Vertical handover techniques over heterogeneous networks

Pangalos, Paul Anthony

Vertical Handover Techniques over Heterogeneous Networks

by

Paul Anthony Pangalos

A dissertation submitted to the University of London
in partial satisfaction of the requirements
for the degree of Doctor of Philosophy

Centre for Telecommunications Research
King's College London
University of London
Strand, London WC2R 2LS
September 2003

Vertical Handover Techniques over Heterogeneous Networks

Abstract of the dissertation

Vertical Handover Techniques over Heterogeneous Networks

by

Paul Anthony Pangalos

King's College London, University of London

Over the last decade, the Internet has grown significantly and undergone an unprecedented evolution. The introduction of a variety of new network technologies and the significant diversification of protocols and end devices has presented problems and challenges that have not been encountered before. The solution to one of these problems, mobility for real time traffic in heterogeneous environments, can be obtained by using the possibilities created by generic IP connectivity, while complying with application requirements. IP as it stands is the ideal choice for "gluing" together heterogeneous wireless environments and hence providing a generic inter-working platform. This thesis, presents the research behind a vision for efficient heterogeneous Internet connectivity for mobile devices and networks; focusing on vertical handovers. The major contribution is to provide real experience of the proposed solutions through their practical realisation on a test-bed consisting of different wireless mediums and associated networks. These solutions aim to provide advanced schemes and mechanisms for inter-working between various protocols assisted by higher layers, as well as enhancements to applications at the sender and receiver.

Inter-working schemes and mechanisms pose a challenge to the current protocols such as the session initiation protocol (SIP) and mobile IP (MIP), because by nature these protocols are designed to perform tasks in a highly efficient manner and independent of each other. The degree of inter-working between protocols is classified into two categories: loose coupling and tight coupling. These coupling techniques are designed and tested through inter-working between MIP and SIP to deal with terminal mobility in both planned and unplanned handover scenarios. The testing reveals weaknesses in the loose coupling approach, while tight coupling results in a trade off between performance and program complexity. The tight coupling technique is also used to design, implement and test a handover technique for a new type of mobility called session mobility. This technique addresses three fundamental challenges posed by this

type of mobility namely: terminal discovery, signalling and mobility support. Furthermore, this research discovers the importance of application support in a heterogeneous environment. Specifically, it demonstrates the significant performance benefits given by media adaptation in the context of vertical handovers, and the advantages of coupling such applications with SIP.

Today, vertical handover protocols and mechanisms are not widely deployed in the Internet despite the availability of a wide range of solutions. One of the main reasons for this is the diverse nature of mobility mechanisms and protocols implemented in end devices. This diversity, combined with the lack of unified inter-working mechanisms, often creates an unacceptable level of incompatibility between terminals and thus hinders widespread deployment. We believe and hope that the proposed inter-working of mobility mechanisms, application enhancements, and most of all the practical realisation of these solutions, will accelerate the widespread development of vertical handovers and make them an integral part of the Internet infrastructure of the future.

Acknowledgments

Have you heard about the Chinese bamboo tree? During the first four years of its life, it grows only a few inches. Then in the fifth year, it grows 90 feet in just 5 weeks. Now the real question is - did it grow 90 feet in 5 weeks - or 5 years? The answer is - 5 years! You see, if at any time during those first 4 years you would stopped watering and fertilising it, it would have died. My PhD course was much the same. It survived on levels and arrived at in seasons. You stay on one level and learn its lessons until you become qualified to move to the next. Completing a PhD is truly a marathon event and I would not have been able to complete this journey without God who sustained and directed me through the best and toughest times of my life.

I must first express my gratitude towards my supervisor, Professor Hamid Aghvami. His leadership, support, attention, hard work and encouragement have set an example I hope to match some day.

My family are also very special people and I owe them a special dept of gratitude. They have, more than anyone else, been the reason I have been able to get this far. Words can not express my gratitude to my parents, Andreas and Nadia, for their love, support and prayers they provided through my entire life. You have instilled in me confidence and a drive for pursuing my PhD by teaching me the value of hard work and how to overcome life's disappointments. Thanks Mum, Dad, Natalie and Angelos. I love you all.

I could have not completed my thesis on time without the support, love and encouragement of my wife Louise. Louise has completely changed my life over the last year. Her love, honesty, intelligence, goodness, humour and beauty is a continual miracle to me. She has supported me in hundreds of ways throughout the development and writing of this thesis. I know it has been a burden with me being in University these many years. Especially the last year I hear you on the phone with our friends "No, we are not having a barbeque this weekend, Paul is writing his thesis". My wife and her parents, Andrew and Peggie Ray, never failed in their support although they must have wondered if I would ever finish my PhD.

Very special thanks go to my uncle Andreas Georgiou for his constant support in the past seven years. He has greatly influenced me and given me his attention, advice and guidance on many practical things in life. He has been very generous in sharing his knowledge and ideas.

I also thank some of my fellow PhD students: Stephane Antoine, Adil Jahan, Piyush Khengar, Nikos Georganopoulos, Andrej Mihailovic, Mischa Dohler, Vasilis Friderikos, Thanos Gkelias, Kostas Boukis, Robert Rümmler, Stan Wong, Bilal Rassool, Oliver Holland, Francesco Ostuni, Nikolas Olaziregi and Lin Wang. They each helped make my time in the PhD program more fun and interesting. Special thanks go to Dave Wisely and Louise Burness who helped me get started with my PhD as well as editing and giving feedback on my work. Lastly, I would like to thank John Pearson, whose editing made the thesis writing flow much more smoothly. Several other people were supportive in helping me finish this thesis. To those individuals I neglected to mention here by name, I still offer my deepest thanks.

In conclusion, I recognise that this research would not have been possible without the financial support of the Engineering and Physical Sciences Research Council (EPSRC) and BTexcact Technologies. I express my gratitude to those agencies.

Let God direct you

(Isaiah 42:16).

You don't need to know all the answers. You just need to know that you've fully obeyed the last instruction God gave you. Once you know that – just drive and let God direct! That's what His Word means when it says, '...along unfamiliar paths I will guide them; I will turn the darkness into light before them and make the rough places smooth. These are the things I will do; I will not forsake them' (Isaiah 42:16).

Dedication

To my parents

Andreas and Nadia Pangalos

without whose sacrifices none of this would be possible

and to my grandparents, who always wanted a doctor in the family

Charalambos and Christala Varoshiotou

Charalambos and Penelopy Pangalos

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Description
3G	Third Generation
3GPP	3rd Generation Partnership Project
4G	Fourth Generation
ASM	Agent Status Module
ATM	Asynchronous Transmission Mode
DMTF	Dual Tone Multi Frequency
DNS	Domain Name System
DSL	Digital Subscriber Line
DVB	Digital Video Broadcasting
FTP	File Transfer Protocol
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
H2E	Handovers in Heterogeneous Environments
HIPERLAN	High Performance Radio LAN
HTTP	Hypertext Transfer Protocol
HUT	Helsinki University of Technology
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISP	Internet Service Provider
JMF	Java Media Framework
MAI	Monitor Active Interface
MIP	Mobile IP
RAN	Radio Access Network
RAT	Robust Audio Tool
RTCP	RTP Control Protocol
RTP	Real Time Transport Protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SMTP	Simple Mail Transfer Protocol
TCP	Transmission Control Protocol
TDP	Terminal Description Protocol
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications Service
WLAN	Wireless Local Area Network

The task ahead of you is never as great as the power within you
"Be strong in the Lord and in his mighty power"
(Ephesians 6:10)

Chapter 1

Introduction

1.1 Motivation

Both first generation mobile (analogue) and second generation mobile (digital) systems were designed primary to offer a single service, i.e., speech. They also have a limited capability to offer low data rates over voice channels using conventional modems. However, third generation mobile systems are expected to offer high-quality multi-media (multi-rate services) to convergent networks of the fixed, cellular and satellite components. 4G is not market or technology driven, it is user driven and will have user controlled services [1]. That means that users are in control of their interactions. As George H. Heilmeier once said [2] *"I think the bottom line is to bring information-age capabilities to everyone, capabilities that enable people and their machines to access information anywhere, anytime, in any medium or combination of media in a cost-effective and secure manner."* To achieve this vision the following challenges must be met:

- Inter-working of fixed and mobile networks
- Mobility among heterogeneous access networks
- Mobile users and network agents
- Reconfigurable terminals and networks

The first two challenges can be realised by IP technology, the third one by agent technology and the fourth one by using middleware technology. The most important challenge is that of mobility among existing and emerging heterogeneous access networks, both public and private. The development of wireless and fixed network access technology is expanding and several new alternatives will be offered in the next few years from those providing broadband wireless hotspots (e.g., HIPERLAN 2) to those providing global coverage (e.g., UMTS, DVB) as well as fixed access such as xDSL. As a result, user demand for continuous connectivity is increasing irrespective of the type of interface or network they are using. Providing continuous connectivity in different types of networks with fixed and wireless segments is itself a challenging task. We believe that there are two fundamental challenges related to mobility that hamper the realisation of this vision.

- Heterogeneity: The Internet today consists of a large diversity of heterogeneous networks. The networks have very different and varying capabilities such as bandwidth, interface speed, edge-to-edge latency and connection types. The introduction of laptops and palmtops have also increased and so have the operating systems, providing a large diversity in the end system capabilities. This diversity combined with the network heterogeneity results in an undesirable level of incompatibility between both the networks and terminals. As a result interpolation of protocols, services, and technologies in this kind of environment remains a challenging task.

- Vertical handovers: It is fairly common today for users to have access to multiple Internet Service Providers (ISPs) and alternate between them. Some of the ISPs offer free low speed dialup connections (i.e. GSM, GPRS) while others high bandwidth, always online connections such as DSL and cable. However, in these cases users tend to only use one connection at a time. With the introduction of overlay networks and constant reduction of connection prices, this is expected to change. Users may wish to handover or use a combination of several networks, each of which is optimised for some particular service and also depending on factors such as time of day and current traffic load, in order to optimise cost versus quality. Vertical handovers is the link that would bridge these heterogeneous access technologies and enable users to inter-work efficiently between them.

This thesis, presents research behind a vision of efficient heterogeneous Internet connectivity for mobile devices and networks; this research focuses on vertical handovers. The essence of our approach is a collaborative partnership between applications, the operating system, and mobility protocols such as the session initiation protocol and mobile IP. Our major contribution is to provide prototype implementation and experimental evaluation of the proposed solutions through their practical realisation on a test-bed consisting of different wireless mediums and associated networks. These solutions aim to provide advanced techniques for inter-working between various protocols assisted by higher layers, as well as enhancements to applications at the sender and receiver terminals. We believe the proposed solutions will accelerate the widespread development of vertical handovers and make them an integral part of the Internet infrastructure of the future.

1.2 Incremental Development

It is essential to recognize the need for an incremental and evolutionary approach towards designing solutions in a large-scale environment such as the Internet in order to make an impact and influence its future evolution. An essential aspect of incremental development is to ensure interpolation between the suggested solutions and existing protocols and applications. Overlooking this constraint would hinder the widespread acceptance and deployment of any proposed solution. This research demonstrates that the flexibility found in protocols, such as SIP and Mobile IP, strongly discourages the need to design a new protocol framework to combat the problems observed in the heterogeneous networks. As a result, the solutions proposed in this thesis, can be successfully and efficiently integrated into the Internet in an incremental fashion.

1.3 Challenges of Vertical Handovers

In the following section, the vertical handover challenges are identified and described in more detailed. A vertical handover is defined as a handover between heterogeneous types of networks, i.e. networks that use different technologies (such as WLAN, GPRS, DVB and UMTS) under different administrative domains, enabling the user to utilise several of these networks in parallel. This type of handover is more complicated than a horizontal one (i.e. between the same access networks) and requires careful consideration and understanding in order to make the transition transparent to the user. From a high level perspective the handover should provide the following.

Firstly, it should support optimum triggering and network access selection. Discovering the right time to perform a handover is a key issue. Only through prompt reactions can a handover be initiated and performed efficiently. A handover can be triggered by the network (i.e. load balancing scenario) or it could also be user initiated (i.e. user requesting a handover to a specific network). Secondly, both real and non-real time applications should be supported. Applications have different performance requirements and need to be handled differently during handovers. Real time applications are sensitive to delay and packet loss requiring fast and reliable handovers, while TCP-based applications must maintain their end-points session by providing continuous connection support during handovers. In order to support vertical handovers, there is a need to exchange information in the form of signalling. This signalling may represent an important overhead in terms of bandwidth and processing requirements, having a significant impact on the performance of handovers. Therefore, a key requirement would be to minimise the signalling overhead as much as possible in order to improve and optimise the handover performance.

Another important aspect of vertical handovers is the need to provide reliable and transparent handovers. As users move between network boundaries, the handover mechanism should enable a transition which is transparent to the application layer with the only visible change to the user being due to the limitations of the new network interface. Furthermore, vertical handovers should provide a safeguard mechanism for failed handoffs or disconnections. Users might experience periods of disconnection that can either be short or long enough to cause severe problems at higher layers. This is also the case when a handover procedure is initiated but fails to complete. Handover fallback mechanisms should be present to support such scenarios. Finally, interoperation is an equally important challenge **Error! Reference source not found.**. Vertical handover mechanisms often make the assumption that some components of the networks can be modified (e.g., extending the sip proxy server software). However, this may not be true for all networks. Therefore, this is an important consideration that should be taken into account when designing solutions to support vertical handovers, that is to keep interoperation with available services and technologies we cannot modify. These challenges arise primarily because of the different characteristics of the network technologies. This thesis focuses on some of these problems in detail and solves them by using a combination of protocol inter-working techniques as well as enchantments to applications and protocols based on the following architecture.

1.4 System Architecture

In a heterogeneous environment users will be able to choose a combination of several networks to receive a particular service and preserve their session under various mobile conditions [4]. This section, presents the architecture, functional entities and protocols that this research is based on. This architecture provides mobility support over heterogeneous networks. Mobility refers to the ability of maintaining session based services (such as real-time services) when dealing with location change (terminal mobility), role change (personal mobility) and changes in devices in use (service mobility) [5]. Work on this architecture is performed in the H2E testbed (Handovers in Heterogeneous Environments) at the Centre of Telecommunication research in King's College London. The solution is based on enhancing a common IP core network with mobility mechanisms as well as adding the equivalent support in both the mobile and correspondent hosts residing in external networks. IP is used as the ideal choice for 'gluing' the various heterogeneous environments and hence providing the generic inter-working platform. The architecture objectives are best described in terms of some of the key challenging problems to which they offer solutions.

- Provide seamless terminal and session mobility across heterogeneous networks. The detection and configuration of new network connections are transparent to the user as is the selection of the best network in range. Seamless mobility means making the transition between different types of network transparent to the application with little or no data loss. The user is able to move between different networks with the only visible change being the limitations of the specific network interfaces. For example the WLAN network supports high bandwidth audio and video while higher overlay networks such as GPRS may only support low quality audio. If the user was part of a full audio and video session, moving to a higher overlay network will result in downgrading the audio stream and terminating the video session. Achieving seamless vertical handovers involves additional signalling in the form of configuration and handover messages. It also requires interaction and co-ordination between various mobility protocols in both the networks and end devices.

- Support for dynamic adaptation of content, tuning the content specifically for the device, the user, and the characteristics of the network connection, often considerably reducing latency and packet loss.

1.4.1 Models of Inter-working Architectures

There are several architectures using multiple heterogeneous networks. These models are illustrated in Figure 1-1 by three heterogeneous networks, network A, network B and network C. The main distinction between these models is the level of inter-working, also referred to as coupling, between them.

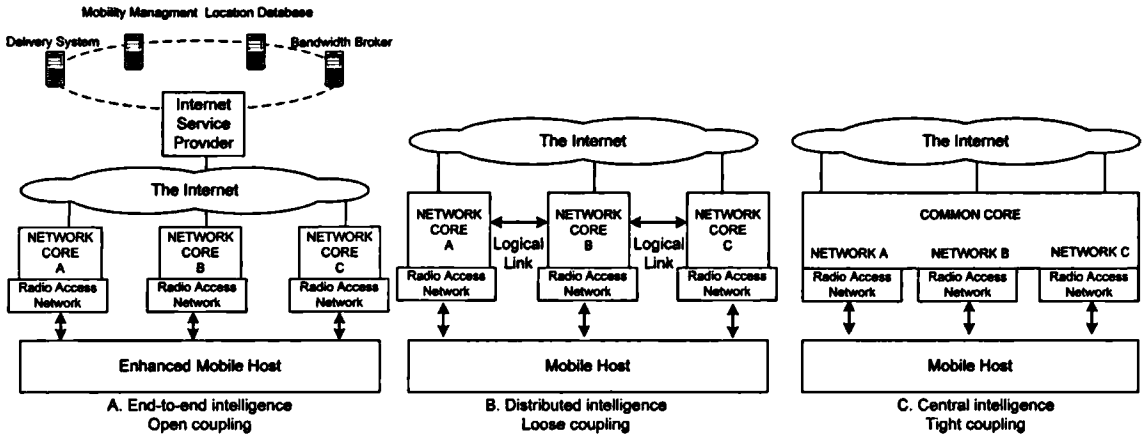


Figure 1-1: Inter-working of networks models

1.4.2 End-to-end Intelligence

In this model all the intelligence is placed in the end terminals and Internet Service Provider (ISP). This is known as open coupling. The networks operate in an independent way requiring no co-operation between them. Each network is regarded as a passive data pipe - bits go in one end and come out from the other end. However, another type of co-operation, a vertical one, is required between the ISP and the individual networks as well as between end devices. The ISP together with the co-operation of the end terminals, are responsible for the majority of functions such as selecting the appropriate access networks for a specific service and application, handover decisions, handling of vertical handovers and re-routing. For billing and authentication one customer database could be used and shared between the networks. This architecture puts users in control of their interactions. Based on user and network profiles, the optimal network is selected for the services requested and for handover initiation. The ISP would send or re-route the traffic across the Internet and over the selected access network to the mobile node. The key advantage of this architecture is that it requires only minor adjustments to existing networks with modifications focusing on billing mechanisms and service level agreements. However, the disadvantage is that connectivity between networks is across the internet, increasing service latency, lowering the security and not having quality of service guarantees.

1.4.3 Distributed Intelligence

In this model, we assume a distributed, non uniform intelligence across the whole network architecture also known as loose coupling. In this model, each network remains as it is while inter-working is achieved via a logical link connecting them together. The networks co-operate effectively in a complementary manner, using their strengths to complement each other and to counteract their weaknesses. Advantages are that networks can inter-work, as well as being function independently (removing the logical link), based on various service level agreements. Inter-domain mobility could also be supported through IETF mechanisms such as SIP and Mobile IP. The disadvantages are that this architecture requires additions as well as possible modifications to the existing networks functions and entities. These include enhancements to billing, authentication, resource and mobility management mechanisms.

1.4.4 Central Intelligence

This is also known as a tight coupling architecture. Inter-working between the networks is accomplished by using a common IP core, (e.g. an enhanced UMTS core.) In this model, all intelligence is concentrated in one network, where the majority of all network functions such as mobility management, resource management, billing and security are controlled and managed. Having the intelligence in the core presents a variety of benefits. It enables fast and seamless handovers to be achieved between the various heterogeneous networks (e.g. broadcast, cellular and WLAN) with the signalling overhead confined in the core network. Billing and security is also much simpler as it is managed by one administrative domain. A centrally intelligent network, however, is anything but simple. For a single application such as a two-way real time voice communication, it has a plethora of services, each with its own systems for operations, provisioning and maintenance. A major obstacle of this model is that it requires convergence, which requires a standardisation effort and business commitment to support it.

1.5 Chosen Architecture Model

A fundamental objective of the architecture is to build an environment that makes the heterogeneous networks transparent to the user. In addition, the main goal is to design a system architecture that is independent of the various other networks providing the user with the same benefits as if the inter-working was managed within one network.

