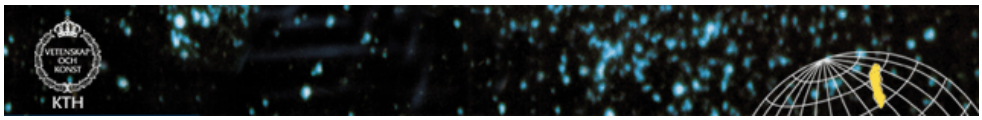
Interactions of Vertical Handoffs with 802.11b wireless LANs: Handoff Policy

GIULIO MOLA

KTH Microelectronics
and Information Technology

Master of Science Thesis
Stockholm, Sweden 2004

IMIT/LCN 2004-01



Interactions of Vertical Handoffs with 802.11b wireless LANs: Handoff Policy

Giulio Mola
{mola@it.kth.se}

3rd March 2004

Abstract

Nomadic computing aims to be a leading short term revolution in the Internet, however to succeed in this the infrastructure, the protocols, the handoff mechanisms have to be designed and implemented to provide mobile computing with both reliability and transparency.

A good deal of the work is addressed by mobile IP itself, but addressing wireless diversity is perhaps the most important next step. Many different wireless technologies are available; while one technology might provide wide coverage, another provides higher bandwidth, but can only be deployed locally, even uni-directional ad-hoc links can be used to carry IP traffic. To take full advantage of the potential infrastructure, a mobile device, with multiple network interfaces, should be able to dynamically switch from one link technology to another; hopefully totally transparently to the user. Moreover, having multiple interfaces allows the device to choose, each time a new connection is established, which interface to select to route the datagrams through, based on the type of service desired.

In this thesis a possible solution is presented, involving both vertical handoff optimization and policy management. Our testing device is the the SmartBadge v4, provided with a GPRS link and a 802.11b WLAN interface.

Keywords*: vertical handoff, mobile IP, SmartBadge, GPRS, 802.11b, Link Layer triggers*

Sammanfattning

Nomadic Computing har för avsikt att revolutionera dagens Internet tillämpningar. För att lyckas måste dock infrastrukturen, IT protokollen och *handoff* procedurer utformas och implementeras med intentionen att förse mobilt dataöverföring med både driftsäkerhet och autonoma processer.

En stor del av de nödvändiga förutsättningarna är tillgängliga tack vare *Mobile IP*, men mångfalden inom de befintliga Wireless teknologierna utgör fortfarande ett problem. Det stora antalet disponibla Wireless tekniker varierar från teknologierer som förser användaren med en stor täckningsgrad, till sådana som förser användaren med högre bandbredd, begränsad till lokala användningområden. Även *Uni-directional ad-hoc links* kan läggas i denna lista av mångfald. För att förverkliga fördelarna med den tänkta infrastrukturen fullt ut måste den mobila enheten kunna utföra omkopplingar från den ena wireless teknologin till den andra, med syftet att vara osynlig för användaren. Utöver detta är det meningen att en apparat med många anslutningar ska kunna välja den anslutning som passar bäst, var gång en ny koppling ska göras, beroende på den tjänst som efterfrågas.

Det här examensarbetet presenterar en möjlig lösning, som utnyttjar både *Vertical Handoff* optimering och *Policy Management*. Prototyp-enheten vi använder oss av för att genomföra undersökningen är *Smart Badge v4*, utrustad med GPRS och 802.11b Wireless LAN anslutningar.

Nyckelord: *Vertical Handoff, Mobile IP, SmartBadge, GPRS, 802.11b, Link Layer triggers.*

Aknowledgements

I would like to express my graditude to all the people in KTH's Center for Wireless System, for their support, their suggestions, and the good time we had together. This project would not have been possible without the assistance and guidance of Prof. Gerald Q. "Chip" Maguire Jr., to him goes my biggest thanks, for teaching the best thing a professor can teach, his passion for his work.

Contents

List of Figures

Chapter 1

Introduction

1.1 Wireless data technology

Wireless technologies are changing users' information access patterns. Enthusiasts claim that the large scale introduction of these technologies will led to a revolution in the Internet, comparable in impact to what the mobile phone represented for telephony. Throughout 2002 the number of wireless LAN hot-spots grew dramatically in western Europe, after a stale period due to regulatory restrictions. In the UK, the market increased by 327% over the previous year from 269 locations at the end of 2001 to around 1,150 locations at the end of 2002.[1]

One can expect a fusion of different technologies. For example, some expect there will be no distinction between mobile phones and IP enabled devices. Voice over IP technology has been spreading among broadband users and campus infrastructures. The introduction of IPv6 and third generation cellular networks is a step further in this direction, the next step will be to unify datacommunications and telecommunications industries, resulting in voice and data traffic traveling on the same network.

Looking at the above numbers, we believe the converged network will be an IP network. Third Generation cellar networks, when and if they become widely available, will deliver data speeds of 1 megabit per second; today IEEE 802.11b already provides 11 megabits per seconds.

1.2 Wireless technologies

The details of the process by which radio waves propagate through the air, the amount of data carried, immunity to interference, coverage of the antennas, and

[1] source: IDC's European Telecommunications and Networking, 11th February 2003.

other characteristic varies from technology to technology. For each possible usage scenario there generally exists one technology that best fits those *particular* needs; for instance a cellular phone's network requirements are different from wireless data requirements, in the former "total" territory coverage is required, because it has become the expected degree of coverage in Europe and increasingly elsewhere, in the latter high bandwidth is generally the key factor, while coverage is a secondary factor.

1.2.1 Classification

There is no one single technology that offers at the same time low cost, high speed, nearly universal coverage, and that suits all the different user needs. We will classify the wireless technologies currently in use to clarify these concepts.

Wireless technologies can be segmented based on the geographic area they cover, one common way is to define three classes: wide-area, local-area, and personal-area networks (respectively abbreviated WANs, LANs, PANs).

A cell of a wireless WAN (WWAN) covers an area which is typically measured in square kilometers. The frequency band, given the range and the relatively high-power required, is licensed, i.e. carriers must have a licence and often pay a fee for their users to transmit within this frequency band. Typical technologies that belong to this category are cellular networks, such as GSM, TDMA, CDMA, and others. The data-rate is relatively slow, typically the order of kilobits to ten of kilobits per second, however the introduction of third generation cellular network promises faster data rates.

Wireless LAN operates over a small, local coverage area, normally about a hundred meters in range and generally uses an unlicensed frequency band. The IEEE 802.11b wireless standards transmits at speeds up to 11Mbps, while 802.11a and 802.11g increase this to up to 54Mbps[9]. As more users share the same WLAN cell, the average per user throughput decreases. However, higher rates generally operate over smaller areas. In Europe WLAN generally use the 2.4 GHz ISM band which is shared with other devices such as microwave ovens thus signal interference can occur. IEEE 802.11a devices use the 5GHz ISM (in Europe this was allocated as the HyperLAN/2 band).

A Wireless PAN cell covers a relatively small area, and is often seen as a cable replacement to interconnect small personal devices, such as mobile phone, laptop, or handhelds. Technologies like BlueTooth, IrDA, IEEE 802.15 belong to this category; for BlueTooth the data rate is about one-tenth that of the current IEEE 802.11 (specifically 802.11b), although Bluetooth power consumption and cost were expected to be significantly lower, although this has not been realized

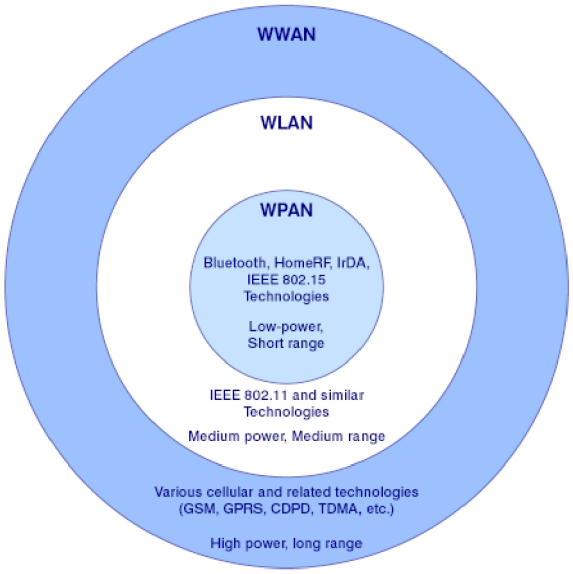


Figure 1.1: Wireless technologies

in practice.[2]

These heterogeneous technologies are not likely to merge into a single multi-purpose standard. We must keep in mind that for a device to be mobile, first we have to eliminate the power cable, hence limited battery life is a major issue. The following rule will always apply: $PowerUtilization \propto F(bandwith, range)$

Thus battery life is generally inversly proportional to bandwidth and range. Therefore it is likely that for the forseeable future we still have to face a heterogeneous wireless scenario.

Overlays Network

Overlay networks are based on the assumption that in the same area several wireless technologies can coexist. We can imagine an environment where higher bandwidth technologies (short range) comprise the lower levels, providing wireless cells that each cover a relatively small area, and these cells together cover a collection of largely disjoint areas. Technologies that provides these high bandwidth local links include 802.11a and 802.11b WLANs, IRDA, and BlueTooth. Higher levels utilize technologies with more limited bandwidth, but that cover a larger area such General Packet Radio Service. The highest level is a very wide-area network, which provides low average bandwidth per user, but almost ubiquitous coverage, such as satellite systems.[1]

[2]In addition interference can occurs also between IEEE 802.11 (also known as Wi-fi) and Bluetooth.

1.2.2 Wireless diversity

Rather than viewing these different technologies as competing, it is more useful to view them as complimentary technologies[2]. Nowadays mobile devices provided with more than one wireless interface are a reality, and the cost of the wireless interface itself affects only a part of the total cost of the device, so that a mobile node can exploit this increasingly wireless diversity.

There are many reasons users to desire a device works over different wireless technologies. First it is a way to combine in one device the best that each single technology claims to offer; for example combining GPRS and WLAN together results in having ubiquitous coverage, via the GSM network, and high speed when a WLAN hot-spot is available. Furthermore highly directional links such as IRDA are the right choice when it is desirable to identify a service from its physical location; imagine our nomad wishes to use a printer, by employing the directional link it is sufficient to localize the closer printer by pointing the device toward it and sending the file to this printer, without needing to know a-priori the printer's IP address.

1.3 Nomadic Computing

Changing from wired to wireless networking is not simply a matter of using or not using cables. It is straightforward to use wireless technology where cables where used before, but the advantage of the wireless comes into its own as new services are introduced, which are not limited to a fixed location.

Imagine a scenario where people access network services anyplace, anytime, and anywhere, with connection to the network being ubiquitous and persistent, even when the mobile node moves from location to location. When a nomad 'settles' in a temporary location he/she should discover services which are available in that particular place, e.g. a printer, speakers, or a wall projector. Today's mobile devices are becoming aware of their movement, velocity and acceleration sensors can be used to help roaming performance, light sensors can be used to automatically adjust the LCD contrast; and biometric sensors used to indentify who is using them.

Despite the new network infrastructure being complex, since it is made from different wireless networks, devices, and protocols, we would like the user to be totally unaware of this. From a user's viewpoint what is important is simply to have connectivity and to be as effective as possible in each new environment, without requiring changes in the manner in which the user operates the device.

We refer to this new paradigm as nomadic computing. We can sum up the requirements of this new paradigm in the following points:

1. ubiquity (connectivity is always available);
2. connections persistent even while the nomad is moving;
3. reasonable bandwidth, i.e. applications that were designed to run over wired networks must work smootly, all be it at perhaps a lower performance.

