location where the network is expected to work and it also reintroduces the problem of power consumption that it present with 802.11 WLAN.

Also PDAs are being sold that support both GPRS, Bluetooth., and IEEE 802.11 WLAN communication standards. These could be used to create tailored solutions for use in corporate offices, etc. There has been research about dual technology telephones switching between the DECT and GSM networks when entering and leaving an office. In a similar manner, IPunplugged has shown that seamless roaming from UMTS to IEEE 802.11 WLAN is possible, meaning that an advanced PDA could switch from cellular (GPRS) to adhoc (IEEE 802.11 WLAN) communication when in range [55].

In order to ensure a minimum adhoc network connectivity in an office, stationary peers could be put in strategically chosen places, an advantage of such a network compared to for example infrastructure IEEE 802.11 WLAN (i.e. ordinary IEEE 802.11 WLAN) is that the capacity of the adhoc network will grow (for internal communication) as more and more people come to work and more and more “routes” become available instead of deteriorating as would be the case of an ordinary IEEE 802.11 WLAN with base stations becoming congested as the available bandwidth does not increase with more users.

Another and perhaps bigger issue that exists in all P2P networks is that there must be some incentive for a user to share his/her resources. Why would a user want to pay for receiving something that he might not even be interested in via the cellular network and waste power in order to let others take part of that data for free over the adhoc interface? In the current infrastructure most users would likely try to receive as much free information as possible while contributing as little as possible and soon there would be no resources which are shared. The problem of freeloading in P2P networks has been well documented for systems like Gnutella [15]. Getting users to work for “the common good” is not an easy task. Some kind of economic or social incentive system has to be created.

A different billing system might be necessary where a user pays a set fee for participating in a group so that the costs are distributed over the group regardless of who is actually receiving the original stream. There are other ways of creating an incentive, perhaps streamed channels could be commercially financed and the viewing applications created so that a user has to share in order to use the application. While Folding@home uses the incentive of helping mankind, SETI@home is perhaps using the incentive of vanity, the possibility of being the one to discover an alien life form?

5. Technical analysis

5.1 Different technologies to realize overlay networks and general end-to-end connectivity

5.1.1 Using JXTA technology

When using JXTA in cellular networks, it is undesirable (though possible) to implement a full JXTA platform in the cellular device itself. This is not only due to the size of a current fullsized JXTA peer which demands a rather large amount of storage space and processing capacity. As discussed in section 2.1 there is another reason why a full JXTA platform is currently not desirable in a cellular device even if it was less capacity demanding. This is the amount of traffic that is actually sent between a peer and a rendezvous in the current JXTA implementation. An XML based JXTA peer advertisement is usually around 2kB, resulting in peer advertisments, group advertisements, etc possibly amounting to several hundred kilobytes when a peer for example joins a group and requests its members. This is not desirable in the GPRS network, which is why the actual peer is currently preferably implemented in the fixed network and the phone only works as a thin client to this peer, translating messages from XML to binary and receiving and sending only what is necessary.

The benefit of having the peer in the actual phone is that any application written for the phone that implements the JXTA protocol gets access to the JXTA virtual network and can send arbitrary data to other applications that also implements the JXTA protocol without the network having to support that particular application. The network only has to support the standardized JXTA protocol (i.e. there has to be at least one rendezvous somewhere in the network). While this is technically possible, as explained above it is not practically possible with the current implementation of JXTA.

As devices becomine increasingly capable and should soon be able to handle a full JXTA platform, bandwidth remains sparse in many cellular networks (excluding hotspots) and the size of JXTA advertisements and the amount of technical traffic would have to be minimized to make this feasible. SUN has developed J2ME APIs that can be used by an application to connect to a special relay that filters traffic and encodes messages from XML to binary. Though this is a very limited peer (and the actual peer is still in the relay) that only uses HTTP, the concept of having a special implementation for mobile networks with a gateway to the non-mobile networks might be necessary for yet some time. The other alternative is to rewrite the existing JXTA implementation which might be a bit more difficult, but could prove a better choice in the long run as total transparency between mobile and fixed networks would be achieved.

The HTTP relay for mobile devices that SUN Microsystems currently offers, can only be implemented using Java2ME APIs. However, as mobile devices are becoming capable of running personal Java and even standard Java2SE applications, APIs for using the mobile relay would be desirable for implementations as well.

Using an HTTP relay that is not application specific, but simply implements a subset of the JXTA functionality, the network would be transparent all the way to the application. While this is in line with the original JXTA concept, an application specific relay (such as the one used in this thesis) could perform services for a user when he is not online, such as caching of messages, etc. (my implementation keeps track of users that are online in order to be able to send this information to the device when it connects).

If an operator provided such relay peers for their users, it would perhaps mean cluster of computers. How many are dependant not only on how many users are expected to be using the relays at the same time, but also on what resources each users peer should have in the form of potential memory and storage space. Since peers and groups automatically distribute themselves in a JXTA network, creating such a cluster would only be a matter of starting as many relay peers as possible on a standard server and then put another server on the same subnet and start another group of relay peers and so forth. If the servers are all on the same subnet, all JXTA platforms will discover each other automatically using multicast and there is not even a need for a rendezvous. If a server goes down, the users on that server loose connectivity, but there is no need to restart the whole system, only the processes that went down.

There are other methods of building distributed servers, for example CORBA [56] or JINI [57] which could very well be used. One of the drawbacks of using CORBA is that the system then has a single point of failure as compared to the distributed JXTA system.

JXTA also differs from JINI in that it is built for Internet communication; meaning that if two operators would want to enable their users to communicate with each other it would simply be a matter of setting up a rendezvous in the subnet of one operator that communicates with the corresponding rendezvous in the other operator's subnet. Since both subnets are now part of the same JXTA network, groups and peer advertisements are distributed automatically between the operators without any additional configuration.

The drawbacks of JXTA are dependant on the sort of applications that are supposed to be implemented on top of it. JXTA is message based rather than stream based as Java, which has benefits as well as drawbacks when creating a solution such as the one discussed here. Naturally, the message based nature of JXTA makes it very suitable for message based applications like instant messaging and sending pictures or perhaps even video clips to other users. As these entities are treated and sent as whole messages in the JXTA framework, it is very easy for an application specific relay to save a message if a user is not online. This system looks very familiar to SMS and MMS messaging and in fact it is. The distributed JXTA rendezous could very well be seen as an MMS server, communicating with the MMS server of another operator in an MMS server network. One big difference is that *arbitrary* data is sent between applications instead of specific data.

While this sort of message based system is exactly what is wanted for many services it could be argued that this is not really direct communication since for example a film clip actually has to be entirely received by the relay before it can be forwarded (in order to be able to create a JXTA message), meaning that the delay might be considerable. Whether communication is **perceived** by the end users as being direct or not boils down to the delay introduced by the system.

Naturally, a message based system is not very suitable for stream based content, if a movie clip or VoIP conversation was to be streamed over the described JXTA framework it would have to be divided into several small JXTA messages which would not only introduce extra delay but also extra overhead. The less buffering delay that is desired, the smaller the first message would have to be before it is sent and consequently the larger the amount of overhead there would be.

5.1.2 Writing a specialized server

The great benefit of a specialized server solution is that it can be tailored to the specific environments of wireless networks as well as services with specific demands. While JXTA is a very general solution, a specialized server can be optimised with regard to different factors such as speed and propagation time depending on the application. The drawback of this is of course that the system has to be designed from scratch. For a small server this may not be a big issue, but when designing a system with an infrastructure as complex and scalable as JXTA (that should be able to communicate with other networks), it is a very big task to develop the entire system. Although using non proprietary building blocks might be advantageous

As discussed previously (see section 5.1.1) a distributed server could also be implemented using CORBA, JINI, .NET or simply standard socket communication. JINI and .NET are suitable for implementing very similar functionality to JXTA for message based systems, but can also support stream based systems since they are stream based. However, unless a stream based system is sought, JXTA is a more complete and versatile platform to use for messaging.

CORBA on the other hand was not intended for implementing the kind of decentralized system described above. Instead it aimed at creating a distributed, but still "centralized" server in the sense that the distributed server does have a central point and therefore also a single point of failure.

Ultimately, the technique to choose depends largely on the size of the system and the type of messaging that is intended to take place. For a system with tens of thousands of simultaneous users, CORBA might not be ideal since the whole system fails if the broker fails. However, for a smaller system with perhaps up to a hundred users or so that require extreme stream replicating efficiency and low delay such as for a VoIP push-to-talk radio channel, CORBA might be used. For these sorts of applications no separate processes acting as "home agent" or caching information in the network is wanted. The server side's only task is to replicate data as fast and efficiently as possible while acting as an ALR. This means that highly optimised C++ or Java code might be preferable.

The implementation should be very similar to that of a Network Address Port Translating (NAPT) gateway which takes the data in a packet, transforms it and sends it in another packet and the reverse. It must remember the mapping of a certain port with a certain user (IP address in the NAPT case) so that further data can be relayed throughout the transaction.

In the case of the specialized server this idea could be extended so that this mapping is kept and can be used by any one that wants to contact the device. But rather than mapping a port to a particular IP address (network layer), a port should be mapped to a username (application layer). When a stream arrives at the server, it will have to look in the data to see what user it is intended for and send the data on the port that the user is mapped to. An application layer protocol (similar to HTTP or any other application layer protocol) would have to be implemented.

The words “application layer” implies that only applications should look into this layer, and originally the idea was that end user applications should communicate with each other in this layer and the network should not examine this layer. However, since the server is acting as an ALR, and in fact is an application, it will access this layer and try to interpret it, thereby partially breaking end to end transparency.

This sort of application level routing is not uncommon though, as many gateways (NAPT, NAT) are actually application level gateways that look into the data that they are address translating in order to accommodate special application layer protocols such as the FTP protocol and several proprietary game protocols.

While a third party would have to place these ALRs somewhere on the Internet, an operator could place the server inside the GPRS network. This has the drawback of the server only being available to users in that GPRS network, but for services that are only aimed at the operator’s customers this might be an advantage (see section 5.2.2) since the operator can guarantee the security of the data path and more importantly Quality of Service (QoS) through the whole network if it never actually traverses the Internet.

Even if an operator wishes to build a network that is interoperable with other operators and the Internet, the operator could still put the server inside its network, it could then communicate with similar servers on the Internet via a highly secured tunnelled network (an idea similar to GPRS tunnelling between operators).

So far, the specialized server solution has been discussed for a specific service, it uses a specific protocol to communicate with others over what could be called an application network. If this protocol for contacting other applications was standardized, other applications could also use it to send any kind of data on top of this application network. Thus a virtual network for arbitrary data has been created on top of the physical network (see section 4.1). Using this standardized protocol, a third party could then write applications that communicate with each other on top of this network, via the ALRs.