These considerations lead to the following essential requirements in designing a **practically deployable, flexible** and **cost-effective** architecture.

- Reuse and extend of existing globally accepted IETF protocols whenever possible, extending and enhancing existing features to achieve new functionalities. Reusing existing protocols also leads to a direct **cost reduction** in innovation processes by limiting the 'reinvention of the wheel' phenomenon.
- Use of existing network entities within the network creating a **practically deployable** architecture.
- Handover support between heterogeneous networks, thus providing maximum **flexibility** in network configuration for operators. This flexibility allows network operators to offer innovative user services anytime, anywhere while users are on the move.

In order to choose the most appropriate architectural model, the following key practical constraints are also considered:

- Minimum alterations to network infrastructures: All networks operators have already invested extensively to upgrade their existing infrastructures – to 3G. As a result the architecture should not propose any major alterations to these networks.
- Ownership and control: Operators want to remain at all times in control of their own network without having any other operator interfering. The architecture must respect the sovereignty of networks involved.
- Security: Inter-working must be carried out in a secure and fair manner protecting the integrity of each network.
- Users' satisfaction: From the user's point of view, the inter-working must provide more, if not the same, benefits to the user, as if managed by one network operator.

As a conclusion, it is clear that model three suggests convergence of networks giving one operator more control over the others. As a result this model will suffer from the issue of ownership and control of customers and services. As mentioned earlier a further obstacle of this model is that it requires a standardisation effort and business commitment to support it.

Model two is not a suitable model either. Each network has its own infrastructure in terms of billing, handover and signalling procedures making it very difficult for existing networks to cooperate efficiently. In order to achieve this inter-working, extensive alterations to these networks are necessary. Model one, however, does not infringe on any of the practical constraints mentioned above, making it the most suitable candidate. In this model the inter-working is based on the Internet, distributing the intelligence between the end terminals and the ISP.

1.6 Handovers in Heterogeneous Environments (H2E) architecture

In this section, the functional entities of this architecture are introduced as well as the associated protocols used. The architecture as depicted in Figure 1-2 consists of three major building blocks: mobile hosts, core network and correspondent hosts.

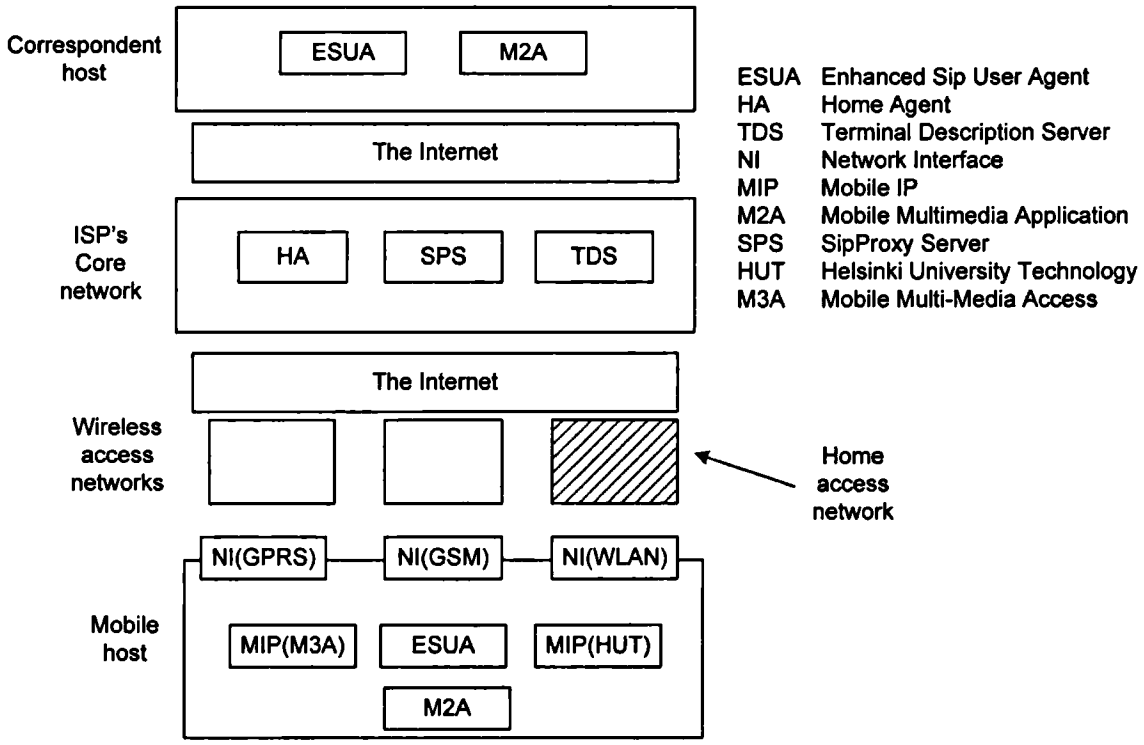


Figure 1-2: Heterogeneous system architecture

The mobile host is a multimode terminal equipped with multiple heterogeneous network interfaces. The wireless LAN is the default user interface, providing a 10Mb connection directly to the user's home network, whereas the GSM and GPRS interfaces offer less bandwidth over a greater coverage area. The mobile host is also equipped with two different versions of mobile IP agents. The M3A [6] developed by BTexact

technologies, is an extended version of Mobile IP that provides the network layer protocol with smart routing and proxy filtering decisions being taken at the Home Agent. The M3A was used as part of the experiments on unplanned handovers described in chapter 3 of the thesis. The mobile host is also equipped with a HUT Mobile IP agent, developed at Helsinki University of Technology, providing a flexible platform for extending the research on vertical handovers. Personal mobility of the user is managed by the SIP user agent on the basis of a personal identifier. SIP was further enhanced to enable it to inter-work with mobile IP, the Linux operating system and application layer. It also listens to handover trigger messages from the link layer, network layer and application layer to initiate handovers when required. A multimedia application, called the mobile multimedia application (M2A) was also developed based on the java media framework. The M2A is a sip enabled application that initiates, modifies and terminates sip enabled sessions. The M2A application inter-works with the session initiation protocol to provide seamless codec change during an ongoing session, in response to a vertical handover.

The core network, also known as the home network of the user, is responsible for the majority of mobility functions. It consists of three important functional entities. The home agent allows the user to move between IP domains, while continuing to be reachable for incoming requests and maintaining sessions across subnet changes. The second entity is the sip proxy server, which acts as a relay between the mobile hosts and the correspondent hosts. It is the component at which the user addresses his requests, and where it is currently registered. Lastly, the terminal description server is the key player for supporting the mobile host during movement to another terminal. To summarise, the components in the core network are primarily responsible for terminal, personal and session mobility functions.

Within the external network is the correspondent node (CN). The Internet connects the external network to the common core network. No mobility mechanisms are assumed within the external network. The correspondent host is equipped with an enhanced sip user agent and a mobile multimedia application (M2A).

The architecture presented shows an approach enabling the efficient use of multiple heterogeneous networks. It is based on the end-to-end intelligence model with all the intelligence placed in the end terminals and the independent Internet service provider core network. The vision of this architecture is that it inter-operates across various heterogeneous wireless and fixed networks proving the user with the same benefits as if the inter-working was managed within one network. This thesis provides solutions that aim to provide advanced schemes and mechanisms for inter-working between various protocols assisted by higher layers, as well as enhancements to applications to achieve this vision.

1.7 Contributions and Structure of Thesis

In this thesis, we analyse the problems posed by the heterogeneous environments and design, implement and evaluate solutions to them that result in significant improvements in vertical handovers. Because of the network and device heterogeneity, and the enormous variation in their characteristics and protocols, there is not a single solution to the problems. We recognise this and developed a suite of solutions and techniques to address a range of scenarios, that together comprehensively solve these problems. These solutions aim to provide advanced schemes and mechanisms of inter-working between various protocols assisted by higher layers as well as enhancements to applications at the sender and receiver nodes [7]. Furthermore, we give a real experience to the proposed solutions through a practical realisation in a test-bed consisting of different wireless mediums and associated networks. The components of this solution suite include the following applications and protocols.

1.7.1 The Mobile Multimedia Application (M2A)

This is a Java based application that optimises the transmission codecs during an ongoing session. The key idea here is the development of a collaborating partnership between the session initiation protocol (SIP) and the M2A application that allows seamless codec change in response to a vertical handover.

This principle of adaptation is not new. Prior work in adapting applications to perform well on mobile or bandwidth constrained hosts has included adaptation in file systems [8], web browsing [9] and more general approaches for client-server systems [10] and resource constraint applications [11]. The same concept can also be applied in vertical handovers where host mobility in heterogeneous environments naturally creates

situations in which network bandwidth variation is frequently encountered. In this work, we develop and refine the idea of media adaptation using the Java Media Framework and the session initiation protocol. Chapter 4 discusses the implementation in greater detail.

1.7.2 An Enhanced SIP User Agent (ESUA)

This is an enhancement to the basic functionality of the session initiation protocol with many additions that enable it to inter-work with mobile IP, the operating system and the application layer. In addition to this, the ESUA is designed to operate in a number of modes that deal with both terminal and session mobility. Each mode consists of its own set of signalling mechanisms and interacts with the appropriate protocols and underlying operating system as required.

This design principle, combined with the various inter-working functions, bridges the diverse nature of mobility mechanisms and protocols and accelerates the widespread development of vertical handovers making them an integral part of the Internet infrastructure of the future. The functionalities of the basic sip user agent implementation are described in Appendix A, section A.2 and A.3.

1.7.3 The Terminal Description Protocol

This protocol plays a key part in session mobility. Its structure resembles that of the session description protocol. Its functions include registration as well as discovery of software and hardware capabilities of terminals on Internet based networks. When combined with the session mobility mechanisms this protocol provides a powerful way by which sessions can be transferred from one terminal to another.

The rest of this thesis is organised as follows.

Chapter 2 discusses the background material in the area of mobility and heterogeneous networks giving an overview of the related protocols and associated extensions.

Chapter 3 describes the details of the loose inter-working coupling approach to terminal mobility with focus on unplanned vertical handovers. The results of several experiments are then presented identifying the weaknesses and limitations of this approach.

After understanding the precise reasons behind the poor performance results, the research is directed towards addressing these problems and describes the details of the proposed solutions to them. Finally the details of the tight coupling solution and the results obtained are presented, comparing it with the other loose coupling alternatives.

Chapter 4 follows on from the tight inter-working approach described in chapter 3 describing the design and implementation of a technique for handling planned vertical handovers. This is achieved by enabling mobile IP and SIP to communicate in a bidirectional manner as well as interacting with the local link layers and the operating system of the terminal. The mechanism employs link layer triggers to assist the mobile node in predicting imminent handovers, and enables it to re-negotiate and establish a new packet flow with its correspondent host prior to the handover. After discussing the mechanisms for both upward and downward vertical handovers, this chapter concludes by discussing various issues related to the tight inter-working between MIP and SIP.

Chapter 5 presents the detailed design and implementation for a new type of mobility called session mobility. It addresses three fundamental challenges posed by this type of mobility namely: terminal discovery, signalling and mobility support. The terminal description protocol is presented for describing terminals in terms of hardware and software capabilities, followed by a detailed description of the signalling mechanism. This chapter concludes with the results obtained from the testbed used to evaluate session mobility.

Chapter 6 describes an approach to media adaptation using the java media framework API. The essence of this model is a collaborative partnership between the session initiation protocol (SIP) and the M2A application that allows seamless codec change in response to a vertical handover.

Finally, chapter 7 gives a summary of the work and contributions of this thesis. This chapter concludes with a discussion of general lessons learned about vertical handovers for heterogeneous networks and outlines directions for future research.

1.8 References

[1] Hamid Aghvami, Paul Anthony Pangalos, "Towards 4G," 2nd WWRF workshop, May 10-11, 2001 in Helsinki.

[2] G. H. Heilmeier. "Global Begins at Home" IEEE Communications Magazine, October 1992.

[3] M. Stemm, R. H. Katz, "Vertical handoffs in wireless overlay networks", Mobile Networks and Applications, vol. 3, no. 4, pp. 335–350, 1998.

[4] Paul Pangalos, Hamid Aghvami, Daniel Morris, Nima Sattari, Kar Ann Chew, Rahim Tafazolli, "Inter-working of Broadcast, WLAN and Cellular networks", WWRF9, Zürich, Switzerland, 1-2 July 2003.

[5] Helen J. Wang, Randy H. Katz "Mobility support in unified communication networks" in Proc of the 4th ACM international workshop on wireless mobile multimedia Rome, Italy, Pages: 95 – 102, 2001.

[6] Caroline Lebre, Richard Titmuss, Pete Smyth, "Handover between heterogeneous IP networks for mobile multimedia services", Acts Mobile Summit, June 1999.

[7] Book chapter: Prof. A. H. Aghvami, Paul Pangalos, Dr. Babak Jafarian, "Mobility Management for an Integrated Network Environment," to be published in summer 2004.

[8] M. Satyanarayanan, "The Evolution of Coda", ACM Transactions on Computer Systems, Volume 20, Issue 2, pages 85-124, May 2002.

[9] A. Fox, S. D. Gribble, E. A. Brewer, E. Amir. "Adapting to network and client variation via on-demand dynamic distillation", In proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems, pages 160-170, Cambridge, Massachusetts, Oct. 1996.

[10] A. D. Joseph, J. A. Tauber, and M. F. Kaashoek. "Mobile computing with the Rover toolkit" IEEE transactions on computers: Special issue on Mobile Computing , Mar. 1997.

[11] B.D. Noble, M. Satyanarayanan, "Experience with adaptive mobile applications in Odyssey", Mobile Networks and Applications, Volume 4, Issue 4, pages 245-254, December 1999.

"Where there is no vision, the people perish"
(Proverbs 29:18)

Chapter 2

Background and Related Work

The purpose of this chapter is to introduce the reader to related work in vertical handovers and associated mobility protocols in a heterogeneous environment. This chapter starts with a description of Internet heterogeneity and the motivation for vertical handovers. It moves on to describe and discuss the various types of mobility found in the Internet. Section 2.2 discusses terminal mobility and gives an overview of mobile IP versions four and six identifying the differences between them. Other terminal mobility protocols are also surveyed. Section 2.3 discusses personal mobility and gives an extensive overview of the session initiation protocol. Section 2.4 describes a new type of mobility called session mobility. Section 2.5 surveys the real time protocol which is currently used to transfer voice and video over the Internet. Towards the end of the chapter, section 2.6 discusses the use of the RTP protocol in Java. It gives an overview of the Java RTP API and sets the background required for the work presented in Chapter 6 of the thesis. Section 2.7 discusses multimedia conversion in a heterogeneous environment. Finally, this chapter concludes with a summary in section 2.8.

2.1 Internet Heterogeneity

The Internet was initially designed to be used with fixed networks. Over the last decade, it has grown significantly and undergone an unprecedented evolution. The introduction of a variety of new network technologies and the significant diversification of protocols and end devices has been remarkable. Figure 2-1 shows four different wireless interfaces attached to a laptop computer to illustrate the point.

Each network interface provides access to a different network (e.g. Cellular, Ethernet). A network interface is defined as the network specific protocols that are required to connect and operate with the network.

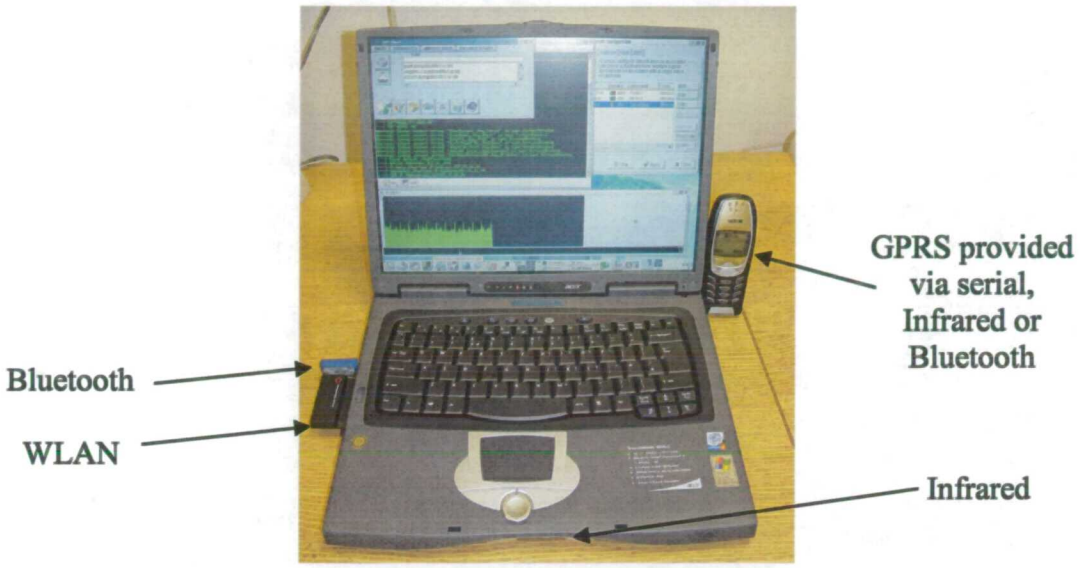


Figure 2-1: Interface heterogeneity. Clockwise from top-left, the picture shows a USB Bluetooth device, a WaveLAN PCMCIA Lucent card interface, an infrared interface and a GPRS enabled cellular phone connected to the laptop via serial, infrared or Bluetooth connection

The wireless technologies in use today belong in one of the following groups.

1. **Infrared:** Infrared, as described in [1] offers an adhoc directional way to communicate to close range devices using direct line of sight. The available bandwidth of Infrared is less than 1Mbps over a relatively small range.
2. **Bluetooth**: Bluetooth [2] was developed and standardised with the intention to provide a simple means of forming ad-hoc networks, where Bluetooth terminals are capable of discovering each other to provide point-to-point and point-to-multipoint connectivity to any device. Bluetooth communication is not limited to line of sight and provides bandwidths of up to 723kbps over a 10m to 100m range depending on the transmitted power used.
3. **Cellular connectivity:** Cellular networks such as [3] provide low bandwidths of less that 56.6kbps today, but their range is several miles long. The third generation cellular networks (3G) such as [4] will increase the bandwidth available and improve throughput and latency. However, cellular will remain slow and expensive compared to alternative local wireless technologies.

4. **Wireless Local Area Networks (WLAN):** Wireless LAN [5] is a technology that is at the top of the group providing high bandwidth connectivity over a large shared area in the form of ad-hoc networks or connected to an infrastructure such as Ethernet. WLAN's offer higher bandwidths at a lower price, but have much more restricted coverage.

None of these technologies however, provides the best solution at all times. The laws of physics still apply, and no solution will cover all possible situations and configurations. These technologies vary widely in terms of coverage, cost, bandwidth, delay, and jitter. As the number of these technologies increases, they are overlaid with each other to form hierarchies. In such environments, users can perform handovers among the different technologies and use a combination of these wireless networks to provide the best possible connectivity based on their requirements. Such handovers are also referred to as vertical. One way of visualising this diversity in access technologies as well as operators and services, is using a 3D diagram as shown in Figure 2-2.

