VAASA POLYTECHNIC

Markku Mikael Hautamäki

USING GPRS AS A WIRELESS CORE NETWORK FOR WIRELESS LOCAL AREA NETWORKS

Technology and Communication
2003

FOREWORD

This final thesis was made for the Wirlab Research Center in Seinajoki between 1st of February 2003 to 25th of April 2003. Wirlab is a research center that does research on future networking real-life environments. Research and development of Wirlab concentrates on combining the traditional wired networks with the wireless solutions and IP-networking on top of them. Wirlab is based on co-operation with research centers, universities, enterprises and networking experts.

The supervisor of the thesis was Mr. Chao Gao on behalf of Vaasa Polytechnic and the contact person was Mr. Kari Virtanen on behalf of Wirlab Research Center.

I would like to thank the employees of Wirlab Reseach Center for all the help I received from them during the project.

Vaasa 27.4.2003,

Markku Hautamäki

VAASA POLYTECHNIC

Degree Programme of Electronics and Information Technology

ABSTRACT

Author	Markku Hautamäki
Topic	Using GPRS as a Wireless Core Network for Wireless Local Area Networks
Year	2003
Language	English
Pages	64+ 9 appendices
Name of Supervisor	Gao Chao

The purpose of this thesis was to find out the possibilities of using the General Packet Radio Service (GPRS) network as a wireless core network for Wireless Local Area Networks (WLAN).

WLAN covers only a limited area. During this project a router was implemented to route the traffic from WLAN into a GPRS network to provide a wireless connection from WLAN to the outside data networks, such as the Internet.

The router was implemented on a normal desktop computer. The GPRS connection was provided using a Nokia D211 GPRS PC card. Network Address Translation (NAT), Dynamic Host Configuration Protocol (DHCP) server, Domain Name System (DNS) server and a proxy server were also implemented on the same computer. The router was tested on a test network that was built in Wirlab Research Center's laboratory. The router was working and the testing was successful. Although GPRS has quite a limited bandwidth, it can be used as a wireless core network for WLAN.

UDK 621.313.13
Keywords: GPRS, WLAN, routing

VAASA POLYTECHNIC

Degree Programme of Electronics and Information Technology

TIIVISTELMÄ

Tekijä	Markku Hautamäki
Opinnäytetyön nimi	GPRS:n käyttö WLAN verkkojen langattomana runkoverkkona
Vuosi	2003
Kieli	Englanti
Sivumäärä	64+ 9 liitettä
Ohjaaja	Gao Chao

Tämän opinnäytetyön tarkoituksena oli selvittää mahdollisuudet GPRS:n (General Packet Radio Service) käyttämiseen langattomien lähiverkkojen (WLAN) langattomana runkoverkkona.

WLAN verkko kattaa rajatun alueen. Tämän projektin kuluessa rakennettiin reititin joka reitittää tietoliikenteen WLAN verkosta GPRS verkkoon ja näin ollen mahdollistaa langattoman yhteyden WLAN verkon ulkopuolisiin tietoliikenneverkkoihin.

Reititin rakennettiin tavallisesta tietokoneesta. GPRS yhteys saatiin Nokia D211 GPRS PC-kortilla. Samalle koneelle liitettiin myös Network Address Translation (NAT), DHCP palvelin, DNS palvelin ja proxy palvelin. Reititin testattiin testiverkossa, joka rakennettiin Wirlab Reseach Centerin laboratorioon.

Reititin toimi testattaessa. GPRS:n melko hitaasta tiedonsiirtonopeudesta huolimatta, sitä voidaan käyttää WLAN verkkojen langattomana runkoverkkona.

UDK	621.313.13
Asiasanat	GPRS, WLAN, reititin

MARKINGS AND ABBREVIATIONS USED

AP	Access Point
ARP	Address Resolution Protocol
BIND	Berkeley Internet Name Domain
BSS	Base Station Subsystem
BTS	Base Transceiver Station
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DSSS	Direct-Sequence Spread-Spectrum
EAP	Extensible Authentication Protocol
ESSID	Extended Service Set Identification
ETSI	The European Telecommunications Standardization Institute
FHSS	Frequency Hopping Spread-Spectrum
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HLR	Home Location Register
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IR	Infrared
ISP	Internet Service Provider
LLC	Logical Link Control
MAC	Media Access Control
MS	Mobile Station
MSC	Mobile Switching Center
NAT	Network Address Translation
OSI	Open System Interconnect

PCMCIA	Personal Computer Memory Card Association
PCU	Packet Control Unit
PDA	Personal Digital Assistant
PDU	Protocol Data Unit
PN	Pseudo-Random Noise
PPP	Point-to-Point Protocol
RF	Radio Frequency Technology
RTS/CTS	Request to Send/Clear to Send Protocol
SGSN	Serving GPRS Support Node
SIM	Subscriber Identity Module
SNR	Signal-to-Noise Ratio
TCP	Transmission Control Protocol
TTL	Time-to-Live
UDP	User Datagram Protocol
VLR	Visitor Location Register
WEP	Wired Equivalent Privacy
WLAN	Wireless Local Area Network

CONTENTS

1. INTRODUCTION

The purpose of this thesis is to find out the possibilities of using the General Packet Radio Service (GPRS) network as a wireless core network for Wireless Local Area Networks (WLAN).

The General Packet Radio Service is a service that allows information to be sent and received across a mobile telephone network. The GPRS allows the user to access external IP networks such as the public Internet or enterprise intranet wirelessly. It is an additional service that provides packet radio access for mobile Global System for Mobile Communications (GSM) telephone network users and this way allows both packet switched and circuit switched traffic in the GSM network. GPRS is included in most of the new mobile telephones and also GPRS PC cards for laptop computers and Personal Digital Assistant (PDA) devices are available

Wireless Local Area Network (WLAN) is a Local Area Network in which the information is sent and received using Radio Frequency (RF) technology. WLAN enables high bandwidth and mobility inside its coverage area and makes the construction of the network easier because the need of cabling is omitted. A number of different WLAN standards exist. The European Telecommunications Standardization Institute (ETSI) has a standardized WLAN called HiperLAN2. The American community, the Institute of Electrical & Electronics Engineers (IEEE) has a more popular standard 802.11b, 802.11a and recently 802.11g. The 802.11 standard is also known as Wi-Fi. Typically a WLAN network consists of the following core components:

1. A WLAN access point (AP), which broadcasts messages on a certain frequency and listens for responses from its clients.
2. The WLAN access card is the client interface that talks to the access point. Typically it is a PC card that can be inserted into a laptop computer or a PDA.

This is so called infrastructure topology of a WLAN network. WLAN can be set up also using so called ad hoc topology, in which the clients send and receive messages between each other instead of through the access point. The WLAN access point can provide a sufficient connection from a 30 to 50 meter indoor location, giving the users the freedom to move within the coverage area. If the users need a connection to data networks outside their WLAN, the Internet for example, there has to be a gateway that routes the WLAN traffic to another data network, just as in a regular Local Area Network.

If we provide a wireless connection to data networks outside a WLAN, we have new possibilities for WLAN technology. We can build a WLAN in the areas where it is not wise or even possible to provide a gateway using the normal cabling, e.g. telephone line, DSL line or optical fiber. We can also have a "mobile" WLAN with a connection to other data networks, for example a WLAN inside a moving vehicle.

At the moment GPRS provides the best data transfer rates in the existing mobile telephone networks, for example GSM network, which covers almost all of Finland. GPRS is available in almost all of the GSM operators´ networks today. These are the reasons why in this thesis the possibilities of GPRS as a wireless core network for WLAN are to be investigated. The limited bandwidth of GPRS is of course restrictive and this restriction has to be kept in mind in finding out how GPRS can be used for this purpose or whether it can be used at all. The overall performance of WLAN with GPRS as a gateway has to be found out and also the cost of the system has to be compared to the performance.

2. THE AIMS OF THE PROJECT

Transmission Control Protocol (TCP) and Internet Protocol (IP) are the two most important protocols used on the Internet. A protocol is a set rules (standards) specifying how computers should communicate in different contexts and how various components of the software and hardware should interact. TCP/IP networking is the technical foundation of the Internet and most modern LANs. It provides mechanisms for communication between computers and for moving information around. TCP/IP is a set of standards for how these mechanisms work, so that software and hardware from different vendors can "inter-operate" [10].

WLAN covers only a limited area of about 30 to 50 meters. If we want to reach longer distances from a WLAN wirelessly we have to route the TCP/IP traffic from WLAN to another wireless network. The aim of this thesis project is to find out a way to route the traffic from WLAN into a GSM mobile telephone network using the General Packet Radio Service of GSM.

In a WLAN a host can talk to another host in the same Ethernet network using TCP/IP on an Ethernet interface. All other hosts can be accessed only through special purpose machines called gateways. A gateway is a host that is connected to two or more physical networks simultaneously and is configured to switch packets between them [7].

To be able to talk to hosts in other networks through GPRS, a gateway with a GPRS connection has to be built and WLAN connected into this same gateway. A computer talks to a GPRS phone through a Point-to Point Protocol (PPP) interface. This means we need an Ethernet interface and a PPP interface in a gateway that routes the traffic from WLAN to GPRS.

IP requires a hardware-independent addressing scheme. This is achieved by assigning each host a unique 32-bit number called IP address [7]. GSM operators use so called dynamic IP addressing for GPRS. This means that you are assigned a different IP address every time you connect through GPRS. To connect WLAN

to other networks through GPRS, the IP addresses inside WLAN have to be isolated from the occasionally changing outer IP address of the GPRS connection. This can be done using Network Address Translation (NAT). NAT translates the internal IP addresses of WLAN or wired LAN into a single outside IP address visible to other networks.

In a WLAN it is important that when a user starts her computer to connect to it, she is assigned an IP address automatically. This can be done using Dynamic Host Configuration Protocol (DHCP). A computer in the network can be configured to work as a DHCP server and this way automate the IP addressing.

The bandwidth of GPRS is limited. We have to find out ways of maximizing the performance of the connection through the GPRS gateway. One effective way to do this is to use a proxy server. Proxy servers can improve the performance by caching the Web pages they retrieve. The next time any of the users on the WLAN wants to retrieve the same Web page, the proxy can send it to the client straight from the cache memory through the WLAN connection. The Web page is loaded much faster this way.

Since we are building a LAN with an Internet connection it is a good idea to use the Domain Name System (DNS). It lets you to refer to the machines inside your LAN by names instead of IP numbers. The DNS is a client/server facility that translates names into IP addresses so that humans can use easily-remembered names, while still allowing TCP/IP internally to use numeric IP addresses as usual [10].

The above elements will be implemented to serve the WLAN in this thesis project. All of them will be implemented on one computer, the router computer. The router will be running Linux operating system because of its complete implementation of TCP/IP networking. Figure 1 illustrates the system that will be implemented.