Stopping for a minute and looking back, this is exactly what JXTA intends to do, creating an overlay network on top of the physical for sending arbitrary data between applications. The JXTA rendezvous are the equivalents of the suggested ALRs so if these rendezvous were placed in the operator’s network (whether inside or just outside the GPRS network, a third party could write applications that implement the JXTA platform and send arbitrary data to other similar applications over the JXTA virtual network. As described in the previous chapter, although this is JXTA’s intent and it is how JXTA works in the fixed network, due to current cellular bandwidth a relay/filter has to be used for cellular devices.

Another, very important difference is that while JXTA implements a random addressing scheme, an implementation of specialized ALRs could use a operator or area bound addressing scheme, making the system much more scalable (although less mobile) because queries do not have to be flooded. This is important if the system is expected to be linked with other operators and possibly grow to a large network.

5.1.3 Using SIP technology

SIP is short for Session Initiation Protocol [29] and as the name tells it is a standardized protocol that can be used for any type of session that is supposed to be initiated. The SIP concept includes a name server function very similar to that of the Domain Name System (DNS) where a hostname is mapped to a particular IP address. A special SIP address of the form sip:username@domain is used. The domain makes it possible to know which SIP domain to ask for the current IP address of a user. The user’s client tells the SIP server when he changes his/her IP address. The mapping thus changes so that his/her username maps to his/her laptop’s IP address when at work and his/hers stationary PC while at home, etc. This approach can also be used to overcome the problem of devices that have dynamic IP addresses that change at every logon or at certain intervals.

This is very similar to the dynamic DNS hostname mapping that several domains currently offer. Here a client is also used that updates the DNS with the current IP address. While this is perhaps not yet extensively used for switching between devices, it is very frequently used by people who wish to set up servers in networks where the ISP only provides dynamic IP addresses.

While the mapping part of SIP can thus be partly achieved in other ways, it is the standardized way of starting a session once this mapping has been done that is the more interesting part of the concept when looking at overlay networks and direct device-to-device communication.

There are a few different ways that SIP could be used to achieve direct communication depending on how the network is implemented. These are described in the following paragraphs.

If devices have globally routable IP addresses, the original concept of a SIP username mapping to this address can naturally be used. This would then not be an overlay network, but instead a part of the ordinary network at the IP level. While some operators offer globally routable IPv4 addresses and others do one-to-one Network Address Translation between private and global addresses (proving that they in fact have enough IPv4 addresses at the moment), these addresses may be suvfficient if always-online services such as the one implemented for this thesis become popular. There is also the fact that while some operators may have enough subnets for all their customers others may not. For example, the whole of India is currently using a single B network (meaning 65536 addresses) so building a system for device-to-device communication using this technique may mean that certain operators cannot take part, which reduces the usefulness of the system. IPv4 addresses might be sufficient for most applications until the long awaited IPv6 protocol is implemented, giving a large enough address space for everybody to use. However, this requires that IPv6 is widely implemented on the Internet by the time that the IPv4 addresses start to become exhausted. The European Commission has stated that this will occur as soon as 2005 [58] and has hosted an initial meeting to form the IPv6 task force that will work to ensure the timely availability of the IPv6 protocol [59].

Even if operators, as described earlier, do not provide globally routable addresses to the devices themselves, but uses NAT mapping of global addresses to private ones in order for the cellular devices not to be completely “exposed” to the Internet, SIP could still be used using an application level NAT gateway. A cellular device that wants to be contacted (perhaps only on a single port) can then register itself at the application level gateway, its SIP name is then mapped to its current global address and messages are relayed to its current private address. However, if the point of the NAT gateway was to make the devices unreachable from the Internet, then the application level gateway makes it reachable again anyway, thus defeating the isolation.

The problem of the IPv4 address space not being sufficient has resulted in wide use of Network Address Port Translation, NAPT. This means that a whole private IP address space is mapped into one globally routable address by the gateway to the Internet, different sessions from different private addresses to the gateway are differentiated by using different ports on the outgoing connections. In the abstract the ports that are supposed to be used to de-multiplex data to different applications are instead used to differentiate different IP addresses (which was supposed to be done using the IP address), hence one level of de-multiplexing is lost. This means that although there are application level gateways also available for NAPT networks, unlike the prior example where IP addresses and ports could simply be mapped to the private IP addresses using the *same* ports, the ports have now been used by the gateway to de-multiplex incoming packets to the corresponding private IP addresses. The result is that a SIP device inside this network could at most ask the gateway to relay all incoming traffic to a single pre-specified port. Meaning that only a single application could be reachable from the outside at a time unless another port on the gateway was also mapped to the same private IP address, but on another port, however, this scheme would mean that if a single device could ask for all it’s 65536 ports to be mapped this way, then no other device inside the network could use the gateway since all it’s 65536 ports would have been allocated. In order for an outside device to be able to address something specific on a device inside the NAPT network, the lost level of de-multiplexing must be added again in the application data. The single open application/port could then be used as a tunnel for another protocol sent in the application data. This means that in fact an overlay network has been created (although only for signalling and session set up).

The above scheme can allow for entities to be contacted on the IP level even if they are inside NAT/NAPT networks by making the gateway to the network aware of an application and allowing a client to “grant” an IP session through the gateway. This is very similar to the UMTS IP multimedia sub system scheme (see section 5.1.5) where a user can be contacted using SIP and grant an IP session through the GGSN). A problem with this scheme on the existing Internet is that just as in the UMTS case, all gateways have to be aware of this, a user with an ISP that does not support this has no chance of joining the network. From an operators point of view this may be desirable, while from an end users it is not.

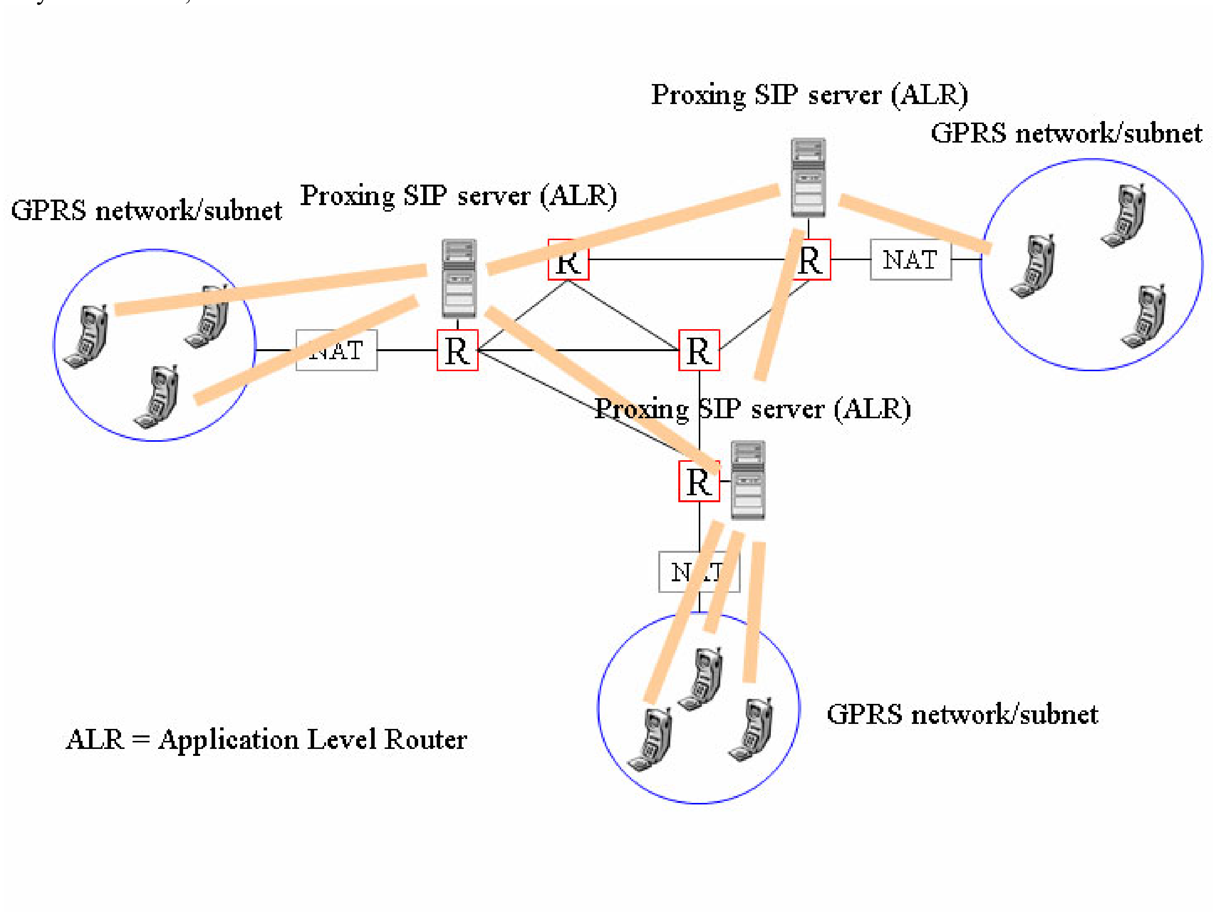


Figure 26. SIP servers acting as ALRs relaying data between users

Instead, a different scheme much more like JXTA could be used that is independent of the ISPs NAT/NAPT implementation. A network of SIP servers could be set up where data is actually sent between devices via the SIP network, i.e. an overlay network where SIP severs (outside the NAT/NAPT networks) act as ALRs. A device with a private address could then connect through the NAT/NAPT gateway to a SIP server on the Internet (a proxing SIP server). Through the established connection the device could then be contacted and receive data. The device could thus be addressed using it's SIP name which might be sip:telephonenumber@operator.com. If someone asks for the IP address mapping to the device's SIP name, the reply will contain the IP address of the SIP server that the device is connected to. If the user then asks the server for a SIP connection on a well-known port, the server will then go ahead and request a connection on a specified port according to the SIP standard for session initiation. The server can then relay everything from the port that the other party was asked to connect on to the device through the established connection. The different devices that have connections established to the server are then de-multiplexed using ports just as in the prior case. Several ports could also be opened as in the prior case, at the same cost. Of course, this scheme demands that the proxing SIP servers are distributed just as in the case of all discussed overlay networks. Hence, this scheme is most interesting for the mobile networks where bandwidths are low.

Messages could now be sent between devices that are both in NAPT networks in two different ways, either a device directly contacts the server responsible for the other device's network using SIP to set up a connection and communicate with the other device through the TCP connection that it has permanently established to the server.