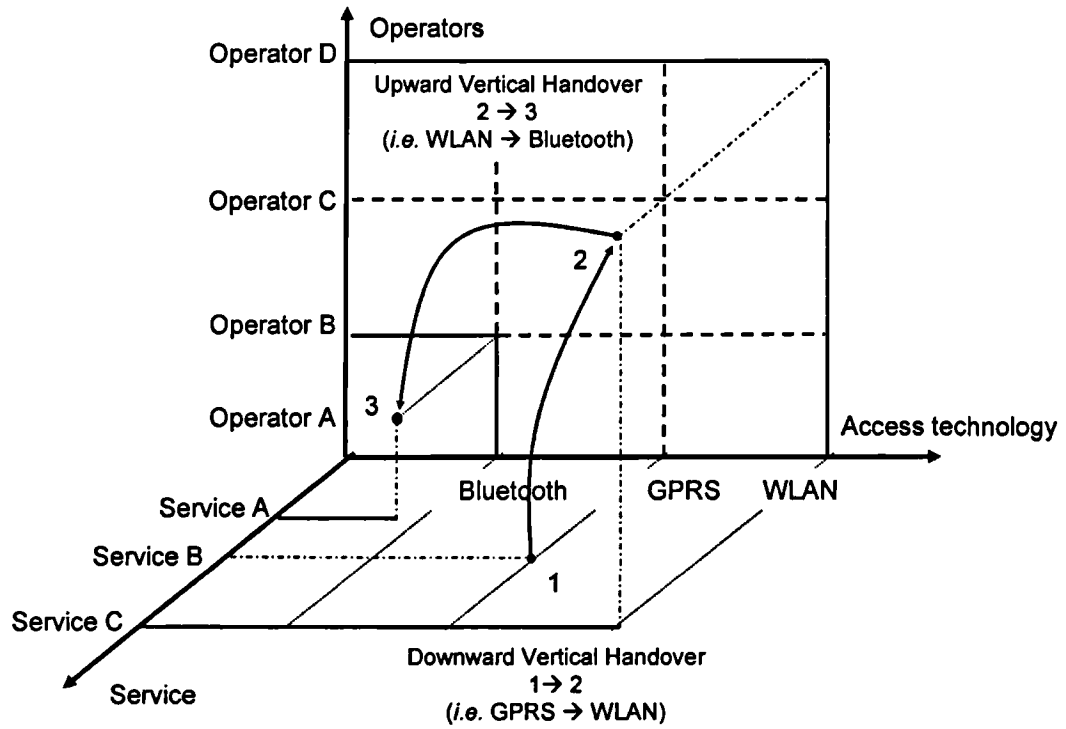


Figure 2-2: The diversity in access technologies, operators and services

This introduction of wireless networks technologies and their constant increase, present problems and challenges that have not been encountered before. Mobility is one of them. Mobility refers to the ability to maintain session-based services in face of changes

to the user environment [6]. These changes are described in terms of terminal location change (terminal mobility) user location change (personal mobility), and session location change (session mobility). Although the Internet has not been designed for a mobile environment, it includes some useful protocols to control movement. The next few sections briefly survey the different types of mobility and the most popular protocols associated with each type.

2.2 Terminal Mobility

The challenge of terminal mobility originates in the way the Internet operates. The Internet enables hosts to communicate with each other across multiple, physically different networks using the Internet Protocol (IP) [7]. To IP it does not matter if the underlying transport is Ethernet, ATM, or even smoke signals, as an IP application will work no matter what the underlying network technology is. IP provides a common language for communication between hosts. As a result end hosts can communicate with each other as long as IP is used for 'gluing' together the heterogeneous link technologies providing a generic inter-working platform as illustrated in Figure 2-3.

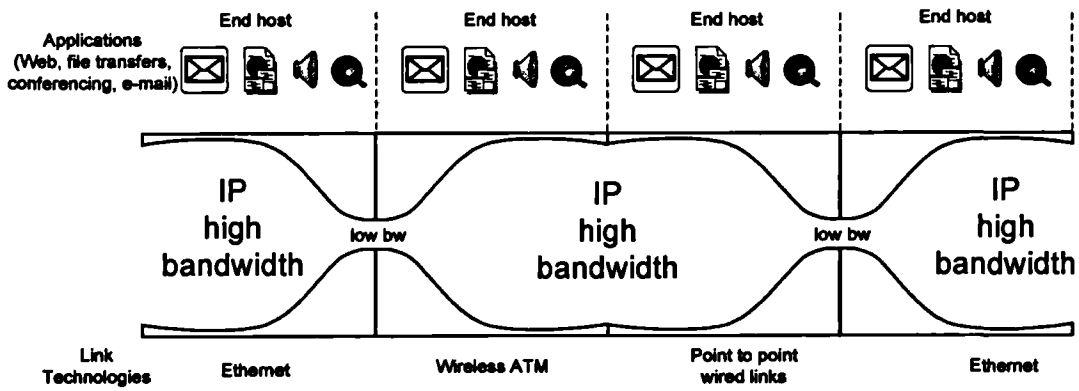


Figure 2-3: A variety of different applications running on end hosts communicating over a variety of different link technologies. The IP layer glues together the heterogeneous link technologies providing a generic inter-working platform

The most fundamental mechanism of IP is the way it routes packets to their destination. This is done according to IP addresses. Each IP address is associated with a fixed network location in the same way a phone number is physically associated with each house in the street. Packets between hosts are routed through a set of routers, as shown in **Error! Reference source not found.**.

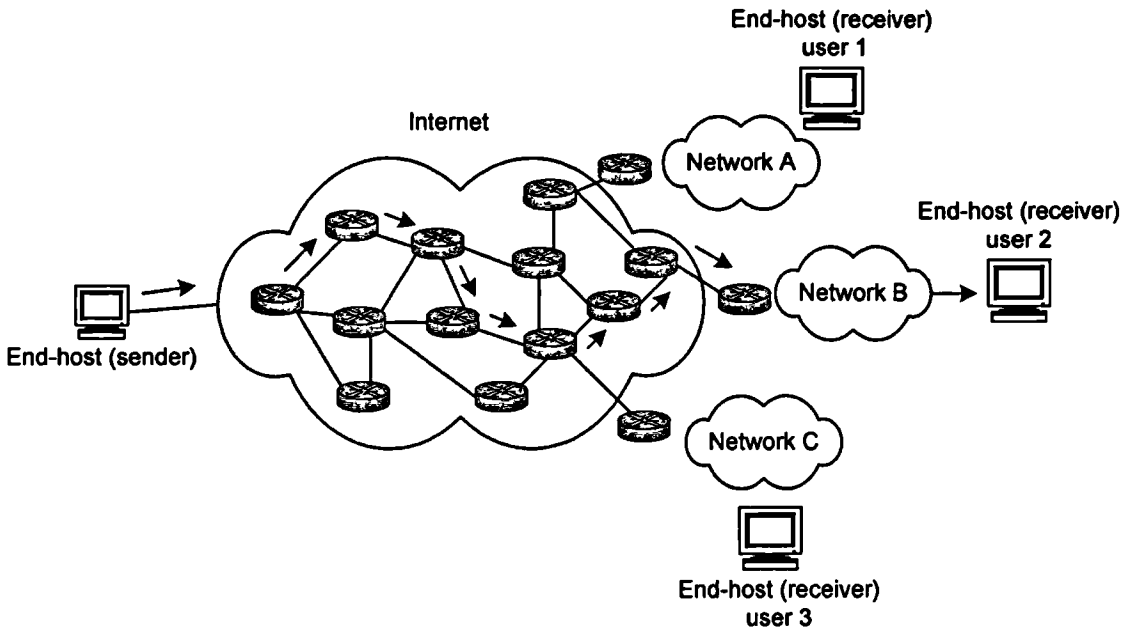


Figure 2-4: Packets between end hosts on the Internet are routed through a set of routers.

The challenge of terminal mobility comes about when the user is mobile. For each new point of attachment, the user obtains a new IP address, making seamless mobility difficult. Terminal mobility refers to the ability of a terminal to move from one location to another while maintaining communication without any service distribution. The Third Generation Partnership Project (3GPP) as well as the Internet Engineering Task Force (IETF) have done a significant amount of research on Mobile IP in recent years [8], [9], [10], [11]. Several implementations are also publicly available on the Internet [12], [13]. The section below describes briefly the operation of Mobile IPv4.

2.2.1 Mobile IPv4

Mobile IP has been proposed as a solution for terminal mobility. It provides users with the freedom to move beyond their home network while consistently maintaining their home IP address. This enables transparent routing of IP datagrams to mobile users during their movement, so that data sessions can be initiated to them while they roam. It also enables sessions to be maintained in spite of physical movement between points of attachment to the Internet or other networks. In the basic operation, when a mobile node changes its location, it is required to notify its home agent of its new point of attachment in the form of a care-of-address. The home agent in turn keeps track of the current location of the mobile node and intercepts and forwards any packets that are destined to the mobile node's home address to the new care-of-address. The home agent forwards these packets to the foreign network by using tunnelling. Tunnelling is a technique in

which a datagram is contained within the envelope of another higher level protocol, in this case IP, during transfer across the internet. The mobile node receives these datagrams and de-tunnels them passing them to the higher layers for processing. All packets sent from the mobile node contain its home address as the source address. However, a well known problem with Mobile IP is the triangular routing which increases the delay and causes an overhead due to tunnelling which increases bandwidth consumption. Measurements [14] show that mobile IP increases the latency by 45% within a campus area network. Tunnelling also adds a typical overhead of 20 bytes (IP to IP encapsulation) [9] for each packet.

In order to avoid triangular routing the route optimisation option is provided as described in [15]. It requires the mobile node to inform the correspondent host of its latest point of attachment to the Internet so as to avoid triangular routing. However, route optimisation has a major drawback too. Correspondent hosts require modifications in the protocol stack so that they can encapsulate IP packets as well as store the care-of-address of the mobile node. One issue still stands out, though. Mobile IP provides a smart solution to terminal mobility by keeping TCP connections alive when the mobile node moves. Figure 2-5 shows the basic routing operation of mobile IPv4.

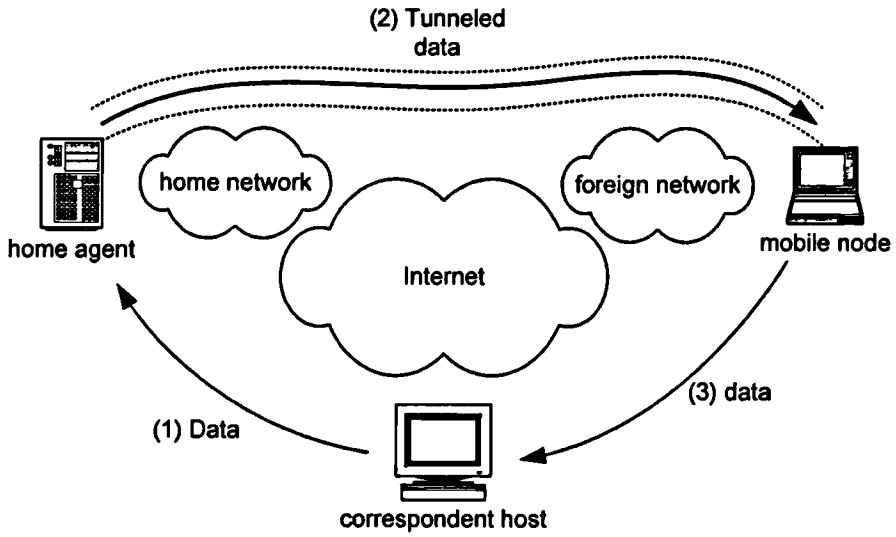


Figure 2-5: Basic mobile IP routing operation

2.2.2 Mobile IPv6

This section discusses Mobile IPv6 briefly and summarises the key advantages offered over Mobile IPv4. Mobile IPv6 borrows many of the concepts and terminology of Mobile IPv4. The high level functions are the same as in Mobile IPv4 and correspond to the three components namely: agent discovery, registration and routing. The basic

operation is shown in Figure 2-6. The full detailed description of the various Mobile IPv6 functionalities and mechanisms can be found in [16], [17], [18], [19] and [20].

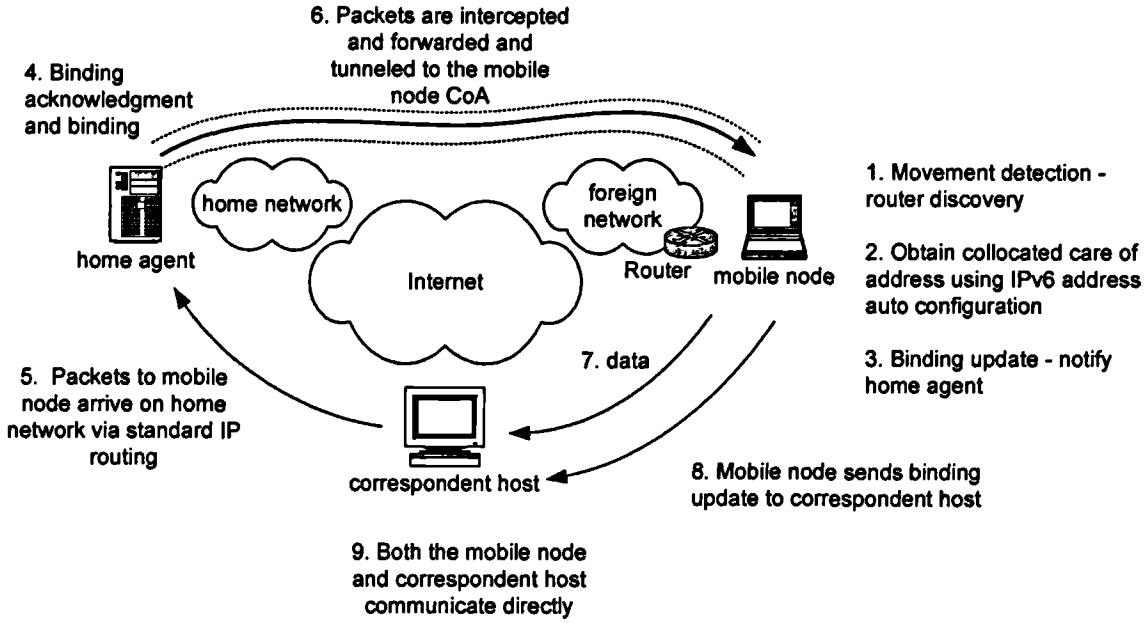


Figure 2-6: Basic operation of Mobile IPv6

The list below summarises the main differences between the two versions of Mobile IP.

- Route optimisation is an optional choice in IPv4 via a separate protocol specification, whereas, it is an integral part of Mobile IPv6.
- For the correspondent host to send traffic over Mobile IPv4, the home agent is required to tunnel the packets to the mobile node. However, in mobile IPv6 this is done using the routing header option that includes the mobile nodes home address, so that the mobile node replaces the care of address of the destination address by the mobile nodes home address before passing the data to the upper layer.
- In order to avoid ingress filtering problems, reverse tunnelling is required in mobile IPv4. In Mobile IPv6 however, the mobile node selects the "care-of address" as source address, which avoids any problem of filtering at the egress point of the foreign network. The "home address" option must also be used so that the receiver replaces the CoA in the "source address" field by the mobile nodes home address before passing on the data to the upper layer.

- Mobile IPv6 enhanced features such as neighbour discovery and address auto configuration, enable the mobile node to function in any location without the services of foreign agents. Therefore, foreign agents no longer exist in mobile IPv6.
- In Mobile IPv4 the care of address is either obtained via agent discovery, a DHCP or configured manually. However, in mobile IPv6 it can be obtained via stateless address configuration [18] as well as DHCP and manual configuration.

Mobile IP is the prime solution to terminal mobility. However, there are other solutions to terminal mobility. Some of these are discussed below.

2.2.3 MSOCKS

MSOCKS specified in [21] implements a transport layer mobility scheme using split-connection proxy architecture. In this approach, a new technique is used called TCP splice that makes a split connection proxy system managing two separate connections, appear as one to the end points. As a result the end points think they are directly connected by a single TCP connection even though they are connected through the proxy server with two separate connections. The MSOCKS approach enables mobility with systems that do not provide access to the network layer source code by intercepting data at the transport layer. Furthermore, it gives the applications the ability to specify which network interfaces should be used for both inbound and outbound traffic.

2.2.4 Session Layer Management (SLM)

The session layer mobility management is a framework for managing connections to mobile hosts in the Internet. It operates above the TCP layer and switches TCP streams between connections. It is based on existing Internet protocols and extends the TCP/IP model by introducing a session layer. SLM is made up of a session management part that operates within the end hosts and supports the dynamic connection establishment and tear down, and a location management part located within the network. This part is used for locating users as well as providing the address resolution. Experimental results have confirmed that SLM is feasible to implement on any platform and behaves well while reconnecting data streams to different networks. This is possible as SLM allows the separate management of different streams belonging to the same process. SLM is described in more detail in [22].

2.2.5 TCP Migrate

TCP Migrate [23] is a new end-to-end framework for Internet mobility developed by MIT. It operates by providing a unified framework to support address changes. The Migrate TCP options support the migration of an active TCP connection across IP addresses. A peer can migrate an open connection to any other IP address simply by sending a new SYN packet from the desired IP address with the Migrate option enabled. Security is provided through the use of a secret cryptographic cookie negotiated through an Elliptic Curve Diffie-Hellman exchange during the initial connection establishment as part of the Migrate Permitted option negotiation. Furthermore, unlike previous approaches, no modification is required to the TCP header or packet format, and no additional information is included in data packets.

2.2.6 The Session Initiation Protocol (SIP)

Although SIP is the primary protocol for personal mobility (discussed in section 2.3) it can also be used for terminal mobility. Terminal mobility impacts SIP in two stages, pre-call, and mid-call. For pre-call mobility, terminal mobility is provided through the use of SIP registrar and redirect servers. As the terminal moves across heterogeneous networks, new temporary identifiers (IP addresses) are assigned to the terminal. These are updated with the SIP proxy server by using the REGISTER method. As a result the current location of the terminal always remains up to date so that new messages can be re-directed successfully to the latest address of the terminal. Mid-call mobility refers to the terminal moving whilst in a session. Once the terminal moves to a new network, an INVITE request is sent to the correspondent host containing the new IP address as well as any new session description. SIP itself does not take care of mobility but rather it facilitates it. Other mobility mechanisms are required to inter-work with SIP to complete the handover.

2.3 Personal Mobility

Personal mobility refers to the idea that the end user is the communication endpoint rather than the devices the user owns. This concept originates from the personal communication systems, described in [24]. Personal mobility involves the network's capability to locate the terminal associated with the user, using a name mapping service for mapping device specific ID's to the user's unique ID. The session initiation protocol provides a solution to personal mobility.

2.3.1 The Session Initiation Protocol

This section gives an overview of the session initiation protocol. This protocol was born in a computer science laboratory less than a decade ago.

2.3.2 Overview

The session initiation protocol (SIP), specified in [25] is a powerful tool for call control and signalling that is gaining support among service providers and vendors. It was originally developed to assist in providing advanced telephony services across the Internet. SIP has also been chosen as the signalling and call control protocol for the Universal Mobile Telecommunications Service (UMTS) R5 and has direct relevance for both UMTS and DSL/cable networks where SIP signalling is also being adopted. The main function of SIP is to establish real–time calls and conferences over IP-based networks. Each session may include different types of data, such as audio and video, although currently most of the SIP extensions address audio communication. As a traditional text–based Internet protocol, it resembles HTTP and SMTP. SIP uses the session description protocol (SDP) [26] for media description. SIP is independent of the packet layer. The protocol is an open standard and is scalable. However, extensions to SIP are needed to make the protocol truly functional in terms of interoperability.

2.3.3 Characteristics of SIP

SIP is an application layer protocol used to establish, modify and terminate calls between one or more users in an IP-based network. A call usually refers to Internet voice calls but is not limited to just that. It also covers Internet multimedia conferences and multimedia distribution for both unicast and multicast sessions. SIP is a pure signalling protocol and it is not tied to any other protocols. To summarise SIP provides the following functions. Firstly it provides name translation and user location also known as personal mobility by providing the capability to reach a called party at a single location–independent e-mail like address 'paul.pangalos@kcl.ac.uk'. The first part of the address is the user name and the second part is the domain name. Secondly, it provides session negotiation, call management and feature changes. It allows users to negotiate and agree on a set of compatible media configurations realising that not all the users support the same features. For example, video may not be supported by one of the users while audio might be the common supported feature giving plenty of scope for

negotiation. With call management a user can add and remove participants to the session as well as being placed on hold or transferred to another session.