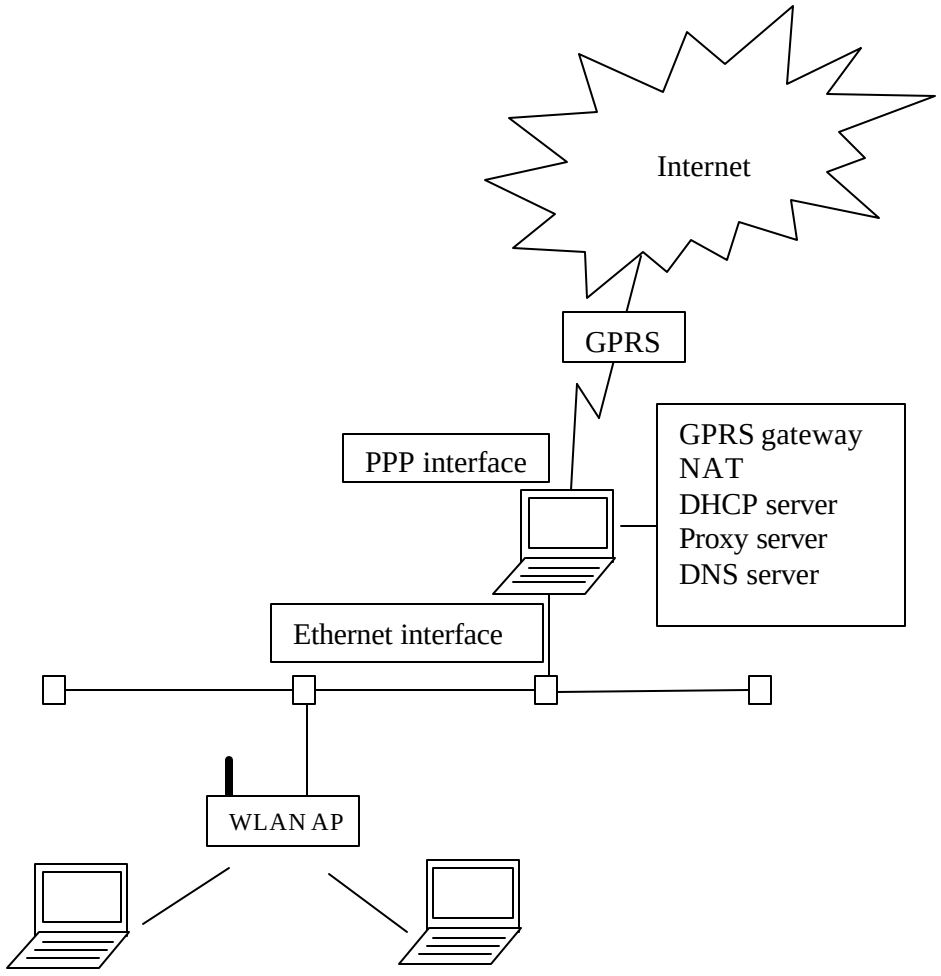


Figure 1. The system to be implemented.

3. GENERAL PACKET RADIO SERVICE

The objectives of the General Packet Radio Service (GPRS) are to enable the packet transmission (to have a communication "always on" and to use the radio channel, or to transmit, only when needed) and to enhance the data rate in the radio interface transmission. [9]

3.1 GPRS Terminals

There are three classes of GPRS terminals, A, B and C. Class A terminal supports the basic GSM services such as SMS and voice simultaneously with the GPRS. Class A terminal users can make or receive calls on both, a packet or switched call type simultaneously.

Class B terminal can monitor GSM and GPRS channels simultaneously but it can support only one service at a time. Class B terminal users can make or receive calls on either a packet or a switched call type one at a time, but not simultaneously.

In a class C terminal user must select the service to be connected to and the other service that is not selected is not available simultaneously. A class C terminal can make or receive only of the selected type of calls.

3.2 GPRS Architecture

GPRS is basically an overlay network on a mobile GSM network. It provides packet data transport at a maximum theoretical rate of 171.2 kilobits per second (kbps). The data rate depends on the radio interface coding and the number of the time slots used. There are four different coding schemes from CS1 to CS4, which represent bit rates from 9.05 to 21.4 kbps, respectively. The maximum number of time slots that can be used for GPRS is 8. At the moment the usual coding scheme used by the operators is CS2, which represents 13.4 kbps and the theoretical maximum data transfer rate of 107.2 kbps when eight time slots are used. Actually only three or four time slots are used.

GPRS attempts to reuse the existing GSM network elements as much as possible, but in order to effectively build a packet-based mobile cellular network, some new network elements, interfaces and protocols that handle packet traffic are required [4].

The Base Station Subsystem (BSS) that consists of Base Transceiver Stations (BTS) and a Base Station Controller (BSC) requires a software upgrade in the BTS and an installation of Packet Control Unit (PCU) and a software upgrade in the BSC.

The GSM core network requires two new components to be able to handle packet transmission. The Mobile Switching Centers (MSC) are using circuit-switched central office technology and hence can handle only the normal circuit-switched GSM transmission that it receives from the BSSs connected to it. The two new components needed to handle packet-based transmission are called Serving GPRS Support Node (SGSN) and Gateway GPRS Support Node (GGSN).

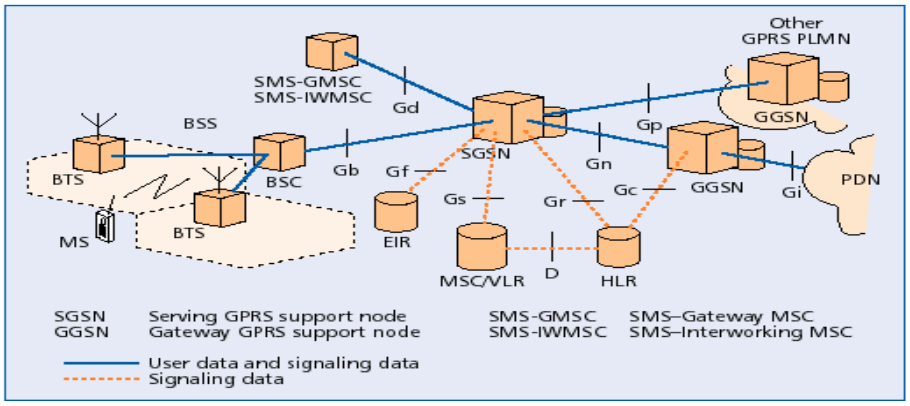
The SGSN can be viewed as a "packet-switched MSC;" it delivers packets to mobile stations (MSs) within its service area. SGSNs send queries to home location registers (HLRs) to obtain profile data of GPRS subscribers. SGSNs detect new GPRS MSs in a given service area, process registration of new mobile subscribers, and keep a record of their location inside a given area. Therefore, the SGSN performs mobility management functions such as mobile subscriber attach/detach and location management. The SGSN is connected to the base-station subsystem via a Frame-Relay connection to the PCU in the BSC. [4]

GGSNs are used as interfaces to external IP networks such as the public Internet, other mobile service providers´ GPRS services, or enterprise intranets. GGSNs maintain routing information that is necessary to tunnel the Protocol Data Units (PDUs) to the SGSNs that service particular MSs. Other functions include network and subscriber screening and address mapping. One or more GGSNs may be provided to support multiple SGSNs. [4]

The databases in the GSM network, such as the Home Location Register (HLR) and the Visitor Location Register (VLR) that handle the mobility management in the network, also require software updates to be able to handle the GPRS functions. The SGSNs communicate with each other and update the user location. Subscriber information and a part of the mobile information to route the incoming calls are stored in the HLR. Also the VLR address is stored in the HLR. The subscriber information is dynamically stored in the VLR. SGSNs can access the VLRs via the GSM MSC. While a Mobile Station (MS) is in the coverage area of a specific SGSN, a logical link is maintained between them to track the location of the MS. When the MS moves out of the area of a specific SGSN, the link is released.

Because GPRS does not require any dedicated end-to-end connection, it only uses network resources and bandwidth when data is actually being transmitted. This means that a given amount of radio bandwidth can be shared efficiently and simultaneously among many users. [2]

GPRS architecture is illustrated in figure 2.



Source: Mohan, Anoop [11]

Figure 2. GPRS Architecture

4. WIRELESS LOCAL AREA NETWORK

A WLAN is a flexible data communications system implemented as an extension of or as an alternative for a wired LAN. Using radio frequency (RF) technology, WLANs transmit or receive data over the air, minimizing the need for wired connections. Thus, WLANs combine data connectivity with user mobility. [2]

4.1 WLAN Network Topologies

WLAN connection between the client and the user is done by wireless medium such as RF or infrared (IR) connection, RF being the most popular medium. Usually the user in a WLAN has a laptop computer with an RF interface adapter, for example a PC card installed in the PC card slot of the laptop. The WLAN adapter can be also in a form of Cardbus, PCI or USB. The connection between the user and the wired network such as Ethernet is accomplished through an access point, which acts as a wireless gateway to the wired network for multiple users simultaneously. This network topology of WLAN is called infrastructure topology.

WLAN can be used also independently of a wired network. Multiple computers with a WLAN adapter installed can form a stand-alone wireless network using wireless peer-to-peer connections to communicate with each other. This network topology of WLAN is called ad hoc topology.

4.2 Modulation Schemes

Most WLAN systems use spread-spectrum technology. It is a wideband radio frequency technique that uses the whole spectrum in a shared fashion instead of dividing it into smaller private pieces. The transmission power is spread over the whole used spectrum, which means that the power levels are lower and the bandwidth is wider. Although spread-spectrum approach is less efficient use of bandwidth than the narrowband approach it has several advantages. The overall Signal-to-Noise Ratio (SNR) is improved. Noise is usua lly bursty and narrowband

in nature, this is why a spread-spectrum signal is more immune to interference. If the receiver knows the parameters of the broadcasted spread-spectrum signal it can easily detect it. If the receiver is not tuned into the correct frequency, the spread-spectrum signal will look like background noise to the receiver. This makes the eavesdropping and jamming of the spread-spectrum signal more difficult. Two modulation schemes are used to encode spread-spectrum signals, Frequency Hopping Spread Spectrum (FHSS) and Direct-Sequence Spread-Spectrum (DSSS).

In frequency hopping, the spread-spectrum signal is generated by "hopping" the carrier frequency in accordance with some pseudo-random noise (PN) code. The same PN code used by the transmitter must be used by the intended receiver [5].

Unlike frequency hopping, in which the signal is spread by periodically changing the frequency, direct sequencing modulates the original base-band signal with a very wide band digital signal. The name direct sequence stems from employing a high-speed PN code directly to the digital information being sent. [5]

In the FHSS system the radio design is simpler than in the DSSS but the transfer rates are limited to 2 Mbps by regulations. Using DSSS transfer rates up to 54 Mbps can be reached.

4.3 IEEE 802.11b WLAN Standard

The most popular WLAN standard today is the IEEE 802.11b standard. The 802.11b standard defines two modes: infrastructure mode and ad hoc mode. Infrastructure mode is the AP-based topology and ad hoc mode is the peer-to-peer topology. 802.11b operates, like most of the radio-based standards, in the 2.4 GHz unlicensed band. In the 2.4 GHz band users can operate without a license. The 2.4 GHz band has been designed license free by the ITU and is available as such in most countries of the world [2].

4.3.1 The OSI Model of IEEE 802.11b

The top five layers of the open system interconnect (OSI) model are the same in both 802.11b and the normal 802.3 Ethernet LANS. The two bottom layers, data link layer and physical layer, are modified in the 802.11b wireless standard. The OSI model of 802.11b is illustrated in figure 3.