1.4 Goal of this thesis

Unlike the wired network, where it is usually the case that the connectivity provided by the network is reliable and high bandwidth, information access via a wireless network to a mobile device is plagued by poor connection maintenance, i.e. link connectivity may be broken. Parameters like bandwidth, reliability, latency, and cost, change dynamically and dramatically as the device moves from location to location (or even as other objects in the environment move). The goal of this research is to permit users and programs to be as effective as possible in this environment of uncertain connectivity.

Furthermore it is important to guarantee backward compatibility, i.e. higher layer protocols designed to run over wired networks should run 'as they are' over wireless links. On the other hand the design of software for mobile devices must consider resource limitations including limited battery power, limited display size, and other constraints often requiring new mobile specific applications to be implemented.

We believe that once wireless connection to the Internet is widely available everywhere and sufficently reliable, the next steps toward the envisioned new paradigm will easily come. The goals of this thesis are to provide a framework (1) for ubiquitous connectivity and (2)to assure connection reliability, or to show how these constraints can be relaxed while still providing suitable service(s) to the user by exploiting multiple wireless interfaces.

1.5 A scenario: walking around the city

Arya, our nomad, is walking around Stockholm, with his handheld PDA in his breast pocket. An incoming call interrupts him from his thoughts; the PDA screen displays his girlfriend's face. He puts on his headset and accepts the call. She asks him to buy Chinese food on his way back home. He hangs up and continues walking while deciding to listen to music from his collection, it is the play-list he likes to listen to on foggy day like today.

Between songs the PDA informs him that Chinese food shop is just few blocks away, in the Galleria. As he enters the shopping mall a number of high band-

width hotspots are available, while in the chinese food shop the PDA downloads the movie he and his girlfriend earlier chose to watch this evening, fetches some large video e-mail he has received, and fills the PDA's buffer with additional content as scheduled by the play-list.

Back home the PDA synchronizes with other devices, video e-mail files are downloaded to the local file server, and the movie is sent to the wall projector to stream.[3]

[3]This scenario was inspired by [6]

Chapter 2

Mobility and its effects on our problem statement

This chapter introduces some of the key concepts of IP mobility, specifically how we define mobility and how this definition affects IP based data communication.

2.1 What is mobility?

Mobility is defined as the ability to move physically. We want to extend this definition to address the case of mobile communication, when two moving actors are exchanging data. In this case mobility means that both sender and receiver are free of the constraints imposed by changes in their physical location.

With static actors, both of them can know how to address the other one in their current location; for instance in the postal system there is a set of cordinates that uniquely identify an actor, e.g. name, street, apartament number, town, country, and postal code. However, these values depend on the static physical location, if while you are traveling and the postman delivers a letter for you, you cannot read it until you return to this address.

2.2 Addressing and mobile communication

It is clear from the previous example that the addressing scheme is the key to enabling mobile communication. If you stay in a hotel room for sometime perhaps you would like to pick up your mail at the reception desk. One possibility is to inform all the people that wish to communicate with you, of your temporary address, i.e. the hotel's address. The other is to ask someone to forward the mail delivered to your home address to the hotel's address.

New address notification is one means of mobile addressing: your address changes as you change location and you have to inform all your friends of the change that has occurred. This solution presents difficulties to people that are **not** in your contact list, i.e. since you did not inform them about the new address, they cannot directly send a letter to you.

A revised version of this mechanism utilizes *directory lookup*. When an actor wishes to contact you, he calls your agent and asks for your current address; then he addresses the letter to your current (temporary) location. Directory lookup is better since new actors can get in contact with you, but every time any actor wants to contact you he has to go through your agent.

Forwarding is the other possibility, your friends are unaware of your current location, but there is a third actor assisting them (and you), the agent, which collects and forwards letter for you. Unfortunately for each letter a delay is introduced and it must generally be forwarded at your expense.[5]

All three solutions have side effects, but this is the price you must pay for mobile communication.

2.3 Movement and IP addressing

IP addresses are not so different from postal addresses. Whereas on a letter we have name, street, town, etc. an IP address is composed of two parts: sub-network identifier and host identifier; the subnet id represents the network where the host is attached, while the host identifier identifies the host among all the others hosts on that network.

Moreover, subnetting builds a multilevel hierarchical structure, which helps the routing algorithms. This is much like employees working in the sorting department of a post office, where letters are first divided based on the district of the destination address; then they are either sent to another district or, when the address belong to this Post Office's district , delivered to the addressee. The same basic mechanism underlies IP routing.

Before going further, it is important to define what is a movement in the case of IP networks. Given a wireless interface, a user is able to physically move around with his device while connected to a certain sub-network; for instance, an IT manager moves his laptop from his desk to the conference room, where he is about to have a meeting. It is likely that this physical movement does **not** entail movement at the IP layer, if the base station the wireless interface was attached to did not change, in other words if the meeting room and the desk are both in range of the same basestation's antenna and the laptop has maintained the same IP address, then it is still connected to the same network, hence there is no change in the IP address or route. We say that movement

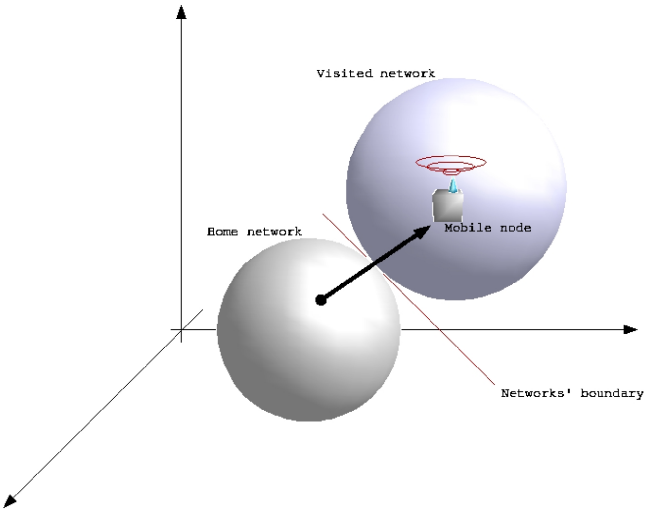


Figure 2.1: IP addresses and node movement

occurres with respect to the IP layer when (due to some change) a mobile node connects via another network (or more properly another sub-network) resulting in the node changes its point of network attachment. Since each sub-network has its own network-prefix, *if a host moves from one network to another, its IP address must change.[3]*

2.4 Addressing for IP mobility

In section 2.2 we described some general mechanisms that can be used to address a nomad (i.e. a mobile actor), not all of these mechanisms are suitable for IP. A brief description of how IP communication works is necessary in order to understand which problems mobility raises.

Transport-level protocols are used for exchanging data between nodes, the most widely used are TCP and UDP, unfortunately both use IP addresses *together* with port numbers as end-point identifiers, i.e. a transport connection within a node is identified by the values: *IP source address, IP destination address, TCP|UDP source port, and TCP|UDP destination port. The* assumption is that all these quantities will remain **constant** for the duration of the transport connection. If one of these underlying values changes, the transport connection would abruptly be dropped. Hence an addressing mechanism is needed to allow hosts to move seamlessly while maintaining existing transport connections.

To summarize, in order to support IP mobility addressing should be modified to solve the following problems:

1. How can I know the current address of the mobile node?
2. How can I preserve an existing connection despite the node roaming to a new subnet (address)?

2.4.1 Directory lookup

Using symbolic names the first problem can be solved. The domain name server binds an IP address to a mnemonic name; when the mobile node moves, it could update its entry in the name server, binding the symbolic name(s) to the new IP address. Before opening a connection to a mobile node, each application must make a name server (NS) query; then once the current IP address is obtained a connection to the mobile node can be established. However, this solution works only far as long as the mobile node retains this IP address.

2.4.2 Forwarding

By means of forwarding both the above problems can be solved. Referring to our postal example, a way to deliver e-mail over the Internet to either mobile or static users has already been in use for many years. The e-mail system supports mobile users since it relies on a **fixed** server to store incoming e-mail; when the user wishes to access his mailbox, he simply contacts his e-mail sever and fetches the stored messages, thus messages are **not** sent directly to the user. Since connections to fetch or deliver e-mail are always initiated by the user(s), either the addressee or the sender (respectively), only the e-mail server is required to have a (fixed) IP address known to the DNS server and bound to the name of the user's mail server.

The same mechanism underlies forwarding. A mobile user is identified not by a temporary IP address, which changes as the user moves; but rather the mobile user is always reached through a fixed IP address known to the DNS server, this is generally the address of a server (usually connected to the wired network) that acts as forwarding agent for this user. The server receives all the traffic addressed to the mobile user and forwards it to his current (temporary) IP address. Whenever the user moves, this server needs to be informed of the mobile's new IP address, then forwarding takes place using this new address as the packets' destination. *This is roughly the solution adopted by IETF for Mobile IP, and the method we will utilize in this work.[4]*

Since the mobile appears to be at a **fixed** IP address, the mobile user is **always** reachable by anyone. If this fixed address is used in the transport end-point specification, then existing transport connection *should* be mantained and thus even long term communication sessions between mobile actors is possible.

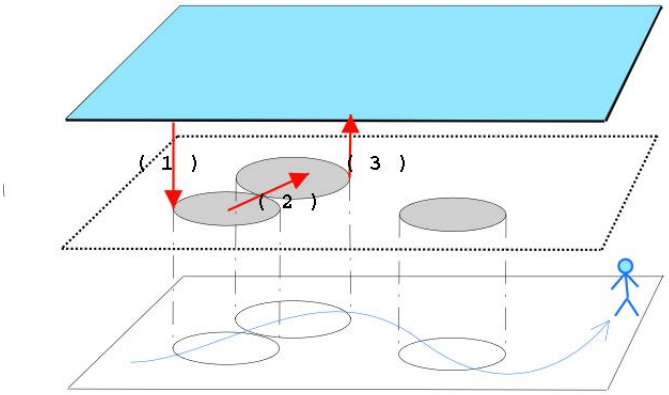


Figure 2.2: horizontal/vertical movement

2.5 Mobile IP in wireless overlays.

The process of migrating connectivity from one base station to another is commonly referred to as handoff, handover, or roaming. We can further refine this definition of into so-called vertical and horizontal handoffs (see figure 2.2).

Horizontal handoff occurs between base stations that use the same type of wireless technology, while vertical handoff involves base stations that utilize different wireless technologies. We can distinguish further between upward and downward vertical handoffs: upwards refers to movement from a lower to a higher level (i.e. from a small sized cell to a larger sized cell), and downward refers to movement from higher levels to lower levels.

Figure 2.3 shows when vertical or horizontal handoff occurs: in (1) the mobile initiates a vertical downward handoff, in (2) an horizontal handoff, in (3) a vertical upward handoff. Notice that horizontal or upward vertical handoff occurs because the device, prior connected to a certain base station, goes out of range and thus the mobile *is forced* to utilize (also referred in WLAN standards as "attach to") a new basestation. A downward vertical handoff procedure, instead, has to be (spontaneously) initiated by the mobile node i.e., since it has connectivity over a wide area, it must *decide* to utilize a cell which is more *local*.

2.6 Analysis: walking around the city

We will analyze the scenario presented in the introduction. First notice that connectivity is **always** available; when Arya accepts the incoming call, he is using VoIP together with some other video streaming standard(s); when a lower level connection is available, as in the Galleria, a downward vertical handoff takes place **if** there is an application that requires high bandwidth. Thus in our example, upon entering the Galleria a sleeping download wakes up and the buffer for the playlist is filled. When exiting from the Galleria an upward (vertical) handoff returns the users' connection back to higher level, and this will suspend the download application if it has indicated that it only wants high bandwidth low cost connectivity. When the user enter their home another downward handoff occurs, this time to synchronize the PDA with the home appliances.