Alternatively, the sending device could send data via the established connection that it has to it's own server instead of directly to the other user's server. The sending device's own server would then establish the connection to the other user's server and relay the data in an overlay network manner. While this may seem like an idea that puts unnecessary extra load on the server, there are several advantages of having data always travelling "inside" the overlay network. One of these is that it can be replicated in the sending devices server and sent to several receivers in a multicast like way without the sender having to send separate streams. The concept of groups similar to JXTA could then be introduced. These could either be propagated between the servers as in JXTA or it could be implemented as a separate centralized function where all servers can ask who is part of a group and if he has permission to send and receive this group.

SIP is already used in this way in some video conferencing applications where streams from separate users has to be either replicated or mixed into a stream that is sent to all users of a group (although the signalling layer and data layer functions could reside in different places in the network).

It should be pointed out that even though the discussion above refers to a single server per operator, there would in fact have to be a group of servers to be able to handle real-time application level relaying large numbers of data streams such as video. This would be easy to implement though if the device also uses SIP when connecting to it's relay server. It could simply contact the server on a well-known SIP port, in accordance with the SIP protocol, the server can then reply and ask the device to reconnect on a different port on a different server. In this way the connections can easily be distributed over several servers. The central server only has to change the SIP username mapping to point to the IP address of the server that the device is connected to.

The question of why it would be of interest to develop these functions based on SIP when JXTA already has them may be asked. The answer is that although this concept of using SIP for an overlay network with servers routing messages on the application level seems very similar to JXTA, there are a few key differences in the implementation that makes the SIP solution more suitable for certain services and less for others. The first is the fact while JXTA is message based, the proposed SIP solution would be connection (or stream) based, making it a lot more suitable for potential real-time applications such as streaming video from a device to another, VoIP applications, and on-line games. However, as discussed in section 2.1, a stream based system is not very suitable for messaging applications where a message should be stored if a user is not on-line. Since a stream based system never in fact has the whole message available at the same time (as is the case with a message based system such as MMS or e-mail), the stream-based system has no real concept of where a message starts or stops. It should be noted that SIP supports (via extensions) notifications for presence and also supports messaging, what is lacks is storage, but voice mail servers demonstrate a potential solution.

The second big difference is the addressing scheme. As was discussed in section 2.4, JXTA has a random addressing scheme that is not location dependant and uses flooding that doesn't scale well for large networks (if all users are still to be guaranteed to reach each other).

SIP on the other hand has a location based addressing scheme that can utilize the underlying network efficiently and scales very well as the number of servers grow. The location based addressing scheme has the disadvantage that roaming to networks other than where the address "belongs" results in very inefficient routing across the underlying IP network, especially if all data is sent via the home server in both directions as suggested in the above overlay network scheme. A network such as most cellular can still utilize this scheme though as a very small number of users are expected to be roaming outside the network at any time.

There is yet another way to implement a SIP version of an overlay network like the one discussed above. Instead of putting the servers on the Internet, the servers that the devices contact could be placed inside the NAPT or NAT network. This server is then either connected out of band to other operator's servers or general servers on the Internet or simply connects through the GPRS gateway to a SIP server on the Internet. This would mean that there is still a need to put a server on the Internet so no gain is made there. However, since the "closest" SIP server is inside the network and can contact and be contacted by the devices directly, using their private addresses, there is no need for every device to keep an open connection to the server.

Which of the two last implementations an operator would choose is actually transparent to other operators since there would have to be a server outside the gateway in any case that can be contacted and that then relays the data to the server inside the GPRS network.

The latter scheme has the advantage that as devices can be contacted using their private IP adresses, there is no need for a device to keep an open connection alive using polling (this is currently necessary due to the implementation of some operators GGSN, which terminates non-active connections after a certain time). Polling affects the GPRS network in that it can make a device switch to a state where they consume more battery and potentially also more radio resources since a device in the ready state updates the SGSN upon every cell change. On the other hand, as this might reduce traffic on the paging channel it is not necessarily bad. This is further discussed in section 5.2.3 chapter.

One big difference between the two is that while only the operator could implement the latter scheme, any third party could implement the prior scheme, since the servers do not have to be physically close to the gateway of the network. They could be anywhere on the Internet, although routing becomes more and more inefficient the larger the detour that the packets have to take.

Also, since a third party has a hard time distributing the system as efficiently as if the servers are placed close to the gateway of each network, the network will become less functional and more susceptible to congestion as more packets pass through the same node.

If a large actor such as a device manufacturer were to build a truly distributed network where users were assigned SIP names depending on which operator they used, the resulting system would probably be adopted by many operators.

5.1.4 IPv6

Internet Protocol version 6 has been talked about for a long time and will most likely be implemented. But it is very uncertain **when** this will happen and how long the transition be before it is widespread enough to replace IPv4. During the transition hosts will have to incorporate both the IPv4 and IPv6 stacks in order to be able to use services that have not yet migrated to IPv6.

The industry that is pushing the hardest for IPv6 to become a reality is the mobile phone industry. It is possible for an operator to completely switch from IPv4 to IPv6 within his network and let the network's Internet gateway tunnel the IPv6 packets in IPv4 packets over non IPv6 supported areas. This would mean that the operator's mobile devices do not need to incorporate double stacks, on the other hand it means that the devices can only communicate with IPv6 enabled services. This may seem like a big loss, but since very few mobile devices actually use the same services as stationary devices (services have to be specifically created for mobile device's resolution and bandwidth) this means that only the services aimed at mobile devices would have to be migrated to IPv6 initially. Since the market is rather new, there are not very many services available and if an initiative was made by the industry that all future services aimed at mobile devices are able to tunnel IPv6 over an IPv4 network, all mobile device networks could quickly be IPv6 enabled and operators could switch to IPv6 completely. This in turn might push the incorporation of IPv6 in the fixed network. Corporations whishing to enable mobile IPv6 devices to connect to their corporate IPv4 LAN would need to set up a gateway that can receive tunnelled IPv6 packets and translate the IPv6 addresses to IPv4. There are standardized schemes for how this translation is done [30].

In order for an IPv6 packet arriving at an operator's Internet gateway to be tunnelled in the right direction, there has to be some sort of IPv6 enabled network (however small - theoretically at least one IPv6 router) that the gateway can tunnel the packets to, which can in turn route/tunnel the packets to another operator's gateway or an IPv6 enabled Internet service. The operators would have to place some IPv6 enabled routers in strategic points in their backbone networks in order for the network to be efficient, only a single or a few IPv6 routers would quickly become congested as the amount of users increase. Some routers are already capable of IPv6 routing although it is not being used.

It becomes clear that building a network of IPv6 tunnels is the same as building an overlay network, the routers will have to look in the data part of a IPv4 datagram to determine if it is an IPv6 datagram and then take action, similar to an ALR. This of course means that the same problems of dimensioning and distribution of the ALRs are the same in this network as in any other overlay network.

There is a very important difference though, the difference that inside the operator's network, which is IPv6 enabled, IPv6 is not an overlay network but is the actual network and packets are routed at the network level (layer three of the OSI stack) rather than at the application level (layer 7 of the OSI stack). This means that as more and more areas of the Internet become IPv6 enabled, the IPv6 network will cease to be an overlay network and become the actual network with all the efficiency gains that this brings.

It could be argued that building a distributed overlay network for arbitrary data is an attempt at building another Internet on top of the existing (but with a larger address space and perhaps more functionality). In order for this Internet to become as efficient as the existing, all existing routers would have to be able to do application level routing.

It would also necessitate the overlay network addresses be divided into subnets similar to IP addresses so that the ALRs do not need to keep tables of every application address in the world or use flooding. The freedom of mobility of an application that is not bound to a particular IP address (see the discussion of services like MSN Messenger and ICQ in section 2.4) is then lost. A scheme similar to Mobile IPv6 could instead be used if devices are expected to frequently leave their subnets (i.e. if tunnelling via the home ALR or so-called triangulation routing is not acceptable for the roaming devices) [60]. Dynamic DNS could also be used, a user registers a user name for a particular application with a dynamic DNS service, for example telephone- number.dynamic-dns.com. Upon connecting to an ALR, the ALR automatically updates the DNS binding of the connecting applications username to point to its own IP address, thus anyone requesting telephone-number.dynamic-dns.com would get the IP address of the ALR that this application is currently connected to. No flooding is needed and the application can still move around between different ALRs.

The notion of groups that exist in some overlay network technologies, such as JXTA, could theoretically be implemented using IPv6 multicast. This remains to be seen though, as multicast has been possible in IPv4 networks for quite some time within ISPs, but has still not been enabled between ISPs since there is no easy way of controlling and verifying who is authorized to join and more importantly send to a group, any user can send to a group without being a member. Because of this, most ISP's routers at the leaves of the networks simply drop multicast packets originating from users. In JXTA, this problem is solved by the user creating the group also being able to create a certificate that other users need to present to the network routers (JXTA relays in the case of JXTA) in order to be allowed to join and use the group. This certificate can then be distributed to users in arbitrary ways. A problem with this technique is scalability. The certificate has to be propagated to all JXTA relays in the overlay network, this only works well for networks up to a limited size. An operator owned overlay network could instead use a dedicated server to store certificates and parameters about groups as described in the SIP chapter.

It may be possible to implement similar functionality for multicast on the IP level, where a DNS like hierarchical system could be used by a router to find out who owns a certain group and what permissions that group has.

One problem that IPv6 does not automatically solve is the problem of cellular devices being exposed to the rest of the Internet. It is likely that operators would still have to implement some sort of protection function where NAT is used today. Using NAT would take away the benefit of having a globally routable IP address and the possibility to use dynamic DNS. A firewall could perhaps be used instead of network address translation to avoid this. The drawbacks of this is that the rules of the operator provided firewall would likely have to be quite general and static, only allowing packets from certain trusted parties such as other GPRS operators. This would severely cripple the use of the network.

The proposed overlay networks have the advantage that as everything is done in the context of a group which acts as a private network, an application that joins a group is only exposed to the users of that group, for example a few friends or co-workers, not to the rest of the Internet. As the creation of these private networks can be dynamic and user initiated, it has greater flexibility than static rules in a firewall.

There are proposals and definitions of how some (but not all) of these problems should be solved in the future UMTS networks. The 3rd Generation Partnership Project (3GPP) [31] has defined a scheme for signalling and establishing IP level sessions between devices using SIP and IPv6 called IP Multimedia Subsystem (IMS). This solution is examined next.

5.1.5 The UMTS IP Multimedia Subsystem (IMS)

The 3rd Generation Partnership Project (3GPP) has defined a scheme for signalling and establishing IP level data communication between devices in the future UMTS networks. In release 6 of the 3GPP specifications this scheme is intended to replace signalling for all multimedia communication such as speech, video conversations, etc. This is done using IPv6 and SIP.