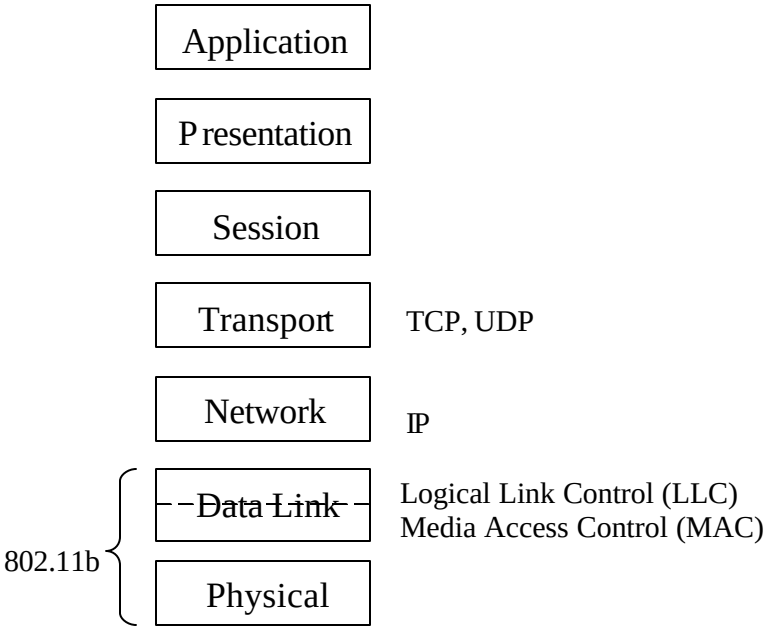


Figure 3. 802.11b and OSI Model

The original 802.11b wireless standard defines data rates of 1 Mbps and 2 Mbps via radio waves using FHSS or DSSS. In 1999 two higher speeds, 5.5 Mbps and 11 Mbps were added to the standard by the IEEE. To be able to reach these transfer rates the DSSS was selected as a physical layer technique of 802.11b since FHSS is regulated to the maximum transfer rate of 2 Mbps.

The DSSS techniques divide the 2.4 GHz band into 14 22 MHz channels. Adjacent channels overlap one another partially, with three of the 14 channels being completely non-overlapping. Data are sent across one these 22 MHz channels. [2]

As can be seen in figure 3, the data link layer consists of two sub-layers: Logical Link Control (LLC) and Media Access Control (MAC). The LLC is the same in 802.11b and other 802 LANs. This allows bridging from the wireless network to the wired. The MAC is different in WLANs. 802.11b uses carrier sense multiple access with collision avoidance (CSMA/CA) protocol, which attempts to avoid collisions by using explicit packet acknowledgement (ACK).

CSMA/CA works as follows: A station wishing to transmit senses the air, and, if no activity is detected, the station waits an additional, randomly selected period of time and then transmits if the medium is still free. If the packet is received intact, the receiving station issues an ACK frame that, once successfully received by the sender, completes the process. If the ACK frame is not detected by the sending station, either because the original data packet or the ACK was not received intact, a collision is assumed to have occurred and the data packet is retransmitted after waiting another random period of time. CSMA/CA provides a way to share access over the air. [2]

802.11b uses also a protocol called Request To Send/Clear To Send (RTS/CTS) at the MAC layer. A sending station transmits an RTS and waits for the AP to reply with a CTS. The CTS causes all stations in the network to delay their transmission, and this way allows the sending station to transmit and receive a packet acknowledgment without a change of collision with other transmission.

Two other features are also provided in the 802.11b MAC layer. These are CRC checksum and packet fragmentation. CRC checksum handles the error checking of each packet. This is different from Ethernet, where higher-level protocols such as TCP handle error checking.

Packet fragmentation allows large packets to be segmented into smaller units when sent over the air, which is useful in very congested environments or when interference is a factor, since large packets have a better chance of being corrupted. This technique reduces the need for retransmission in many cases and improves overall wireless network performance. [2]

4.3.2 802.11b Client Association

Association is the process of a WLAN client establishing the connection to an access point. When a client enters the coverage area of one or more access points, it chooses an AP to associate with based on the signal strength. The association is handled by the MAC layer of 802.11b. When the AP accepts the client, the client tunes itself to the same radio channel that the AP is using. The client surveys the radio channels periodically to check whether some other access point would provide a better signal strength for it. Once it finds an AP with a better signal strength it associates with this AP. This feature allows the user to move to the range of another AP while connected to the 802.11b network.

4.4 WLAN Security

4.4.1 Wired Equivalent Privacy

IEEE 802.11 standards use media access control and encryption mechanisms called wired equivalent privacy (WEP) to provide information security. An AP can have so called extended service set identification (ESSID) programmed to it. When a wireless client wants to associate with the AP it has to provide an AP with the ESSID to get associated with it. Also the AP can have an access control list of the allowed clients´ MAC addresses.

For data encryption, the standard provides for optional encryption using a 40-bit shared-key RC4 PRNG algorithm from RSA security [2]. This shared key is also known as the WEP-key. All data that are sent and received are encrypted using this key. WEP-key is also used for access control. When a client wants to associate with the AP, it is challenged with an encrypted challenge packet. The client must use its WEP-key to encrypt a correct response to the challenge if it wants to associate with the AP.

4.4.2 802.1X Authentication and Key Management

A major underlying problem with the existing 802.11 standard is that the WEP keys are cumbersome to change. If you do not update the WEP keys often, an unauthorized person with a sniffing tool, such as AirSnort or WEPcrack, can monitor your network for less than a day and decode the encrypted messages. [3]

The IEEE 802.1X standard can be used for effective authentication, access control and dynamic varying of the encryption keys. The 802.1X standard passes a protocol called extensible authentication protocol (EAP) over a wireless or wired LAN. EAP provides options for different authentication methods. Three parties are involved in the 802.11X authentication. The client that wants to connect to a wireless LAN is called a *supplicant*. For example Windows XP client can act as 802.1X supplicant because Windows XP ships with 802.1X. The server doing the authentication is called the *authentication server*. Usually the authentication server is a RADIUS server. The device between these two, an access point in case of WLAN, is called the *authenticator*.

Initial 802.1X communications begin with an unauthenticated supplicant (i.e., client device) attempting to connect with an authenticator (i.e., 802.11 access point). The access point responds by enabling a port for passing only EAP packets from the client to the authentication server located on the wired side of the access point. The access point blocks all other traffic, such as HTTP, DHCP, and POP3 packets, until the access point can verify the client's identity using an authentication server (e.g. RADIUS). Once authenticated, the access point opens the client's port for other types of traffic. [3]

802.1X makes it possible for the client to automatically change encryption keys periodically to minimize the possibility of eavesdroppers to crack the key that is in use. The key should be changed as often as every 5 to 10 minutes.

5. TCP/IP NETWORKING

In modern networking the applications need a way to carry data from one machine to another. Users of one network want to be able to access other data networks simultaneously without interfering one another. Most of the networking protocols today use a method called packet switching. A packet is a small chunk of data that is transferred from one machine to another across the network. A packet-switched network uses a single link to send packets of many users alternately and this way the users can share the link. Transmission Control Protocol/Internet Protocol networks, or TCP/IP networks are packet-switched networks.

5.1 The Internet Protocol

To be able to communicate with a host computer in any kind of physical network, e.g. Ethernet or point-to-point data link etc, you need Internet Protocol (IP). IP is independent of the hardware used.

The main benefit of IP is that it turns physically dissimilar networks into one apparently homogeneous network. This is called internetworking and the resulting "meta-network" is called an internet. [7]

5.1.1 IP Addressing

IP requires an addressing scheme that is independent of the hardware. In IP each host is assigned a unique 32-bit number that is called the IP address. IP address is usually written in a so called dotted decimal notation, which means one decimal number for each 8 bits (octet) of the address and the decimal numbers separated by dots. An IP address in dotted decimal notation can be for example 192.0.1.7.

An IP address can be viewed in two parts: the network part of the address identifies the network the machine is connected to and the rest of the address, the host part, identifies the particular machine on that network.

Routing on your machine uses the network part to decide whether the destination machine is on the same network as your own. If the network part of the destination address is the same as the network part of your own address, then the two machines are locally connected; otherwise they are not. [10]

The IP addresses are divided into five classes, A, B, C, D and E. Classes A, B and C are shown in Table 1.

Table 1. IP Address Classes A, B and C

Class	Address range	Network part	Number of networks	Host part	No. hosts/ network
A	0.0.0.0-127.255.255.255	1+7 bits	128	24 bits	16,777,216
B	128.0.0.0-191.255.255.255	2+14 bits	16,384	16 bits	65,536
C	192.0.0.0-223.255.255.255	3+21 bits	2,097,152	8 bits	256

Even though a class A-address has an 8-bit network part, one of the bits is always 0. Therefore, there are only 7 variable bits in the network part, which is why there are only 128 class A networks, not 256 as you might first expect from an 8-bit network part. Similarly, class B- and class C-addresses have 2 and 3 bits respectively taken up with identifying the class type. [10]

Classes D, E and F are in the range of 224.0.0.0 through 254.0.0.0 and they are reserved for special purposes, they do not specify any network. For example class D is used for IP multicasting, which is a service that allows material to be transmitted to many points on an internet at one time.

In the host part of an IP address, octets 0 and 255 are reserved. An address where all host part bits are 0 refers to the network. An address where all host part bits are 1 is a broadcast address, which refers to all hosts on the network simultaneously.

IP addresses 0.0.0.0 and 127.0.0.0 are called the default route and loopback address, respectively. These addresses are reserved for the special purposes. The default route specifies the route to the gateway machine on the network.

Network 127.0.0.0 is reserved for IP traffic local to your host. Usually, address 127.0.0.1 will be assigned to a special interface on your host, the loopback interface, which acts like a closed circuit. Any IP packet handed to this interface from TCP or UDP will be returned to them as if it had just arrived from some network. [7]

Certain IP address ranges from each IP address class are reserved to be used for private networks and are not routed on the Internet. These address ranges can be used by anyone for building their own intranet or a small network. These address ranges are listed in table 2.

Table 2. IP Address Ranges Reserved for Private Use

Class	Networks
A	10.0.0.0 – 10.255.255.255
B	172.16.0.0 – 172.31.0.0
C	192.168.0.0 – 192.168.255.0

5.1.2 IP Netmasking

IP networks can be subdivided into several smaller networks called subnets. Subnets can be generated by the network administrator, by using a 32-bit number called netmask.

A netmask combined with an IP address specifies a range of IP addresses of a particular size, by selectively extracting the bits of the IP address corresponding to the network part of the address. [10]

Let us say that we have an IP address 192.168.100.101 and a netmask 255.255.255.0. The netmask in binaries look like: 11111111.11111111.11111111.00000000. The 1-bits in the netmask tell us which part of the IP address we should look at and the 0-bits tell us which part we should ignore. In our case the first three octets of the netmask are 1-bits, so we look at the first three octets of the IP address and replace the last octet with 0-bits. This way we can see that the subnet in this case is 192.168.100.0.

5.1.3 Address Resolution

Each machine in a network, for example Ethernet, has its own unique IP address. Each machine also has its own unique Ethernet address for its Ethernet interface. This address is also called the medium access control (MAC) address and it is usually hardwired to the network interface card. Communication inside an Ethernet network uses the MAC addresses to specify each machine. A mechanism for mapping these two addresses together is needed to find a machine whose MAC address corresponds to a specific IP address. This mechanism is called the address resolution protocol (ARP).