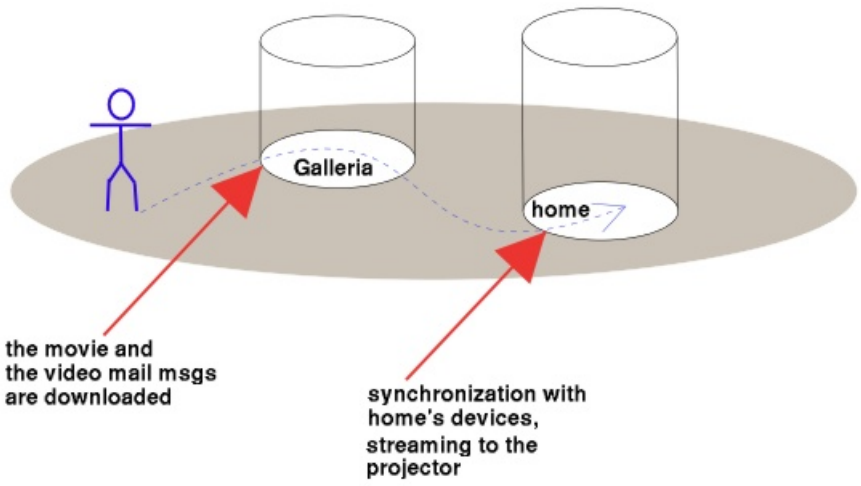


Figure 2.3: Arya is walking

2.7 Problem statement

We stated that while the mobile node (MN) is on the move, existing transport connections should be maintained, even in the case of real-time communication. We want to analyze under what circumstances this is feasible.

We have discussed how Mobile IP can solve the addressing problem, but switching from one network to another introduces two other issues besides the change in the IP address:

1)horizontal/vertical handoff delay

2)bandwidth bottleneck(s)

Both of these factors can affect existing TCP and UDP transport connections and in the worst case may make them unusable.

During the upward and horizontal handoffs, the MN is actually *not connected to any network* and thus is unreachable, this is especially true during a horizontal handoff using a single interface[1].

After the handoff has been completed, if the new network has less available bandwidth then the old one and the MN node was generating more traffic then the bandwidth now available, then the reduced bandwidth can be an impediment for existing data flows. Such bandwidth bottlenecks are less likely to happen in horizontal handoff, but are more insidious since not easily predictable; and they are common, but more easily predicted in upward vertical handoffs.

In this research we will address handoff optimization, we believe that this is urgently needed in order to achieve successful nomadicity. Moreover bandwidth bottlenecks can be avoided at the application level: since changes in bandwidth can happen at other times, not only when a node moves, thus a well engineered application should always address the problem of available bandwidth and its utilization of bandwidth. For example, some multi-media and in particular VoIP implementations can dynamically change the amount of data flow by adjusting the compression level, therefore dynamically adapting to available bandwidth.

[1]This is assuming that the single interface only supports a single wireless link at any given time

Chapter 3

Methods

Handoffs optimization has been a fashionable topic for research, in recent years many papers have been published on vertical handoffs [1, 7, 8, and others], hence one might wonder why anyone should invest additional time in further research. The reason is that we will follow a slightly different approach in our research: instead of an analytical investigation and mathematical simulation we have built a prototype of our vertical handoffs framework. This make possible to test how well actual "every day" applications run on this prototype, and how they are affected by handoffs.

Following this approach required a great deal of work to implement the necessary software. We believe the resulting software will be useful for other researchers and thus it will be released under GPL. We also believe that by measuring actual usage we gained additional insight (these are decribed in chapter 7).

3.1 The Platform

The hardware platform we used for our testing was the Smart Badge v4[1]. As the name suggests it is roughly the size of an ID card and measures 12x6x1.5 cm; the basic system includes sensors for light, humidity, temperature, and acceleration. The accelerometer can be used for movement detection in order to help the handoff procedure. The processor is an Intel Strong Arm 1110 and an SA 1111 co-processor which supports the PCMCIA interface and the IRDA interface; it offers enough processing power to properly test our handoff implementation with interesting applications.

The Badge represents a good choice for our work; as it is a portable device

[1]Badge v4 is a prototype from HP labs, for up to date information see http://www.hpl.hp.com/personal/Mat_Hans/research/badge4/

that is movement aware; we can plug many wireless interface into it; and it runs Linux (kernel version 2.5 beta). In addition, the Badge provides a high quality (16bits per channel, 44k samples per second) audio encoder/decoder, thus allowing working with entertainment quality audio streams.

The Badge was also equipped with both a GPRS card and WLAN card for testing our vertical handoff framework between these two types of networks. The WLAN card, in compact flash format, was manufactured by Socket and uses the PrismII chipset and the GPRS card is a Sony/Ericsson model GC75. Another wireless card used in our testing was a Lucent Technologies silver 802.11b card, equipped with the well known Orinoco chipset.

3.2 Wireless LAN 802.11b

The first WLAN standard was introduced by IEEE in 1997 as IEEE 802.11. It defines both Media Access and Control (MAC) and Physicial (PHY) Layers. It is similar in many respects to the well known 802.3 (Ethernet) standard. However IEEE 802.11 must take into account the additional physical limitations of the wireless media, e.g. range limitations, unreliable media, dynamic topologies, and interference from other sources.

The 802.11b PHY defines data rates up to 11Mbps, with fall back rates of 5.5Mbps, 2Mbps, or 1Mbps under noisy condition or for backward compatibility with earlier 802.11 products. IEEE's 802.11b standard uses direct sequence spread spectrum coding.

The 802.11 MAC defines asynchronous, best effort, connectionless delivery of MAC layer frame data. The access method is CSMA/CA, similar to the CSMA/CD deployed in 802.3 (Ethernet) LAN, but with collision avoidance (CA), since the trasmitting station can **not** listen while trasmitting, which would be required for collision detection (CD). Moreover the MAC also provides authentication and encryption services, the later via a wire equivalent privacy (WEP) mechanism.[9]

IEEE 802.11b is often marketed under the trade name "Wi-Fi", which was a marketing phrase designed to suggest wireless fidelity.

3.2.1 WLAN in the real world

The reason behind the incredible growth number of hotspots is due to many factors, but as usual, it is strongly related to low prices of the components (i.e. WLAN cless). A basestation costs nowadays from US 150 to US250, depending on the specific WLAN standard(s) implemented and additional functionality such as Network Address Translation or firewalling it might support. A wireless

PCMCIA card costs the end user between us$ 50 and us$ 70, a PCI version for desktop use costs even less, and Wi-Fi is increasingly built in into PDAs and laptop computers.

This makes WLAN a convenient alternative to standard Ethernet for small office / home office (SOHO) user(s) that wants to setup an intranet. In fact the total cost for Ethernet hardware components, e.g. hubs/switches, Ethernet card, and cabling is often higher than an equivalent WLAN. The ease-of-use of some WLAN devices, such as the many basestations with integrated NAT, make them attractive for home users who want to access the internet from their couch or read the news headlines in the kitchen while having breakfast; all they have to do is just plug the basestation device into their network jack (often aan ADSL or cable modem) and they are all set. WLAN is also used as a point-to-point link to provide Internet access in areas with few inhabitants, where cabling would be too expensive to install.

The fact that there are many Wi-Fi hotspots does not mean that they are available to everyone. Today most hotspots are for private use. Even if system administrators have not protected their WLAN network, it may be illegal in many cases to connect to their networks without authorization. Others hotspots are meant to be open to larger numbers of users, such as those in airports, train stations, hotels, universities etc. Some private user have purchased a basestation with the intention of letting other people connect via it, in the hope that if everyone does so, there will be high speed WLAN coverage in densly populated centers[2].

3.3 GPRS and GSM

The General Packet Radio Service (GPRS) was designed to offer packet-switched radio services over GSM (a TDMA based cellular system). GSM is one of the most widely used technologies for mobile phones. It is a circuit-switched cellular telephony network. When packet-switched data reaches the GSM basestation, the packet is sent via a packet switched network to a Gateway GPRS Support Node (GGSN) where it can leave the GPRS network and be transfered to a TCP/IP network such as the Internet.

GPRS theoretically offers maximum speeds of up to 171.2 kbps, unfortunately this speed is not achievable in practice. Accord to the ETSI specification, the highest speed for a single time slot allows 21.4 kbps, thus theoretically by using all the 8 time slots available a node could achieve a maximum troughput of 171.2 kbps, but it is unlikely that an operator will allow one user to consume

[2]There are many groups that work on this idea. For Sweden see http://www.elektrosmog.nu, for a list of similar see groups http://www.seattlewireless.com

all the available bandwidth in a channel as this would block other users' calls.

What make GPRS interesting to us is that it provides roaming below the network level, thus a user can move freely in the GPRS operator's coverage area while sending/receiving data over a TCP connection. To give an idea of the coverage of the GSM network, in Sweden operators must cover 95% of the country (449960 km^2) to obtain a license.[3]

3.3.1 GPRS today: costs and performances

In Sweden and in others part of Europe GPRS has nearly ubiquitous coverage; this was the main reason why we decide to adopt it in our framework for providing wide area connectivity.

In Europe GPRS is provided as an extra service by cellular operators. It is (of course) not free of charge and billing is done not based on connection time, but rather based on amount of traffic sent and received. The connection to the internet is generally through a NAT, for two main reasons: first it is cheaper for the operator, as they need fewer IP addresses; second it avoids unwanted connections from outside networks, this is important since the user's bill is based on the **total** amount of traffic (i.e. both transmitted and received traffic, while most European GSM customers are used to being billed only for class they make). By blocking incoming traffic unless the mobile initiates it, the operators avoids attacks where attacker bombards a mobile node with traffic which the mobile subscriber would be charged for. This is a limitation we have to face, since it must always be the mobile device that initiates a GPRS connection, e.g. with a home agent, and IP-in-IP encapsulation will not work.

For our test we connect to GPRS through Telia's Swedish network[4], the prices were:

- 300 SEK / month, for total traffic below 25 megabytes and
- 0.018 SEK for each additional kbyte

The throughput of the connection we measured was far below the theoretical speed of 171.2 kbps. To measure the throughput we sent a burst of 1000 blocks, each 1024B long, over a TCP connection; packets were received by an end host, located within Sweden, that simply discarded them. The average up-link speed over 10 measurements was 12.93 kbps. We do not have information of how many time slots were reserved for the communication, but since the average speed was lower than the theoretical speed for a single slot, we presume that only one slot in each frame was utilized. This is similar to the results of others.

[3] http://www.pts.se "Post & Telestyrelsen"
[4] http://www.telia.se

Chapter 4

Handoffs optimization

4.1 Handoff definition

In sections 2.5 and 2.7 we defined handoffs as the process of changing from using one base station to using another, we also stated that the delay in performing such a handoff is critical since the mobile node is generally unable to communicate until the process has completed. We therefore define handoff latency as the interval of time between the last user data packet sent on the old network and the first user data sent on the new network. Optimizing handoffs means **both** reduce this latency time and avoiding unnecessary handoffs.

Looking closer at handoffs they consist of three phases:

1. the mobile node discovers that connectivity was lost or that it is about to loose connectivity;
2. the mobile node determines what other network or networks are reachable; and
3. the mobile node selects and connects via one of these networks.

Each phase can be independently optimized and perhaps even reordering some of these phases might result in lower handoff latency. In this chapter we will focus on phase #1, namely detection/prevention of connectivity loss. Both Héctor Velayos [10] and Jon-Olov Vatn [15] have examined the delais of phases 2 and 3, hence we refer interested readers to their work.

4.2 Link Layer vs. Network Layer

The roaming process in a technology such as WLAN, where handoffs are controlled by the mobile node, i.e. it is the mobile node in charge to initiate the

handoffs, begin by detecting that connectivity is lost or has changed.