2.3.4 Architecture

From an architecture standpoint, the physical components of a SIP network can be grouped into two categories: clients and servers. Figure 2-7 illustrates the architecture of a SIP network. The user clients are installed in the end devices while the sip proxy servers are located in the network. The SIP user clients have two further components. The user server client, listening for incoming SIP messages and the user agent client, sending messages based on user action or in response to incoming messages. SIP user agents are also used to start applications based on the received session description of the invite message. A SIP proxy server however, serves a rather different function. It receives SIP requests, uses the identifier to obtain the IP address or name of the host using a location server, determines where to relay the requests, and passes them to the next server using the IP routing protocol. SIP proxy servers exist in three forms – a SIP stateful proxy, a SIP stateless proxy and a SIP re-direct proxy. A stateful proxy server has memory capabilities being able to remember all incoming and outgoing requests, information that can be of use for future requests. On the other hand a stateless server, just relays all requests and forgets all information once it has left the server. These type of servers are likely to be much faster than the stateful servers. Lastly, redirect servers returns the location of the user rather than relaying the message. Together these components make up a basic SIP architecture. Application servers can sit above these components delivering SIP services to end-users such as instant messaging and presence, third party call control and user profiling. Finally as part of the signaling procedure SIP proxy servers need a mechanism to locate each other. In the proposed solution, SIP uses DNS procedures to allow a client to resolve a SIP Uniform Resource Identifier (URI) into the IP address, port, and transport protocol of the next hop to contact. It also uses DNS to allow a server to send a response to a backup client if the primary client has failed. DNS procedures are described in more detail in [27].

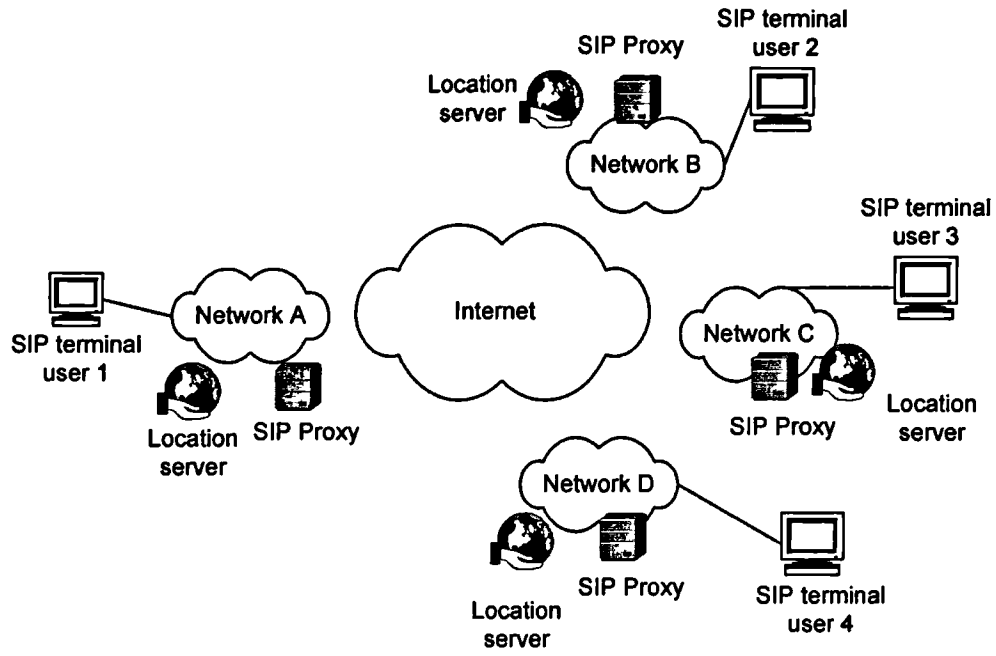


Figure 2-7: SIP architecture

2.3.5 SIP Signalling

A simple operation of the session initiation protocol is illustrated in Figure 2-8. Users register with the sip proxy server using their assigned SIP addresses. This information is then passed on to the location server within the user's domain. In the example shown in Figure 2-8 user 1, who resides in Network A, wants to call user 2. Therefore, User 1 sends an INVITE request for user2@networkB. The INVITE is received by the local SIP proxy of Network B which in turns attempts to identify the current location of user 2 from the location server. The sip proxy determines that user 2 is no longer residing in Network B but has moved to Network C with the new address user2@networkC. The proxy server at Network B can then either return a 302 response directly to user 1 for them to try to contact user 2 or it can try to forward the INVITE on user 1's behalf. In this example the sip proxy of Network B modifies the original INVITE and sends it on. This is received by the sip proxy of network C which then completes the routing of the message to user 2. User 2 accepts the INVITE and responds with a 200 OK following the same path that was taken by the INVITE back to user 1. In the final leg of the call user 1 confirms it received this response by sending an ACK. This is directly send to user 2 bypassing the sip proxy in Network B.

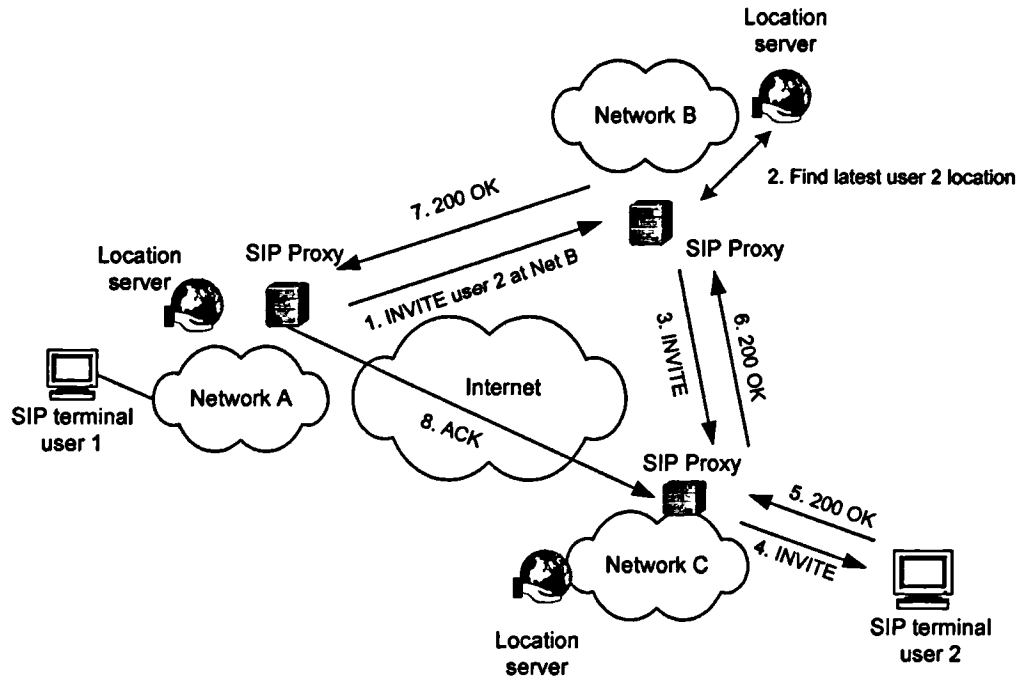


Figure 2-8: SIP operation

2.3.6 SIP Messages

A sip message request consists of three main elements: the request line, the header and the message body. An example is shown in Figure 2-9. A response message also contains three main elements; the status line, the message header and the message body. A method is the main function that a request is meant to invoke on a server and is carried in the request message itself. Example of SIP methods are summarised in Table 2-1

```
INVITE sip:networkB SIP/2.0
Via: SIP/2.0/TCP
From: user 1<sip:user1@networkA>
To: sip:user2@networkB
```

Request

```
Call-ID: 341708
CSeq: 1 INVITE
Content-Type: application/sdp
Content-Length: 98
```

Header

```
v=0
o=user 1 341 252 IN IP4 193.113.150.120
c=IN IP4 193.113.150.120
m=audio 5036 RTP/AVP 1 8
```

Body

Figure 2-9: An example of a request message

SIP Method	Description	SIP Method	Description	SIP Method	Description
INVITE	*Invites a user to join a call*	CANCEL	*Terminates a request, for a user but not the call*	INFO	*Used for mid-session signalling*
ACK	*Confirms that a user received a final response to an invite*	OPTIONS	*Requests information about a server's capabilities*	REFER	*Used for call transfer*
BYE	*Terminates a call between two users or declines a call*	REGISTER	*Registers a user's current location*	MESSAGE	*Used for instant messaging*

Table 2-1: List of SIP methods

2.3.7 SIP Extensions

There are a number of extensions to SIP. This section gives a brief overview to the extensions that are relevant to the work presented in this thesis.

2.3.7.1 The REFER Method

This section gives a brief overview of the REFER extension used for call transfer that is relevant part of the work presented in chapter 6. REFER is a SIP method defined in RFC 3515 [28]. It indicates that the recipient (identified by the Request-URI) should contact a third party using the contact information provided in the method. A success response indicates that the recipient was able to contact the third party. The RFC document only defines the Refer event package and the Refer-To request header. The REFER method may also contain a body message. However, [28] does not assign a meaning to such a body as it is outside the scope of the RFC document. A REFER request is shown in Figure 2-10.

```
REFER sip:paul@kcl.ac.uk SIP/2.0
Via: SIP/2.0/TCP
To: sip:paul@kcl.ac.uk
From: natalie.pangalos <sip:natalie.pangalos@kcl.ac.uk>
Cseq:1 REFER
Call-ID: 208232@ee026
Refer-to=angelos@kcl.ac.uk
Refer-By:<sip:natalie.pangalos@kcl.ac.uk>
Contact:sip:natalie.pangalos@kcl.ac.uk
Content-Length:0
```

Figure 2-10: An example of a REFER message

Once the target user successfully receives the REFER request, it will respond to the sender using the NOTIFY mechanism defined in the event notification in SIP. The NOTIFY message contains a body with a response status line indicating the success of the referred action.

2.3.7.2 The INFO Method

The SIP INFO method is a new extension to SIP that is used for carrying mid-call signaling information. The INFO method could be used for both telephony and non telephony purposes. Some examples include, carrying account balance information, wireless signal strength information and carrying DTMF digits generated during a SIP session. This information can be communicated in either an INFO message header or part of a message body. The definition of these messages however is outside the scope of [29]. Once an INFO request is sent, it must be acknowledged with a 200 OK response indicating that the request was successfully received for an existing call. Each time an INFO message is sent the CSeq header field is incremented. However, this should not be used to determine the sequence of the INFO information as there could be gaps in the sequence numbers caused by other SIP messages. A different mechanism is required.

2.4 Session Mobility

In a typical future environment, users will be able to receive multimedia services via multiple network devices (i.e. mobile phone, desktop phone, public IP terminals, and desktop computer). As people have access to these devices they may desire to switch from one device to another in the middle of a session.

Users might also want to shift parts of a session separately. This can also be described as another type of mobility in which users move from one network to another by changing devices. This type of mobility is called session mobility and it refers to the user's ability to maintain an active session while switching between terminals. The operation of maintaining the session across the different networks and devices is called session mobility and the ability of switching between the terminals session handover. To successfully carry out a session handover three mechanisms need to be present, namely: terminal discovery, signalling, and mobility support. Previous work on the area of session mobility has resolved many issues regarding the basic session handover signalling mechanism [22],[28], and[32], but did not consider mechanisms for terminal

discovery and mobility support. Chapter 5 investigates and presents all three mechanisms, comprehensively addressing several issues that are required for a complete solution for session mobility. The basic signalling mechanism specified in [28] is also extended in order to interact with the terminal discovery and mobility support protocols.

2.5 The Real Time Protocol

To send or receive voice and video between two end points in an IP network, a media transport mechanism is required. This mechanism is responsible for handling packet loss, packet jitter, and delay. Within the Internet voice and video transport are provided by the Real-Time Transport Protocol (RTP). This section introduces streaming media concepts and describes the basic concepts of the RTP protocol that is used throughout this thesis.

2.5.1 Streaming Media

The Internet is on its way again to bring the biggest change in the way people communicate and access information. But this time around it is for voice not data communication. Streaming media is everywhere on the Internet. Examples include live radio, television broadcasts, cinema trailers, audio and video conferences, phone calls, etc. The term streaming media refers to the technique of delivering content over a network without having to wait for the complete stream to download.

TCP based protocols such as HTTP and FTP are designed for reliable data communications. Consequently, these protocols do not work well for streaming media for the following reasons. TCP does not support multicast. Using TCP for multicast is not affordable as hundreds or thousands of connections have to be established. Both the sending computer and your network would collapse. TCP also provides 100% reliability which is not an essential requirement for streaming media. Furthermore, when a packet is lost or corrupted, TCP retransmits it and reduces the window size. As a result the overall transmission rate is reduced. Therefore, TCP protocol may not be suitable for streaming media and other protocols are used instead. One of them is the User Datagram Protocol (UDP). It is an unreliable protocol that does not guarantee delivery of each packet or that the packets will reach their destinations on time and in the order they were sent. This is also the transport protocol that RTP uses. RTP is defined in [31].

2.5.2 RTP Architecture (Protocol suite)

RTP provides end-to-end network delivery services for the transmission of real-time data such as interactive audio and video. The RTP architecture is shown in Figure 2-11. RTP is transport-protocol independent, but a commonly used protocol is UDP. RTP does not provide any mechanism to ensure timely delivery or provide other quality of service guarantees but relies on lower layers services to do so. The Real Time Control Protocol (RTCP) is also an important part of RTP and its main function is to provide feedback reports on the quality of the multimedia session. Applications may use this feedback to adapt to different network conditions, e.g. initiating a vertical handover. Feedback about transmission quality is also useful to locate problems and diagnose faults. The primary function is to provide feedback on the quality of the data distribution. This is an integral part of the RTP's role as a transport protocol and is related to the flow and congestion control functions of other transport protocols. The feedback may be directly useful for control of adaptivity.

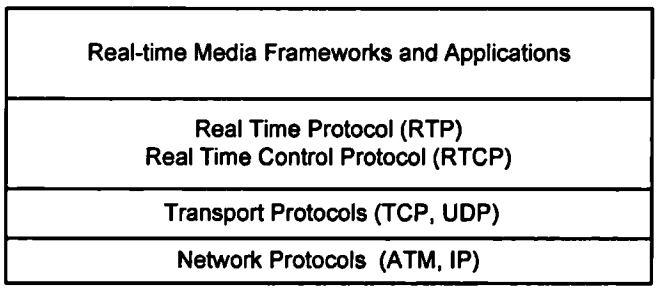


Figure 2-11: RTP architecture

2.5.3 RTP Packets

An RTP session is described by a network address and a pair of ports. An even port number is used for the media data and an odd one for the RTCP reports. Each media type is sent over a separate session enabling the users to choose what type of stream they are interested in. For example, someone who has a low-bandwidth network connection might only want to receive the audio portion of a conference and not the video. An RTP stream is transmitted as a series of packets consisting of a header and the payload. An example of an RTP packet header is shown in Figure 2-12.

```
1057756645.432799 RTP len=124 from=80.225.174.58:5066 v=2 p=0 x=0
cc=0 m=0 pt=7 (LPC ,1,8000) seq=21089 ts=1006668211
ssrc=0x4e451348
```

Figure 2-12: An example of an RTP packet header.

The fields of highest importance in the RTP packet header are:

- The synchronisation source (SSRC) identifier. It is a 32 bit value that a receiver uses to identify the session. Each identifier is unique within an RTP session.
- Payload type (PT): It is a 7 bits field index into a media profile table that describes the payload format. The payload mappings for audio and video are specified in [32].
- The packet sequence number (Seq). This is a unique 16 bit packet number that identifies each packet's position in the stream. The packet number is incremented by one for each packet sent. This number can also be used by a receiver to restore the sequence of packets disordered during the transmission, to detect packet loss, and to compute the number of expected and lost packets.
- The timestamp (ts). It is a 32 bit number that points the sampling instant of the first byte of the payload data carried by the packet. A receiver can use the timestamps for stream playout synchronisation. For example, several consecutive packets can have the same timestamp if they are logically generated at the same time that is if they are all part of the same video frame.

2.5.4 RTP Performance

The performance of an RTP session can be measured in terms of packet inter-arrival time, jitter, and packet loss. In voice over IP (VoIP) jitter is defined as the variation in time between packets arriving, caused by network congestion, timing drift or routing changes. Packet loss on the other hand is an error condition in which data packets appear to be transmitted correctly at one end of a connection, but never arrive at the other. This might be caused due to poor network conditions or packets deliberately dropped at routers because of network congestion. An example of an RTP connection is shown in Figure 2-13 and the corresponding jitter measurements shown in Figure 2-14. The regions labelled A, B, C show three different areas of inter-arrival times. In regions A and C packets arrive with a constant inter-arrival time. The packets' inter-arrival time

in region A is smaller than that of C. Since the inter-arrival times are constant the jitter is zero. Region B however, shows a rather large gap in the reception of packets but no packet loss is experienced. This might be caused by a handover and results in an increase in jitter. Point D shows a period of packet loss which results in an increase in jitter as shown in Figure 2-14. Finally region E, illustrates the arrival of out of order packets. At any given time there are two packets arriving having a different sequence number. In such cases, the RTP application should discard the packets with the lower sequence number in order to avoid disturbing the user's session.

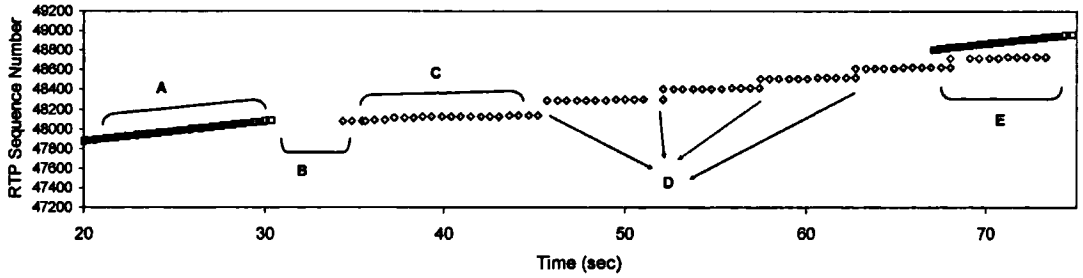


Figure 2-13: A dynamic time evolution of an RTP connection

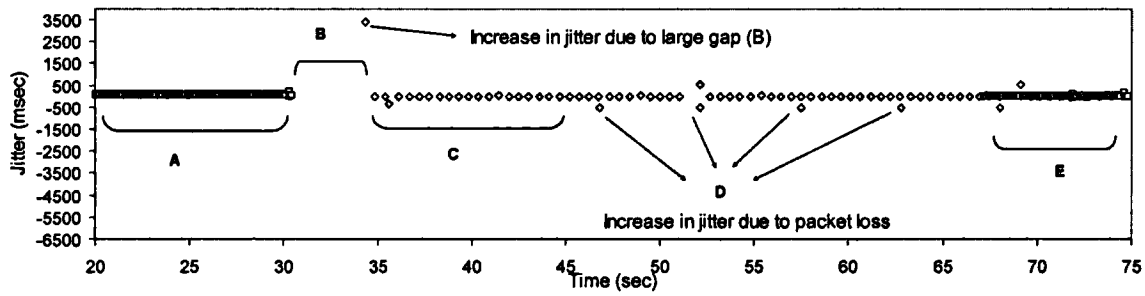


Figure 2-14: Jitter of the above RTP connection

2.6 RTP in JMF (Java Media Framework)

This section briefly discusses the use of the RTP protocol in Java. It gives an overview of the Java RTP application programming interface (API) and sets the background required for the work presented in Chapter 6.