When ARP wants to find the Ethernet address corresponding to a given IP address, it uses an Ethernet feature called broadcasting, in which a datagram (a packet) is addressed to all stations on the network simultaneously. The broadcast datagram sent by ARP contains a query for the IP address. Each receiving host compares this query to its own IP address and if it matches, returns an ARP reply to the inquiring host. The inquiring host can now extract the sender's Ethernet address from the reply. [7]

5.2 The Transmission Control Protocol

The internet protocol handles the routing of the packets from one machine to another. Once the packets arrive to a certain machine, IP does not know which packets are for different applications on this machine. This is handled by the transmission control protocol (TCP).

IP is an unreliable (or best-effort) protocol. It does its best to deliver the data but packets can get lost, e.g. if a router fails or a connection breaks, or if the underlying Ethernet network, which is also only best-effort, fails to deliver a packet. When IP drops a packet like this, the packet is lost, silently – no error message is generated anywhere. For many applications this is not acceptable, and that's where TCP comes in. TCP is a reliable network protocol: it goes to great

lengths to guarantee to deliver all your data, and in order you sent it, so I know that you receive exactly what I sent. [10]

TCP enables the data communication between applications and it works on top of the IP layer. Applications like the Web, e-mail, file transfer protocol and database systems use TCP. For two applications to be able to communicate, a TCP connection has to be established between them. The TCP connections are set up using a client/server model. In the client/server model, the server is running on an application, listening for requests of incoming connections from the clients. When a request is detected, the server establishes the connection with the client. This connection is a reliable data channel that the applications in both ends use for exchanging data. The server listens for client's requests, services them and sends responses back to the client using this connection. The TCP connection is released when either the client or server requests it. One machine and even one application can have multiple TCP connections simultaneously.

TCP connections are established between machines using ports. A port is an address to distinguish between different TCP connections on the same machine [10]. The TCP port is a number that tells the packet where to go inside a specific machine. When a client sends a TCP connection request to the server, it specifies the destination port number and the source port number to establish a connection between them. For example, a Web server and a client establish a connection between them using the servers port 80 because it is so called well-known port used by the HTTP application for HTTP requests. Also the source IP number and the destination IP number have to be specified for the connection. When multiple TCP connections are made between two machines, the source IP number, destination IP number and destination port number can be the same for all connections, but the source port number has to be different for all of them. TCP does not allow a client application to use the port number that is already in use by another client application.

5.3 Routing

When a host of a LAN wants to send a packet to an IP address that is not in the same network according to the local network IP address and the netmask, the packet has to be sent to a router, which then forwards the packet to the correct network.

The host compares the network part of the destination IP address of the packet to its own IP address to see if they exist on the same network. If the destination IP address of the packet is not in the same network as the host, it sends the packet to the router using the routers Ethernet address. The host can have the routers Ethernet address specified as its default gateway router or it uses ARP if needed. Usually a LAN PC knows only about its own LAN and the default gateway. The router does the same comparison of the IP addresses and if the destination IP address is not the routers own, it will forward the packet towards the packets destination. A router can pass packets only to hosts or routers that are on a network that is directly connected to it, this means that a router has to have a connection to at least two networks to be useful. The packet is forwarded by the hosts or routers of other networks until it reaches its destination.

A router is a general-purpose computer or specially built box with two or more network interfaces that forwards packets from one network to another [10]. For example a general-purpose computer with Linux or Windows NT can be set up as a router.

6. NETWORKING SERVICES

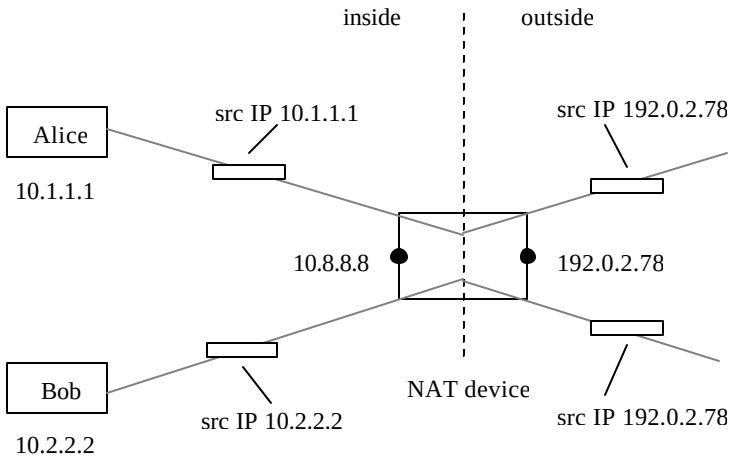
6.1 Network Address Translation

When a local area network is built, there are two possibilities for IP addressing. One possibility is to request, for example, class C static IP addresses for each computer in the LAN and use a router to connect the LAN to the Internet. This means that you have to pay for the IP addresses that are assigned to your LAN. This solution is good, for example, for companies who will probably want to have static IP addresses and are willing to pay for them.

Another possibility is to use the IP addresses that are reserved for private use, get an Internet connection from an Internet Service Provider (ISP) and share this connection to the Internet between the machines in a LAN. The private addresses can be used freely by anyone. The problem here is that the private range IP addresses cannot be routed to the Internet.

Network address translation (NAT) provides a solution to this problem. NAT lets you use IP addresses inside your LAN that are different to those visible from outside on the Internet. The IP address that your ISP gives you for the Internet connection may be dynamic. This means that every time you connect to the Internet, you are assigned a different IP address. This does not matter when you are using NAT between your private network and the Internet. NAT isolates the private addresses inside a LAN from the changing IP address that is visible to the outside world. This type of network address translation is also called IP masquerading. IP masquerade is the name given to one type of network address translation that allows all of the hosts on a private network to use the Internet at the price of a single IP address [7]. A computer running for example Linux operating system can be set up as a NAT device.

The NAT device and a router can be combined, which means that one machine handles the routing and the network address translation. Figure 4 illustrates how a NAT device works.



Source: Mansfield, Niall [10]

Figure 4. How NAT translates IP addresses

When a workstation inside the private network wants to establish a connection to a remote host outside its LAN, it routes the packets to the router with a NAT device, the masquerade router. The masquerade router identifies this connection request as requiring NAT services. The router accepts the packets and allocates a port number to use. The router substitutes its own outside IP address and port to the IP address and port number of the originating host and transmits the packets to the destination host. This way the destination host believes that it has received packets from the masquerade router. The local host also believes that it is communicating directly to the destination host. Only the masquerade router knows these two hosts are communicating with each other and it performs the address and port translations that are necessary for the communication between the hosts.

None of the hosts on the supported network behind the masquerade router are ever directly seen; consequently, you need only one valid and routable IP address to allow all the hosts to make network connections out onto the Internet. [7]

Although network address translation has many good features, it has also some drawbacks. None of the hosts inside the private network are accessible from the outside. Only the masquerade router can be accessed from the outside. Some network services do not work through the network address translation. For example, video or audio multicasting services, such as video conferencing usually do not work.

6.2 The Dynamic Host Configuration Protocol

Dynamic host configuration protocol (DHCP) allows a machine to obtain its IP address, netmask and other network configuration settings from a DHCP server at boot time. This is useful especially in large networks. Without DHCP, the network configuration settings of every workstation have to be done manually. When the settings are done manually, errors are always inevitable. Finding the errors that occur during the configuration can be very time consuming.

DHCP is also useful for example in WLAN networks where people connect to it using for example a laptop computer or a PDA. If you are visiting a site for the first time and you want to connect to a network, first you have to find the network administrator to find out the network settings, then you have to configure them to your machine. If the WLAN network has a DHCP server running, all you have to do is boot your computer and the network setting are configured automatically.

An IP address that a host receives via DHCP is called a dynamic IP address. The user has the same IP address as long as she is connected to the network or the computer is booted. DHCP is a client/server application. The DHCP client broadcasts a request for server and the server replies with requested network parameters using broadcast.

DHCP uses user datagram protocol (UDP) as its transport. UDP is a transport layer protocol like TCP. UDP sends data also in packets through ports like TCP. These packets are called datagrams. UDP is connectionless, which means that the client does not establish a connection to the server, it sends the data straight away. UDP is unreliable in the same way as IP, if packets are lost, UDP cannot detect it.

UDP is suitable for small data transfers, intermittent transfers, e.g. where a client contacts a server rarely, but where there may be a very large number of clients per server, and for multicast or broadcast applications [10]. For example domain name system (DNS) client and server use UDP to exchange queries and requests also.

The DHCP conversation between the client and the server works as in the following. The DHCP server listens on UDP well-known port 67. The DHCP client broadcasts a DHCPDISCOVER request, using IP address 0.0.0.0 as its source address, to UDP port 67.

The server sends a DHCPOFFER reply to the requesting client on UDP well-known port 68 using the client's MAC address. The server can usually unicast or broadcast the replies depending on the client. In DHCPOFFER the server already sends the network configuration values to the client. The IP address is chosen from a pool of addresses that the network administrator has configured for the server to use.

The client accepts the DHCPOFFER by broadcasting a DHCPREQUEST that contains the IP address of the accepted server. The IP address is specified because there may be many servers offering the DHCP service.

The server knows it has been selected to serve the client when it notices its IP address in the DHCPREQUEST and replies with a DHCPACK acknowledgment.

The client sets its network parameters to the values it received in the DHCPOFFER.

The conversation between the DHCP client and the server is illustrated in figure 5.

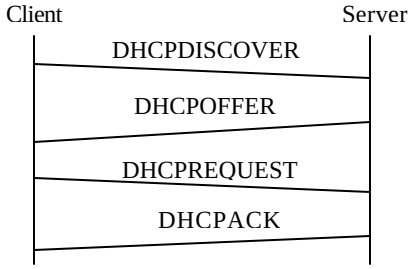


Figure 5. DHCP Conversation

The DHCP server serves the client as long as the client is connected to the network. The client contacts the server periodically according to a renewal time, and renews the service lease. This way the client can keep the same IP address as long as it needs. DHCP tries to allocate the same IP address to the same client even if the client shuts down and boots up again. After a client shuts down, the server can preserve a record of the IP address it had allocated to this client [10]. If the server runs out of IP addresses that are configured to it, it has to give a different address to the client.

6.3 The Domain Name System

The Domain Name System (DNS) lets you refer to machines by name instead of IP number. DNS converts machine names to IP numbers and vice versa. [10]

It is easier for humans to remember the names than the IP numbers. For example, it is easier to remember a name www.sjoki.uta.fi than an IP number 198.98.80.1.