Referring to the ISO OSI protocol stack there are three alternatives: detection can take place either at the Physical Layer (L1), the Link Layer (L2), or the Network Layer (L3). The alternative adopted in IETF's Mobile IP is to detect connectivity loss at L3: the mobile node listens for IGMP Router Advertisements to discover that the point of attachment has changed and consequently it initiates the handoff procedure.[4] An evident bound to handoff optimization in this case is the frequency at which Router Advertisements are broadcasted. Basically detecting a missed beacon requires 1.5 beacon intervals, since at least one miss has to be detected, on the other hand some systems only initiate handoff after missing **several** beacons. The lower bound to beacon interval in the first IETF Mobile IP RFC was 1 seconf, this leads to handoffs time in the order of seconds, which is disruptive for many data flows. More recently the interval has been lowered to milliseconds.

The reason behind this choice is to be found in layering, TCP/IP must work above any kind of wireless technology; opting for L2 or L1 detection would unquestionably decrease handoff latency, but it would violate layering. As a compromise a common API for L2 triggers has been proposed.[12] Such L2 triggers would reduce the detection delay to below 100ms.

4.3 Horizontal handoff triggers.

Because L1 and L2 triggers utilize information from the physical and link layers (respectively), rather than having to wait for a loss of connectivity, we utilize parameters such signal to noise ratio (SNR) to **predict** connectivity loss. This enables the system to initiate the handoff process *while(1)* there is still connectivity and *before(2)* there is a loss of connectivity.

Nevertheless it is not as straightforward as it might seem. The easiest approach is to compare the SNR of the basestations in range and trigger a handoff to the strongest one:

$SNR_{new} > SNR_{old}$ (Eq. 1)

or alternatively if the SNR of the new basestation is greater than the SNR of the old basestation plus an hysteresis margin H:

$SNR_{new} > SNR_{old} + H$ (Eq. 2)

The use of hysteresis prevents oscillations between two basestations.

A more accurate approach, is to trigger the handoff only if the SNR of the current basestation fall below a threshold SNR, in order to avoid unnecessary handoffs:

$SNR_{old} < T_{SNR}$ (Eq. 3)

All these approaches fail to take into account that the mobile node could actually be repeatedly crossing the network border, where we can define this border by a temporal threshold. The ping-pong effect is avoidable by introducing a dwell timer: the timer is started when the condition is true and the handoff is triggered only if the condition continue to be true when the timer expires, where the condition can be one or a combination of the above, i.e. (1),(2),(3),(1)+(3),(2)+(3).

4.4 Handoffs in a hybrid network

The mechanisms described in the previous section do **not** apply when we consider dissimilar networks, thus we can not use the above trigger to vertical handoffs; this is because it is not meaningful to compare, for instance, SNR from different technologies.

One possibility would be to let the user trigger the vertical handoff, but this would violate the transparency of the model we are attempting to build, as one of the requirements was that the users should be unaware of the network structure. It is clear that we **can** trigger a *upward* vertical handoff when we can no longer communicate with the link in the current level and there are no other basestations detected in this level. However, we can not use this approach to trigger a downward vertical handoff.

Thus the solution is to a priori define policies to trigger vertical handoffs. A straight forward policy might state that WLAN is preferred over GPRS when both are available, because of the higher throughput via the WLAN. Nevertheless there are other aspects that we might want to consider when defining our policy, e.g. cost of the link, reliability, etc. In the next chapter we will analyze how policies that take into account all these characteristics can be defined.

4.5 Our framework

We can refine the defining of triggers to utilize these policies. If we need connectivity, then *we just need to know if a particular wireless link is available or not*; if not, then the software will ask the policy manager what link should be used, thus for example a vertical handoff might be initiated. In conjunction with this, horizontal handoffs will occur based on the conditions in section 4.3. For ease of reading we will suppose that when the MN is connected via 802.11b, the condition for horizontal handoffs is:

$SNR_{old} < T_{SNR}$ and $SNR_{new} > T_{SNR}$ (Eq. 4)

This means that if the SNR of the current network falls below the threshold

and another network is in range, for which the SNR is higher than the threshold; then we will initiate a horizontal handoffs. Along with this threshold based rule we will also make use of a dwell timer.

In Eq.4 we did not take into account policy management; it could be the case that the available network(s) do not fit user's requirement, e.g. cost, and thus a vertical upward handoff would take place instead. Also when more networks are available, the policy manager must chose among them.

Vertical upward handoffs will occur in the following situations:

$SNR_{old} < T_{SNR}$ and $SNR_{new} < T_{SNR}$ for all possible WLAN (Eq.5)

In this case the software will initiate a vertical handoff to the GPRS network.

Vertical downward handoffs, e.g. when the mobile node switches to a WLAN link from a GPRS link, are entirely policy management based.

4.6 Threshold and dwell time estimation

The value of T_{SNR} and of the dwell timer significantly affects the performance of the model. The goal is to trigger the handoff *just* before we are about to loose the current link, in other words before experiencing any packet loss. A big value for T_{SNR} would result in handoffs being triggered before the node approaches the network border, i.e. before it is about to loose packets, introducing *additional* packet loss due to handoff latencies that could be avoided. If T_{SNR} is too small, instead, handoffs are triggered too late and thus packet loss is likely to occur before the handoff procedure is initiated.

The dwell timer value should be related to the speed at which the mobile node moves, the higher is the speed of the mobile node the lower the value of this timer.

To find the optimal value for T_{SNR} and dwell timer we set up a test to relate SNR and packet loss. The MN generates a constant flow of UDP packets that are sent to a server on the same network segment as the base station, this to reduce the probability of packets being lost in the wired network. Each packet has a payload of 92bytes[1], the only relevant data in the payload for this experiment is a sequence number, that the server uses to detect packet loss. The interval between two sequential packets is 200 ms; for each packet sent the mobilenode stores the current value of signal strength and noise. Examining these values in the packets received at the server it is possible to see for which SNR value packets were lost.

Originally we wish to set the rate for our experiment to 20ms, we did several measurement but we were unable to collect any data since, for some reasons

[1] 92 byte was chosen as the average payload of a RTP packet, used for voice with a simple codec in VoIP

probably related with the wireless card driver, the Badge halted when we tried to read the value from the wireless card at such a rate. Reducing the rate to 1/10 we were able to collect the following data.

The next figures show the result of our test. The MN moves away from the base station until it looses link level connectivity (samples 150 - 350); then it changes direction, moving back towards the basestation following the same path (350-480).

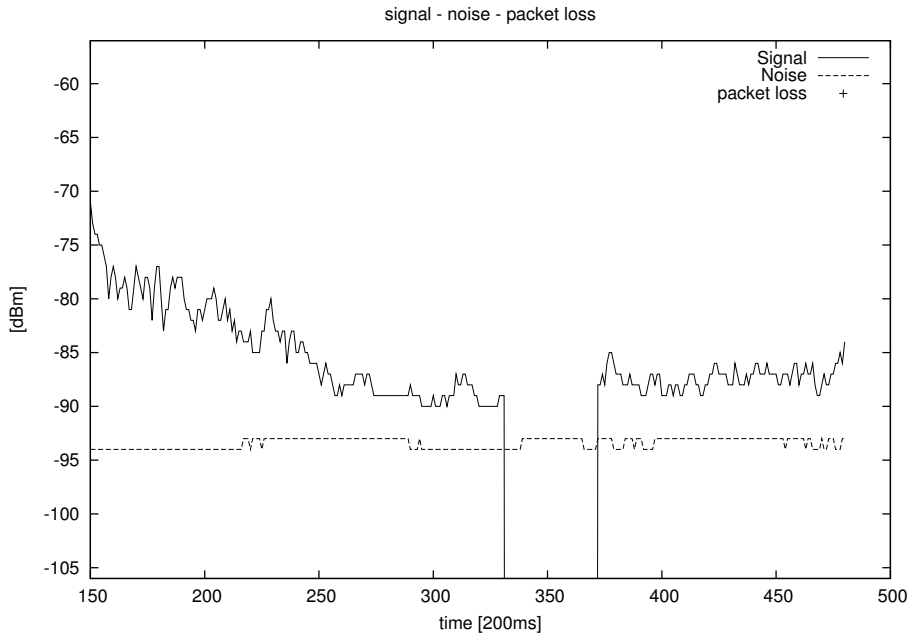


Figure 4.1: Signal, noise, and packet loss

In figure 4.1, the signal strength and noise are plotted, together with packet loss. It is remarkable that while the signal decreases as the MN moves away from the base station, the noise is almost constant. The signal strength abruptly fall to 0dBm when the wireless card reports that there is no network connectivity; in reality the signal is above 0dBm, but it is not possible to receive WLAN beacon because of the really low SNR.

We had to modify the driver for the Prism-II wireless card to report a fake value of 0 for the signal, since the original version of the driver reported incorrect values for signal and noise when link level connectivity was down, misleading

our software. (Specifically these values were not updated properly when there was no traffic getting through.)

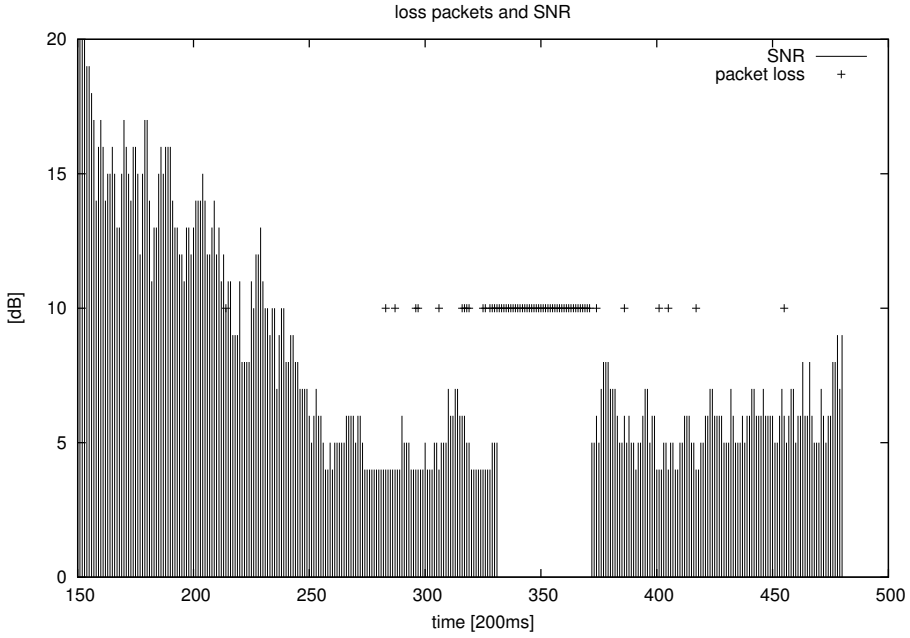


Figure 4.2: SNR and packet loss

In figure 4.2 we can clearly see the relation between SNR and packets loss. For values of SNR above 6 we have sporadic loss of packets, which is to be expected in a wireless network. For SNR below 6 packets are lost because of poor signal quality, thus T_{SNR} must be at least 6 to avoid packet loss.

The dwell timer value will be optimized according to this test. A better approach would be to dynamically change the value of the timer according to the current speed of the MN, but this is beyond the scope of this work. We will optimize it according to this test, in a scenario where the MN is moving at the average speed of roughly 5km/h, e.g. the one of a man walking; this was the speed of the MN when we run the test.

For this experiment reasonable values for T_{SNR} and dwell timer are:

$T_{SNR} = 7dB$

dwell = 1s

As it is shown in figure 4.3 these values fit our scenario well. Referring to the

same set of data (altough just a portion is shown in figure 4.3) at sample 236 an unnecessary handoff is avoided; since during the dwell time the SNR became higher than 7, so the condition for the handoff is false. At sample 251 a vertical handoff should take place since the SNR had been lower than 7 for the dwell time, and this is exactly what we want, since as shown the SNR continued to decrease.