The IMS system is similar to the overlay networks described above, although the overlay network in IMS is mainly used for signalling and set up of a connection. Data is then sent on the IP level (hence NAT is not used).

The IMS system is implemented as a group of functions called the IP Multimedia Core Network Subsystem (IM CN SS). This is situated outside the GGSN node of the GPRS network.

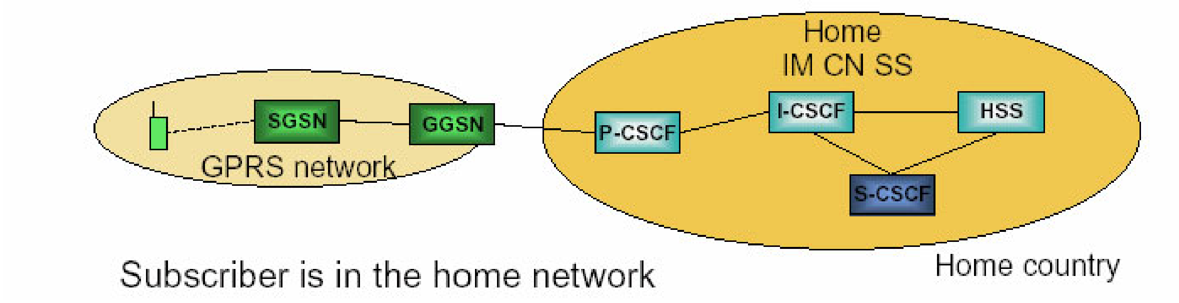


Figure 27. Overview of the IMS Core Network in relation to the GPRS nodes

A device that wishes to attach to the IMS network would establish a GPRS connection and then perform a discovery using a special multicast address (meaning that the GGSN must be multicast enabled) in order to find a proxy call state control function (P-CSCF). This is a proxy SIP server. The device sends a SIP register request to the P-CSCF where the device's home network is specified (if its roaming). The P-CSCF then forwards this request to the interrogating call state control function (I-CSCF) in the home network of the device (in this case the same network as the device is in). The P-CSCF adds its own address to the request before sending it to the I-CSCF in order that reply traffic can travel the same way back. The I-CSCF (which is intended to hide the configuration of the home network to the outside) then forwards the request to a servicing call state control function (S-CSCF), also including itself in the reply path. The home networks S-CSCF then interrogates the home subscriber server (HSS) for subscription details and policys and registers the users SIP username mapping. The S-CSCF replies along the reversed route to the P-CSCF that registration is complete, possibly including certain policys that should be invoked for this subscriber.

If there is no security function in the GGSN and no NAT functions in the network, the device is now reachable on the IP level using it's SIP username which will be mapped to its current IPv6 address.

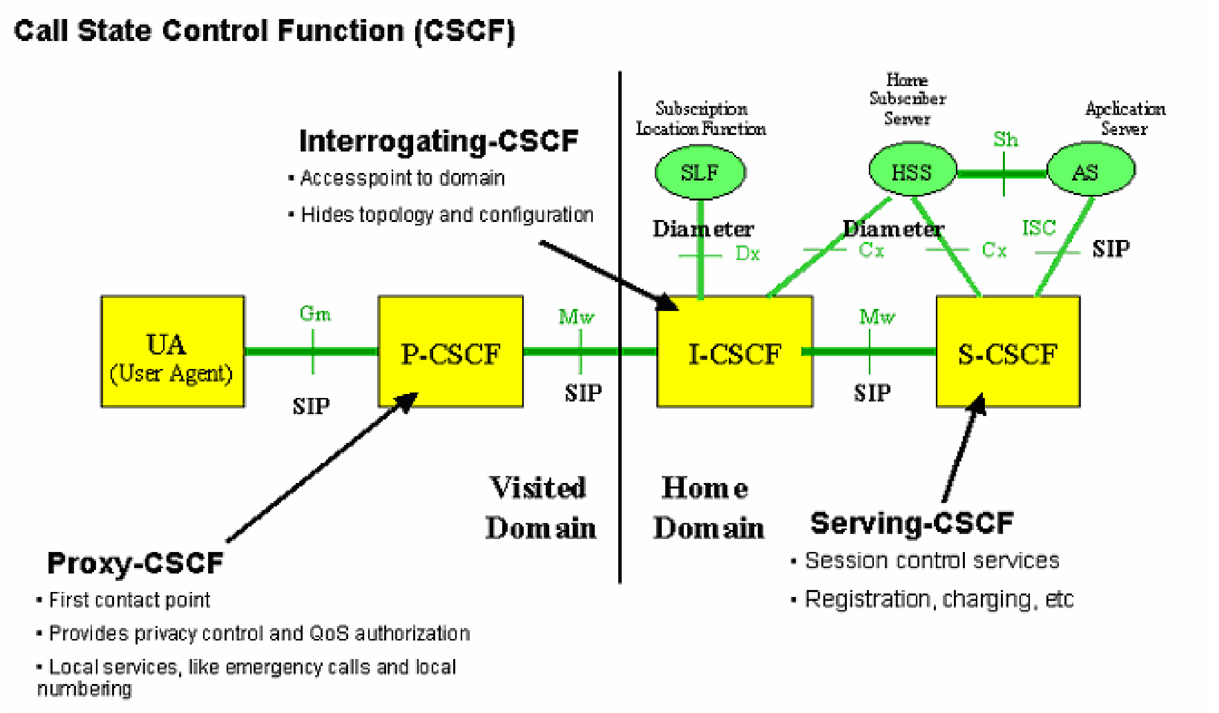

Call State Control Function (CSCF)
Interrogating-CSCF
• Accesspoint to domain
• Hides topology and configuration
Subscription Location Function
SLF
Home Subscriber Server
HSS
Aplication Server
AS
Sh
Diameter
Dx
Diameter
Cx
Cx
ISC
SIP
UA (User Agent)
Gm
SIP
P-CSCF
Mw
SIP
I-CSCF
Mw
SIP
S-CSCF
Visited Domain
Home Domain
Serving-CSCF
• Session control services
• Registration, charging, etc
Proxy-CSCF
• First contact point
• Provides privacy control and QoS authorization
• Local services, like emergency calls and local numbering

Figure 28. Description of the nodes and the protocols that are in IMS signalling

However, it is not likely that there will be transparent IPv6 connectivity through the GGSN due to the security concerns discussed earlier in this report. Instead of using NAT which would destroy IPv6 transparency, policys are suggested to be used in the GGSN. These policys are then enforced per transaction by the P-CSCF using for example COPS [61] to communicate with the GGSN. In other words, when the mobile device sends or receives a SIP invite via the P-CSCF, the P-CSCF, looking at the information in this invite, can setup that particular temporary transaction through the GGSN (only if the subscriber wishes to accept the invite of course). In this way the device cannot be reached by malicious traffic on the IP level without the device authorising it.

It is possible that IMS could be used to achieve the functionality that is whished for the overlay networks that this report proposes. The similarities between the proposed overlay networks and the IMS system is that an application registers itself with a server in the network, this application can then be reached and communicated with using a application level protocol (SIP in the case of IMS, JXTA in the case of JXTA).

However there are two distinct differences. The proposed overlay networks implement the overlay protocol stack in every application, registering each application with a separate overlay ID corresponding to a certain IP address + port pair. Each application parses the information in the overlay protocol in it's own way. IMS is intended to register only one SIP client application using for example an overlay address on the form of SIP:telephonenumber@operator.com. This application is then intended only as a dispatcher and is used to set up another IP level transaction and start an application. This means that if for example Microsoft's messenger (which uses SIP in the latest version) is used as the SIP client, any other application wishing to use the SIP overlay addressing network in order to be able to be contacted on the IP level would have to be written as a plug-in to the messenger application. The messenger application will parse the received SIP invite message to determine which application to use, i.e. demultiplexing of the applications is done in the application layer. In the proposed overlay networks, applications are demultiplexed by having separate overlay IDs instead (i.e. different ports).

Although it may not have been the initial intention of the IMS system, the SIP protocol does allow for several SIP clients on the same device to register in a SIP server (the P-CSCF in IMS) specifying different ports to be contacted on. This means that as in the proposed overlay networks, separate applications (if the operator allows it) could register with separate SIP usernames, mapping to their separate SIP clients. So the SIP:telephonenumber@operator.com username may map to the messenger program that the user uses for voice conversation while SIP:MyWebserver@operator.com may map to a web server application on the user's device. Each application would then implement its own SIP stack (and thus be able to extend it to suit the particular application) and parse the SIP message information as it sees fit. This means that any functionality could be implemented in a specific application's SIP parser, independent of other applications and developers.

If this was enabled in IMS, an application could be addressed using a specific SIP username, very similar to using a JXTA or other overlay ID. The difference is that while the overlay networks transport the data inside the overlay network, the IMS provides an IP level connection between the applications. While IP level connectivity enables more efficient routing as well as lower overhead and delay, and is certainly desirable for one-to-one connections, IMS in itself is intended to provide IP level data sessions between entities. However there are 3GPP specifications for generic IMS group service management [8]. This specifies a group, how it is identified, created, and the rights and attributes of the members of such a group. It does not specify anything about the application that uses the group.

Services such as a chat or conference call could then be created using a single server which implements the group functionality. This would work for groups up to a certain size at which point the server application in the network would have to be distributed. However the group management could be used to distribute the group while keeping it consistent. The servers of such an application would then form an overlay network for the specific applications. Indeed if the servers in such a network do nothing more than transparently relay data to all other members of the group, it would in effect be an ALR and form a general application-independent overlay network similar to JXTA. Device applications could implement the group management functions in the same way as the JXTA APIs are implemented and get access to the group network.

While this could provide distributed group functions on the application level, it may be possible to implement group functions on the IP level using multicast as has been discussed before in section 5.1.4 IP multicast could be used if the problems of creating secure dynamic groups are solved. While IP multicast groups devices on the IP level, the proposed overlay networks group separate applications. Sharing a single application means that other applications and possible vulnerabilities in them are not exposed to a group. It may be possible that the IMS group management could be developed to be able to configure the GGSN to join a certain multicast group and forward messages from a specified subset of users (i.e. source differentiated multicast). If the GGSN only relays traffic aimed at a certain port (i.e. application) then the other applications running on the device would not be available to the group.

If the GGSNs only accept multicast messages from routers that it trusts (for example other GGSNs), the multicast network can be “closed” similar to the overlay networks. In the overlay networks the idea was that all entrypoints to the network (ALRs) were trusted, and thus any information received from them can be relayed. If the GGSN required an application to have some sort of centralized permission using IMS group management before letting it join and send to a multicast group, then the same kind of effect could be achieved provided that the network of GGSNs could be trusted. Other parties on the Internet could join through trusted routers (that require login).