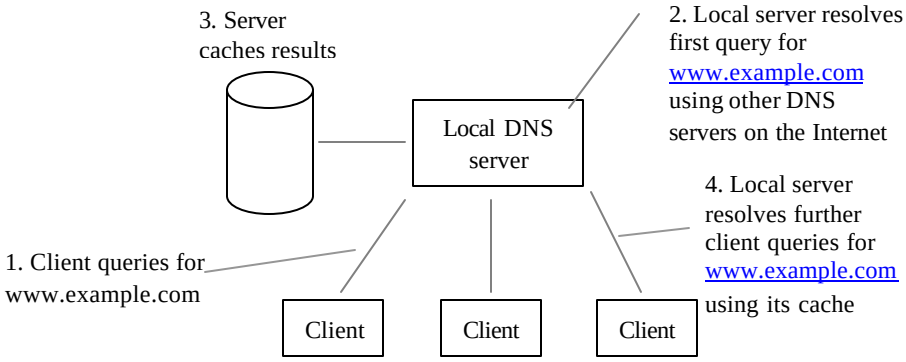
The DNS is a client/server facility. When the end-user specifies a host address as a name, the application sends the name to a DNS server and asks for the IP number that corresponds to the name. The DNS server sends back the IP address, which the application then uses. This process is called resolving the hostname or address. The end-user does not have to know anything about this process.

Basically, the DNS server holds a table of names and their corresponding IP numbers. It uses this table when a client requests an IP number corresponding to a hostname or vice versa. If the table does not have a record of a hostname and IP

number requested, it either returns an error message or forwards the request to another DNS server it has knowledge about. The DNS as a whole is a database that is distributed all over the Internet, there is no single machine with a database that holds a list of all hosts in the Internet.

When a LAN is built it is possible to use either a DNS server outside the LAN, for example your ISPs server, or use a private DNS server. When you run your own private DNS server, you can refer to the internal machines in your network by their hostnames that you give to them.

The private DNS server also improves the performance of your network. DNS servers cache the DNS queries. This means that when a DNS query for a specific host is made for the first time, the DNS server forwards the query to other DNS servers outside the LAN. When the correct query result is found the DNS server saves, or caches the result in local memory. Next time this same query is made, the server can retrieve the result from the cache instead of querying it from the outside. This improves the overall performance of the network. Figure 6 illustrates the caching of DNS results.



Source: Mansfield, Niall [10]

Figure 6. DNS Results Cached on Server

A DNS server that is used for caching in a private network, and is not used by anyone else than the local hosts, is called a caching-only server.

Each record that is stored in the cache of the DNS server has a time-to-live (TTL) field included. TTL is a value that corresponds to time in seconds that the record is kept in the server's cache. After the TTL has expired, the record is removed from the cache. The value of TTL can be for example 86,400, which corresponds to 24 hours.

6.4 The Proxy Server

Like the DNS server, also the proxy server can improve the performance of the LAN. The proxy server provides for caching of the requests, which helps to access data from local resources rather than fetching the data from the Web thus reducing access time and saving bandwidth. For HTTP, the proxy server is a server that your browser connects to for some or all of its transfers; your browser sends requests to the proxy server, and then the proxy server retrieves the pages from the real servers (which are called origin servers) [10].

The proxy server is an intermediate server between all clients on the LAN and the remote Web sites. The proxy is a server to the clients in the LAN and a client to the origin servers.

The proxy server can improve the performance of the LAN by caching the Web pages it retrieves. When the same Web page is requested the next time by any host on the LAN, the proxy server can send it to the client straight away without having to retrieve the page again from the origin server. This reduces the load on the Internet connection and the average response time to the Web page requests is improved.

A proxy server improves also security on the network since the Web clients do not have direct access to the Internet, the proxy makes the connections for them. The contact between the LAN and the Internet is reduced.

Proxy servers have also special access control features. Using these features the network administrator can, for example, specify the hosts who have access to the Internet, specify a particular group of sites to which the access is denied or even make the Internet access possible only during certain times during the day.

7. THE IMPLEMENTATION OF THE PROJECT

7.1 The Project Implementation Plan

A test network was set up for the project. This network consisted of an IEEE 802.11b standard WLAN network and a default gateway router for the network. The WLAN network was set up using infrastructure topology with one access point and a few computers. This network was a private network. IP addressing was done using the private range IP addresses.

To be able to provide a GPRS connection for the WLAN network, a router with a connection to the GPRS network and WLAN network was implemented. A normal desktop computer was used for this purpose. The router was running the Linux operating system. The Linux operating system was chosen because it provides such good networking capabilities. The Linux operating system is so called open source system, which means that it is practically free. All the network services that were implemented were either included in the Linux operating system distribution or they were downloaded from the Internet free of charge. Also because Linux operating system has gained such a wide popularity all over the world, good documentation for it can be found on the Internet and in the literature.

The GPRS connection was established using a GPRS PCMCIA (Personal Computer Memory Card Association) card, also called a PC card. GPRS PCMCIA cards are usually used in laptop computers or PDA devices. A GPRS PCMCIA card can be inserted in the PCMCIA slot of the computer and it makes GPRS/GSM connections possible for the user. The GPRS PCMCIA card needs also a Subscriber Identity Module (SIM) card of some mobile operator to work. The SIM card has to have the GPRS service enabled.

The Linux operating system provides the IP masquerade network address translation networking features. These features were used to provide the NAT for the test network. IP masquerade features are ready in the “core” of Linux, the

Linux kernel. IP masquerading had to be configured when the Linux kernel was configured. Also rules for the IP masquerading had to be configured so that it performed the address translation according to our wishes.

Dynamic Host Configuration Protocol service is also provided in the Linux operating system. The DHCP server was configured on the router computer. It provides the network configuration settings for the hosts that connect to the WLAN network.

The DNS server application was downloaded from the Internet. The most popular DNS server used in the Linux environment is the Berkeley Internet Name Domain (BIND) service. A recent version of BIND was downloaded from the Internet, installed and configured to work as a caching DNS server for the WLAN network.

The proxy server was also downloaded from the Internet. The *Squid* proxy server is a popular proxy server with a lot of features. A recent version of the Squid proxy server was downloaded, installed and configured to work as a proxy server for the WLAN network.

Tests were made after each of the features mentioned above were implemented. All of the features were implemented on the same computer: the router. Tests were made to make sure that the features were working as they were supposed to. After all of the networking features were implemented successfully the network was tested. The data transfer rate of the Internet connection was tested and traffic with multiple machines on the WLAN was generated to the outside using the GPRS connection to find out the usability, for example Web page response times.

7.2 The Description of the Implementation

7.2.1 The Desktop Personal Computer Used for Routing

The machine used for the implementation of the default gateway router, NAT, DHCP, DNS and proxy server was a normal desktop PC with 266 MHz Intel Pentium processor, 64 Megabytes of RAM memory and a 15 Gigabyte hard drive. It has the 3Com 3C900B-Combo Ethernet network interface card and Matrox

Graphics MGA G200 AGP video card installed. To be able to install a PCMCIA card for the GPRS services, a Nokia PCI-to-PCMCIA adapter was installed to the PCI slot of the computer.

Slackware 9.0.0 distribution of the Linux operating system was installed on the PC. Slackware 9.0.0 has the Linux kernel version 2.4.20 and the GNU c-compiler (gcc) version 3.2.2 in it. The operating system was installed from a CD-ROM and the installation was quite straightforward. In the kernel configuration the kernel was configured according to the hardware of the computer and also the PCMCIA support, the PPP support, the networking options, the IP netfilter options and the Wireless LAN support had to be configured because all of them were needed later.

7.2.2 Nokia D211 Multimode PCMCIA Card

Nokia D211 multimode PCMCIA card was chosen for the project. Nokia D211 can be used in GSM mode to provide the GPRS connection or in WLAN mode to connect to the WLAN network. In the project it was used in GSM mode together with DNA mobile operator's SIM card for the GPRS connection. The price of the Nokia D211 is about 450 euros. It has a device driver for Linux, which can be downloaded from www.forum.nokia.com. The Support Guide for the Linux Device Driver [12] that was needed for the configuration of the device driver is also available there.

Nokia D211 provides the maximum theoretical data transfer rates of 13.4 kilobits per second for uplink (upload) and 40.2 kilobits per second for downlink (download).

7.2.3 Installation of the Nokia D211 Device Driver on Linux

The Nokia D211 device driver requires a working PCMCIA support to work. In the newer 2.4 Linux kernel series the needed PCMCIA support is included in the kernel. It is hence also included in the 2.4.20 kernel that was used in this case. Also support for the /proc file system in the in the kernel is required.

The Linux device driver of the Nokia D211 was provided partly in the binary and partly in the source code form. The driver itself was in the source code form and a user space daemon (daemon is a process that runs on the background) that is used for the kernel module management was in the binary form.

Installing the device driver was quite straightforward and the instructions were provided on the Nokia's website [9]. The device driver was packed in two packages. The driver packages were copied to the /usr/src directory. Then the packages were unpacked with the command "`tar –zxvf package_name`". The device driver sources and binaries were now located in the /usr/src/d211 directory.

After unpacking the device driver package, it had to be compiled. First, a copy of config.mk.orig file was made into the /usr/src/d211 directory with a command "`cp config.mk.orig config.mk`". The config.mk had to be edited to match the system. Only the OS_RELEASE value had to be changed to 2.4.20 to match the kernel version used in the project. The config.mk file looked like in the following after editing:

```
LINUX = /usr/src/linux
OS_RELEASE = 2.4.20
ROOTDIR = /
SMAC = d211fw.bin
```

Next the device driver was compiled with a command "`make all`". The compilation was successful and it created nokia_cs.o file in the /usr/src/d211/src directory.

Since the device driver was installed for the first time, command "`make config_install`" was issued to install the default configuration. This has to be done only once, even if the device driver is compiled again for some reason.

The last command that had to be issued was "`make install`" which installed the device driver. It installed the kernel module and the user space utilities of the device driver. After this the PCMCIA system was shut down and restarted by

issuing commands "`/etc/rc.d/rc.pcmcia stop`" and "`/etc/rc.d/rc.pcmcia start`"

7.2.4 Starting the GPRS Connection

To start the GPRS connection, the Linux device driver had to be configured first. The configuration file that needed to be edited was /etc/pcmcia/nokia_cs.opts, which defines which functionality, GSM or WLAN is activated in the startup of the card. When the card is used in GSM/GPRS mode, nokia_cs.opts has to contain only one line, which is: `MODE = GSM`.

GSM connection with the card requires a working Point-to-Point Protocol (PPP) setup in the computer. This means that the Linux kernel had to be configured to support PPP. The PPP support was configured already during the installation of the Linux operating system. PPP is a protocol used to send datagrams, or packets across a serial connection. When a GPRS connection is established, the computer establishes a PPP connection between itself and the GPRS device. The packets are sent to the device using this connection. The GPRS device then sends the IP packets to the mobile operators network and they are delivered to for example the Internet from there.

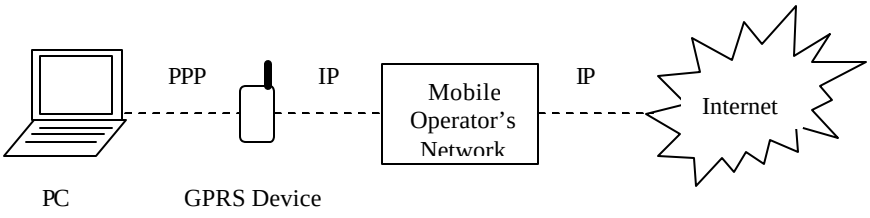


Figure 7. Communication between the computer and the GPRS device with a PPP connection

The first problem with the Nokia D211 Linux device driver was noticed right in the beginning. When the D211 PCMCIA card was inserted to the PCI-to-PCMCIA adapter for the first time, nothing happened. If it had been working normally, the operating system should have created a system log entry to the /var/adm/messages file and should have given two high beeps if it had successfully identified and configured the device. In Linux there is a PCMCIA

card daemon called cardmgr, which monitors the PCMCIA sockets for devices that have been added or removed. When the card was inserted to the adapter, the cardmgr daemon was “killed” i.e. terminated from the background.