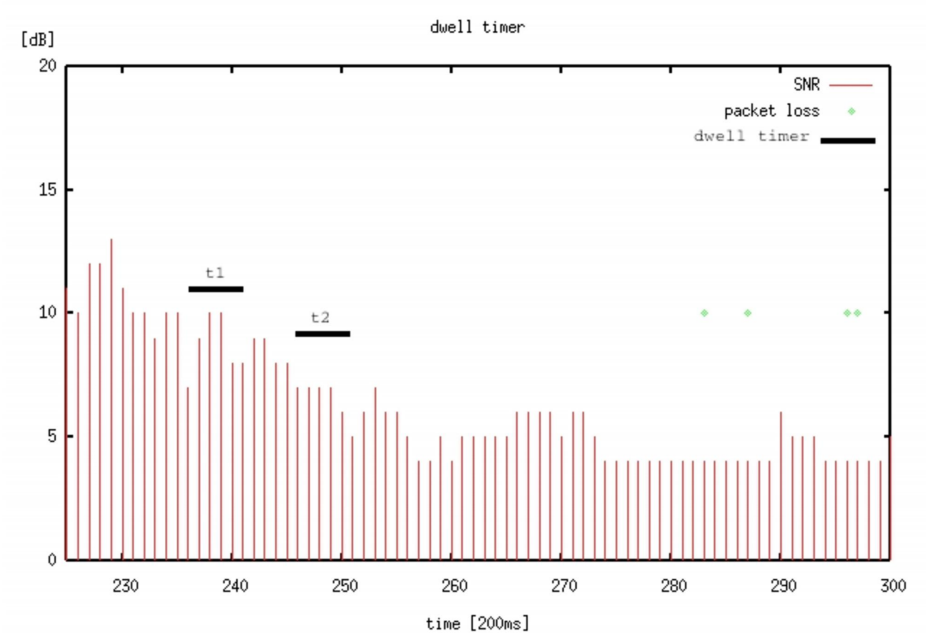


Figure 4.3: Dwell timer

Chapter 5

Policy based handoff management

Our belief is that handoff optimization is not simply a matter of reducing delays, but rather the approach we will follow is to ***avoid*** *handoffs through policy based handoff management. We identify two alternative methods of policy based management: data driven or link driven.*

5.1 Data driven

To exploit wireless diversity, when two or more interfaces are available at the same time, it is possible to *choose* the interface that best fits the requirement of the data flow the mobile node is **about** to generate. This implies that it is likely that, at some given point in time, many interfaces can be in use, with each one potentially serving a different flow.

5.1.1 Data Flows

It is reasonable to assume that a MN will generate many different data flows[11]. Let say that a user is moving around with his or her PDA. The PDA constantly receives information about the stock market from server.company.com; this flow is approximately 5kbps and of urgent importance, since missed or delayed data can led to misinterpretation of the whole flow and hence misinformation about the market. Once in a while this user initiates a VoIP session with one of his or her customers to inform them about a large potential for profit or loss. Sporadically this user accesses the web to find extra information, look at news headlines, and to execute trades.

In this simple scenario it is straightforward to identify three data flows:

- mobile node ↔server.company.com
- mobile node ↔customer's cell phone
- mobile node ↔ generic web server

Each of these has radically different needs. We can easily describe these flows according to a set of parameters, as shown in the following table:

Destination	Throughput	Reliability	Fixed IP src address
server.company.com	5kbps	urgent	yes
customer's cell phone	5-13kbps	relatively high	yes
generic web server	bursts of data	not urgent	not necessary

5.1.2 Interface specification

Referring to the overview of different wireless technologies described in section 1.2, a similar table can be made for the interfaces:

	Throughput	Reliability
WLAN 802.11b	high	low
GPRS	low	high
IrLAN	low	low

The parameters we chose to describe the interfaces are simply to give an idea of our approach, for example we could to add other parameters such as cost of the link, billing, power consumption, probability of availability, etc.

5.1.3 Binding

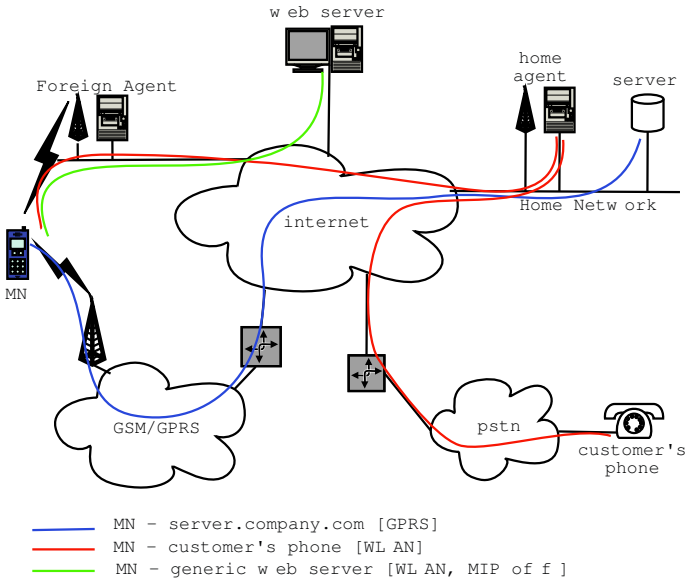
Once data flows are identified and weights have been assigned to each interface's parameters, then binding each flow to an interface is straight forward. The interface chosen for a given flow will be the one that best fulfills its requirements.

Referring to the previous example, the flow mobile node↔server.company.com will be assigned to the GPRS link, this prevents any handoffs, since the link is always available. This was done because of the importance of the data carried. However, it comes at a high cost. Therefore we might consider making a downward vertical handoff if the node is stationary **and** WLAN is available.[1]

The mobile node↔VoIP flow could be bound to the WLAN interface; data lost due to handoffs might be tolerated by the end users who will experience in

[1]For movement aware devices as the Badge we can determine that the node is stationary.

Figure 5.1: data driven policy management



the worst case a short break in this conversation **if** the losses are infrequent and of a sufficient shortly duration. If they are not, then we may have to trigger a upward vertical handoff to GPRS or even a circuit switched call via HSCSD data over GSM.

The last flow is particularly relevant since it shows that in some cases Mobile IP is not needed at all. Web surfing is sometimes one of these cases[2]; an average HTTP session needs to exchange (get/receive) only few messages, and because it takes only a few seconds to load most web pages we might use the temporary IP address for the HTTP traffic. Changing the IP address during a page loading in many cases does not disturb the end users, as they will experience only a delay in the page loading; this because the HTTP get / receive sequence can be re-initiated[3] if handoffs are infrequent. Moreover the probability that handoffs occurs during the sequence is fairly low.

5.2 Link driven

Link driven policy management does not take into account data flows: at any time only one interface is chosen to be optimal, i.e. over this interface all the

[2]HUT's Dynamic, a mobile IP implementation from Helsinki University of Technology (HUT) introduced this concept.

[3]However SSL/TLS will not tolerate a change in the source address, since the protocol will have to re-key, thus re-initiation may not be possible or may be very slow.

incomings/outgoings packet will be routed. The decision is based on a set of parameters that describe the interfaces, but which are **not** link dependent. As we discussed earlier in section 4.4, choosing simply based on SNR is not a suitable criteria since comparing SNR of different technology does not give any useful information.

Therefore it is important to select a suitable set of parameters that can describe a (general) wireless link. The following table shows the parameters we have chosen and the weights of these parameters for the case of the pair (WLAN and GPRS) interfaces we have utilized:

	WLAN 802.11b	GPRS link
max link speed (Sp)	10 Mbps $\rightarrow Sp = 7$	170kbs $\rightarrow Sp = 4$
reliability (Re)	medium $\rightarrow Re = 5$	high $\rightarrow Re = 10$
power utilization (Pu)	x mAh	x mAh
billing	traffic / packets / free	traffic / packets / free
cost	none/ x	none/ x

We can discern two classes of parameters: *internal* parameters that describe the link characteristic,and *external* parameters related to the service provided via this interface.

Internal parameters are: maximum link speed and reliability; external parameters are: billing, cost, and power utilization. While internal parameters are identical for a class of wireless links, external parameters might vary with the service provider and even by location and/or time of day.

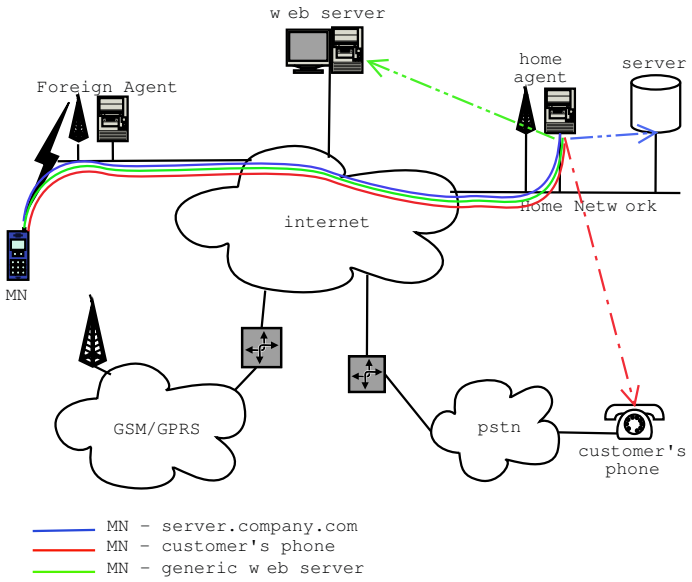
Together with this set of parameter values a trigger is specified for each interface, so that at any time it is possible to poll the interface and check whether it has connectivity or not. Triggers are defined as described in section 4.3, i.e. a trigger for WLAN might be:

$(SNR > T_{SNR})$ or $(SNR_{old} < T_{SNR} \wedge SNR_{new} > T_{SNR})$

5.2.1 User preferences

Which interface is optimal at a given time is evaluated relative to the user's preferences, the user interacts only with the external parameters, specifying for instance what he is willing to pay. User preferences can be redefined at any time, to provide flexibility to the policies and to allow the user to adapt the behavior to their needs and desires in their current context. For example a stingy, or rather smart, user does not want to pay for WLAN access as a general policy and prefers VoIP over making a circuit switched GSM call since the former is cheaper; when such a user is about to initiate a call, he/she might be willing to pay for WLAN access for the duration of the conversation because

Figure 5.2: link driven policy management



of convenience. The user simply changes his/her preferences according to their current desires and changes them back when the call is over.

5.2.2 Decision making

Decision making begins by looking at the internal parameters, thus the available interfaces with connectivity are sorted using the following criteria: {Power utilization can also be included in the formula}

$\alpha(Sp_i - Sp_j) + \beta(Re_i - Re_j) \geq 0 \Rightarrow If_i \geq If_j$

Where (α, β) are weights assigned a priori, and If_i and If_j the interfaces to be compared. After the interfaces are sorted, external parameters are taken into account and compared to the user's preferences. If the external parameters of the first interface in the list fits the user's preferences then that interface is selected, otherwise the second interface in the list is considered, and so on.

5.3 Rationale for focusing on link driven policies

In the remainder of this report we will discuss, implement, and evaluate only link driven policy management. Data driven policy will be left for future research; however, some of the results we obtained while testing the prototype are relevant to both methods.

One could expect that data driven is always best; since as a general principle lower level wireless layers provide high bandwidth, but serve smaller areas, thus for example every time the MN enters or leaves the WLAN's coverage area, it is forced to handoff. Data driven policy would prevent many handoffs by assigning each flow to the highest wireless layer, i.e. the most widely available and thus most reliable, generally without any loss of performance as seen by the end user.