2.6.1 RTP in JMF Architecture

JMF is a set of multimedia Java Programming APIs that enables audio, video and other time-based media to be added to Java applications and applets. The RTP API is an optional package that enables the playback and transmission of RTP streams. It can be used for media-on-demand as well as interactive services such as Internet telephony. JMF can also be enhanced and extended through the standard plug-in mechanism. Advanced developers and technology providers can implement JMF plug-ins to support dynamic payloads and additional RTP formats as well as implement custom packetisers and de-packetisers. The high level RTP in JMF architecture is illustrated in Figure 2-15.

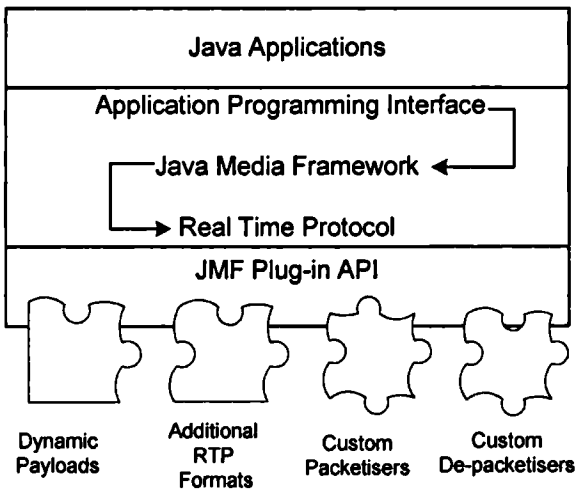


Figure 2-15: RTP in JMF high level architecture

The main components of the RTP API are processors, players, session managers and data sources. Each component within the RTP API has a unique function. Players and processors are used to present and manipulate media streams. The Data Source is used to transmit a media stream already captured whereas the Data Sink is used to store the stream to a file. Another key component is the Session Manager used to coordinate an RTP session by keeping track of the session participants and the active streams. Furthermore, it provides more methods that enable an application to start, close, monitor and modify RTP streams.

2.6.2 RTP based Transmission and Reception

The following diagram Figure 2-16 shows the arrangement of the above components for the transmission and reception of RTP streams.

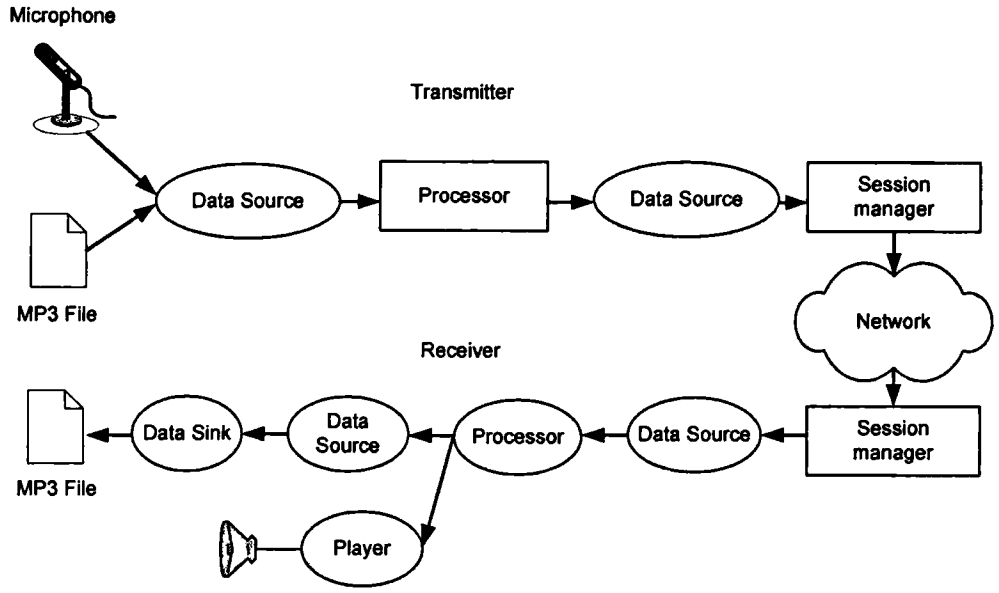


Figure 2-16: RTP transmission and reception

The transmitter can use captured or stored media streams to send these streams across the network. They can originate from a file (e.g. MP3 file) or be captured directly from a capture device such as a microphone. The Data Source is a buffer of the type PushBufferDataSource that acts as a storage place to handle the data in each stage of the transmission. The data then enter the processor component which converts the input stream into an RTP capable stream. Once the conversion is complete, the output of the processor is passed to a Data Source which is then used as the input to a session manager. Similarly, the RTP stream can be received as shown at the bottom of Figure 2-16. On the receiver side, the stream is received by a Session Manager that will provide the stream as a Data Source. The reverse procedure is followed in order to play the stream through a speaker or save it back to a file. A more detailed description of the RTP classes and architecture can be found in [33].

2.7 Multimedia Conversion

As mentioned before, the introduction of wireless networks and technologies and their constant increase, present problems and challenges that have not been encountered

before. The first few sections gave an overview on mobility issues looking at the various types of mobility and associated protocols. However, an equally important challenge is that of multimedia conversion.

2.7.1 Intelligence Distribution in Applications

A user moving through a heterogeneous transmission environment, as shown in the example in Figure 2-2, can experience rapid and large-scale changes in bandwidth availability. Therefore, it is desirable for an application to adapt in response to these changes. There are a range of strategies for adaptation that range from two extremes as shown in Figure 2-17. On one extreme, the applications are kept simple with no modifications or built in intelligence, passing the entire responsibility for adaptation to other entities within the user's terminals such as the operating system or even the network. The other extreme, puts all the intelligence in the application avoiding the need for operating system support. Between these extremes lies another solution where adaptation is achieved by inter-working the applications with the operating system. This inter-working enables applications to adapt as well as allowing the system to monitor and determine adaptation decisions and pass the relevant information to the application. The next section gives a short overview of previous work in adapting applications focusing on the Remos and Odyssey projects.

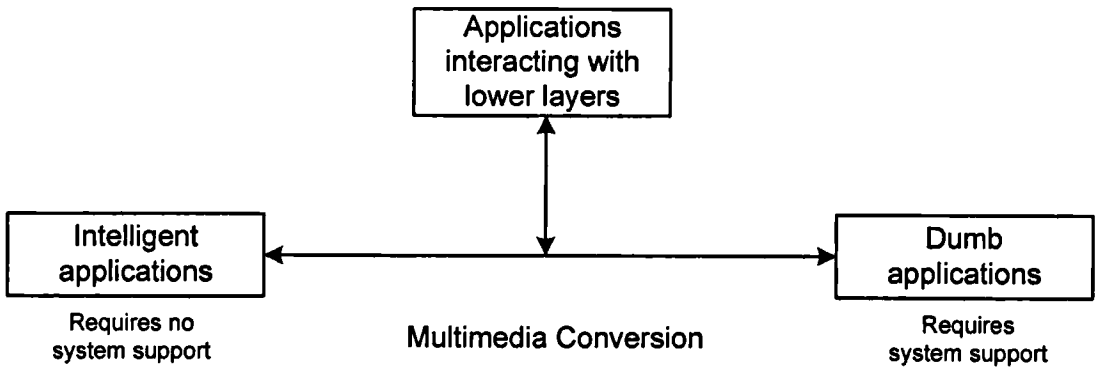


Figure 2-17: Distribution of intelligence

2.7.2 Remos

The Resource Monitoring System [34] also referred to as REMOS is an API that was developed to allow applications to collect information about networks and host conditions across different network architectures. Based on this information the

application adapts in order to run efficiently and predictably under a broader range of conditions. Remos focuses on the interface between the application and the operating system to provide the information that applications need to adapt efficiently to their environment. The Remos API was developed with the goal of providing a query-based interface that allows clients to obtain "best-effort" information on network conditions. Furthermore, the application specifies the kind of information it needs, and Remos responds by supplying the best information. A more detailed description of the Remos API can be found in [35].

2.7.3 Odyssey

Other previous work in adapting applications included adaptation in web browsing [36], client-server systems [37], resource-constrained applications [38] as well as file systems [39]. These systems are based on what is called modal adaptation in which applications have a number of different modes they operate on. For example if a user moves to a lower bandwidth network, a web browser will adapt by fetching low resolution images from a web server. A more recent project such as [40] looked at exploration of application adaptation in Odyssey – a platform for mobile computing.

In order to incorporate adativeness into an application it is essential for the system to detect and inform the application about the environment. Odyssey, demonstrates a collaborative relationship between the operating system and applications to offer the most general and effective approach to adaptability. Some of its features include monitoring resources such as bandwidth, CPU cycles and battery power and interacting with various applications to best exploit them. For example, when the available bandwidth is reduced it notifies the application and in response the video application will skip frames, while the web applications will respond by displaying degraded versions of large images. This system demonstrated that inter-working between the applications and the operating system provides an effective approach to application aware adaptation.

Chapter 6 investigates the design and performance of an adaptable application based on the real time protocol. An advanced application is designed using the Java Media Framework API to provide seamless adaptation in response to a vertical handover. The underlying concept is the same as Odyssey and Remos. In order to incorporate adaptiveness in an application, it is essential that underlying protocols detect and inform

the application about the change in the user's environment. In our approach this was done by inter-working the application with the session initiation protocol to provide efficient and transparent mobility in response to vertical handovers.

2.8 Summary

This chapter started with a description of the Internet heterogeneity and motivated the need for vertical handovers discussing the various types of mobility as well as their associated protocols. Section 2.5 introduced streaming media concepts and described the basic concepts of the real time protocol in terms of its architecture and packet structure. It also discussed and gave examples of how the performance of an RTP session can be measured. Section 2.6 described the Java Media Framework API with emphasis on the implementation of the real time protocol. Section 2.7 studied the challenge of multimedia conversion in heterogeneous network environments giving an overview of prior work as well as introducing the background required for the work presented in Chapter 6.

The material described in this chapter sets the stage for the work to follow. The next chapter focuses on various ways in which mobile IP and the session initiation protocol inter-work to handle unplanned vertical handover for real time traffic.

2.9 References

[1] Patrick J. Megowan, David W. Susak, Charles D. Knutson. "IrDa Infrared Communications: An overview" www.irda.org.

[2] Bluetooth Special Interest Group, "Bluetooth Baseband Specification Version 1.0 B" http://www.bluetooth.com

[3] C. Bettstetter, H.-J. Vogel, J. Eberspacher, "GSM Phase 2+, General Packet Radio Service (GPRS): architecture, protocols and air interface", IEEE Communications Surveys 2(3) (1999) available at http://www.comsoc.org/pubs/surveys/

[4] K. W. Richardson, "UMTS overview", available at: http://www.roke.co.uk/download/articles/UMTS_Overview.pdf

[5] Pablo Brenner, "A Technical Tutorial on the IEEE 802.11 Protocol", http://www.sss-mag.com/pdf/802_11tut.pdf

[6] Helen J. Wang, Randy H. Katz "Mobility support in unified communication networks" in Proc of the 4th ACM international workshop on wireless mobile multimedia Rome, Italy, Pages: 95 – 102, 2001

[7] J. B. Postel, "Internet Protocol", RFC 791, Information Sciences Institute, Marina del Rey, CA, September 1981.

[8] Charles Perkins, ed., "IP Mobility Support", IETF RFC 2002, Oct. 1996

[9] Charles Perkins, "IP Encapsulation within IP", IETF RFC 2003, Oct. 1996.

[10] J.D. Solomon, "Mobile IP: The Internet Unplugged", Prentice-Hall, 1998.

[11] C.E. Perkins, " Mobile IP Design Principles and Practise", Prentice-Hall, 1998.

[12] Mary G. Baker, Xinhua Zhao, Stuard Cheshire, Jonathan Stone, "Supporting Mobility in MosquitoNet", Proceedings of the 1996 USENIX Technical Conference, January 1996.

[13] Dynamics, "HUT Mobile IP" , http://www.cs.hut.fi/Research/Dynamics/

[14] X. Zhao, C.Castelluccia, and M. Baker, "Flexible network support for mobility", in Proc of Mobicom, (Dallas,Texas), Oct. 1998.

[15] C. Perkins, D. Johnson, "Route optimisation in mobile IP", Internet Draft, Internet Engineering Task Force, Dec 1997.

[16] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specifications", RFC 2460, Internet Engineering Task Force, December 1998.

[17] T. Narten, E. Nordmark, W.Simpson, "Neighbour Discovery for IP Version 6 (IPv6)", RFC 2461, Internet Task Force, December 1998.

[18] S. Thomson, T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, Internet Engineering Task Force, December 1998.

[19] A. Conta, S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 Specification", RFC 2463, Internet Engineering Task Force, December 1998.

[20] A. Conta, S. Deering, "Generic Packet Tunnelling in IPv6 Specification", RFC 2473, Internet Engineering Task Force, December 1998.

[21] David A. Maltz, Pravin Bhagwat, "MSOCKS: An Architecture for Transport Layer Mobility", pages 1037-1045, IEEE INFOCOM'98.

[22] Björn Landfeldt, Tomas Larsson, Yuri Ismailov, Aruna Seneviratne, "SLM, A Framework for Session Layer Mobility Management", In Proc. IEEE International Conference on Computer Communications and Networks, pages 452-456, Natick, Massachusetts, October 1999.

[23] Alex C. Snoeren, Hari Balakrishnan, "The Migrate Approach to Internet Mobility", Technical report, MIT Lab for Computer Science, available at http://www.cs.helsinki.fi/u/kraatika/Courses/sem02s/Migrate.pdf

[24] Mohammed Zaid, "Personal mobility in PCS", IEEE Personal Communication, Fourth Quarter 1994.

[25] M. Handley, H. Schulzrinne E. Schooler, J. Rosenberg, "SIP: Session Initiation Protocol", RFC 2543, Internet Engineering Task Force, March 1999.

[26] M. Handley, V. Jacobson, "SDP: Session description protocol", RFC 2327, Internet Engineering Task Force, April 1998.

[27] J. Rosenberg, H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 2543, Internet Engineering Task Force, June 2002.

[28] R. Sparks, "The Session Initiation Protocol (SIP) Refer Method", Request for Comments 3515, Internet Engineering Task Force, April 2003

[29] S. Donovan, "The SIP INFO Method", RFC 2976, Internet Engineering Task Force, October 2000.

[30] J. Rosenberg, H. Schulzrinne, J. Peterson, "Third party call control in SIP", Internet Draft, Internet Engineering Task Force, March 2000.

[31] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Application, RFC 1889, Internet Engineering Task Force, January 1996.

[32] H. Schulzrinne, "RTP Profile for Audio and Video Conferences with Minimal Control", RFC 1890, Internet Engineering Task Force, January 1996.

[33] The JMF API documentation available at http://java.sun.com/products/java-media/jmf/2.1.1/documentation.html

[34] Thomas Gross, Peter Steenkiste, Jaspal Subhlok, "Adaptive Distributed Applications on Heterogeneous Networks", Eighth Heterogeneous Computing Workshop April 12 - 12, 1999, San Juan, Puerto Rico.

[35] Bruce Lowekamp, Nancy Miller, Dean Sutherland, Thomas Gross, Peter Steenkiste, Jaspal Subhlok, "A Resource Query Interface for Network-AwareApplications", In 7th IEEE Symposium on High-Performance Distributed Computing, pages 189–196, Chicago, July 1998.

[36] A.Fox, S. D. Gribble, E. A. Brewer, E. Amir. "Adapting to network and client variation via on-demand dynamic distillation", In proc. of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems, pages 160-170, Cambridge, Massachusetts, Oct. 1996.

[37] A. D. Joseph, J. A. Tauber, M. F. Kaashoek. "Mobile computing with the Rover toolkit", IEEE transactions on computers: Special issue on Mobile Computing , Mar. 1997.

[38] B.D. Noble, M. Satyanarayanan, "Experience with adaptive mobile applications in Odyssey", Mobile Networks and Applications, 1999.

[39] M. Satyanarayanan, "The evolution of Coda", ACM Transactions on computer Systems, 20(2):85-124, May 2002.

[40] Brian D. Noble, M. Satyanarayanan, "Agile Application-Aware Adaptation for Mobility", in the proc. of the 16th ACM Symposium on Operating System Principles, St. Malo, France, October 1997.

[41] http://www.google.com

"Trust in the Lord with all your heart and lean not on your own understanding in all your ways acknowledge him and he will make your paths straight"
(Proverbs 3:5)

Chapter 3

Inter-working of MIP and SIP for Unplanned Vertical Handovers

This chapter focuses on the various ways in which mobile IP (MIP) [1] and the session initiation protocol (SIP) [2] inter-work to handle unplanned vertical handover for real time traffic. Different signalling mechanisms are used to assist the mobile node to recover during unplanned vertical handovers considering both the network and application layers. The key mechanisms for supporting disconnections include inter-working of SIP and MIP to signal and preserve communication during a reconnection.

3.1 Introduction

Unplanned handovers are a significant area of inter-domain mobility which is often ignored. During such a handover a terminal might experience a disconnection period that can either be short or long enough to cause severe problems at higher layers. Although significant work has been done in the area of disconnected file systems [3], [4], less attention has been paid to the issues of preserving application communication during a reconnection when resuming after a vertical handover. The primary characteristic of an unplanned handover is that it happens suddenly and is not predicted, causing severe disruption to the IP layer [5]. This can be caused by the user (i.e. unplugging a cable or a network interface), or a blackout in the signal level.

In this chapter, we will be concerned primarily with avoiding congestion during a reconnection. In particular, we focus on two different techniques. The first one, studies the inter-working between the SIP user agent (SUA) and MIP in a loose coupling scenario. In this approach an extended version of MIP, based on the M3A project [6], is modified to inter-work with the SUA. The MIP agent, residing in the user's terminal, communicates in a unidirectional manner with the SUA. This connection is used to allow MIP to send status related information to the SUA. Such information includes connection management and handover related information. The second technique looks at the tight coupling approach between the two. Each agent is able to access directly the functionalities of the other enabling a bidirectional flow of information providing a tight co-operation between them. The term coupling refers to the measure of the strength of the inter-working between the two agents. Table 3-1 summarises the two approaches.

	Loose coupling	**Tight Coupling**
Basic mobile IP	Based on M3A project	HUT Mobile IP version 0.81
Sip User Agent	Developed to work with both coupling techniques	
Communication type	Unidirectional	Bidirectional
Dependence	Mobile IP (autonomous mode) SIP (reliant on MIP)	Mobile IP (reliant on SIP) SIP (reliant on MIP)
Exchange functions (Mobile IP to SUA)	Status information such as mobility and connectivity information	Status information such as mobility and connectivity information
Exchange functions (SUA to Mobile IP)	None	Handover initiation Update interfaces Disconnect Select tunneling mode

Table 3-1: The two coupling techniques

3.2 Loose Coupling between Mobile IP and SIP

Loose coupling refers to the limited number of interactions, and the direction of information flow is between Mobile IP and the SIP agent only. Mobile IP is designed to function as an autonomous module independent of all other layers and SIP functions. Its main function is to hide the IP address change during a handover. Once a handover is initiated a series of consecutive events will follow that are independent of the SUA functions. On the other hand, the SUA is responsible for establishing and tearing down sessions as well as session re-negotiation during a handover. The functions and interactions between the two agents are shown in Figure 3-1.