The answer to this problem was found from the www.forum.nokia.com discussion forum for the multimode cards. The Linux device driver has been tested only on the new Red Hat Linux distributions. The device driver creates a script called nokia_cs to the /etc/pcmcia directory when it is installed. This script has a line for starting the nokia_cs daemon:

```
#start nokia_cs daemon
initlog -c "$NOKIA_CS" &
```

The first line is just a comment and the second line starts the daemon. The second line had to be changed to:

```
#start nokia_cs daemon
"$NOKIA_CS" &
```

This worked for the Slackware 9.0.0 distribution of Linux that was used in the project. Now when the card was inserted again, the cardmgr daemon successfully identified and configured the device, gave two high beeps and created a system log entry to the /var/adm/messages file. Also the nokia_cs.o module was loaded to the kernel. By issuing a command “`lsmod`”, all the loaded modules were listed:

```
nokia_cs                82320  2
ds                      6792   4 [nokia_cs]
i82365                  34912  2
pcmcia_core             38880  0 [nokia_cs ds i82365]
```

The device driver of the D211 creates a serial port /dev/ttyNC0. This serial port is used together with PPP for the GSM/GPRS connections.

The Linux device driver of D211 has a simple user interface called `nokia_ctl` for Linux to control the radio card. To be able to use the GPRS, the following commands have to be issued:

```
nokia_ctl enable_gsm
```

- This command asks for the PIN code of the SIM card and enables GSM.

```
nokia_ctl gsm enableGSMradio
```

- This command enables GSM radio access and it is necessary if GSM data connection is desired.

The second problem was discovered at this point. When the first command for enabling GSM was issued, the nokia_ctl user interface waited for a while and then responded with an error message: `TPC Timeout`. It should have responded `Command Succeeded`, if it had been working properly.

The problem proved to be the timing parameters of the PCI-to-PCMCIA adapter. The PCMCIA needs a socket driver called i82365 to work. A script rc.pcmcia for starting the socket driver is in directory /etc/rc.d. In this directory it is possible to define also PCMCIA socket driver timing parameters. When a PCI-to-PCMCIA adapter is used, these parameters have to be defined. The PCI-to-PCMCIA adapter started working correctly when the following PCMCIA socket driver timing parameters were defined:

```
# Put socket driver timing parameters here
PCIC_OPTS="irq_mode=1 pci_csc=1 poll_interval=100
wakeup=1"
```

After defining the timing parameters of the socket driver, when the `nokia_ctl enable_gsm` and `nokia_ctl gsm enableGSMradio` commands were issued, the respond was `Command Succeeded`. The D211 card and the PCI-to-PCMCIA adapter were working correctly now.

The PPP connection with the Nokia D211 card was established using the PPP daemon, pppd in Linux. The GPRS connection with the card was established using the command:

```
pppd /dev/ttyNC0 115200 debug connect 'chat -v -t 30 ""
"AT" OK-AT-OK-AT-OK ATZ OK "AT
+CGDCONT=1,\"IP\",\"Internet\""
"OK" "atd*99#" "CONNECT"' modem crtscts defaultroute
noipdefault
```

`/dev/ttyNC0` is the serial device that the D211 device driver created and that is used with the PPP.

`115200` is the transfer speed used on the serial port, 115200 bits per second.

`connect` is a command line option of pppd and it is used for giving the command to be executed to the pppd.

```
'chat -v -t 30 "" "AT" OK-AT-OK-AT-OK ATZ OK "AT
+CGDCONT=1,\"IP\",\"Internet\"" "OK" "atd*99#"
"CONNECT"'
```

Chat is a simple program of pppd, which is used to automate the login sequences. The above is a chat dialup script that is used for establishing the DNA mobile operator's GPRS connection.

`modem` is a pppd keyword that makes it perform modem-specific actions on the serial device, like disconnecting the line before and after the call.

`crtscts` option turns on hardware handshake on the serial port. This option has to be used with serial port transfer speeds above 9600 bits per second.

`defaultroute` option causes any IP packet destined to a non-local host to be routed to the other networks using the PPP interface as a default route. pppd creates a PPP interface called ppp0 when the connection is established and assigns it to be a default route. This one option makes the computer work as a default gateway router for us.

`noipdefault` option is needed because we are connecting in to a server that will assign us an IP address, we do not want the pppd to attempt to negotiate one for itself.

Once the GPRS connection was up, it was tested first using a "`ping sjoki.uta.fi`" command and then using a Web browser to connect to a Web page in the Internet. It was working fine otherwise but the connection was broken

every two minutes. The GPRS connection was up for exactly two minutes and then came down. The solution to this problem was in the /etc/ppp/options file. In this file there are two lines for PPP echo requests, `lcp-echo-interval 30` and `lcp-echo-failure 4`. These echo requests had to be turned off and it was done by commenting out the two lines as in the following:

```
#lcp-echo-interval 30
#lcp-echo-failure 4
```

This solved the problem and the GPRS connection was not broken every two minutes after this.

The file rc.local in the /etc/rc.d directory can be used to initialize local systems when the computer is booted. A shell script called startD211 was created under the /usr/local/bin directory for starting the GPRS connection when the system is booted. The rc.local and the startD211-script can be found in the Appendices 1 and 2. The following was added to the /etc/rc.d/rc.local to run the startD211 script at the boot time:

```
# Script for initializing Nokia D211 and GPRS connection
sh /usr/local/bin/startD211
```

It is not unusual that the GPRS connection is terminated occasionally. A shell script called ppp-up was created under the /usr/local/bin directory for "monitoring" the GPRS connection. If the connection is broken, the script will start it again. The script can be found in the Appendix 3. The following was added to the /etc/rc.d/rc.local to run the ppp-up script on the background at the boot time:

```
# Script for GPRS-connection "monitoring" and re-initialization
sh /usr/local/bin/ppp-up &
```

7.2.5 Setting Up the Test Network

The test network was set up using the following components:

- Two desktop computers with Ethernet network interface cards; the router desktop computer (Linux operating system) and one host desktop computer (Linux operating system)
- One host laptop computer (Windows 2000 operating system) with Orinoco 802.11 a/b ComboCard WLAN PCMCIA card
- One Lucent Technologies WP-II E WLAN access point
- One Allied Telesyn FS708 switching hub
- Three Ethernet cables with RJ-45 connectors

The router, host desktop PC and the WLAN access point were connected through the switching hub with the Ethernet cables. The laptop PC was connected to the network through the WLAN. The network is illustrated in figure 8.

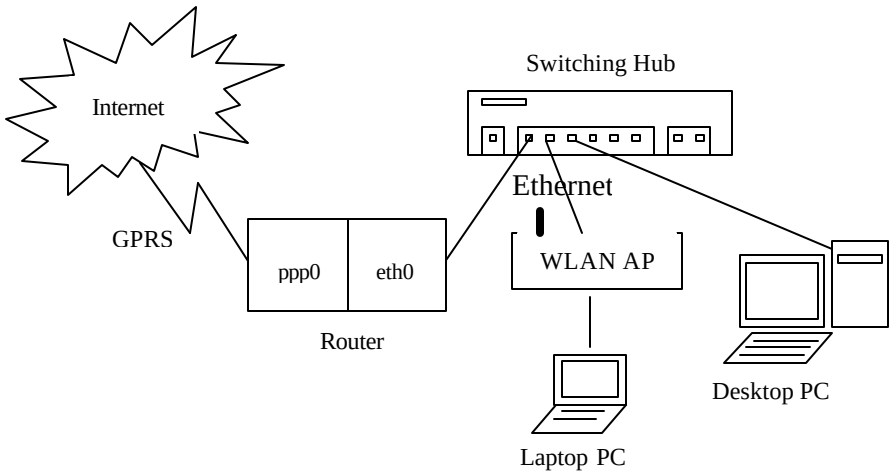


Figure 8. The Test Network

The network was set up as a private network and for this reason the IP addressing was done using IP addresses that are from the private range. The IP addresses were chosen from the private range of Class A addresses. The Ethernet interface, eth0, of the router was given an address 10.101.1.1 with a broadcast address 10.101.1.255 and netmask 255.255.255.0. This was done using the following command:

```
ifconfig eth0 10.101.1.1 broadcast 10.101.1.255 netmask
255.255.255
```

By issuing a command "`route`", the routing table of the machine can be viewed. It had now the eth0 interface with the address 10.101.1.1 and the ppp0 interface (created when the GPRS connection was established) with the IP address 10.6.6.6 as a default gateway:

```
Kernel IP routing table
Destination Gateway   Genmask         Flags Metric Ref  Use Iface
10.6.6.6    *         255.255.255.255 UH    0      0    0   ppp0
10.101.1.1  *         255.255.255.0   U     0      0    0   eth0
loopback    *         255.0.0.0       U     0      0    0   lo
default     10.6.6.6  0.0.0.0         UG    0      0    0   ppp0
```

The file rc.inet1 under the /etc/rc.d directory is used for starting up the base networking system when the computer is booted. The following was edited in rc.inet1 to configure the Ethernet interface at the boot time:

```
# Edit these values to set up your first Ethernet card (eth0):
IPADDR="10.101.1.1"  # REPLACE with YOUR IP address!
NETMASK="255.255.255.0" # REPLACE with YOUR netmask!
```

The IP addressing of the other machines in the network was done using the DHCP server after it was configured.

7.2.6 Configuration of the Network Address Translation

The Linux kernels 2.4.0 and later versions provide a networking option called netfilter, which is used for network packet filtering. The netfilter has a so called iptables utility included in it. The iptables utility is a tool that is used to build the IP firewall and IP masquerade NAT rules, i.e. it is used to configure the netfilter filtering rules. The instructions for configuring IP masquerade NAT using iptables were taken from Kirch and Dawson [7].

The iptables utility is used with the "`iptables`" command. It has two tables of rules called `filter` and `nat`. The filter-rules are used for the IP firewalls and nat-rules are used for the IP masquerading. There are three chains that are used for the `nat` table; `PREROUTING`, `POSTROUTING` and `OUTPUT`. The general

syntax of most iptables commands is: `iptables command rule-specification extensions.`

In our test network the IP masquerading is enabled using the command:

```
iptables –t nat –A POSTROUTING –o ppp0 –j MASQUERADE
```

`-t nat` option is used to specify that the `nat` table is used.

`-A POSTROUTING` specifies that the `POSTROUTING` chain is appended

`–o ppp0` specifies the interface on which the packet is to be transmitted

`–j MASQUERADE` indicates that the packets that match this rule specification should be masqueraded.

After the IP masquerading was enabled on the default gateway router of the network using the above command, whenever any of the hosts tried to connect to a host outside the private network, they were masqueraded by the router.

The effect of IP masquerading was tested in the following way: The desktop host computer, that was also running Linux operating system, was given the IP address 10.101.1.2 with broadcast address 10.101.1.255 and netmask 255.255.255.0. After this a remote host shell.puv.fi was contacted using SSH program.