Nevertheless there are negative aspects of this approach, especially when dealing with many flows: having several interfaces powered up at a any given time can be detrimental to long battery life. Measuring power consumption with both the interfaces up and comparing this to the power needed when turning on one interface weighted by the probability of doing so, one can estimate which policy is better in terms of battery life.

It is also important to notice that the two approaches are complementary. Data driven policies must use link driven policies to select a network among those available on an interface, so choosing to first implement link driven policy seemed reasonable.

Chapter 6

Implementation

In this chapter we will give an overview of the Mobile IP platform we developed. Architectural and conceptual issues will be treated. The latest version of the source code can be found at http://wintermute.hopto.org/resources/mip2.tar.gz

6.1 Background and previous work

When this research project began only two open source implementations of the standard Mobile IP protocol were available for the Linux platform: Mosquito IP from Stanford University[1] and Dynamics from Helsinki University of Technology[2]. The former had not been updated since 1999 and it was only available for Linux kernels up to version 2.2.5; unfortunately the Smart Badge runs a newer kernel version and the Mosquito implementation proved to be unusable. Although the latter (Dynamics) is entirely implemented as a user space daemon and thus could be run on the Smart Badge, it lacks co-located care of address support, which is a strict requirement for our model. Co-located care of address support was required for two main reason: first, since we do not have access to operators gateway of the GPRS network we used, hence we cannot set up a foreign agent there; second, using co-located care of address makes our system more usable, since we can connect to any open WLAN network.

Another necessary feature needed, but unavailable in both implementation was IP-in-UDP tunneling for NAT-traversal.[13] This was necessary since the particular GPRS network we used was NAT'd.

The solution adopted was to re-implement Mobile IP for Linux in a simplified version, to serve our purpose only. Nevertheless one of the goals of our implementation was to maintain full compatibility with Mobile IP, so that our

[1] http://mosquitonet.stanford.edu/mip/
[2] http://www.cs.hut.fi/Research/Dynamics/

mobile node is able to speak with any standard home agent.

6.2 Software structure

The software we implemented is a collection of shell scripts, kernel modules, and a user space daemon. It was implemented with a special eye for portability, since, as we mentioned earlier, other MIP implementations became outdated because of too strict dependence upon a certain version of the Linux kernel. For this reason we did not modify the kernel at all, rather new kernel modules were implemented to serve our purposes.

Apart from the shell scripts, whose main purpose is to set up the modules and adjust the routing table, the software can be logically divided in two parts:

1. the virtual interface (Mobile InterFace, MIF), and
2. the Mobile IP daemon.

The first is implemented as a kernel module, the latter is entirely in user space. The idea is to move as many components as possible into user space, and leave for the kernel module only those operations that cannot be performed in user space. The implications of this are discussed in section FIXME.

6.2.1 The Virtual Interface

The goal of the virtual interface is to hide the details of mobility from user space applications. Outgoing packets are sent to the MIF independently of the current location; moreover incoming packets always come from the MIF, so that applications have only to deal with this interface.

For outgoing packets the MIF plays an active role and processes outgoing packets differently based on the current location of the MN. There are two cases to consider:

* *Mobile node is Visiting a foreign network:* An actual interface is chosen for this outgoing packet, it is the user space daemon that selects the interface based on policies and link availability. The packet is encapsulated using IP-in-IP (or IP-in-UDP).

* *Mobile node is at home*: The packet is routed to the actual interface that has link connectivity to the home network. Encapsulation is **not** needed.

When Mobile IP is set up, another interface is brought up, to decapsulate packets tunneled from the home agent to the mobile node. Linux provides IP tunnel

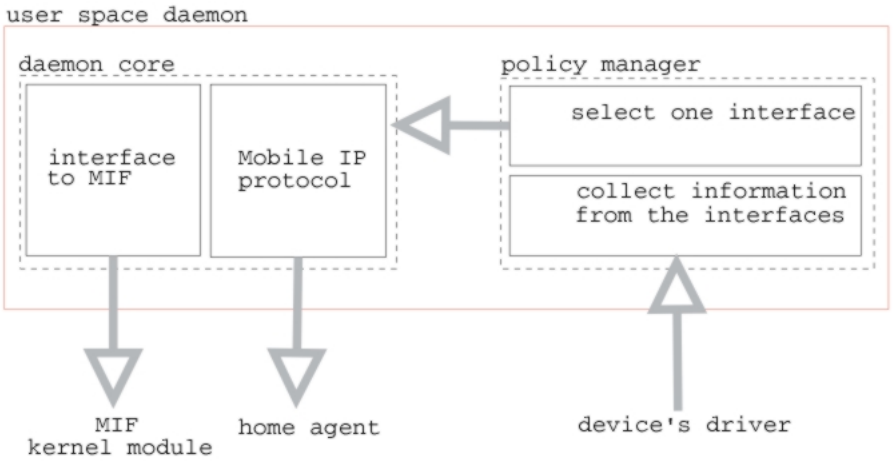


Figure 6.1: daemon structure

support either into the kernel or as a kernel module, depending on user preference at compile time. We did not compile IP tunneling in the kernel, rather we left it as kernel module that we slightly modified. For applications it is important that incoming packets are seen as coming from the MIF, if they were sent through this interface. Our modified IP-in-IP tunnel decapsulates incoming packets and sets the incoming device as if they were received from the MIF.

6.2.2 User Space Daemon

The user space daemon (MHD) has two functions: it is responsible for exchanging registration messages between the mobile node and home agent and it constantly checks the status of the available interfaces in order to select the current interface based on policies and link availability.

This logical distinction between these two functions is reflected in the daemon's structure, as shown in Figure 6.1. The daemon comprises a *daemon core* and the *Policy Manager*. We will examine policy management in the next section, since it is central to our work.

The daemon core can be further divided in two components, one handles the mobile IP protocol, i.e. communicates with the home agent to update the care-of-address when the MN moves. *As our MN must be able to speak Mobile IP,*

all the protocol's messages must be correctly handled. Thus registration with the HA, authentication, registration of care of address are all in the format described in RFC 3344. For further details refer to this RFC[4]. We tested this part of the daemon with the home agent implementation by the Mosquito group of Stanford University and all the messages proved to be correctly handled. The other component communicates with the virtual interface, and each time the selected interface changes it sends an update to the virtual interface.

The method adopted to check the status of each interface is polling: it proved to be the easiest method since it does not require any modification of the network driver to generate some kind of soft interrupt when the interface could provide or not provide connectivity. This gives great flexibility to our software; if we add a new interface to our mobile device, all what we need to do is to write software that reads the parameters we wish to monitor, e.g. typically entries in the /proc file-system, without needing to understand how the device works or needing to modify its driver. Nevertheless polling introduces some latency: our system cannot respond to changes faster that the time interval between sequential measurements, even if the time spent reading the value(s) from /proc or sending/receiving an ioctl plus the time spent computing the triggering algorithm were zero. However, we estimate that these operations take only a few milliseconds, thus we can considered it as zero. For the two interfaces and the simple policies we have used, the polling plus the decision making latencies amount to approximately 20 ms. How this value affects the performance of the system could be an interesting subject for further investigation.

Every polling interval (delta) the daemon executes the following steps:

- Poll the Policy Manager.
- If the status of the interfaces is unchanged, there is nothing more to do.
- If the status changed, ask the Policy Manager to select one interface.
- The Policy Manager selects one interface among those that have connectivity and passes this choice to the daemon's core.
- The daemon's core updates the binding with the HA when the current interface is changed and sends an ioctl to the MIF, to update the current interface.

6.3 The Policy Manager

In order to understand how the policy manager works, and how it exchanges informations with the daemon core, it might be useful to look at its header file.

File: policy.h

```
#ifndef _POLICY_H
#define _POLICY_H


#include <asm/types.h>  // for typedef _u32


/*
   check the status of the current interface
        return value:
        1-> working
        0-> trigger handoff !
 */
int trigger();

/*
 *  select an interface,
 *  pre-triggering ...
 *      The function performs the
 *      following tasks:
 *      -select an interface
 *      -turn it on
 *
 *      information about the selected
 *      interface are returned by:
 *      __u32   get_current()
 *      char*   get_current_ifname()
 *
 */
int select_interface();

/*
 *  get the name of the current interface
 */
char* get_current_ifname();


/*
 * get the IP address of the current interface
 */
__u32  get_current();

/*
 *  start_up interfaces
 *  read user's preferences
 */
int policy_init();

#endif /* _POLICY_H */
```

Figure 6.2: policy manager header file

As it can be seen in Figure 6.2, the interface to the rest of the software is straightforward, the idea is to separate the policy manager code from the daemon core. This makes it possible to redefine the policy manager at any time; in the same way a redefinition of the daemon core would be possible without rewriting the policy manager code.

The daemon core periodically calls the function *trigger()* to check whether the status of the interfaces has changed. The delay between two sequential calls to the *trigger()* function is the polling time we described in the previous section. It is important to underline that it is only the policy manager that keeps track of the status of the interfaces; the daemon core polls the policy manager to retrieve each interface's information, calling either *get_current_if_name()* or *get_current()*, and consequently sends an update to the virtual interface. This makes the core of the system independent from the type and the number of supported wireless devices; if new device is to be added, only the policy manager needs to be modified.

6.3.1 Interface selection

For each interface the policy manager stores a set of parameters that describe its external parameters (as defined in section 5.2). Decision making should be based upon these parameters, figure 6.3 shows how the interfaces are declared.

Since our system utilizes only two interfaces, the definition of the policies can be further simplified. During testing we found that the time spent by our system to turn on the GPRS card, send the configuration parameters and attach to the network was between 5 and 6 seconds; because this value was so high we were forced to redefine the policies so that the GPRS connection is brought up at initialization time, and it remains always up, even when the node is transmitting over WLAN. This is not in contradiction with our policy definition, since billing in GPRS is done based on the amount of traffic, and having the interface up without sending any traffic does not incur any extra cost (but does use battery power). Figure 6.4 shows how our policy manager selects the current interface: WLAN is always the preferred interface, when available; packets are routed through the GPRS card only when no WLANs networks are in range.

6.4 IP-in-UDP encapsulation

IP-in-UDP encapsulation is implemented in the virtual interface. Before sending a packet the software check the address of the current interface: if the address is public the packet is sent to the home agent using standard IP-in-IP encapsulation; if the address belongs to a private network, the packet is encapsulated

File: interfaces.h

```
/*
 * interfaces.h
 * interfaces definition
 */

#ifndef _INTERFACES_H
#define _INTERFACES_H

#include <linux/if.h>
#include <linux/wireless.h>

/* defines for cost_type */
#define NONE    0
#define TIME    1
#define TRAFFIC 2


#define         ifname  if_req.ifr_ifrn.ifrn_name /* interface name */

struct _internal  {

        struct          ifreq   if_req;
        unsigned        speed;    /* in kbps */
        unsigned        reliable; /* scale 1..10 */
        unsigned        power_consumption; /* scale 1..10 */


};

struct _external {
        unsigned        cost;
        /*
           if cost_type == TIME -> cost = $/min
           if cost_type == TRAFFIC -> cost = $/kbps
         */

        int             cost_type;
};

struct wlan {
        int                     active; /*status*/

        struct _internal        internal;
        struct _external        external;

        /* wlan specific */
        char                    essid[IW_ESSID_MAX_SIZE+1];
        struct iw_quality       qual;
        struct iw_freq          freq;  /* channel */
};

struct gprs {
        int                     active;/*status*/

        struct _internal        internal;
```

Figure 6.3: interfaces specification

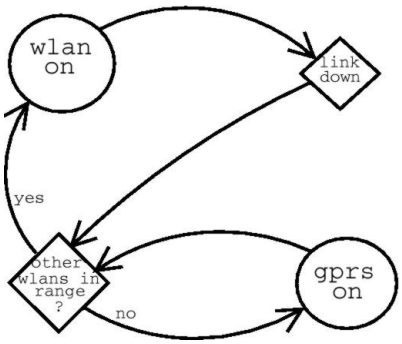


Figure 6.4: Simple policy

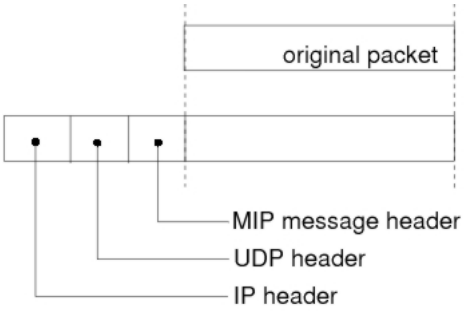


Figure 6.5: IP in UDP encapsulation

in UDP and sent to the home agent.