An overlay network functioning exactly like JXTA where devices open up connections through the GGSN to an ALR on the Internet and keeps it open for incoming traffic using an application level ID would still be possible in IMS, just as in the networks of today. However, since IMS itself could provide an application ID in the form of a SIP username as discussed previously, the application can be contacted by an ALR without having to keep a connection alive (which could save radio resources) it seems that there are reasons to investigate synergy effects and efficiency gains by incorporating these functions in the IMS signalling infrastructure.

5.2 Possibilities and limitations with regard to network and device hardware.

5.2.1 Streaming video in existing GSM networks

Contrary to what may be the general opinion amongst users of the current GSM networks, services such as streaming video and audio are actually possible in existing networks. This has been shown by several companies such as Ericsson using their P800 device and Nokia using their 7650 device. This is due to the extreme compression of images and audio that is possible with MPEG-4 (Moving Pictures Expert Group – 4) and CELP (Code Exited Linear Prediction) techniques. While the technology has been around for a while, the problem has not been the bandwidth, but rather the processing power needed to code and decode these low bandwidth streams in real-time.

With new smart phones having large processing capacity and sometimes even hardware support (e.g. Nokia and Ericsson ARM based phones) for MPEG-4 code and decoding, streaming of low resolution clips, suitable for a phone display are possible with the bandwidths available in the existing GSM networks. There are already operators offering streamed news services to cellular customers.

However, what device manufacturers and operators have shown, is only streaming video and audio on the **downlink**. Currently existing phones do not have symmetric down- and uplinks. A GPRS phone can use more than 1 timeslot in a GSM frame (a circuit switched call in GSM uses 1 timeslot, 1 GSM frame has eight timeslots [32]). Because of hardware limitations, most phones today only utilize a total of 4 – 5 total timeslots (downlink + uplink). These are usually divided as 4 or 3 on the downlink and 1 on the uplink (called 4+1 or 3+1) or sometimes with an optional symmetric but low bandwidth setting like 2+2. For acceptable MPEG-4 streaming with audio, at least 3 – 4 timeslots are needed, depending on what is considered acceptable. This means that while streaming video to a cellular device is possible with acceptable quality, streaming video from that device is not possible at the same quality with current devices.

The asymmetric implementation of the timeslot allocation in most existing devices is partially a result of devices and technology having been developed for typical client – server applications such as WAP browsing, and the kind of streaming news services discussed above. In these applications, downlink bandwidth is always preferable to uplink bandwidth. Thus, not much energy has been put into schemes for being able to dynamically switch from for example 4 + 1 to 1+ 4 depending on the application used.

However, this may change since many new phones are supposed to generate a lot of data such as pictures and video clips rather than simply downloading, making symmetric, or at least dynamically reconfigurable allocations more important.

As hardware development continues, devices may be able to allocate all eight timeslots of a GSM channel in both directions. This would be possible for only very few simultaneous users. Currently, a user can allocate a maximum of 4 timeslots in frame.

Calculations on existing mobile device based MPEG4 CODECs such as Hantro's, [26] provides a (very rough) average estimate of 3 – 3.5 Kbits/s/frame. It is important to note that this depends on the level of detail of the scene being filmed as well as the level of detail that the particular CODEC is using.

Given that a GPRS/GSM timeslot can carry about 10.7 Kbits/s of effective data using TCP and 12.8 Kbits/s using UDP [32], this suggests that frame rates of up to 12 fps would be theoretically possible if the maximum of 4 timeslots were granted by the network and no sound was needed. If sound is desired, ½ - 2 timeslots are needed depending on whether speech or music is to be sent. Streaming implementations such as Packet Video's MPEG4 player [33] for the Ericsson P800 device, use 2 timeslots for video and 2 for sound where sound is prioritised if timeslots are not available. According to the above calculations this would give a frame rate in the vicinity of 7 fps (depending on Ericsson's particular implementation) with rather high quality sound.

The above discussion shows that the currently available single uplink timeslot in most devices could only carry about 3 fps without sound while devices capable of 2 uplink timeslots can theoretically carry about 4.5 fps with speech if a speech CODEC such as AMR capable of bit streams down to 4,8 Kbits/s is used (see section 5.2.2).

While this suggests that services like low frame rate video streaming between users is possible with some devices today, further overhead due to an application level streaming protocol pushes the practical needs towards at least three timeslots.

There is also the non-negligible problem of delays and congestion in the GPRS networks, making duplex services that cannot use large buffers difficult to implement. The problems of delays and how they may be dealt with are discussed in section 5.2.2.

The implementation of such services in existing networks is also very dependant on the dimensioning of the radio resources and the GPRS core network resources, even if devices can handle the necessary number of timeslots, the cellular network may not be able to provide this due to the current traffic load. This is discussed further in section 5.2.3.

5.2.2 Voice over IP and push-to-talk services in GSM and UMTS networks

While videostreaming from a cellular device reaches the limits of existing devices, streaming of sound, and especially speech does not. The Adaptive Multi Rate (AMR) CODEC [39] proposed by 3GPP (3rd Generation Partnership Project) for UMTS networks is capable of bit streams between 4.8 – 12.2 Kbits/s with speech quality close to GSM even for the lowest rates.

This means that even 1 timeslot of 12.8 Kbits/s is more than sufficient to send VoIP data. The reasons for using a VoIP implementation rather than the circuit switched one depend on whether you are the user or operator. For a user the potential gain is the cost. However, with current GPRS prices, a VoIP conversation of 4.8 Kbits/s would give a price of 3.46 SEK/minute using Telia's most favourable GPRS subscription (GPRS high). Since data is only sent (and hence paid for) when a user actually says something (a threshold has to be used), the above figure is calculated with the assumption that users seldom speak at the same time, resulting in only one stream in either direction at a time. It should also be considered that a certain percentage of a conversation is often quiet, consisting of pauses, etc. which the user does not pay for. Users could have a call ”online” for a very long time without having to pay for more than what is actually said.

A duplex service such as the one above not only requires a certain bandwidth, it also requires low delay in order to be useful.

Delay in a wireless IP system can be divided into five factors:

- Propagation time, i.e. the time that the signal propagates on the wire,fiber, and in the air.
- Packet transmission, the time it takes to send the contents of a package on the media.

- Queuing/Congestion delay, delay due to nodes on the along the way that may be congested and interference over the air interface generating retransmissions It can be argued whether a retransmission should be considered a delay since it is actually a failure to reach the recipient at all, but since for example the GSM hardware generates a retransmission automatically and such are rather frequent, it may be viewed as a delay.

- Packetization delay, the time needed to collect the data that is going to be sent in a package.

- Signal processing delay, coding/decoding and compression/decompression.

While congestion can be solved by the operator since he controls the data paths all the way from one device to the other (or to the server if an overlay network is used) and can dimension his/her network to handle this, propagation time and interference are more difficult to handle. The quite large propagation time in GPRS networks is partially a result of the hardware interleaving of the data that takes place over the air interface.

Unlike in a fixed link, data sent over wireless links is often subject to a lot of bit errors. Because of this forward error correcting code is used to produce a packet that the receiver can use to detect and correct bit errors (if they are not too many). This means that the receiver can handle small randomly distributed bit errors (i.e. limited interference) without having to ask for a retransmission. The problem is that interference on wireless links are often not short and evenly distributed, but instead occasional and quite large (up to single milliseconds) meaning that a large part of a packet or even whole packets disappear a time. Even if the error correcting code reaches the receiver it is of no use if a large percentage of a packet is faulty.

Because of this interleaving is used. Several packets are buffered before being sent over the air interface, instead of sending the packets consecutively, the first part of each of the buffered packets are sent, then the next part of each packet and so on until all packets are sent. This means that even if a disturbance lasts the length of a whole packet, it will only wipe out a small part of each packet and when the receiving hardware reconstructs the original packets, a small part will be missing from each packet instead of a whole packet missing. In effect, the error is distributed over the data and the error correcting code of each packet can hopefully correct these small errors. While this technique is essential to being able to transfer data with high integrity, the necessary buffering unfortunately introduces a delay of 20 milliseconds in the GSM network, this is not desirable in full duplex applications.

This interleaving is implemented in hardware in the network and cannot be turned off when reducing propagation time is more critical than data integrity.

This means device-to-device communication will have a built in delay of 40 milliseconds due to interleaving alone since interleaving has to be done twice when sending to another cellular device.

Measurements of perceived quality of VoIP performance [34] show that users experience good duplex performance until delays exceed roughly 180 milliseconds when perceived performance drops sharply. While interleaving results in a 40 millisecond delay, a typical ACELP CODEC such as the AMR CODEC specified by the 3GPP to be used in UMTS networks, implements a 20 millisecond frame size. This introduces another 20 milliseconds of delay. This sums up to 60 milliseconds leaving 120 milliseconds for delays introduced by the fixed network nodes, radio interference, and resource congestion. While this may seem like sufficient time, practical studies [35] and simulations [36] indicate that in fact delays of 200 -300 ms should be expected for device to device IP communication.

Another factor that introduces a lot of delay is that the hardware layers of GPRS generate retransmissions when data is faulty even if there is no use for the retransmissions, as is the case of a duplex streaming service where old data is discarded anyway. This cannot be easily worked around. There are basically two scenarios. If the sending application can detect that data is not yet sent (this depends on the IP stack implementation, whether it reports a packet as sent when it is handed to the GPRS layer or when it is actually sent) it can then skip sending some data in order for the receiver to “catch up”. The second solution is to have excess bandwidth, thus if data is retransmitted such that new data is buffered, then the channel must have enough bandwidth to transmit faster than new data is being produced, making it possible to “catch up” to real-time. The receiver must then be able to detect that data is coming faster than real time and utilize the appropriate received data. This could be implemented with timestamps such as used in the Real Time Protocol (RTP) [37].

The suggested delays imply that VoIP currently cannot compete with regular circuit switched full duplex traffic. However, the suggested delays may be very acceptable in a simplex system where users speak one at a time using a protocol to declare when he/she is finished.

A service that has become very popular in America is the so-called push–to-talk service using simplex VoIP (for example Nextel's direct connect service). Users in this system can communicate with each other using simplex VoIP. If the users are physically close to each other, this can save the operator a lot of resources compared to a circuit switched call since packets are being routed locally. How large this saving is naturally depends on how the operator's network is implemented, although a packet switched system is always more resource efficient than a circuit switched one. This service is offered both for one-to-one conversations and for larger group conversations.

The benefit for Swedish operators and users is perhaps not for one-to-one simplex VoIP systems (because of currently existing delays and GPRS prices as discussed above), but instead for group based communication where a cellular phone can be used as a walkie talkie type communications radio, but with national coverage. Traffic from a device is sent to a server that replicates the data and sends it to everyone in the group. Large numbers of users can listen to the "radio channel" and push a button to talk just like in an ordinary communications radio system.