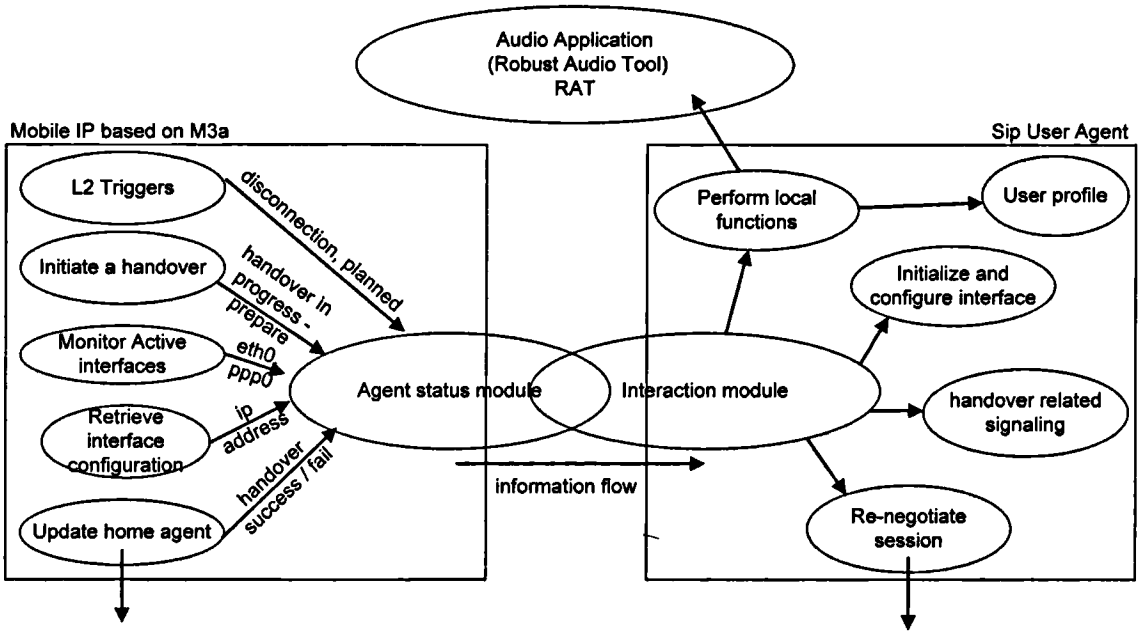


Figure 3-1: Inter-working of SIP and MIP in a loose coupling scenario

3.2.1 Experimental Environment

In this section, we introduce our experimental environment used to test and evaluate the loose coupling technique for handling unplanned vertical handovers within a mobile IP environment.

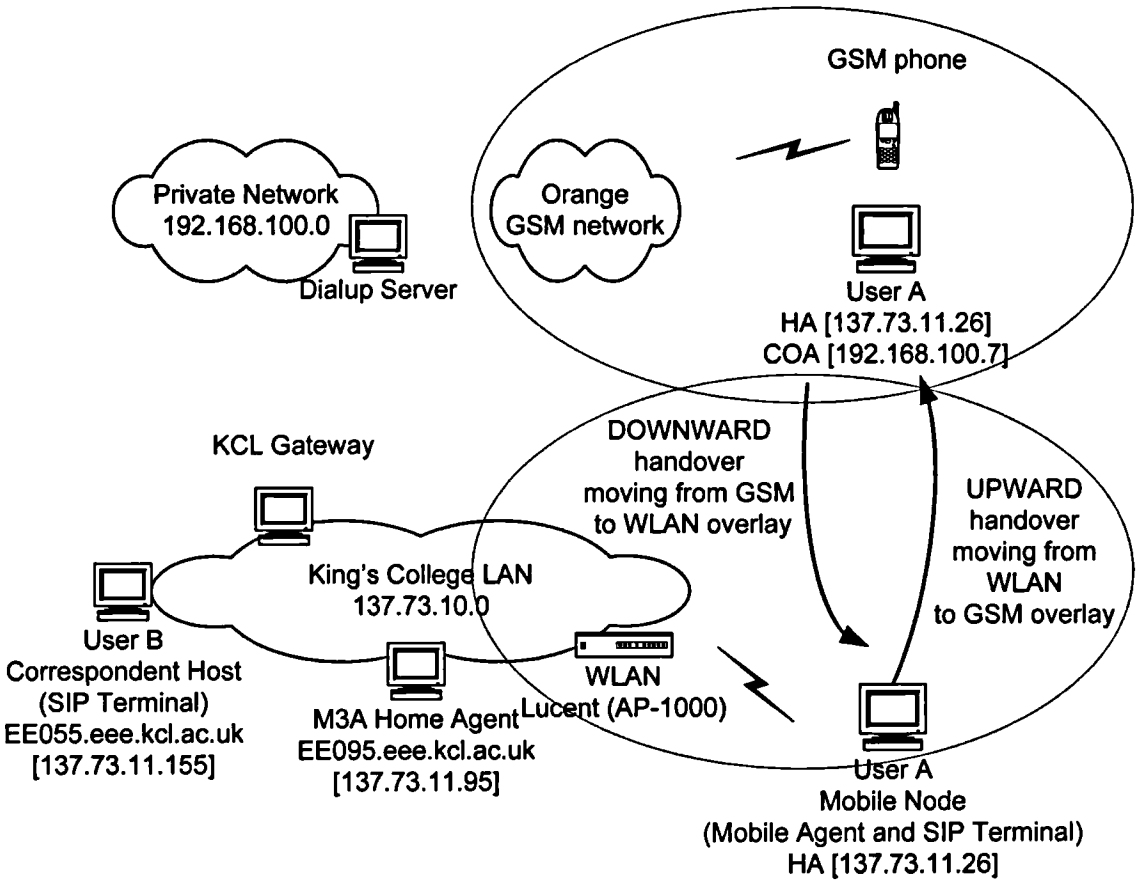


Figure 3-2: Experimental environment for the loose coupling technique

The experimental architecture has been implemented on a testbed, shown in Figure 3-2. The implementation consists of the following three software modules: An SUA, an extended version of Mobile IP (M3A), and the robust audio tool (RAT). The SUA and MIP extension modules were both developed on the basis of standard PCs under LINUX. The SUA is used in combination with the SIP proxy server enabling TCP based SIP communication between any registered nodes. MIP is used to handle terminal mobility as users move from one network type to another. SIP is used to re-negotiate an existing session when there is a disconnection. Finally, RAT is used as an audio conferencing and streaming application tool that allows users to participate in audio conferences over the Internet [7]. RAT is based on IETF standards, using RTP [8] above UDP/IP as its transport protocol. For tracing, parsing, and printing RTP/RTCP streams, RTP tools [9] are used. RTP tools comprise a number of small applications for

processing RTP data. The main hardware parts of the testbed are an AP-1000 lucent wireless access point (IEEE 802.11) and two main servers, a SIP proxy server and a Mobile IP home agent. There are currently two overlay networks shown in Figure 3-2. The local area network (LAN) covered by the AP-1000 wireless access point and the private domain which is covered by the public GSM network. The mobile node (referred to as user A) is equipped with an IEEE 802.11b WLAN card and a GSM modem. The correspondent host (referred to as user B) is located inside the King's LAN and is used to establish a real time session with user A. Table 3-2 gives a summary of the experimental environment.

Summary	Loose coupling technique
Home agent: Using M3A mobile IP with full tunnelling mode, running on Linux Redhat – kernel-2.2.14. Ethernet connection on King's LAN. Requires security authentication between home and foreign network	
User B: Using RAT for audio on Linux Redhat – kernel 2.2.10. Source connected to KCL LAN	
Home Network: 137.73.10.0, Netmask 255.255.254.0	
Multihomed mobile terminal(User A): Acer travelmate 402 with WLAN and GSM interfaces	

Table 3-2: Summary of experimental environment

3.2.2 Inter-working between SIP and MIP

This section gives a technical explanation of the loose unidirectional inter-working between mobile IP and SIP. Multithread programming was used since the two agents need to operate independently and only communicate when necessary. A multithread program is one that contains two or more parts that can run concurrently. Each part of such a programme is called a thread. Thus, multithreading is a specialised form of multitasking. Unlike other programming languages, Java provides built in support for multithread programming. The signalling communication between the two agents was implemented using TCP and later on replaced by UDP. For the TCP implementation, Mobile IP initiates and performs the call for initiation of the connection as shown in Figure 3-3 step 1. However, if the sip user agent is not already loaded, it cannot listen for any incoming request for connections. Thus, a run time error occurs, terminating the initiating thread, and breaking down the communication between mobile IP and the sip

user agent. This could be avoided by first loading the program that implements the sip user agent, but this is not very flexible and it is not desirable to have such restrictions. Even with this method, if the sip user agent ceases its execution and has to be restarted, then the program that implements the mobile IP must be restarted as well. To avoid this situation TCP was replaced by the UDP protocol.

Mobile IP is required to inform the sip user agent about two states; Interface disconnections (step 3) and the successful registration of the mobile node with the home agent (step 5). The first state is identified by the message 'nowhere interface' indicating the interface that has been disconnected. The second state is conveyed by the message 'mip:default route interface' specifying the new interface is registered with the home agent (steps 4 and 6). Both messages are received and processed within the dialog box class of the sip user agent. Upon receiving the first message, the sip agent selects and loads an interface based on the user's profiles, taking into account parameters such as time of day, cost, and required bandwidth. Once the new interface becomes available, Mobile IP automatically registers that interface with the home agent. If registration is successful, the mobile agent would notify the sip user agent by sending a second message indicating the name of the registered interface. In response, the sip user agent will send a sip info message to the correspondent node requesting suspension of the session, followed by a re-invite message in order to re-negotiate the old session based on the characteristics of the new interface. In the worst case, the application of the correspondent host may be unable to suspend the session, thus having to terminate the application.

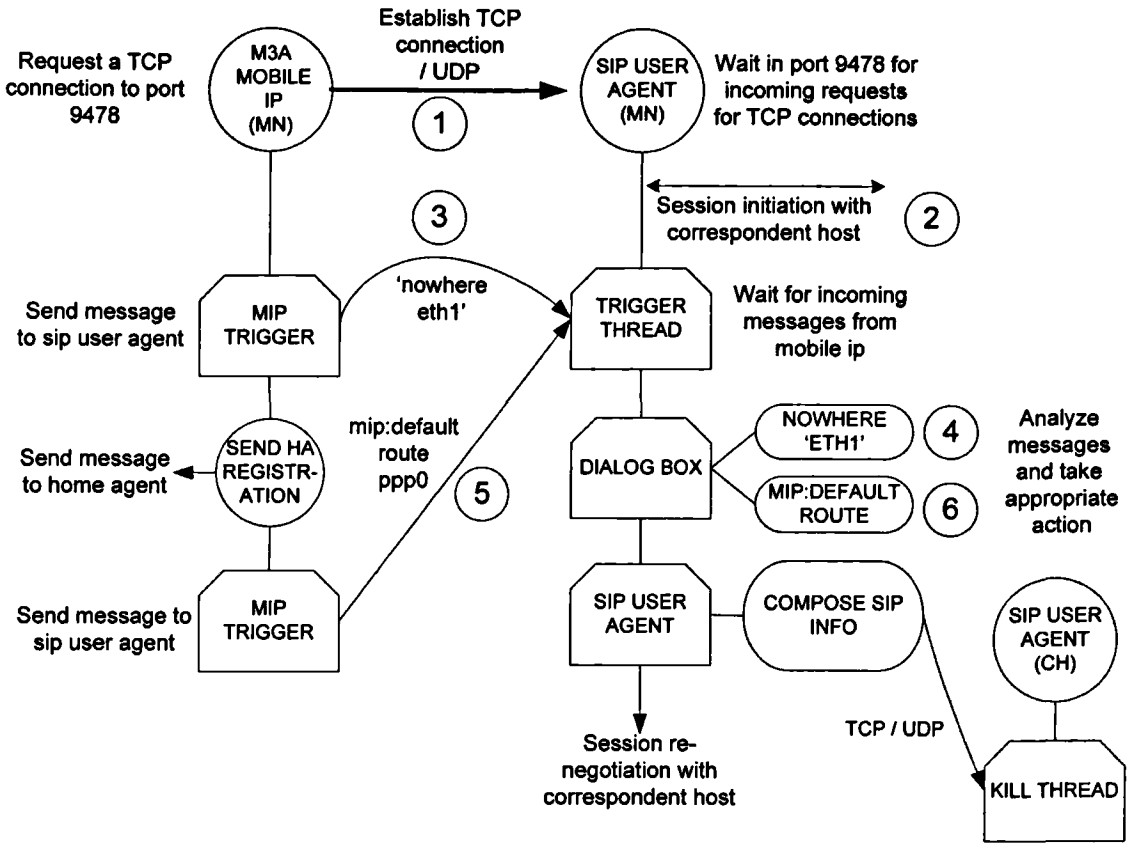


Figure 3-3: Software implementation - inter-working of Mobile IP and SIP

3.2.3 Experimental Scenario

In this section, we describe, test and evaluate a method of providing support for disconnections in a Mobile IP environment in a loose coupling approach. The signalling mechanism is shown in Figure 3-4. User A is initially connected to its home network through the wireless LAN interface. Sip is used to initiate a two-way real-time audio session with User B at a bit rate of 64kpbs. Then at some point, User A loses its current network connection. The disconnection is initiated by unplugging the wireless network interface or switching off the WLAN base station. This change is detected by the local link layer (L2 Trigger) module of the MIP agent triggering the 'nowhere eth1' message, informing the SUA that a disconnection is detected at the eth1 interface.

The SUA will then initiate a process by which it will detect other available interfaces on the terminal and, based on the user preference characteristics (i.e. bandwidth, cost, and services), will select an appropriate interface to connect to. In this case the GSM overlay network is selected. In the future, various networks will be available to the user e.g. Hiperlan 2, UMTS, and GPRS. However, the basic idea, concepts and procedures

described in this chapter remain the same whatever network the user wishes to handover to. Connecting to the GSM network causes an initial delay of approximately thirty seconds for dialling, authenticating and configuring the GSM interface. During the configuration process, the interface is assigned a collocated care of address (COA) from the private network, and becomes the new default interface of the terminal. This change is promptly detected by the monitor active interface (MAI) module of M3A, triggering MIP to send a registration request to the home agent who in turn responds with a successful registration reply message. The success of the registration is also relayed from the agent status module (ASM) to the SUA in the form of 'mip:default route ppp0'. On receiving this message user A stops transmitting data at a high bit rate, in order to avoid overloading the slow GSM link and, at the same time, sends a SIP INFO message to user B to do the same. The purpose of the INFO message [10] is to carry application level information along the SIP signalling path.

The INFO method is not used to change the state of SIP calls, or the parameters of the sessions SIP initiates. It merely sends optional information, related to the session. The aim of this message is to inform user B about the vertical handover, in order to stop the transfer of data at a bit rate that is beyond the bandwidth capability of the receiving end, by either pausing or terminating its application. Following the SIP INFO message, user A generates and sends a re-invite message to B, using the same CALL-ID, describing the new required media session based on user A's profile and current network connection. Both users then re-negotiate the session, load the appropriate application(s) and resume transmitting using the newly agreed codecs (LPC@5.6kbps). If the new session being negotiated is of a similar nature to the old one then applications need not be terminated but just paused - stop sending data, until the new session description is received. Once received, applications re-adjust the codecs and resume transmitting using the same session ID. In RTP, this is identified by the synchronization source (SSRC) identifier. However, if the required new session is of a different type compared to the previous one, then applications need to be terminated, the session destroyed and new types of application loaded (i.e. text messaging tools, whiteboard), according to the new network connection.

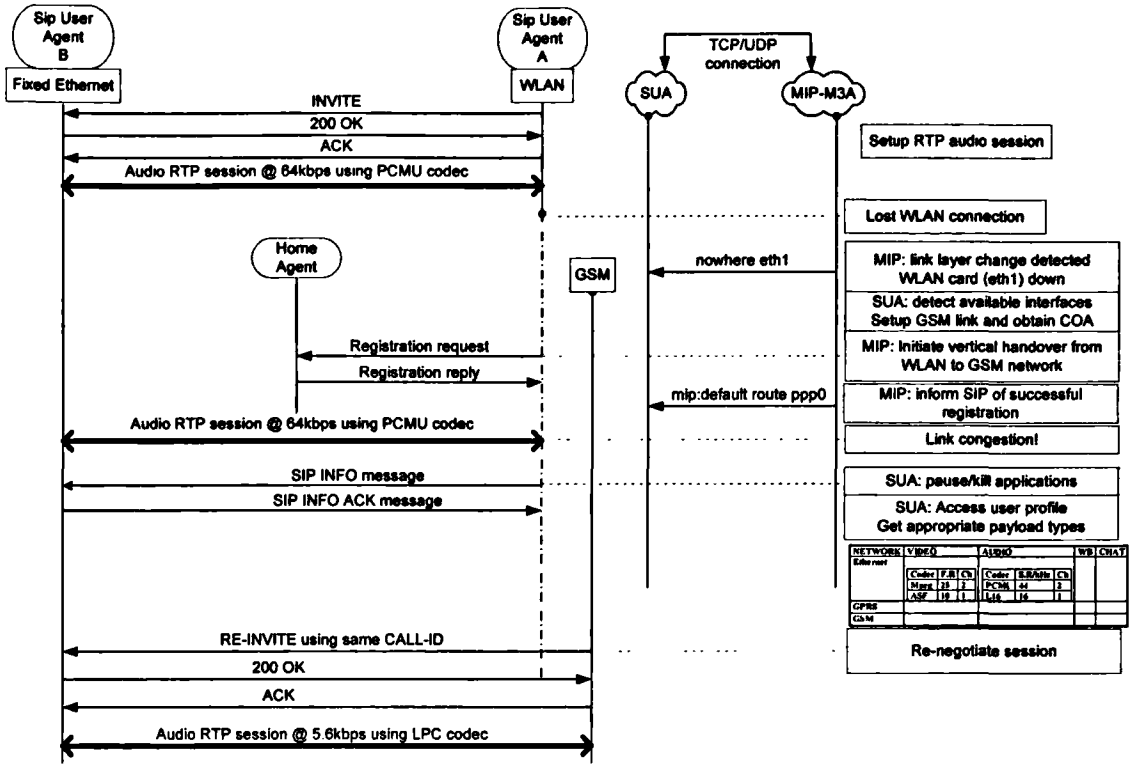


Figure 3-4: Loose coupling - signalling during a disconnection

3.2.4 Test Results

The results obtained from this scenario are presented in

Figure 3-5. The graph was obtained by monitoring the RTP packets received by User A while responding to a disconnection from the WLAN to the GSM overlay. It is clearly seen from the graph that there is a significant delay (t=11 to 85 sec) before the new session is re-established.

At t=0sec, user A receives an RTP audio stream from user B at 64kbps with a packet size of 1280 bytes. At around t=11sec the disconnection is detected and the GSM interface is setup. The modem uses the point-to-point protocol (PPP) to setup the interface and obtain an IP address. This takes approximately 36 seconds to complete. Once an IP address is obtained from the private domain, MIP registers with the home agent using that as the care-of-address (COA) of the terminal. The mobile IP registration takes approximately two seconds to complete over the GSM network. An interesting part of the graph is the region between t=50sec to t=80sec. This is the time taken for user A to send the SIP INFO message to user B. Once the signalling gets through both users re-negotiate the session (at t=79 to 84 sec) loading the appropriate

applications. Finally, user A receives the new low bitrate RTP packets at t=84sec with a packet size of 124bytes, reducing the received bitrate from 64kbps to 5.6kbps.

Since the packet size of the new received stream has decreased no change in the gradient is observed.