IP-in-UDP encapsulation is performed according to RFC 3519 [13]; figure 6.5 shows how it works: an extra header is appended to the original packet, then the resulting packet is sent in a UDP datagram.

The format of the extra header is shown in figure 6.6; the header is necessary since after encapsulation, the resulting datagrams are sent to the home agent, to port 434, which is the same port where registration request and reply are sent to. Thus the header is needed to distinguish between the messages; the type value for UDP encapsulation is fixed to 4. The field "next header" describes which protocol follows, in our case it is IP, i.e. the IP header of the original message.

As we said earlier the only reason for using IP-in-UDP encapsulation is for NAT traversal; supporting both types of encapsulation makes things more complex, so one might wonder why not use only IP-in-UDP instead. The reason is that this would increase the length of control data in the packets when it is not needed, since IP-in-UDP encapsulation adds 16 bytes more than IP-in-IP (i.e. 8 bytes are used for the UDP header and 8 bytes for the Mobile IP message

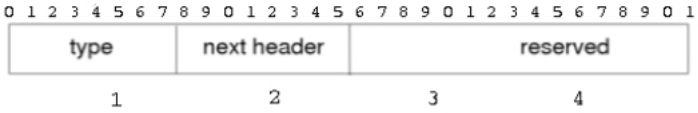


Figure 6.6: MIP message header

header).

6.5 General implementation issues

Implementing our software we ran into various problems. In this section we will describe some of the most relevant as well as the solutions that were adopted.

6.5.1 Device Drivers

In Linux, as in any OS, to make a device work the first step is to load into the OS core a piece of software, the device driver, that interfaces the physical device with the system.

The first problem we run into was that for the Sony/Ericsson GC75 GPRS card there was no Linux driver available. The theory of operation for most GPRS PCMCIA cards is the following: once the card attaches to the network, the system sees the GPRS connection as it were a Point-to-Point connection, and PPP is used to exchange packets. In reality PPP is used just to exchange packets between the system and the card, and the card uses the GPRS protocol over the radio link. We modified a driver for serial PCMCIA cards, and used the 'chat' program to set the configuration parameters needed to attach to the GRPS network. When a GPRS connection is established the system sees a PPP connection to the Internet.

In Linux, Wireless LAN cards are fully supported, moreover Linux provides an abstraction to communicate with the card, which is a software layer between the card's driver and the rest of the system. This software layer, called "the wireless tools"[16], permits us to read some values from the card in a way which is independent from the type of card used; parameters such as signal strength, noise level, and network's ESSID (network name) can be read in the /proc file system or by mean of an ioctl system call. The software also allows us to perform an active scan (i.e. look for other WLAN basestations) in an easy way, when the device driver supports this functionality. The device drivers for both the cards we tested are based on the Orinoco ver.13d device driver code, which does not natively support active scanning, so we had to patch this driver to add this

functionality.

The Prism-II card caused a bit of trouble because the device driver is poorly written. Because at power on the card lacks of a firmware, the device driver must upload one at boot time. The device driver we found on the net[3] accomplished this initial firmware loading without problem. However, the card manifested various malfunctions that we could not completely solve. One of these was that after the card looses link connectivity to a network it was connected to, e.g. because of low SNR, the values reported for signal strength and noise were incorrect, leading to errors in our software that uses these values to compute the SNR and trigger the handoff. We modified the driver to force it to report a signal level equal to zero when there is no link connectivity; even if this is not the real signal strength, it worked well enough for our software and also, once it is not possible to establish link connectivity, there is no use in reading a specific value since it is not possible to use this basestation since no link can be estabilished. This can be seen in the plots in figure4.6, where the signal level abruptly falls to 0dBm.

6.5.2 Link probes

Having L2 connectivity or even L3 connectivity, does not necessarily mean that it is possible to send and receive packets over the link. During testing of our software we ran into this problem: on the WLAN interface everything was properly set up, but no traffic was seen going through it. This might be due to many reasons, some of these are out of the scope of this work; below we will examine those within our scope.

Once the WLAN interface is brought up and an IP address obtained via DHCP, the software assumes that everything is working fine. Following probes of the interface L2 trigger confirms connectivity so that no handoff is executed; however, if the mobile node was not able to successfully send a registration request to the home agent, then the result is that the mobile node is unreachable.

Another scenario we encountered is when the MN is able to register, but cannot send/receive any packet. The MN becomes reachable again only when it moves to another physical location, since this causes an handoff triggered by a change in L2 connectivity.

One of the reasons is that some wireless networks make use of proprietary authentication systems, that take place **after** DHCP. Since the node is not allowed to send/receive any packets until the authentication procedure has been carried out successfully, our software will fail if the authentication fails, since once an IP address is obtained for the interface and L2 connectivity is up, no

[3]http://www.handhelds.org/~nils/socket-cf-wlan.html

additional operations are performed to test the **usability** of the link.

To overcome these problems the software should be slightly redesigned, introducing a traffic monitor in the MIF module. The idea is to monitor the traffic and when no traffic is detected the MN should send a probe packet, for example addressed to an echo server. If no answer is received then the MN should migrate to another network, since the current network is not utilizable. We did not include link probes in our software; we leave it as a possible subject for further research since we think it would represent a useful improvement to our software.

Chapter 7

Findings

7.1 Measured performances

We set up a test to investigate how well our system performs with horizontal handoffs. The handoff was made between two basestations, one connected to the home network and the other to a private network; the private network was connected to the home agent that in this experiment acts also as a router / NAT box between the two networks.

These are the IP addresses for our test:

1. home agent: 130.237.15.208;
2. home address: 130.237.15.16; and
3. foreign network: 192.168.1.0/24.

During the test the mobile node was generating a flow of UDP packets at a rate of roughly one packet per 100ms; each packet had a payload of only 20 bytes, since the size is not really important in the estimation of the handoff time.[1] Figure 7.1 was obtained using TCPdump on a computer connected to the home network: it is possible to see the rate of the flow and that no encapsulation takes place, since the mobile node is at the home network. The last packet seen on the home network is packet number 753[2], after the mobile node starts the handoff procedure.

Figure 7.2 is a dump from the private network 192.168.1.0/24. The first packets arrives at time 21:13:41.446977, this is roughly one second after the last

[1] if the size of the original packet plus the size of the extra IP (UDP) header is below the MTU, and thus fragmentation does not occur; otherwise extra packet loss can occur.

[2] Packet 751 has nothing to do with the test; it was sent by the destination since the port we addressed the UDP traffic to was closed.

packet seen on the home network (time-stamp 21:13:40.414196); but the first useful packet is packet number 21 (time-stamp 21.13.45.207659). The handoff latency time is therefore 4.8 seconds, which is much higher than the expected value and could be disruptive for many connections.

Notice that the packets sent through the private network are encapsulated with IP-in-UDP and the filter we used to read the dump[3] properly recognized them as Mobile IP packets.

7.2 Consideration

Looking closer at the dump shown in figure 7.2 we can identify the different phases of the handoff procedure and the relative times.

The first two packets are Ethernet frames that the filter could not recognize. They are probably part of the scan phase. We did not control the scan, which was delegated to the wireless tool for Linux. Héctor Velayos in [10] has described a mechanism to reduce scanning time that could be adopted in our system.

During the scan phase the mobile node cannot send packets but it is able to receive. Since in our model the scan phase is triggered before L2 connectivity is lost, the mobile node will be able to receive traffic from the correspondent nodes during this phase.

The DHCP phase, from #4 to #12, significanlty affects the total time. It takes about 2.5 seconds from the first BOOTP request to the first Mobile IP registration request. DHCP time could be substantially reduced following the method described by J.O.Vatn in [15]. For example the ARP request issued in #5 by the server could be removed assigning a previuosly polled IP addresses to mobile nodes by the DHCP server, thus the ICMP ping request after the address has been assigned could be removed.

The Mobile IP registration phase is rather fast, it takes only 30 milliseconds to register with the home agent. Notice that the first registration request is denied because the mobile node and the home agent need to synchronize their clocks in order to successfully exchange messages. This happens only for the first registration request, the following registrations require only two messages.

7.3 Performance in vertical handoffs between GPRS and IEEE 802.11b

A vertical handoff, either to upper or lower layers, currently takes less time than horizontal handoffs, since the GPRS connection is kept constantly available.

[3]Ethereal, http://ethereal.sf.net

	time	source	destination	protocol	description
743	21:13:39.424304	130.237.15.216	213.100.40.206	UDP	Source port: 1024 Destination port: echo
744	21:13:39.534238	130.237.15.216	213.100.40.206	UDP	Source port: 1024 Destination port: echo
745	21:13:39.644294	130.237.15.216	213.100.40.206	UDP	Source port: 1024 Destination port: echo
746	21:13:39.754227	130.237.15.216	213.100.40.206	UDP	Source port: 1024 Destination port: echo
747	21:13:39.864284	130.237.15.216	213.100.40.206	UDP	Source port: 1024 Destination port: echo
748	21:13:39.974216	130.237.15.216	213.100.40.206	UDP	Source port: 1024 Destination port: echo
749	21:13:40.084272	130.237.15.216	213.100.40.206	UDP	Source port: 1024 Destination port: echo
750	21:13:40.194205	130.237.15.216	213.100.40.206	UDP	Source port: 1024 Destination port: echo
751	21:13:40.197077	213.100.40.206	130.237.15.216	ICMP	Destination unreachable
752	21:13:40.304263	130.237.15.216	213.100.40.206	UDP	Source port: 1024 Destination port: echo
753	21:13:40.414196	130.237.15.216	213.100.40.206	UDP	Source port: 1024 Destination port: echo