The SSH-connection to shell.puv.fi was of course established through the router's GPRS connection. After connecting to shell.puv.fi the command "`finger`" was issued to list all the users connected to it. The following information was issued about user t0103021:

```
Login    Name             Tty    Idle  Login Time

t0103021 Markku Hautamäki pts/10       Apr 10 09:02
(anneal135.gprs.suomen2g.fi)
```

`anneal135.gprs.suomen2g.fi` is the domain name of the DNA mobile operator's server that was providing the GPRS connection. Since this connection

was possible to establish from a local host through the router's GPRS connection, the routing and IP masquerading had to be working as expected.

The iptables command for the IP masquerade was added to the /etc/rc.d/rc.local file to run it at the boot time.

7.2.7 Configuration of the BIND Domain Name System

The source code of the BIND domain name system version 9.2.2 was downloaded from ftp://ftp.funet.fi /pub/mirrors/ftp.isc.org/isc/bind9/ to the /usr/src directory. The instructions for configuration of BIND 9.2.2 were taken from Langfeldt et al. [8]. The package was called `bind-9.2.2.tar.gz`. The package was archived and packed so it had to be extracted and unpacked with the command "`tar –zxvf bind-9.2.2.tar.gz`"

Before installing the BIND 9.2.2 a command "`export CFLAGS = -O0`" had to be issued. The CFLAGS environment variable affects the compilation of BIND. `CFLAGS = -O0` was used to disable the optimizer of the gcc 3.2.2 c-compiler. The BIND 9.2.2 was installed without this command for the first time and the `named` server program of BIND did not work correctly. Every time when a DNS query was forwarded to a DNS server outside the LAN, the `named` server program was terminated.

A command to configure the installation options was issued next: "`./configure --prefix= /usr –-sysconfdir= /etc –-localstatedir= /var`". The `--prefix= /usr` option sets the installation into /usr directory, `–-sysconfdir= /etc` option sets the configuration files into /etc directory and `--localstatedir= /var` sets the default parent directory of "run/named.pid" to /var.

The source code of BIND 9.2.2 was compiled with command "`make`" and the compilation was successful.

BIND 9.2.2 was installed with a command "`make install`" and the installation was also successful.

A directory called nameserver was created under /etc directory using the command "`mkdir nameserver`". The ownership of the nameserver directory was changed to group `daemon` and user `daemon` using the command "`chown daemon.daemon nameserver/`. The user `daemon` was given write permission, group `daemon` was given read, write and execute permissions, and others were given read and execute permissions using the command "`chmod 2755 nameserver/`". With `chmod` command, the permissions can be specified using a three-digit octal number: 4 = Read, 2 = Write and 1 = Execute. The ownership and permissions were changed to protect the configuration files of BIND.

The first file that was created under the /etc/nameserver was named.conf. named.conf is used by the named server program and it tells the name of the directory where to find the data files that map domain names to addresses, the IP addresses of the forwarding DNS servers outside the local network and the data files that are used for different zones. The contents of the named.conf file are in the following:

```
options {
      directory "/etc/nameserver";

      forwarders {

      192.98.80.1;
      //sjoki.uta.fi dns
      };

};

zone "." {
      type hint;
      file "root.cache";
};

zone "0.0.127.in-addr.arpa" {
      type master;
      file "rev-localhost";
      notify no;
};

//
zone "operator.fi" {
      type master;
      file "operator.fi";
      notify no;
```

```
};

zone "1.101.10.in-addr.arpa" {
      type master;
      file "rev-10.101.1";
notify no;
};
```

The first zone (".") points to the file root.cache. The name server needs root.cache data file to know where the name servers are for the root domain.

The second zone ("0.0.127.in-addr.arpa") points to the file rev-localhost. This data file is used by the name server for reverse mapping, i.e. address-to-name mapping of the local loopback network 127.0.0.0 that the hosts use to direct traffic to themselves.

The third zone ("operator.fi") points to the file operator.fi. This data file is used by the name server for name-to-address mapping of the hosts of the private network. operator.fi was chosen to be the domain name of the name server. The name could be anything because the name server is serving only our private network and it is visible only to the hosts inside it. In the operator.fi file the name of the local name server was defined. The name-to-address mappings of the name server (operator.fi with IP address 10.101.1.1), the local loopback host (localhost.operator.fi with IP address 127.0.0.1) and the hosts were also defined. Name-to-address mappings of 20 hosts were defined as: dh101.operator.fi with IP address 10.101.1.101, dh102.operator.fi with IP address 10.101.1.102 and so forth.

The fourth zone ("1.101.10.in-addr.arpa") points to the file rev-10.101.1. This data file is used by the name server for reverse mapping (address-to-name mapping) of the hosts on the private network.

The data files root.cache, rev-localhost, operator.fi and rev-10.101.1 were created under /etc/nameserver directory. The data files can be found in the Appendices 4, 5, 6 and 7.

The named server program looks for the named.conf file under the /etc directory. For this reason a symbolic link from /etc/nameserver directory was created for named.conf under /etc. The symbolic link was created by issuing a command "`ln –s nameserver/named.conf`" under /etc directory.

DNS uses a resolver program on the client side. It is responsible for translating a program's request for host information into a query to a name server, and for translating the response into an answer for the program [1].

The resolver uses a resolv.conf file under /etc directory to look up the address and the domain name of the name server that is used. The local name server was added to the resolv.conf file:

```
nameserver 10.101.1.1
search operator.fi
```

The DNS was now configured. The named server program was started using a command "`named –u daemon &`". This command started the named as a daemon - a process in the background. A command "`ps axugwww |grep named`" can be used to verify that the named daemon is running.

The local DNS was tested using the nslookup command. When a command "`nslookup sjoki.uta.fi`" was issued, the response was as in the following.

```
Note:  nslookup is deprecated and may be removed from future
releases. Consider using the `dig' or `host' programs instead.
Run nslookup with the `-sil[ent]' option to prevent this message
from appearing.
Server:         10.101.1.1
Address:        10.101.1.1#53

Non-authoritative answer:
Name:   sjoki.uta.fi
Address: 192.98.80.1
```

The IP address 192.98.80.1 corresponding to sjoki.uta.fi was found using the local DNS server that was configured. The local DNS was working. The local IP address of operator.fi was found also using a command "`nslookup operator.fi`". The response was as in the following.

```
Note:  nslookup is deprecated and may be removed from future
releases. Consider using the `dig' or `host' programs instead.
Run nslookup with the `-sil[ent]' option to prevent this message
from appearing.
Server:         10.101.1.1
Address:        10.101.1.1#53

Name:   operator.fi
Address: 10.101.1.1
```

The rc.inet2 file in the /etc/rc.d directory is used to start networking services and daemons when the computer is booted. This file can be used to start the named daemon during the system boot. To enable this, the following lines were uncommented in the /etc/rc.d/rc.inet2 file:

```
# Start the NAMED/BIND name server:
   if [ -f ${NET}/named ]; then
      echo -n " named"
      ${NET}/named -u daemon &
   fi
```

A shell script called named-up was created under the /usr/local/bin directory for "monitoring" the named server program. If the program is terminated, the script will start it again. The script can be found in the Appendix 8. The following was added to the /etc/rc.d/rc.local to run the named-up script on the background at the boot time:

```
#Script for monitoring and re-starting named
sh /usr/local/bin/named-up &
```

7.2.8 Configuration of the Dynamic Host Configuration Protocol Server

The DHCP server program is called dhcpd in Linux. The dhcpd was included in the Linux 2. 4.20 kernel but it had to be configured to be able to use it. The dhcpd uses a dhcpd.conf file in the /etc directory for looking up its configuration settings. The dhcpd.conf was configured to match our network settings:

```
# dhcpd.conf
# Configuration file for ISC dhcpd (see 'man dhcpd.conf')
#

ddns-update-style none;

subnet 10.101.1.0 netmask 255.255.255.0 {
  range 10.101.1.101 10.101.1.120;
  default-lease-time  600;
  max-lease-time 7200;
```

```
  option subnet-mask 255.255.255.0;
  option broadcast-address 10.101.1.255;
  option routers 10.101.1.1;
  option domain-name-servers 10.101.1.1;
  option domain-name "operator.fi";
}
#
```

The dhcpd uses a pool of IP addresses that it can assign to the clients. This pool of addresses was configured by "`range 10.101.1.101 10.101.1.120`", i.e. 20 private range addresses were assigned for the dhcpd to use. The subnet-mask and broadcast addresses were configured according to our private network IP addressing, the IP address of the router and the domain name server, 10.101.1.1, was configured and the domain name of our network was configured to operator.fi.

An iptables command was needed to make the DHCP work with the IP masquerade NAT. The following command was issued:

```
iptables –t nat –A POSTROUTING –o eth0 –j ACCEPT
```

The iptables command was added to the /etc/rc.d/rc.local to run it at the boot time.

The dhcpd was started to the background with a command "`dhcpd &`". The DHCP was then tested using the WLAN connection of the network. The WLAN access point and the laptop computer with the WLAN PCMCIA card were started up. The DHCP requests and acknowledgments could be viewed from the router computer by issuing a command "`tail –f /var/adm/messages`". The DHCP requests and acknowledgments of the access point (WirlabGOLD) and the laptop computer (LIFEIT-M2QUOQU) were as in the following.

```
Apr 11 10:27:59 slack dhcpd: DHCPREQUEST for 10.101.1.118 from
00:02:2d:15:49:8b (WirlabGOLD) via eth0

Apr 11 10:27:59 slack dhcpd: DHCPACK on 10.101.1.118 to
00:02:2d:15:49:8b (WirlabGOLD) via eth0

Apr 11 10:31:34 slack dhcpd: DHCPREQUEST for 10.101.1.117 from
00:20:a6:47:bd:55 (LIFEIT-M2QUYOQU) via eth0

Apr 11 10:31:34 slack dhcpd: DHCPACK on 10.101.1.117 to
00:20:a6:47:bd:55 (LIFEIT-M2QUYOQU) via eth0
```

The WLAN access point was assigned the IP address 10.101.1.118 and the laptop computer the IP address 10.101.1.117. The DHCP was working well.

The following was added to the /etc/rc.d/rc.local file for initializing the dhcpd during the system boot.

```
echo "Initializing dhcpd.."
dhcpd &
```

The connection to the outside of our private network using the WLAN connection to the router and the GPRS connection to the Internet was also tested. The ping-test to the sjoki.uta.fi server, "`ping –w 2000 sjoki.uta.fi`" was done on the laptop computer:

```
Pinging sjoki.uta.fi [198.98.80.1] with 32 bytes of data:

Reply from 198.98.80.1: bytes=32 time=1762ms TTL=239
Reply from 198.98.80.1: bytes=32 time=1052ms TTL=239
Reply from 198.98.80.1: bytes=32 time=1392ms TTL=239
Reply from 198.98.80.1: bytes=32 time=1422ms TTL=239

Ping statistic for 198.98.80.1:
      Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
      Minimum = 1052ms, Maximum = 1762ms, Average = 1407ms
```

The ping-test was successful, which meant that the routing from the WLAN to the Internet through the GPRS connection was working. The routing was a tested successfully also by contacting several Web sites using the laptop computers Internet browser.