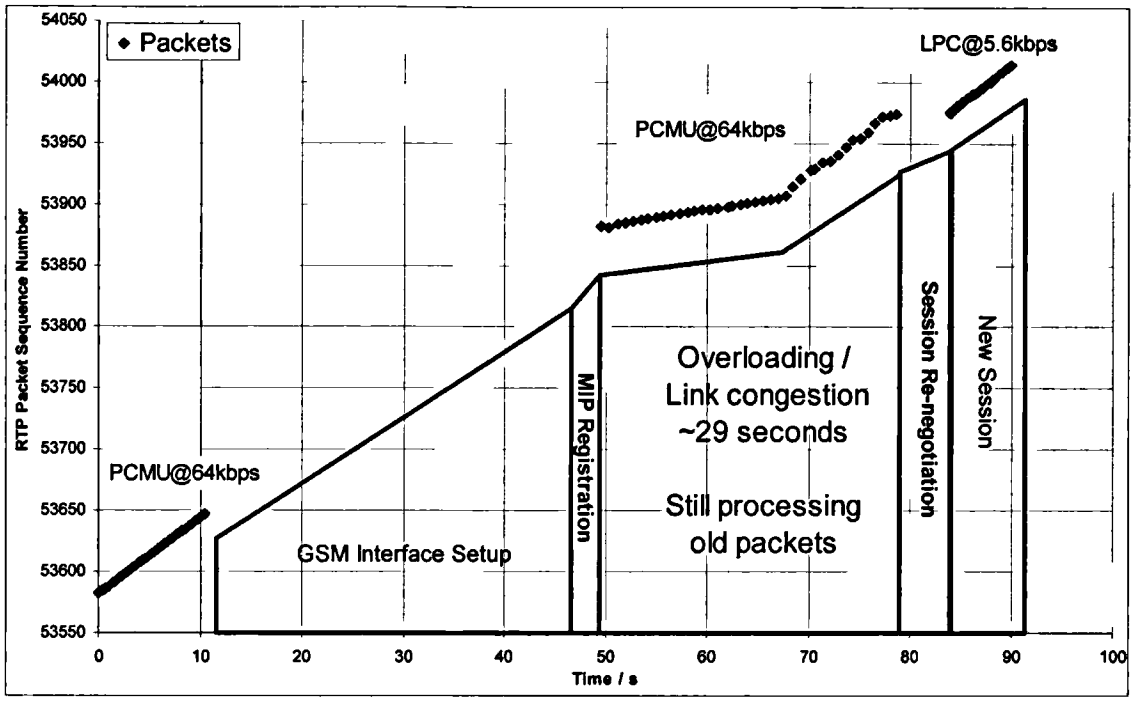


Figure 3-5: Upward unplanned vertical handover
ETH (PCMU - 64kbps) to GSM (LPC - 5.6kbps)

To summarise, the four main causes of the delay experienced during unplanned vertical handover are – dialup connection, the registration with the home agent, the signalling delay and the session re-negotiation. This is shown more clearly in the timing diagram in Figure 3-6.

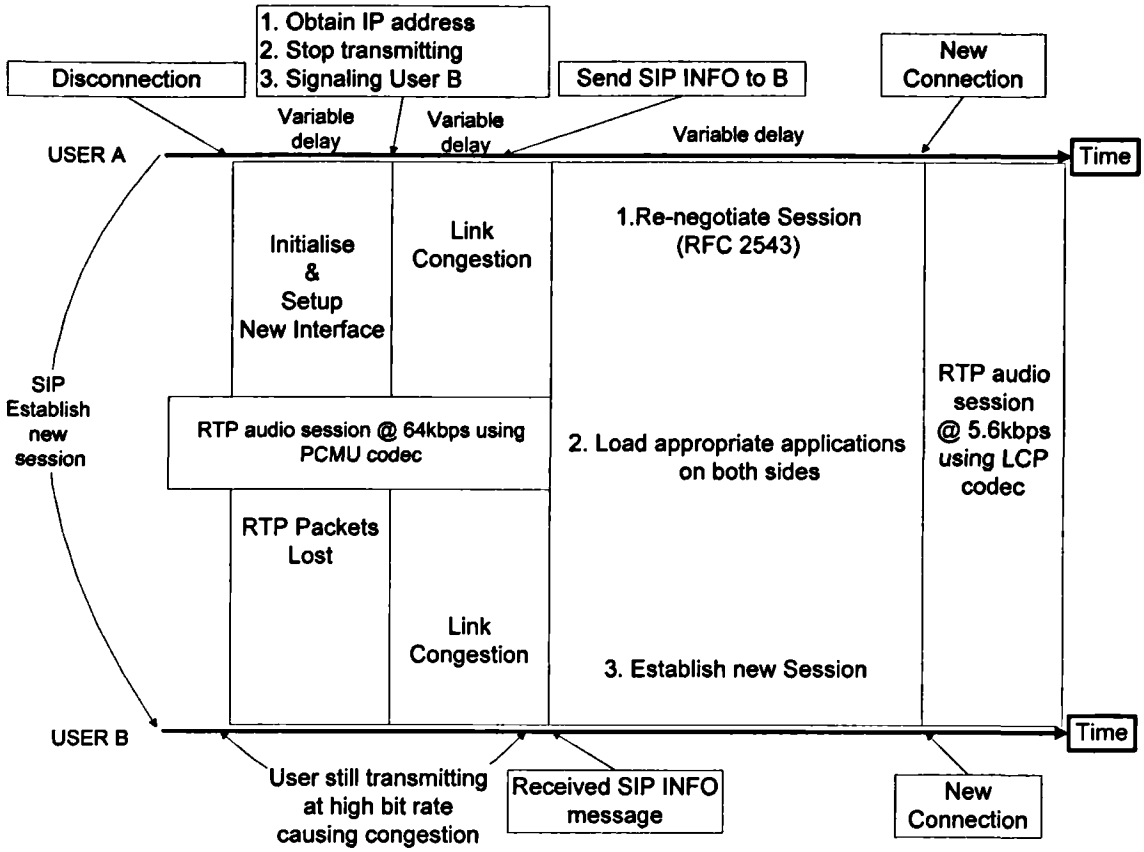


Figure 3-6: Timing diagram during an upward unplanned vertical handover

Dialup connection delay

This delay is specific to the GSM network and in particular to the dialup negotiation. This is the time taken for the laptop to initialise the GSM interface modem, dialup, and setup the connection. In the future, many other networks will be available e.g. HIPERLAN 2, UMTS, and GPRS. As a result, this connection time will vary according to the network to which the user connects. Currently this time is over 30 seconds.

Home agent registration delay

Once user A is assigned an IP address from the private network, the dialup interface will be set active, prompting mobile IP to send a registration message to the home agent. Until the registration is completed all RTP packets send from user B are lost. The home agent GSM registration takes approximately two seconds to complete.

Signalling Delay

This phase makes up one of the largest part of the delay and is caused by link congestion. Ideally, user A has to notify user B about the current vertical handover before the terminal registers with the home agent. This will inform user B to stop transmitting data that is beyond the receiving capability of A. Unfortunately the nature of the loose coupling scenario does not allow this kind of synchronisation between mobile IP and the SUA. Once an IP address is assigned to user B and the GSM interface becomes active, MIP automatically and independently registers the terminals new interface with the home agent. All packets destined for the home address of A are intercepted and forwarded to the care-of-address. This causes link congestion preventing user A from making further communications to user B through which to inform B of the handover. This delay is variable and depends mainly on the initial data rate between the two users.

SIP re-negotiation delay

The re-negotiation delay is the time taken for both users to terminate/suspend their session, re-negotiate the new one and load/resume the appropriate applications. In the SIP INFO message, user A includes its original SIP user identifier and the same call ID into the 'from'-and 'call-id' fields of the SIP header. It also includes information about the applications that need to be paused or terminated within the existing session. This information is placed in the m field of the SDP header containing a description of applications that user B needs to suspend or terminate. An example of such a message is shown in Figure 3-7. When the SIP INFO message is received, user B stops transmitting by terminating its audio application, and waits for a SIP re-invite message from A. Instead of terminating the session the correspondent host could suspend it and resume using the new agreed codecs when the new session re-negotiation is complete. However, this is only possible if the application supports such a function. The experimental results presented here are based on the first option; that of terminating the application and re-loading it. Once the sip message, containing the new session description is received, both users re-negotiate the session and load the appropriate application(s) as described in [2]. This time is expected to vary from one session to the next and from one user to the other. This is because the terminal itself takes some time to terminate and re-load the required applications depending on the processing power available. A slower terminal will take longer to process and re-negotiate the session

resulting in a higher re-negotiation delay. Experimental results have shown that using RAT audio application takes approximately an average of two to three second to reload on a PIII 500 MHz terminal.

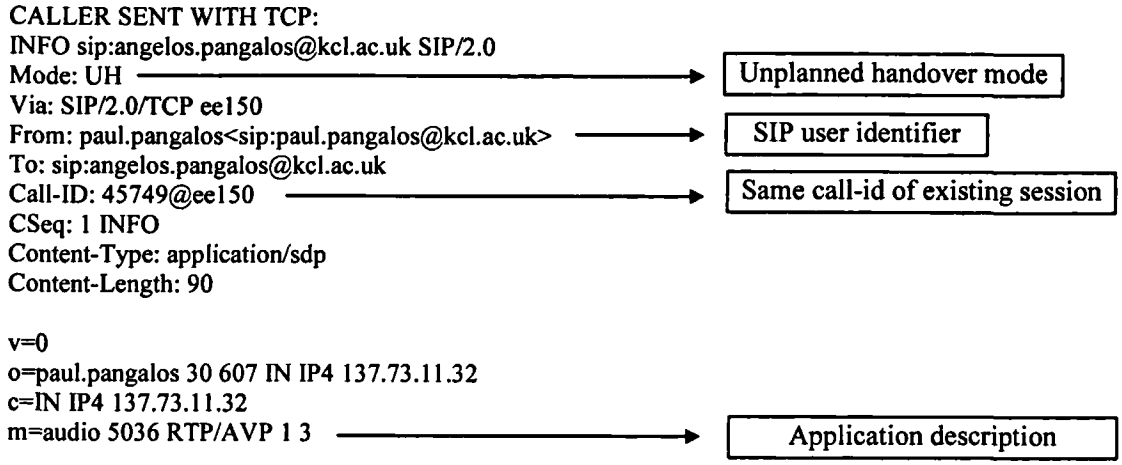


Figure 3-7: SIP INFO message for unplanned vertical handover

3.2.5 Varying the Initial Data Rate

In this section, we investigate how the signalling delay varies with respect to the initial session of the users. The scenario was repeated using various initial data rates between the two users. The results are shown in Table 3-3.

USER A / Data Rates		Interface Setup	Registration + Obtain COA / ~ Time (seconds) [A]	Link Congestion / ~ Time (seconds) [B]			Session re-negotiation / ~ Time (seconds)[C]			Total Delay / Approximate ~ Time (seconds) [A+B+C]
Initial Session	After Handover	GSM		Upward Vertical Handover						
		Initialization		1	2	3	1	2	3	Range
1.4Mbps	5.6 kbps	~ 4 sec		Greater than 52 seconds			Connection not established (CNE)			> 54
512kbps	5.6 kbps	Dialup		Greater than 52 seconds			Connection not established (CNE)			> 54
256kbps	5.6 kbps	~ 4 sec	~ 2 sec	> 52	37.3	> 52	CNE	5.8	CNE	45.1 – > 54
128kbps	5.6 kbps	PPP		32.1	26.9	30.8	5.2	5.2	5.9	34.1 – 39.3
64kbps	5.6 kbps	~ 28 sec		30.5	29.0	30.2	5.1	5.1	5.3	36.1 – 37.6
32kbps	5.6 kbps	Total		16.3	15.0	15.5	5.9	5.2	6.0	22.2 – 24.2
13.2kbps	5.6 kbps	~ 36 sec		4.6	4.7	5.6	5.2	5.4	5.2	11.8 – 12.8

Table 3-3: Varying the initial data rate

It is clear that link congestion during upward handovers is approximately proportional to the data rate of the initial session between the two users. Figure 3-8 shows an upward handover from an initial rate of 1.4Mbps down to 5.6kbps. Once the new GSM link is established it immediately gets congested with a huge amount of traffic arriving from user B. This makes it almost impossible for user A to establish a connection to user B. On the other hand, user B keeps on sending packets at a high data rate making the situation worse. The root of the problem seems to be that the SIP INFO message is not

received in time, causing link congestion. It is vital that during the handover both users stop transmitting until the new SIP session description is received. In this example, the key reason why the SIP-INFO signal is not received fast enough to inform User B about the handover, is due to the long latency times found in slow wireless links like the GSM network. On these kind of links, even a small packet exchange causes TCP to wait for an acknowledgment, taking a full round-trip before the signalling is finished. A round-trip over GSM is typically around 1.4 seconds. The SIP INFO message requires a full round-trip before it is sent when using TCP. The graph shows clearly the congestion delay caused by not receiving the signalling on time.

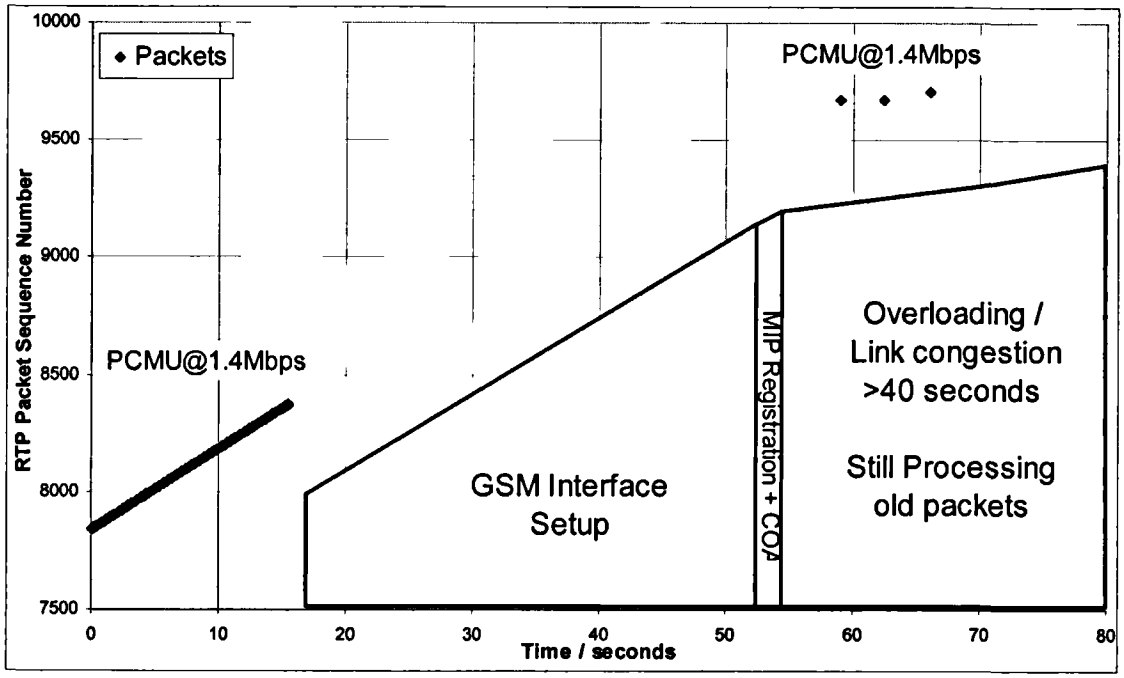


Figure 3-8: Upward unplanned vertical handover
ETH (PCMU – 1.4Mbps) to GSM (LPC - 5.6kbps)

The congestion is improved when the experiment is repeated using a lower initial data rate. Figure 3-9 shows the results of a handover from an initial data rate of 32kbps down to 5.6kbps. In this case it is clear that the congestion time is significantly reduced to around 16 seconds resulting in a total signalling delay of 22 seconds.

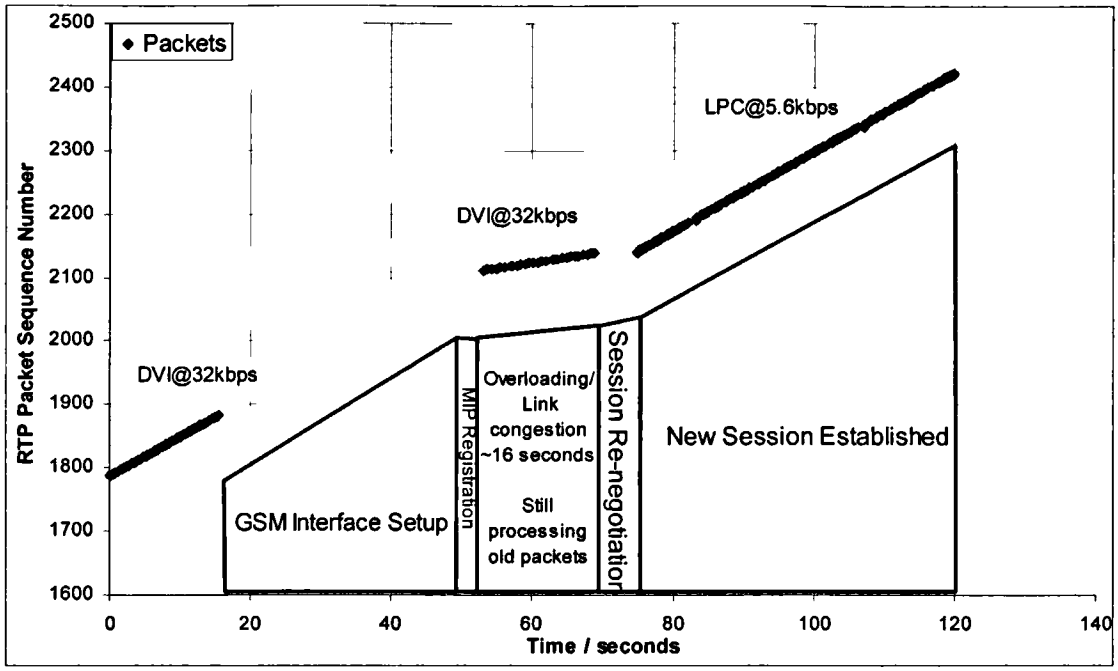


Figure 3-9: Upward unplanned vertical handover
ETH (DVI – 32kbps) to GSM (LPC - 5.6kbps)

Overall, the results show that the link congestion is proportional to the initial session between the two users. Figure 3-10 summarises the results presented in Table 3-3. For high data rates between 256kbps and 1.4Mbps a congestion of over 50 seconds was observed. During this time a TCP connection could not be established and as a result, the SIP-INFO was not sent. However, for lower data rates, the SIP-INFO did go through eventually. The lowest congestion delay was recorded at 13.2kbps measuring five seconds on average.

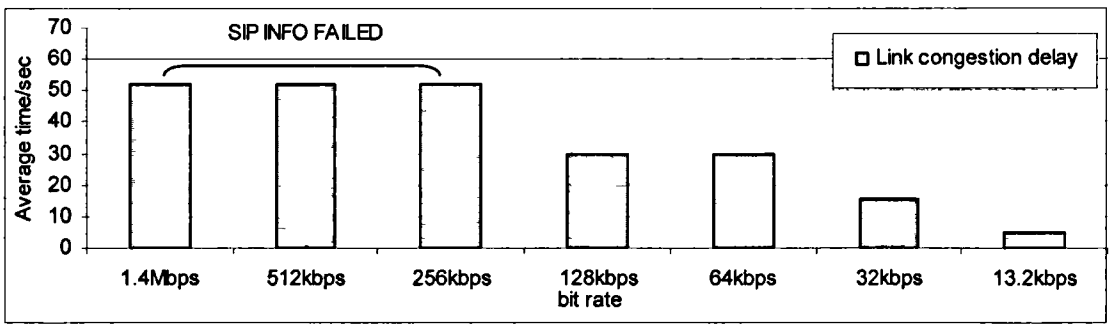


Figure 3-10: Link congestion delay

3.2.6 Downward Vertical Handovers

In this section, we test the performance of our scenario during downward vertical handovers. This is a handover to an overlay with a smaller cell size (high bandwidth per unit area) e.g. GSM (9.6kbps) to WLAN (10Mbps). In this scenario user A is initially connected to the GSM network through the private domain as shown in Figure 3-2, and is in a two-way, real-time RTP audio call with user B. Both users are using the LPC codec and transmitting at 5.6kbps.

A disconnection is initiated forcing user A to perform a downward vertical handover to its home network. Figure 3-11 shows the results. The graph was obtained by monitoring the RTP packets received by user A while performing the downward vertical handover. We can see from the graph that the whole process takes around 4.2 seconds to complete. This is the total time taken for user A to register with its home agent, re-negotiate the session and establish the new connection with user B. This is significantly better than the upward handover because all communications are speeded up – it is quicker to register with the home agent, and it is easier to signal the SIP-INFO message to B.