Users can have the channel open continuously since it is only virtual and does not lock any resources in the network. This would be both practically (for the operator) and economically (for the user) impossible using circuit switched technology if the groups are large.

Such services have taken a lot of the traffic that traditionally used proprietary networks (frequently called specialized mobile radio) in the USA. A proprietary network is very expensive to build and maintain, especially in urban areas where coverage similar to that of the GSM networks (in Europe) would mean huge investments.

Since a data connection is already established between a group of devices, the speech data could be mixed with other data that is interesting to the group (if there is sufficient bandwidth). As an example, the police might utilize such a system. As a crime is reported, a group could be created for that particular mission, the group then not only functions as a private radio network for the officers involved in the mission, but can also be used to send data such as pictures of suspects, maps, etc. to the group.

The Swedish government has begun investigating the different possibilities of how to implement new public safety services to replace the legacy radio systems. Systems such as TETRA, GSM, EDGE and UMTS are being considered. On the government's request, the consultant AB Stelacon produced a report regarding this matter in 2002 [38].

5.2.3 Radio and core network resource limitations

There are two factors that should be considered along with the dimensioning of a possible overlay network. It is important to consider that even though a certain operator may choose not to implement such a network, a third party could and thus the effects on the operator's network should still be investigated.

The first consideration is the available bandwidth in the fixed part of the GPRS network. As has been mentioned previously most operator's SGSN and GGSN configurations were not dimensioned for the potential traffic of an overlay network where a user might stream data continuously for a while, they were dimensioned for WAP browsing and occasional picture messaging. While there are currently GGSN solutions such as the Siemens @vantage CPG-3000 and Alcatel's Evolium that are capable of throughputs around 150 – 250 Mbit/s, Alcatel's GGSN solution from 2001 only handled a throughput of 20 Mbit/s. Depending on an operators current dimensioning, these nodes may need to be upgraded if services are launched that are expected to utilize large bandwidths, such as streaming services. This could either be done using more powerful SGSN and GGSN solutions or by finer granulation of the GPRS subnets, i.e. by using more GGSNs that handle physically smaller subnets.

Hopefully, this increase in traffic would happen at a pace such that it is possible to upgrade the network as revenue grows.

The above consideration is mainly important if a distributed overlay network that can handle large data loads is created or alternatively if some sort of streaming service on the Internet suddenly became very popular.

In the very near future, the kind of overlay networks that are expected to appear in phones in large numbers are instant messaging overlay networks such as MSN and ICQ. While these do not (at the moment) generate large amounts of traffic, they utilize the technique of keeping a connection alive through the GPRS and NAT system just as the overlay network implementation of this thesis does. The effect on the radio resources if such services become very common should be investigated. The amount of traffic needed to keep a connection alive depends on the specific implementation of an operator's GGSN as well as the NAT function if present. The idle time interval before a connection is terminated by either the GGSN or the NAT function is operator specific, meaning that it could be changed in order to minimize the necessary amount of polling traffic.

Another factor that may have an even larger impact on radio resources is if polling is used to minimize propagation time for a certain application. The GPRS standard specifies that a GPRS device that receives or sends data will move into a ready state for a certain time, listening to a timeslot in each GSM frame for incoming data, after this time the GPRS devices mobility management context (MM_Context) in the SGSN is set to standby state where the network has to page the device in order to find it and make it listen for incoming data again [32]. The paging procedure introduces a lot of propagation time which is why third party software may send keep alive messages very frequently in order to stay in the ready state and maintatin minimum propagation time. The time interval before an MM_context returns to standby state is specified by the SGSN when a device attaches to the GPRS network. This timer can be set to values ranging from all zero, which makes a device immediately return to standby state, to all 1:s which disables the standby state, making a device continuously update the SGSN about it's cell position, hence no paging is necessary.

The cellular clients in the network implemented for this thesis receive keep alive messages from the network ALRs every 15 seconds if no other data is sent, in order for the MM_Context to remain in ready state.

If this time interval is not considered and such applications were to become popular it may have an impact on the statistically available GPRS resources in urban areas. While the amount of polling traffic needed for a device to stay in ready state may cause problems, a device in ready state always updates the SGSN with it's new cell ID when changing cells, which means that the SGSN never has to page the device. As paging is done over an entire routing area (RA) in order to find a device, the potential radio resources that are saved by data only being sent in a single cell might balance the negative effects. As an interesting side effect, having nothing to do with overlay networks, is that the device could be continuously tracked at cell-level by simply interrogating the SGSN over the Gs interface (i.e. "pretending" to be a Mobile Switching Centre (MSC)). The protocol used by the MSC over the Gs interface specifies how to request the user's cell ID. Existing schemes for tracking of a device currently involves continuous paging over an entire Location Area (LA, an RA is a subset of an LA).

It is also worth considering that the radio resources in general may be to small to launch any large scale services that require 3 or 4 timeslots to function. A user utilizing such a service would hence need to allocate 3 or 4 times the amount of radio resources that a user making a circuit switched phone call would. Since the current network of GSM cells was initially dimensioned to handle a certain number of users making circuit switched calls (i.e. allocating 1 timeslot), it may be unlikely that a user would get the necessary resources for the service more than occasionally.

6. Market analysis

6.1 Implementing the technology

6.1.1 Adding functionality to the existing networks, operators or application developers?

Using the technology of overlay networking, functionality can be added to the existing cellular networks. As have been discussed earlier in section 4.1 1, most existing services that enable end applications to communicate with each other, such as for example ICQ and MSN Messenger in the fixed network, and MMS in cellular networks can be considered to use overlay networks on top of the existing IP network (SMS cannot really be viewed as an overlay network, at least not on top of the IP network). While the MSN Messenger and ICQ overlay networks are implemented with centralized servers or ALRs, MMS is implemented with a network of distributed servers.

The functionality that can be added by these networks is large. Apart from the end to end data connectivity which is the primary gain; persistent ID and groupcasting functionality can be implemented.

MSN Messenger, ICQ, and IRC clients are already available in smart phones from Microsoft, Nokia, and Ericsson. These clients connect to the same overlay network as stationary clients and can communicate with these clients as well. It is obvious that building application specific overlay networks works just as well for cellular devices as for stationary computers when it comes to these services, so should an operator bother or just sit back and wait for application developers to create applications that generate traffic? This depends on where an operator expects to get revenue: for applications or for bitshuffling.

When looking at services such as instant messaging, even if they do generate traffic in the GPRS networks, this could be largely at the cost of operator proprietary and very profitable solutions such as SMS and perhaps even MMS being replaced (see section 6.3.1).

As mentioned in section 2.4, Yahoo Messenger already offers users the ability to send files up to 1.5 Mb “inside” the Yahoo Messenger network, as opposed to normal file transfers in most messaging applications which is done on the IP level, i.e. it is possible to send files between devices in cellular networks today using this service although this would at the moment require a laptop to be able to run the PC Yahoo client. Since files using this system are temporarily stored on Yahoo’s servers, delays may be considerable showing that the kind of real-time services that are discussed in this thesis may be difficult to implement. However, as a competitor to the MMS network, such a messaging application for existing cellular devices could prove devastating for operators who expect to make high profits on MMS traffic.

As another example of third party implementation possibilities, the e-mail system could be used. Large centralized mail systems such as Hotmail allow sending and receiving of 1 Mb attachments. This is not limited primarily due to lack of storage space (which has become very cheap over the last couple of years), instead this limit is to avoid the servers becoming congested. This means that the hotmail service can handle the load of receiving and sending (i.e. relaying) sporadic messages up to about 1 Mb and hence these servers should be able to do so in a cellular overlay application as well.

If a user connects to his/her corporate or university mail system using his/her phone, he will likely be allowed to send much larger attachments since these systems do not have the same number of users and hence do not face the bandwidth constraints of the hotmail system. The same would be true for a cellular messaging application, if it was only directed towards a specific group, such as only one operator or perhaps a rather small country as Sweden, thus it could likely offer higher bandwidth services than current Microsoft or Yahoo applications can.

While a third party developer is able to offer instant messaging services and small file messaging services such as for pictures, video clips, etc. without the active support of an operator, they will still have a hard time offering higher bandwidth services such as streaming and real-time services since all data has to be relayed via a central point. Even if these servers are very powerful, the service will sooner or later become congested as the number of customers grow. For example, a user in the proposed UMTS networks can get a maximum of about 400Kbit/s (on the downlink). This means that a third party developer who wants to dimension his/her network for 25000 simultaneous **maximum bandwidth** users would need at least a 10Gbit link, for an international service such as MSN with many million users this is not a simple problem to overcome.

A company could build a network of geographically distributed servers that cater to particular groups or regions. This could make it possible to handle more bandwidth demanding services. Whether it would be enough to provide for example real-time video conversations also depends on the propagation time of the system, but it may be possible.

An operator could build a very efficiently distributed overlay network for high bandwidth services, where each operator's GPRS network has local routing functions, similar to the MMS network. If this overlay network was implemented using well known and standardized protocols, it could be offered to third party developers just as the MMS network is today, but could be used to send arbitrary data in real-time rather than only specific data in non real-time.

The main differences between a third party implementation and an operator implementation may be expected to be the bandwidth and possibly propagation times and reliability. Although these factors seem to be few, they may be very crucial for specific applications.

The incentive for an operator to create such a network is that it can be charged for as a service like SMS or MMS. If a third party implements the network, the operator will only get income from IP traffic.

There also has to be some incentive for a third party to implement such a network, what these incentives might be is discussed further in section 6.3.1.

6.1.2 Necessary market factors

In order for the above scenario to become a reality, several factors need to coincide, some of which are more likely to happen by themselves than others.

In order for instant messaging to become popular and replace SMS and possibly MMS, no particular initiative has to be taken since this is already possible. What is necessary is that development continues in the direction of smart phones that are able to install third party software that can access the Internet. Although these devices are currently perhaps a bit too expensive to become the device of choice for ordinary users and especially the younger crowd, prices will probably decrease as production in high volumes is begun and technology development continues.

In order for an operator driven large scale overlay network for higher bandwidth services to become a reality, agreements within the mobile industry would most likely have to be made and standards created. A single operator could implement a proprietary solution if the customer base of the operator was large enough to create sufficient income. Whether it is or not may be difficult to foresee since it depends on the future popularity of the service.

Using standards that are already available when creating these networks could provide great advantages. The chances of succeeding with an inter-operator spanning network greatly increase if a neutral standard such as JXTA (for a message based system) or source routed SIP (for a stream based system) is used. Even though JXTA was developed by SUN Microsystems, it is an open source project and developed in a community form, so possible discontinued support from SUN would not be a major factor if an initiative was taken by the mobile industry to develop it.