7.2.9 Configuration of the Squid Proxy Server

The source code of the Squid proxy server version 2.4.STABLE7 was downloaded from ftp://ftp.clinet.fi/pub/unix/squid/squid_2/STABLE to the /usr/src directory. The name of the downloaded package was `squid-2.4.STABLE7-src.tar.gz`. The package was extracted and unpacked with a command "`tar –xvzf squid-2.4.STABLE-src.tar.gz`". The instructions for configuring the Squid proxy server were taken from Kiracofe [12].

To configure the installation options a command "`./configure --enable –linux-netfilter`" was issued. The Linux netfilter had to be enabled because the proxy server would be used on the same computer with the IP masquerade NAT.

The source code of the Squid proxy server was compiled with a command "`make`" and the compilation was successful. It was then installed successfully with a command "`make all`". The proxy server was installed under /usr/local directory by default.

First a new user group called squid and a new user called squid were created into the system with commands "`groupadd squid`" and "`adduser squid`". These were created because Squid has to be run as a user squid under the user group squid.

The Squid proxy server was configured by modifying the squid.conf file, which was installed under /usr/local/squid/etc directory. The squid.conf has a lot of parameters that can be modified, but for our purposes only a few of them had to be modified.

In squid.conf file, under ADMINISTRATIVE PARAMETERS header, two parameters were modified: `cache_effective_user squid` and `cache_effective_group squid`.

Under ACCESS CONTROLS header after a line: `# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS,` two lines were inserted:

```
acl network1 10.101.1.0/255.255.255.0
http_access allow network1
```

These two lines allow the proxy server to be accessed only by our private network 10.101.1.0/255.255.255.0. These modifications are enough to configure the Squid proxy server.

The root user, usually the network administrator, was set as the owner of the /usr/local/squid directory with a command "`chown root:root /usr/local/squid`". The Root user was given all permissions and others were given read and execute permissions to the same directory with a command "`chmod 755 /usr/local/squid`". These actions were done to secure the configuration and log files of the Squid.

The Squid proxy server needs hard drive space for caching. A Command "`/usr/src/squid-2.4.STABLE7/src/squid –z &`" was issued to reserve space from the hard drive for caching.

Before starting the Squid, an iptables command had to be issued:

```
iptables –t nat –A PREROUTING –i eth0 –p tcp --dport 80
–j DNAT –-to 10.101.1.1:3128
```

This command forces all the TCP traffic inside our network with the destination port 80, to the port 3128 of the router machine. This means that all of the HTTP traffic is forced to the Squid proxy server because Squid uses the port 3128 on the router. The iptables command was added to the rc.local file under the /etc/rc.d directory for issuing the command at the boot time.

The Squid proxy server was started by running a shell script RunCache in the /usr/local/squid/bin directory: "`sh RunCache`". The proxy server was tested using the Internet Explorer browser on the WLAN-laptop computer to contact www.wirlab.net. The squid_access.log file in the /usr/local/squid/logs directory records the pages that are contacted if the Squid is working. The following is a part of the log that was recorded when www.wirlab.net was contacted:

```
1050042388.699   6942 10.101.1.1 TCP_MISS/200 5785 GET
http://www.wirlab.net/ - DIRECT/192.98.80.4 text/html
1050042388.699   1251 10.101.1.1 TCP_HIT/200 5796 GET
http://www.wirlab.net/ - N ONE/- text/html...
```

This proved that the Squid was working. "`sh Runcache &`" was added to the /etc/rc.d/rc.local file to start Squid when the system is booted.

A shell script called squid-up was created under the /usr/local/bin directory for “monitoring” the Squid proxy server. If the program is terminated, the script will start it again. The script can be found in the Appendix 9. The following was added to the /etc/rc.d/rc.local to run the named-up script on the background at the boot time:

```
#Script for monitoring and re-starting Squid proxy server
sh /usr/local/bin/squid-up &
```

8. TESTING OF THE IMPLEMENTED NETWORK

8.1 The Data Transfer Rates

The data transfer rates were tested by uploading data to another network outside the test network and downloading data from the same network. The GPRS gateway limits the uploading rate to a theoretical maximum of 13.4 kilobits per second and the downloading rate to a theoretical maximum of 40.2 kilobits per second. The tests were performed using the desktop computer (with Linux operating system) that was connected to the test network. The data was transferred between the test network and shell.puv.fi using sftp file transfer protocol. The size of the file that was transferred was 1,081,623 bytes (1.0315 Mbytes) and it was called `pcmcia-cs-3.1.16.tar.gz`. The data transfer rates were calculated using the formula $\frac{\left(\frac{1081623\ bytes}{The\ transfertime\ in\ seconds} * 8\right)}{1000}$, to get the transfer rate in kilobits per second.

8.1.1 The Uploading Data Transfer Rate

Five uploading tests were performed during different times of the day between 8 a.m. and 4 p.m. on the 10th of April 2003. The test results are in the following table.

Table 3. Results of Uploading the Data

Time of the Day	Transfer Rate
8.30 a.m.	9.69 kilobits/s
10 a.m.	9.71 kilobits/s
12.30 p.m.	9.73 kilobits/s
2 p.m.	9.51 kilobits/s
3.30 p.m.	9.60 kilobits/s

The average uploading data transfer rate was 9.65 kilobits per second.

8.1.2 The Downloading Data Transfer Rate

Five downloading tests were performed during different times of the day between 8 a.m. and 4 p.m. on the 10th of April 2003. The test results are in the following table.

Table 4. Results of Downloading the Data

Time of the Day	Transfer Rate
9 a.m.	30.58 kilobits/s
10.30 a.m.	16.39 kilobits/s
1 p.m.	21.80 kilobits/s
2.30 p.m.	20.42 kilobits/s
4 p.m.	22.94 kilobits/s

The average downloading data transfer rate was 22.43 kilobits per second.

8.1.3 The Analysis of the Results

The results of the uploading and downloading tests show that the GPRS bandwidth can have significant variations over time. The reason for this may be the signal strength fluctuations and the traffic generated by other users in the operators GSM network. The average uploading and downloading data transfer rates show that in reality the transfer rates are lower than the maximum theoretical data transfer rates.

8.2 The Performance of the Network

The performance of the network was tested using two laptop computers with WLAN PCMCIA cards connected to the WLAN AP and the desktop computer that was connected to the test network with the Ethernet cable. The laptop computers were used for Web browsing and at the same time the desktop computer was downloading a file using file transfer protocol. The test was made to see how the GPRS connection in the router handles the traffic generated by multiple users and how the Web browsing is affected.

A user on the other laptop was browsing the Web and the desktop computer was downloading data when a Web site with a lot of dynamic content and images,

www.cnn.com, was accessed from the other laptop. The loading of the www.cnn.com took 2 minutes and 42 seconds. After this, another Web site was accessed and then again www.cnn.com. This time www.cnn.com was loaded from the cache of the proxy server because it had been loaded before. The loading of it took now only a few seconds.

The download transfer rate of the desktop computer during the test was measured. The transferred file was `pcmcia-cs-3.1.16.tar.gz` (1,081623 bytes) and it took 6 minutes 57 seconds to download. The download transfer rate was 20.75 kilobits per second.

According to this test the GPRS can be used as a gateway for a network of few computers. For example the loading of the web pages is of course quite slow, but the proxy server improves the overall performance quite significantly. The most important result of this test, from the point of view of the project, was that it is possible to use GPRS as a wireless core network for WLAN networks.

9. CONCLUSIONS

The aims of this final thesis project were attained. The router for routing the traffic from the wireless local area network to the GPRS network was implemented and the all the planned networking services were included in the implementation. The tests performed during the project prove that GPRS can be used as a wireless core network for wireless local area networks. Although the bandwidth of the GPRS is limited, the overall performance of the system can be improved. An example of this is the use of proxy server and the local DNS server that were implemented during the project. The DHCP server that was implemented to the system makes the configuration of the network settings easier for the users and the network administrator. This is an important feature because it makes the construction of the WLAN network easy and users can easily access the network.

The cost of the router is low. The GPRS PCMCIA card and the computer together cost around 1000 euros, depending on the computer, and the software is basically free since the Linux operating system is used.

The results of this project can be used for several purposes. Wireless local area network can be set up inside a moving vehicle, a bus for instance, and provide the Internet access for the users or WLAN can be used to gather data in a remote location and use the GPRS network to upload the data to the people who need it. A WLAN network with the GPRS gateway provides a mobile wireless local area network that can be set up easily in a short time. The cable data connection to the outside networks, such as the Internet, is omitted because the GPRS provides the data connection.

10. SUMMARY

This final thesis project was made to investigate the possibilities of using GPRS as a wireless core network for wireless local area networks. The WLAN network provides a wireless connection to the network but it covers only a limited area. The networks outside the WLAN can be accessed wirelessly if the access to them is provided by the GPRS. The aim of the project was to implement a router that would provide a GPRS default gateway for a WLAN network. Network address translation, dynamic host configuration protocol server, domain name system server and a proxy server were also to be implemented on the same router computer.

All of the above elements were implemented and a test network was built to test them. The implemented system was tested. The system worked as planned and the GPRS worked as a default gateway for the WLAN network. The GPRS bandwidth creates some limitations to the implemented network but it can provide for example the Internet access for a small WLAN network.

REFERENCES

[1] Albitz, Paul and Liu, Cricket 1992. DNS and BIND. 1. edition. Sebastobol, CA, The United States of America. O'Reilly and Associates.

[2] Garg, Vijay K 2002. Wireless Network Evolution : 2G to 3G. 1. edition. New Jersey, The United States of America. Prentice Hall.

[3] Geier, Jim. 802.1X Offers Authentication and Key Management [online]. Available on the Internet:
<URL:http://www.80211-planet.com/tutorials/article.php/1041171>.

[4] GPRS White Paper. Available on the Internet:
<URL:http://www.cisco.com/warp/public/cc/so/neso/gprs/gpr_wp.htm>.

[5] Hioki, Warren 2001. Telecommunications. 4. edition. Prentice Hall.

[6] Kiracofe, Daniel 2002. Transparent Proxy with Linux and Squid mini-HOWTO [online]. Available on the Internet:
<URL: http://www.linux.org/docs/ldp/howto/mini/TransparentProxy.html>.

[7] Kirch, Olaf and Dawson, Terry 2000. Linux Network Administrator's Guide. 2. edition. Sebastobol, CA, The United States of America. O'Reilly and Associates.

[8] Langfeldt, Nicolai, Norrish Jamie and others 2001. DNS HOWTO [online]. Available on the Internet: <URL: http://langfeldt.net/DNS-HOWTO/BIND-9>.

[9] Lempiäinen, Jukka and Manninen, Matti 2001. Radio Interface System Planning for GSM/GPRS/UMTS. 1. edition. Boston, The United States of America. Kluwer Academic Publishers.

[10] Mansfield, Niall 2003. Practical TCP/IP: Designing, Using, and Troubleshooting TCP/IP Networks on Linux and Windows. 1. edition. London, Great Britain. Addison-Wesley.

[11] Mohan, Anoop. Introduction to General Packe t Radio Service (GPRS) [online]. Available on the Internet:
<URL: http://www.eas.asu.edu/trace/eee459/Anoop%20Mohan.doc>.

[12] Nokia 2002. Support Guide for Nokia D211 Linux Device Driver [online]. Available on the Internet: <URL:http://www.forum.nokia.com>.