Figure 7.1: horizontal handoff dump 1

	time	source	destination	protocol	description
2	21:13:41.446977	LucentTe_1e:d3:c0	01:a0:f8:f0:f0:04	0x8781	Ethernet II
3	21:13:42.717212	LucentTe_1e:d3:c0	01:a0:f8:f0:f0:04	0x8781	Ethernet II
4	21:13:43.732921	0.0.0.0	255.255.255.255	BOOTP	Boot Request from 00:60:1d:1e:d3:c0[Short Frame]
5	21:13:43.734634	XnetTech_18:35:04	Broadcast	ARP	Who has 192.168.1.2? Tell 192.168.1.1
6	21:13:44.312961	Intel_c3:7b:45	01:a0:f8:f0:f0:02	0x8781	Ethernet II
7	21:13:44.733861	XnetTech_18:35:04	Broadcast	ARP	Who has 192.168.1.2? Tell 192.168.1.1
8	21:13:44.735199	192.168.1.1	192.168.1.2	BOOTP	Boot Reply[Short Frame]
9	21:13:44.739643	0.0.0.0	255.255.255.255	BOOTP	Boot Request from 00:60:1d:1e:d3:c0[Short Frame]
10	21:13:44.744275	192.168.1.1	192.168.1.2	BOOTP	Boot Reply[Short Frame]
11	21:13:45.058096	LucentTe_1e:d3:c0	Broadcast	ARP	Who has 130.237.15.208? Tell 192.168.1.2
12	21:13:45.058245	192.168.1.1	192.168.1.2	ICMP	Echo (ping) request
13	21:13:45.058343	XnetTech_18:35:04	LucentTe_1e:d3:c0	ARP	130.237.15.208 is at 00:05:1c:18:35:04
14	21:13:45.062612	LucentTe_1e:d3:c0	Broadcast	ARP	Who has 192.168.1.1? Tell 192.168.1.2
15	21:13:45.062699	XnetTech_18:35:04	LucentTe_1e:d3:c0	ARP	192.168.1.1 is at 00:05:1c:18:35:04
16	21:13:45.063354	192.168.1.2	130.237.15.208	MobileIP	Reg Request: HAddr=130.237.15.216 COA=192.168.1.2
17	21:13:45.065148	192.168.1.2	192.168.1.1	ICMP	Echo (ping) reply
18	21:13:45.065304	192.168.1.1	192.168.1.2	MobileIP	Reg Reply: HAddr=130.237.15.216, Code=133
19	21:13:45.092057	192.168.1.2	130.237.15.208	MobileIP	Reg Request: HAddr=130.237.15.216 COA=192.168.1.2
20	21:13:45.094445	192.168.1.1	192.168.1.2	MobileIP	Reg Reply: HAddr=130.237.15.216, Code=0
21	21:13:45.207659	192.168.1.2	130.237.15.208	MobileIP	
22	21:13:45.317565	192.168.1.2	130.237.15.208	MobileIP	
23	21:13:45.427543	192.168.1.2	130.237.15.208	MobileIP	
24	21:13:45.537539	192.168.1.2	130.237.15.208	MobileIP	
25	21:13:45.647521	192.168.1.2	130.237.15.208	MobileIP	
26	21:13:45.757506	192.168.1.2	130.237.15.208	MobileIP	
27	21:13:45.867480	192.168.1.2	130.237.15.208	MobileIP	
28	21:13:45.977462	192.168.1.2	130.237.15.208	MobileIP	

Figure 7.2: horizontal handoff dump2

Even while transmitting user traffic over GPRS, it is possible to perform over the WLAN for those phases that introduce high delays, i.e. network scan and DHCP.

In vertical upward handoffs the latency is the sum of the time required by the mobile node to realize that L2 connectivity is lost, which can be reduced using L1 triggers, as described in section 4.3, plus the time required for Mobile IP registration.

For vertical downward handoff the only latency is Mobile IP registration.

Under these conditions it becomes convenient, in the case of horizontal handoffs, to temporary triggers an upward vertical handoff and remain in the upper layer until the mobile node as obtained an IP address for the WLAN interface. However a problem with the GPRS card prevents us from testing this scenario and measuring the values for these latencies. Unfortunately the GPRS card requires so much power that we were not able to send packets through both the interfaces at the same time. When we tried to do so, the Badge could not provide the required power to the interfaces and therefore an interrupt was sent to the Linux kernel that put the system in sleep mode. From this state it was not possible to bring the Badge back to a working state without powering off the system or generating a hardware interrupt.

Chapter 8

Conclusions and Future work

8.1 Conclusions

This thesis work has represented for me a great learning opportunity. When I started to work on it, I knew C programming and Linux because I had taken courses in the university, but I never had the chance, and the time, to study them so deeply as this work required. Therefore I had to study them by myself, while writing this thesis. Sometimes I was annoyed by all the details of the programming and the difficulties in reading the Linux kernel; I wondered if there is any use in knowing so many details. But exactly by knowing these details, one understands what is possible to do in networking and what is not, why a protocol should be designed in a certain way, as well as why things should be simple and standardized. These are probably the most important lessons I have learned from this work.

8.2 The Prototype

Setting up the prototype and implementing the software represented a great deal of work in this thesis project: my hope is that the work I have done, regardless of the results I did or I did not achieve, can be useful for further research. Having a working prototype can move mobile IP investigations from something theoretic to more concrete testing and use; it was surprising for me that despite so many papers being published on the topic, almost no open source software was released.

Even though some commercial platforms exist, they do not help researchers

outside the companies who implemented these platforms; unfortunately the developers in these private companies are the same people that earlier implemented the first mobile IP platforms. As a consequence many open source projects were left with no developers working on them and have silently died. I hope that the software I have implemented, licensed under GPL[1], can help advance research in this field and could facilitate other students/researchers work.

The main achievements of the prototype can be summed up as follow:

1. The virtual interface, meant to hide the detail of mobility to the applications, can do packet by packet decision making, choosing which physical interface to send each packet on;

2. at any time the mobile node can receive packets from all the interfaces which are operating, independently from the interface selected for outgoing traffic. This is particularly beneficial during handoffs, since theoretically only outgoing traffic should be delayed;

3. the policy manager controls the virtual interface; the simple policy we implemented can be easily modified without any modification in the rest of the software.

8.3 Data driven policy management

The natural continuation of this work is to further implement and test data driven policy management. My opinion is that this could be done in two ways: redefining the socket's API or allowing the IP layer to make decisions for each flow. In the first approach applications could ask for certain link characteristics before initiating a flow, much as they can ask for specific Quality of Service (QoS). However, only specifically implemented applications could take advantage of this. The latter solution requires a simple modification of the software I have implemented: it would be sufficient to modify the MIF virtual interface driver so that it keeps state information for each flow and routes packets according to defined policies.

Given such an implementation it would be interesting to compare how much better data driven policy management performs when compared with link driven policy management.

[1] http://www.gnu.org/copyleft/gpl.html

8.4 Wireless diversity

The wide variety of wireless devices makes this project interesting, since we will always face different wireless technologies that coexist and a clear goal is to make them work together. My software currently works only with WLAN and GPRS; although support for many other technologies could be added. I have worked on adding IRDA support, using the IRLAN protocol to send datagram over the infrared link; Bluetooth could be the next obvious technology to add, since Bluez[2] provides Linux with a Bluetooth protocol stack.

8.5 Movement and location awareness

Movement detection could be used for fine-tuning handoff parameters. For example the dwell timer could be tuned according to the speed of the MN, while the accelerometer on the Badge could be used to estimate the speed. The test we described in section 4.6 should be repeated at different speeds in order to find which timer values gives the best performance at a given speed.

Movement awareness could also be used to trigger network scanning. When the node, while connected to the GPRS network, remains in a area where there is no WLAN coverage, it does not make sense to (rapidly) scan for WLAN networks until the node is on the move again. This would save computational resources and battery power. This and other applications of movement awareness should be investigated.

It would also be interesting to examine how location awareness could reduce handoff latencies. It is reasonable to imagine our nomad often moving within the same area, and sporadically away from it, e.g. the path from work to home, with the subway stations in between, and the restaurant where he/she usually has lunch. The mobile node could keep information about the presence of networks in these places, thus once the mobile node detects that it is in one of these locations, scanning could be avoided (or substantially reduced) and information collected during previous visits could be used instead.

There are many means to detect the current location. GPS is probably one of the best, if such a device is available; otherwise infrared beacons could be used, for example they could be placed at the entrance of a building, so that when the user crosses the door the mobile node is informed of the current location and maybe of the services available in that building. Cellular basestation IDs and WLAN AP MAC addresses could be used to recognize that you have returned to a previous area. Yet another solution could be to store (in the mobile node) network topology information in terms of adjacent networks, e.g.: from network

[2] http://bluez.sourceforge.net

A moving leads to net B or C; from network C movement leads to networks B, D or E; and so on. Each network in the 'map' is identified by the network name or IP address and the net-mask obtained during a previous visit. This information could be collected and even exchanged by mobile nodes, or sent to / received from some server.

The above section are some suggestions for further research, but many other aspects could be investigated. We hope the prototype we set up during this thesis project will be useful for this research.

Abbreviations and Acronyms

API	Application Program(ming) Interface
CDMA	Code Division Multiple Access
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
	CSMA/CD Carrier Sense Multiple Access/Collision Detection
DNS	Domain Name Server/Service
ETSI	European Telecommunications Standards Institute
	GPL General Public License
GPRS	General Packet Radio Service
GSM	General System for Mobile Communications
HTTP	Hyper Text Transfer Protocol
IEEE	Institute of Electrical & Electronics Engineers
IETF	Internet Engineering Task Force
IGMP	Internet Gateway Message Protocol
IP	Internet Protocol
IrDA	Infrared Data Association
IrLAN	Infrared Local Area Network
ISM	Industrial, Scientific, and Medical
LAN	Local Area Network
LCD	Liquid Crystal Display
PAN	Personal Area Network
PCMCIA	Personal Computer Memory Card International Association

PDA	Personal Digital Assistant
PPP	Point-to-Point Protocol
SNR	Signal-to-Noise Ratio
SOHO	Small Office/Home Office
SSL	Secure Socket Layer
TCP	Transport Control Protocol
TDMA	Time Division Multiple Access
TLS	Transaction Layer Security
UDP	User Datagram Protocol
VoIP	Voice over IP
WAN	Wide Area Network
WEP	Wired Equivalent Privacy
WLAN	Wireless Local Area Network

Bibliography

[1] Mark Stemm and Randy H.Katz, *Vertical Handoffs in wireless overlay networks*, ACM MONET 3(4), 1998.

[2] HP Labs, *Wifi and Bluetooth - interfarence issues*, White Paper, http://www.hp.com/sbso/wireless/images/WiFiBlue.pdf, 2002.

[3] Douglas E.Comer, *Internetworking with TCP/IP*, Volume 1, 1991.

[4] C.Perkins, *IP Mobility support,* RFC 3344, August 2002.

[5] C.Perkins, *Mobile IP: Design Principles and Practice*, Addison Wesley Longman, Reading, Mass., 1998.

[6] Bo Karlson, Aurelian Bria, Jonas Lind, Peter Lönnqvist, and Cristian Norlin, *Wireless Forecast: Scenarios of the Mobile World in 2015*, KTH Center for Wireless Systems, Sweden, 2003.

[7] Ye Min-hua, Liu Yu, and Zhang Hui-min, *The Mobile IP Handoff between hybrid networks,* 2002.

[8] Jean Tourilhes and Casey Carter, *P-handoff: a protocol for fine grained peer-to-peer vertical handoffs*, Mobile System and Service Laboratory, HP Lab, Palo Alto, 2002.

[9] IEEE Computer Society LAN MAN Standards Committee, *Wireless LAN medium access control (MAC) and physical layer (PHY) specification*, IEEE Standard 802.11-1997, The Institute of Electronics Engineers, New York, NY, USA, 1997.

[10] Héctor Velayos and Gunnar Karlsson, *Techniques to Reduce IEEE 802.11b MAC Layer Handover Time*, Laboratory for Communication Networks, IMIT KTH, Stockholm, 2003.

[11] Xhinua Zhao, Claude Castelluccia, and Mary Baker, *Flexible Network Support for Mobile Hosts*, Computer Science Departement, Stanford Univeristy, USA, 2004.

[12] M. Scott Corson, *A Triggered Interface*, Internet Draft 2002

[13] H. Levkowetz, *Mobile IP Traversal of Network Address Translation (NAT) Devices*, RFC 3519, 2003.

[14] Mary G. Baker, Xinhua Zhao, Stuart Cheshire, and Jonathan Stone, *Supporting Mobility in MosquitoNet,* Stanford University, 1996.

[15] Jon-Olov Vatn, *Supporting real-time service to mobile Internet hosts* Dissertation Proposal, Dept. of Microelectronics and Information Technology (IMIT), Royal Institute of Technology (KTH), Sweden, June 5, 2002.

[16] Jean Tourrilhes, Wireless Tools for Linux, http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html

IMIT/LCN 2004-01

www.kth.se