6.2 Who controls the network?

6.2.1 Operators or application developers?

The question of who will control an operator's data network largely depends on the strategy that the operator has. Because of the previously discussed factors, in most cellular networks making direct contact between devices on the IP level is impossible, hence an overlay network of some sort is necessary for these services. This can either be implemented as per application networks like MSN, ICQ, or MMS or it could be implemented as a general network for arbitrary data.

So the question of who controls the operator's network is really a question of who controls the overlay network since this is necessary. If the trend of standalone application networks continue, operators will naturally loose control since there is no way for them to differentiate services and bill for them and they are ultimately reduced to shuffling bits which many believe is not a very lucrative business. In the fixed networks however, this has been the case for quite some time. As an example, the core network part of Swedish operator Telia, called Skanova, was forced by the PTS [62] to form a separate company and let the rest of Telia buy bandwidth in competition with all other parties. The same thing may happen to the operator's cellular access networks in the future.

Simply because a third party **could** implement a proprietary network for text and small file messaging doesn't necessarily mean that he will though. The developer of the service has to have an economic incentive. Depending on whom the developer is, this incentive might differ. For a device manufacturer, the incentive may be that his/her devices become more popular with this service. For developers who do not have this incentive and need paying customers, the operator could offer to bill the customer via his/her cellular subscription (using for example SMS payment) on behalf of the third party, and thus the operator could gain a percentage share on the service in this way.

The easiest way for an operator to control a network is of course to provide it himself as is the case with MMS, where the operator can charge on a per message basis regardless of who created the messaging application. However, since the MMS network is designed for casual messaging which could be implemented quite well by a third party overlay network, the incentive for using the quite expensive MMS network is not strong.

As long as operators are not providing an overlay network that either offers superior performance (as the stream based high bandwidth arbitrary data overlay networks discussed in section 6.1.1) or is cheaper than building and maintaining your own network, developers have ample incentive to use their proprietary solutions.

6.3 Threats and possibilities from an operators point of view

6.3.1 Proprietary solutions such as SMS and MMS in danger

As has been discussed in section 6.1.1, a third party could implement overlay networks for mobile users to communicate data such as text, pictures, and video clips. This would be a direct competitor to the SMS and MMS services offered by the operators, and as such is very serious.

As an example to illustrate the potential impact of third party messaging applications, the following figures are used. A high rate GPRS subscription at the Swedish mobile operator Telia at the time of this thesis writing offers a user 25Mbytes for 300 SEK , yielding a cost of 0.0015 SEK/Kbit. An SMS message consists of 160 characters and a standard camera phone produce pictures that are usually in the vicinity of 20Kbit (if the pictures are taken for the phone's screen resolution which is likely for a phone to phone picture messaging application). Thus a picture sent as general data over the GPRS network rather than as an MMS would generate an income of 0.03*2 =0.06 SEK (times two because the data would be charged both on the uplink and downlink) for the operator. This is almost 1/100th of the cheapest suggested price for an MMS in Sweden, which is currently 4.80 SEK (Vodaphone) .

The 160 characters of an SMS would generate 0.00192*2=0.00384 in pure GPRS revenue, giving a factor of 1/500th to 1/1000th the price of an SMS depending on the operator subscription, and time of day.

The above is also true if the user chooses to send the text or picture as email, if the phone is capable of this (some extra overhead is added for the email protocols, but it is still very inexpensive compared to SMS and MMS).

The problem of an operator not being able to differentiate one type of IP traffic from another becomes apparent. Furthermore, if operators want to incorporate future high bit rate services such as video streaming in GPRS, prices must decrease dramatically, resulting in low bit rate services such as those above generating very little income at all.

Although the SMS and MMS networks are efficient for one to one messaging, they are currently less useful for group messaging due to the implementation of the billing system where the sender always pays for the transaction rather than the sender and receiver both paying, This means that the whole cost of sending a message to a group is put on the sender and not distributed over the group. If the message is instead sent over GPRS as opaque data, replicated in a server (in the case of email for example) and sent to the rest of the group, the sender is charged for sending one message and the receivers are each charged for receiving their copy.

6.3.2 Device manufacturers posing the greatest threat

While most application developers do not have an incentive to develop and maintain a large and expensive solution for data messaging that could compete with SMS and MMS if they cannot charge for the messages being sent, device manufacturers do have this incentive if it give their devices will have a competitive edge because services exist for **their** device.

Even though Microsoft does not get paid directly for the traffic that MSN Messenger is relaying, the service greatly increases the value of their operating systems for PCs as well as their phones, which in turn generates income.

In the new Symbian based camera phones from Nokia, a user taking a picture has the choice of sending the picture to Nokia's photo zone, where it can be shared with others, making the step to a full featured picture exchange application quite small. The user also has the choice of sending the picture as an email.

In a country like Sweden, the few and quite dominant device manufacturers also have a competitive edge compared to the operators because all of their devices can be equipped with the application at the time of purchase. If an operator is to create such a service, customers must be persuaded to download and start using the new application. Alternatively, since phones are often bought bundled with an operator subscription, the operator could add their own applications. Depending on the operator's dominance in the market this may work to create a de facto standard or not.

Operator branding of phones is beginning to take place with operators like Vodaphone in the lead. The Symbian OS was developed to be able to brand. The question remains whether it is the manufacturers or the operators that will brand the phones in the future. Operators must most likely still cooperate with each other in order to be able to create de facto standards in a country like Sweden.

Although it will be difficult to stop these kinds of services, an operator still has a number of advantages compared to the device manufacturers that would make it interesting for a manufacturer to cooperate when developing services. These advantages are discussed in the section 6.4.

6.4 Means of competition

6.4.1 Providing a billing system to third party service providers

While a device manufacturer may decide that enhanced functionality alone is enough to justify the investment in device to device communication services that compete with the operator's own services, other third party developers need revenue. Creating a billing system is not only an expensive task, but it can also be difficult to convince customers to sign up and pay yet another bill every month.

However, if the service is billed for via the existing (operator) subscription without the user having to do anything, the chances of the service becoming a success may increase greatly compared to competing services.

6.4.2 Providing network specific services to third party service providers

An operator can also use the fact that he has access to the physical network in order to create so called value added services. These could be used either to make the operator's own services more competitive or sold to a third party in order to get revenue.

A typical example of such a service is to provide positioning data or certain network statistics to third parties.

6.4.3 Providing reliable QoS and security to customers with special needs and critical systems.

Perhaps the greatest means of competition for an operator is to provide security and reliability to customers. While users of an application such as instant messaging may not be very concerned with security and reliability this may be crucial in a company solution.

Owning the access networks means that the operator can guarantee security and Quality of Service all the way to the customer. As soon as data reaches the operators overlay network, he can encrypt it, tunnel it, etc. before it reaches the Internet. A third party such as the device manufacturers can never guarantee this without the support of the operator since their servers have to be on the Internet. The security issue can instead be solved by the end devices encrypting the information end to end at the cost of some added complexity in the end devices as well as some extra overhead.

Another benefit of being able to implement a service totally inside the operator's network is that services that are aimed at a certain group of subscribers can be totally independent of the Internet and whether or not delays and congestion appear elsewhere.

While delays and temporary congestion may not be crucial to an instant picture messaging application, a service such as the push-to-talk service discussed in section 5.2.2 necessitates low delays as well as extremely high reliability, if for example being used by a security company.

Closely linked to the reliability issue is the fact that an operator can distribute a service physically in his/her network in order to handle a heavy traffic load while a third party cannot. This has been thoroughly discussed in section 6.1.1 chapter where it has been discussed why a third party will have a very difficult time providing any services that necessitates higher bandwidth than casual pictures and text messages do.

7. Conclusions

Distributed overlay networks could be used to create general data connectivity and the ability to form generic groups of applications in the existing GPRS networks. This could enable networking services such as personal groups of a user's applications running on different machines (e.g. home PC, work PC, laptop, car, phone, etc.) and groups of trusted applications in chat, picture messaging, video messaging, gaming, and streaming voice and image scenarios.

Whether such overlay networks should be implemented specifically for a service (as this thesis has) or as a general overlay network similar to the MMS network depends on whether or not third party development is desired.

For services such as VoIP push-to-talk services, a specific overlay network may provide a better solution since it could be optimised for low propagation delay. However, if it is possible to create a general platform (although perhaps not externally standardized) this has the advantage of being reusable for other applications and can be offered to third party developers.

For a global "real-time" version of the MMS overlay network to become possible a large amount of standardization work is needed. However, since no low level reconstruction of the physical network is needed, a small scale overlay network could be initiated by a single or a few operators or even third parties at rather low initial investment costs (see section 6.1.1). In Sweden for example, three GSM operators currently cover the entire population. This has the advantage that services may be launched within a matter of months. It may also be possible for a standard to evolve over time as a de-facto standard. In order to increase the possibilities of a de-facto standard evolving (i.e. that others will adopt), existing open, non-operator proprietary standards should be used (such as SIP, JXTA, etc.) if possible.

A transition to IPv6 is likely to take place in the future as the UMTS networks develop. The 3GPP release 6 specifies the use of IPv6. It also specifies the use of a signalling system called IP multimedia Subsystem (see section 5.1.5) intended to create IP level data connectivity between devices. This system uses SIP to inititiate IP sessions. Although IMS in itself does not provide generic group services there are also specifications that such a system should be available in these networks (though it is not specified whether these should be implemented as a single server, a network of distributed servers, or using IP level multicast).

Even though large scale UMTS networks using IMS may be several years into the future an overlay system designed with IMS in mind may be able to take advantage of this functionality as current systems are migrated to UMTS. As an example, if SIP was used to initiate a session to an operators overlay network (i.e. an ALR) and the evolving 3GPP specifications for creating generic groups were used to create groups in this overlay network, applications may be migratable to UMTS and are more likely to be adopted by others.

The overlay networks discussed in this thesis could be implemented by different parties. Operators have the advantage of owning the network and being able to dimension it effectively, device manufacturers have the advantage of owning the devices in the network and being able to deliver preinstalled and configured applications creating de-facto standards. Operator branding of phones is becoming increasingly popular. Using branding an operator can achieve both the above benefits.

An overlay network and related applications could also be offered by a third party that does not own devices or physical networks. It is possible that a free distributed overlay network for messaging, and information sharing similar to those used for file sharing in the fixed network could evolve. Although such a network may not be as physically/geographically well dimensioned as an operator implementation it may serve very well for traffic such as casual picture (or any other file) messaging that does not have hard real-time constraints. The implementation in this thesis shows that such a distributed network is possible without the support of operators. If users register in such network with their phone number as username, the similarities with MMS would be large and potential loss of MMS revenue could be severe (see section 6.3.1).

Although overlay networks may be considered a threat to certain services that are expected by some to make up for a large part of their UMTS investments, they can also be considered the potential enabler that produces GPRS traffic and hence gives an incentive to invest in the proposed higher bandwidth networks.

8. Future work

There are several areas within overlay networking that should be examined further. Both in the case of a possible implementations and for the sake of being prepared for how a third party implementation may affect the network.

Security issues in overlay networks have not been discussed in any detail in this report. This area must be extensively investigated if services are to be offered to corporations or organizations such as police and fire departments.

Market analysis should be performed to establish what potential consumers request and demand of such a network in order for business cases to be created.

The possibilities of integrating an overlay network with possible future technology such as the IMS should be further investigated and participation in IMS standardization initiated, it may be possible to influence the standardization of such functionality in future 3GPP releases.

The effects of a third party overlay network implementation should be more thoroughly investigated. Free overlay networks for file sharing on the Internet have managed to attract large numbers of users very quickly. What would the expected loss of income for MMS and SMS services be if a similar network was developed for information sharing between wireless devices. What category of customers may be expected to start using such a system and more importantly, could currently available radio and GPRS core network resources handle a potential sudden and vast increase in GPRS usage? Such a service might congest the network for users of operator provided services such as email and WAP browsing.

9. Bibliography

[1] B. Shrader "A proposed definition of 'Ad hoc network'", Investigative paper, Royal Institute of Technology (KTH), Stockholm, Sweden, May 8th 2002

[2] JXTA community, http://www.jxta.org/, accessed on January 17th 2003

[3] Internet Protocol version 6, RFC 2460, December 1998

[4] Internet Protocol version 4, RFC 791, September 1981

[5] General Packet Radio System, http:// www.etsi.org/, accessed on January 17th 2003

[6] K. Anderson "Analysis of the traffic on the Gnutella network" CSE222 Final project paper, University of California San Diego, March 2001

[7] JXME implementation, http://jxme.jxta.org/, accessed on January 17th 2003

[8] 3GPP TS 22.250 v2.0.0 "IMS Group Management, Stage 1> (Release 6)" November 2002

[9] MSN Messenger http://www.msn.com, accessed on January 17th 2003

[10] ICQ, http://www.mirabilis.com, accessed on January 17th 2003

[11] SETI@home, http://setiathome.ssl.berkeley.edu/ , accessed on January 17th 2003

[12] Folding@home, http://folding.stanford.edu/, accessed on January 17th 2003

[13] Direct Connect, http://www.neo-modus.com/, accessed on January 17th 2003

[14] Yahoo Messenger, http://messenger.yahoo.com/, accessed on January 17th 2003

[15] E. Adar and B. Huberman, "Free Riding on Gnutella," *First Monday*, vol. 5, no. October 10th 2000

[16] Freenet, http://freenetproject.org/cgi-bin/twiki/view/Main/WebHome, accessed on January 17th 2003

[17] Abacast Streaming media software, http://www.abacast.com/, accessed on July 10th 2002

[18] Blue Falcon Networks, http://www.bluefalcon.com/, accessed on July 10th 2002

[19] Nitin Desai, "KONARK -An Adhoc Service Discovery Protocol.", A Thesis presented to the graduate school of the university of Florida , 2002

[20] Y. Ko and N. H. Vaidya, "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks" In Proceedings of the 4th Annual International Conference on Mobile Computing and Networking (MOBICOM'98), October, 1998.

[21] Siemens AG, "Adhoc Awareness", CT SE2 (Corporate Technology, Software & Architectures department), 2002

[22] Nextel\Blackberry, http://www.nextel.com, http://www.blackberry.net/, accessed on July 10th 2002

[23] Nextel Direct connect, http://www.nextel.com/services/directconnect.shtml, accessed on January 17th 2003

[24] MOBITEX, http://www.ericsson.com/network_operators/mobitex/about.shtml, accessed on January 17th 2003

[25] Symbian OS, http://www.symbian.com/, accessed on January 17th 2003

[26] Hantro, "MPEG4 Video Codec overview" 2002

[27] Bluesoft inc., http://nt.bluesoft-inc.com/index_e.asp, accessed on July 10th 2002

[29] Session Initiation Protocol , RFC 3262, June 2002

[30] An IPv6-to-IPv4 transport relay translator, RFC 3142, June 2001

[31] 3rd Generation Partnership Projet, http://www.3gpp.org/, accessed on January 17th 2003

[32] Lin Y., Chlamtac I., "Wireless and Mobile Network Architectures" first edition, 0471-39492-0, John Wiley and sons, Inc. 2001

[33] Packet Video, http://www.packetvideo.com, accessed on January 17th 2003

[34] R.G. Cole and J.H. Rosenbluth, "Voice Over IP Performance Monitoring", Computer Communication Review, a publication of ACM SIGCOMM, volume 31, number 2, April 2001. ISSN # 0146-4833. is available from: http://www.acm.org/sigcomm/ccr/archive/2001/apr01/ccr-200104-cole.html, accessed on February 6th 2003

[35] Parantainen J., Hamiti S., "Delay Analysis for IP speech over GPRS", IEEE VTC 1999, 829-833

[36] Bellalta B. Oliver M. Rincón D. "Capacity traffic analysis of VoIP services over GPRS mobile networks" Technical University of Cataluña (UPC) and Universitat Pompeu Fabra (UPF) Barcelona Spain 2001

[37] Real-time transport protocol, RFC 1889, January 1996

[38] AB STELACON 2002, "Finansiell och funktionell analys av alternativa tekniker för gemensamt radiokommunikationsnät för skydd och säkerhet" September 23rd 2002

[39] ETSI TS 126 090 V.3.1.0 2000-01 AMR speech transcoding functions 3G-TS 26.090 Technical specification

[40] M. Papadopouli and H. Schulzrinne. "Network Connection Sharing in an Ad Hoc Wireless Network among Collaborative Hosts", *Proceedings of NOSSDAV*, Jun.1999.

[41] Maria Papadopouli and Henning Schulzrinne. "Seven Degrees of Separation in Mobile Ad Hoc Networks", In *IEEE GLOBECOM*, , San Fransisco Nov. 27th-Dec. 1st, 2000.

[42] S. Banerjee and P. K. Chrysanthis, "Peer Support in a MobileWorld" Position paper (invited), Proceedings of the NSF Workshop on Context-Aware Mobile Database Management, Providence, January 2002.

[43] R. Schollmeier and I. Gruber. Routing in Peer-to-Peer and Mobile Ad Hoc Networks: A Comparison. In *Proc. of the Workshop on Peer-to-Peer Computing, held in conjunction with IFIP Networking 2002*, Pisa, Italy, May 24, 2002.

[44] D. Brookshier, D. Govoni, N. Krishnan, J. C. Soto, "JXTA: Java P2P Programming" first edition, ISBN: 0672323664, SAMS Publishing, March 2002

[45] Stevens A. "ARM 3D Graphics solution" ARM, 2002

[46] The IP network adress translator (NAT), RFC 1631, May 1994

[47] Java 2 Second Edition, http://java.sun.com/j2se/, accessed on February 2nd 2003

[48] Mobile Information Device Profile, http://java.sun.com/products/midp/, accessed on February 2nd 2003

[49] Dynamic DNS service, http://www.dynu.com/, accessed on February 2nd 2003

[50] Jeode Java virtual machine by Insignia, http://www.insignia.com/, accessed on February 2nd 2003

[51] Mobile IP, RFC 2002, October 1996

[52] Telephone Number Mapping (ENUM), RFC 2916, September 2000

[53] Secure IP (IPsec), Security Architecture for the Internet Protocol, RFC 2401, November 1998

[54] Eugene Shih, Seong-Hwan Cho, Nathan Ickes, Rex Min, Amit Sinha, Alice Wang, and Anantha Chandrakasan, "Physical Layer Driven Algorithm and Protocol Design for Energy-Efficient Wireless Sensor Networks", Proc. MOBICOM 2001, Rome, Italy, July 2001. Available from: http://www-mtl.mit.edu/research/icsystems/uamps/pubs/eugene_mobicom01.pdf, accessed on February 6th 2003

[55] IP Unplugged, http://www.ipunplugged.com/, accessed on February 2nd 2003

[56] Common Object Request Broker Architecture (CORBA) 3.0.2. Spec. available from: http://www.omg.org/technology/documents/formal/corba_iiop.htm, accessed on February 6th 2003

[57] JINI, Sun Microsystems, Spec. available from: http://wwws.sun.com/software/jini/, accessed on February 6th 2003

[58] Information Society, European Commission, "Commission calls for rapid move towards new generation Internet Protocol to secure success of future wireless services", http://europa.eu.int/information_society/topics/ebusiness/ecommerce/3information/keyissues/epolicy/01_05/text_en.htm, accessed on February 6th 2003

[59] IPv6 Task Force, http://www.ipv6tf.org/, accessed on February 2nd 2003

[60] David B. Johnson and Charles Perkins, "Mobility Support in IPv6", Internet Draft (work in progress), draft-ietf-mobileip-ipv6-20.txt, January 20th 2003

[61] Common Open Policy Service (COPS), RFC 2748, January 2000

[62] Post- och Telestyrelsen (PTS). http://www.pts.se/, accessed on February 2nd 2003

[63] D. Dittrich "Distributed Denial of Service (DDoS) Attacks/tools", http://staff.washington.edu/dittrich/misc/ddos/, accessed on February 2nd 2003

[64] Integrated Digital Enhanced Network (IDEN), http://idenphones.motorola.com, accessed on February 2nd 2003

[65] Wireless Application Protocol Architecture, WAP-210-WAPArch-20010712-a, http://www.wapforum.org, accessed on February 2nd 2003

[66] Post Office Protocol version 3 (POP3), RFC 1939, May 1996

[67] Simple Mail Transfer Protocol (SMTP), RFC 2821, April 2001

[68] Multipurpose Internet Mail Extensions (MIME), RFC 1341, June 1992

[69] Multimedia Messaging Service architecture overview, WAP-205-MMSArchOverview-20010425-a, http://www.wapforum.org, accessed on February 2nd 2003

[70] Digital Video Broadcast (DVB), http://www.dvb.org/, accessed on February 2nd 2003

[71] Digital Audio Broadcast (DAB), http://www.worlddab.org/, accessed on February 2nd 2003

[72] The Open System Interconnect model (OSI), http://www.etcs.ipfw.edu/~lin/cpet355/S2000/Lectures/Lect3OSIModel_files/frame.htm, accessed on February 2nd 2003

[73] Telia, http://www.telia.se, accessed on February 2nd 2003

[74] Forum Nokia, http://www.forum.nokia.com/, accessed on February 2nd 2003