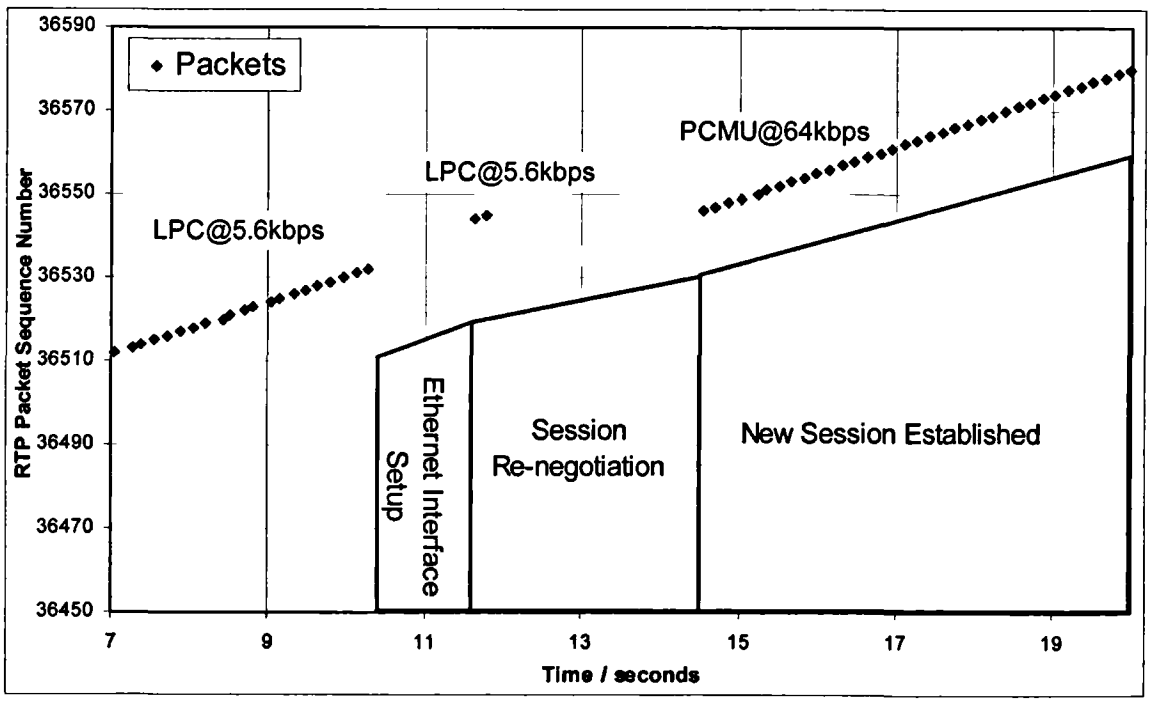


Figure 3-11: Downward Vertical Handover
GSM (LPC – 5.6kbps) to ETH (PCMU - 64kbps)

Varying the initial data rate

Table 3-4 also shows the results obtained during disconnections when varying the initial data rate. No link congestion is observed and the total re-negotiation delay equals the time taken for user A to register at its home network and load the appropriate applications. In our case this was found to be 4.2 seconds for the downward vertical handovers. The session re-negotiation is more or less the same and approximately four to six seconds, depending primarily on the processing power of the end terminals. The re-negotiation involves shutting down the application and re-loading it with the requested audio codecs.

USER A / Data Rates		Interface Setup	Registration + Obtain COA / ~ Time (seconds) [A]	Link Congestion / ~ Time (seconds) [B]	Session re-negotiation / ~ Time (seconds) [C]	Total Delay / Approximate ~ Time (seconds) [A+B+C]
Initial Session	After Handover	Ethernet	Downward Vertical Handover			
5.6kbps	Any	1.3		0	2.8	4.1

Table 3-4: Downward vertical handovers

3.2.7 Summary

We have looked at disconnections in a mobile IP environment and how the session initiation protocol is used to assist terminals to re-negotiate the session during both upward and downward vertical handovers [11]. The key issue is that of congestion caused by the signalling not being received on time.

In the following sections, we focus on two other methods of improving this delay for the loose coupling scenario. In particular, the first method involves using UDP signalling instead of TCP, and the other uses the home agent to take care of the signalling messages during unplanned handovers in an attempt to reduce this delay even more.

3.3 Method A: UDP Based Transport Signalling

In this section, we look at the approach of using UDP signalling with the aim of reducing the congestion delay.

3.3.1 UDP Signalling

TCP signalling requires a three-way handshake to set up a connection whereas UDP requires no handshaking between sending and receiving transport-layer entities before sending a packet. This is shown in Figure 3-12. A TCP connection establishment requires the following steps. In the first part, user A sends a TCP packet with the SYN flag set. B then responds with SYN/ACK packet. When A receives the response it completes the final part of the three-way handshake and returns an acknowledgement to B. On low bandwidth wireless links such as the GSM network, even a small packet exchange would cause TCP to wait for an acknowledgment, taking a full round-trip before the signalling is finished. A round-trip over GSM is typically around 1.4 seconds.

UDP signalling is expected to improve the signalling delay since it is a connectionless protocol with no flags and no acknowledgements of receipt. By using the UDP protocol for the SIP-INFO signalling we expect to improve the link congestion significantly by providing a faster signalling response to the correspondent host during an upward vertical handover. The operation of the SUA was modified to use the UDP protocol to transport all the SIP-INFO messages. Retransmission mechanisms and acknowledgements are also required to ensure the SIP-INFO message arrives at the destination.

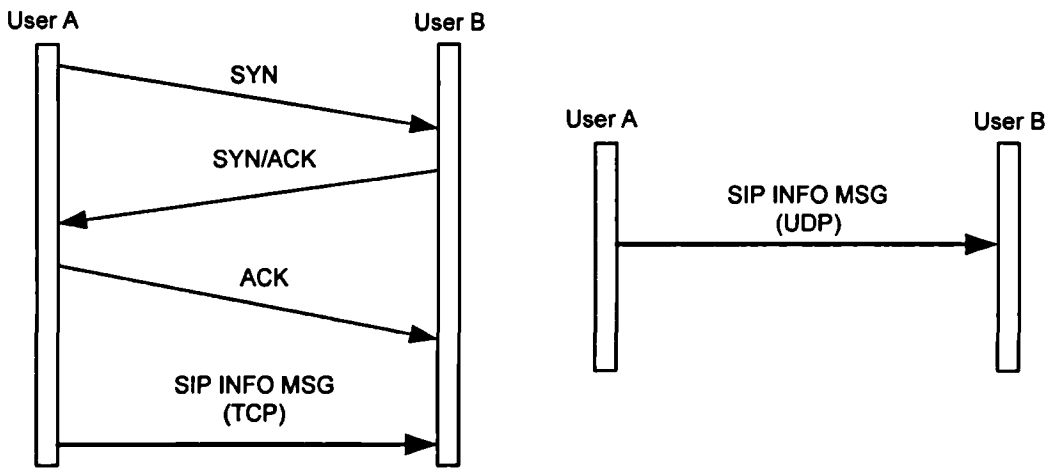


Figure 3-12: TCP versus UDP signalling

3.3.2 Results

The experiment described in Section 3.2.3 was repeated three times for a range of initial data rates between the users. The results obtained for both upward and downward unplanned vertical handovers are illustrated in Table 3-5

USER A / Data Rates		Interface Setup	Registration + Obtain COA / ≈ Time (seconds)[A]	Link Congestion / ~ Time (seconds) [B]			Session re-negotiation / ~ Time (seconds) [C]			Total Delay / Approximate ~ Time (seconds) [A+B+C]
Initial Session	After Handover	GSM	Upward Vertical Handover							
		Initialization		1	2	3	1	2	3	Range
1.4Mbps	5.6kpbs	≈ 4 sec		16.4	18.8	17.2	5.5	5.8	5.7	23.9 –26.6
512kbps	5.6kpbs	Dialup		10.1	12.3	12.8	5.4	5.7	6.2	17.5 – 21.0
256kbps	5.6kpbs	≈ 4 sec	≈ 2 sec	12.1	14.6	13.5	5.8	5.2	5.6	19.9 – 21.8
128kbps	5.6kpbs	PPP		14.8	14.4	14.6	5.9	4.9	5.8	21.3 – 22.7
64kbps	5.6kpbs	≈ 28 sec		6.8	13.2	15.5	5.6	5.0	4.8	14.4 – 22.3
32kbps	5.6kpbs	Total		3.8	7.4	5.6	5.1	6.1	5.4	10.9 – 15.5
13.2kbps	5.6kpbs	≈ 36 sec		1.4	1.7	1.5	5.9	5.3	5.8	9.0 –9.3
		Ethernet	Downward Vertical Handover							
5.6kbps	Any	1.3		0			2.7			4

Table 3-5: Summary of results - using UDP signalling for SIP INFO.

The results when compared to Table 3-3, clearly show an improvement in the congestion time for all data rates. Significant improvements were especially observed for the higher data rates. User B receives the signal message much faster, and promptly stops sending data to user A, significantly improving the link congestion delay. The graph shows the UDP signal is received much sooner than the TCP, recovering the session faster. During downward vertical handovers, both TCP and UDP methods perform the same. The time taken to handover to the Ethernet network after a disconnection is approximately equal to 2.7 seconds. This is the total time taken for user A to update its home agent, send the SIP INFO message and re-negotiate the session with user B. As with the previous method there is no link congestion since the new network will always support the existing session coming from the lower bandwidth network (i.e. GSM).

Figure 3-13 shows the results obtained for one of the experiments during an unplanned vertical handover for an initial data session of 64kbps. The congestion delay was measured at 9.8 seconds compared to 29 seconds when the TCP based signalling was used. Although this approach reduces the congestion delay significantly, the user still experiences unacceptable congestion delays of over ten seconds for data rates above 64kbps. Ideally, the only delay the user should experience is the one caused by the session re-negotiation. Figure 3-14 compares both the UDP and TCP based signalling

approach averaging the results obtained in Table 3-3 and Table 3-5. A significant improvement in link congestion was recorded throughout all the data rates with most improvements observed at data rates of 1.4Mbps, 512kbps and 256kbps. The maximum congestion for UDP signalling was found to be 18 seconds at a data rate of 1.4Mbps while less than two seconds delay was recorded for the lowest data rate session of 13.2 kbps.

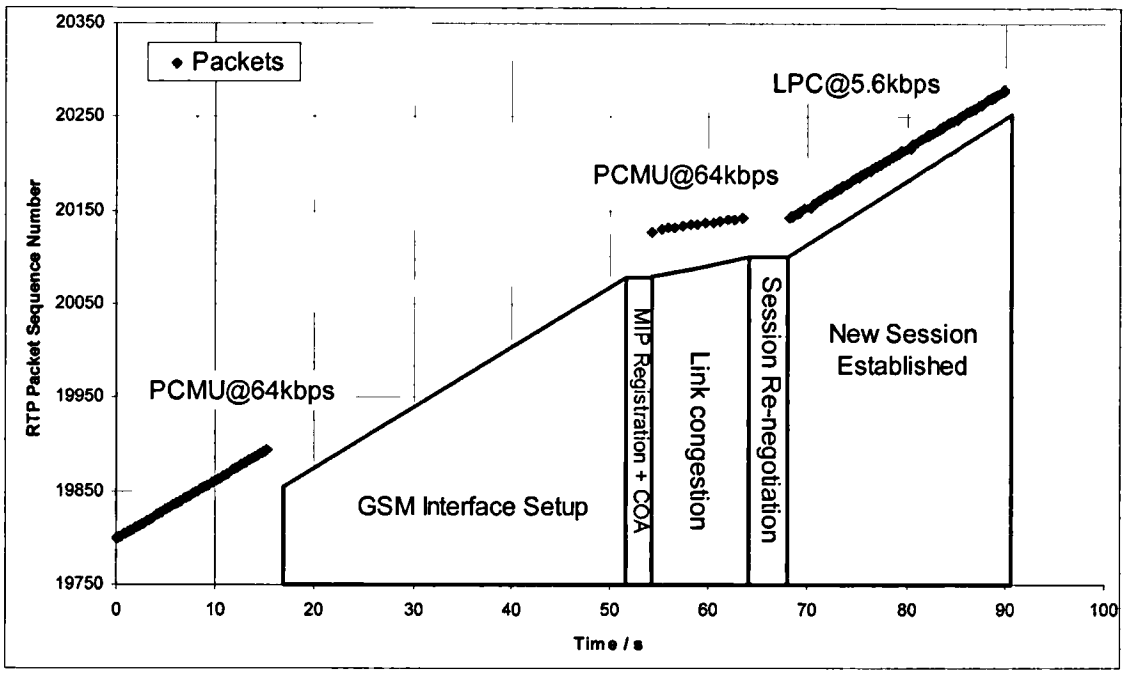


Figure 3-13: Upward unplanned vertical handover – UDP SIPINFO
ETH (PCMU - 64kbps) to GSM (LPC - 5.6kbps)

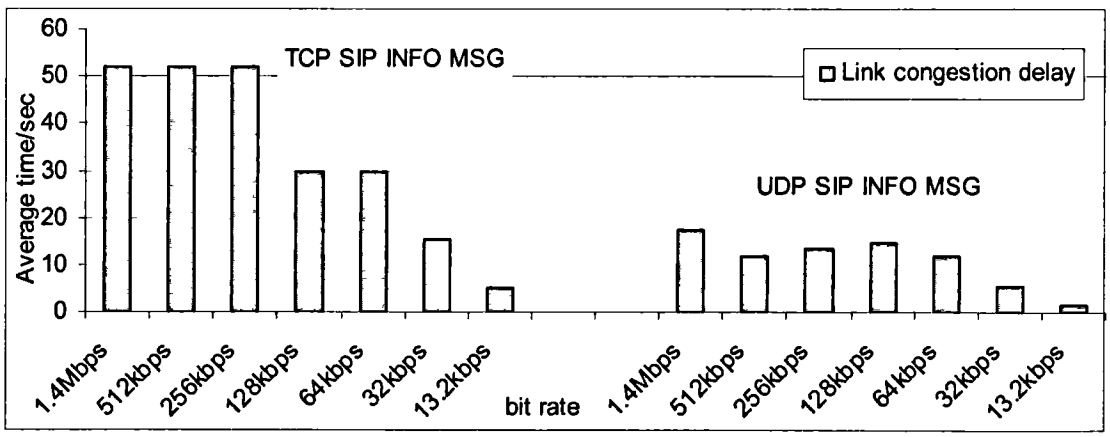


Figure 3-14: Congestion delay for TCP and UDP SIP INFO signalling

3.4 Method B: Using the Home Agent as a Signalling Proxy

The idea here is to use the home agent for sending the SIP INFO messages during an unplanned handover. This approach is an attempt to reduce further the link congestion problem even at high bit rates.

3.4.1 Implementation Description

This section gives a technical explanation of the loose inter-working approach between mobile IP and sip using the home agent as a signaling proxy. In this approach when a disconnection occurs the mobile node is no longer responsible for informing the correspondent host. This function is instead performed by the home agent. The overall low level functionality is illustrated in Figure 3-15. In order to provide this functionality to the home agent some additional methods were developed.

The first method called 'add element' is located at the home agent and listens for incoming messages from the mobile node. Once the mobile node initiates a session it sends a message to the home agent containing its own IP address and the IP address of its correspondent host. This is done through the method called 'sendip_to_homeagent' located in the mobile node. The home agent maintains a cache table that keeps information for each mobile node registered with it. The table is responsible for storing the IP address of a mobile node and the address of nodes with which the mobile node is communicating. During an unplanned handover, the mobile node configures and initialises the new interface. This is followed by a registration message sent to the home agent. Upon receiving this message the home agent, consults the cache table and sends automatically the SIP INFO message to the correspondent host using the 'sendkill' method.

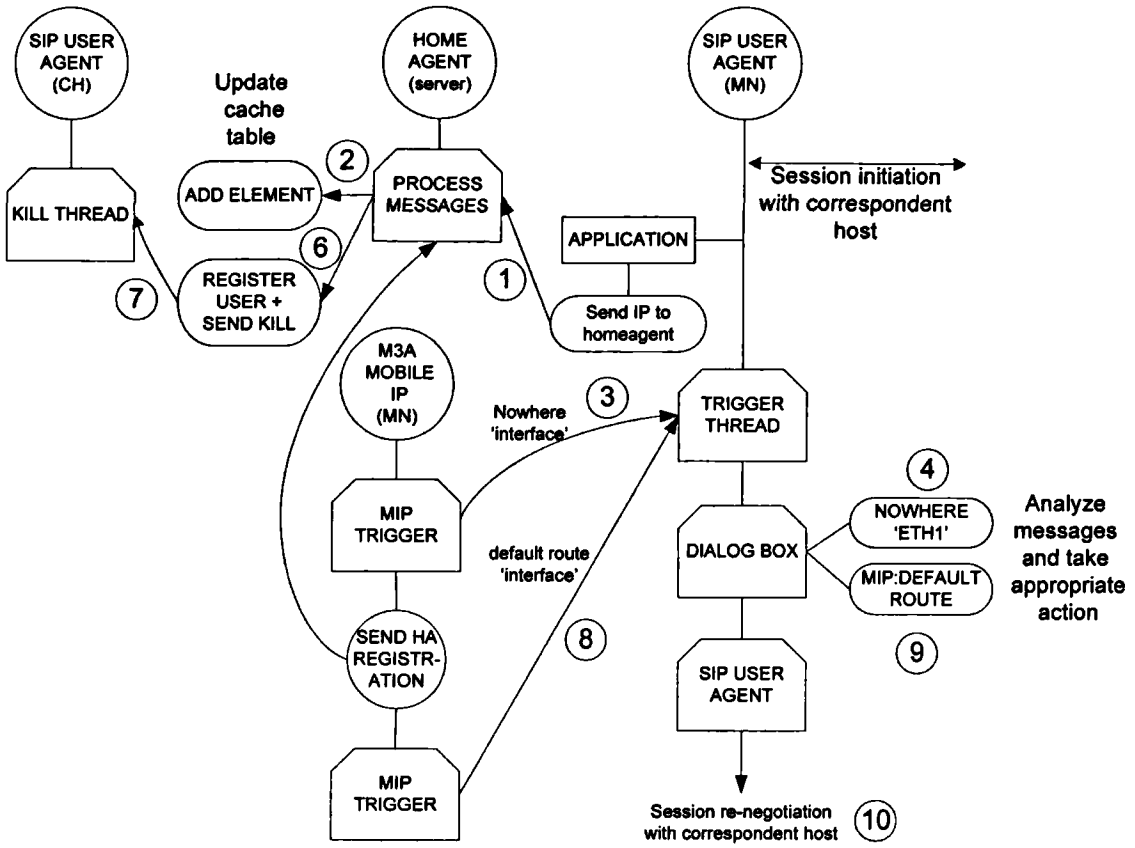


Figure 3-15: Software implementation – home agent as a signalling proxy

3.4.2 The Signalling Mechanism

In this new revised design a SIP mapping cache table is added to the home agent. This is used to monitor and keep track of all active SIP sessions of its registered users. Figure 3-16 shows user A connected to its home network, and about to initiate a two-way, real-time audio call with user B. Once the call has been established A signals its home agent to update the cache table by adding the IP address of its correspondent host user B. An unplanned handover is initiated by disconnecting the Ethernet interface. During the handover, a request is send to the home agent to register the care of address. After successful registration, the home agent consults the cache table, obtains the IP address and sends the SIP INFO message directly to B. Meanwhile the SUA of user A receives up to date information about the new network connection, matches the appropriate payload types and generates a re-invite message to describe the new session. A new session is established and both users start transmitting using the agreed codecs. Once the session is finished User A signals the home agent to clear its entry from the SIP cache table.

In this approach the signaling messages are always sent on time irrespective of the initial bit rate of the Users. As a result User B always receives the signaling on time and promptly stops transmitting, avoiding congestion altogether [12].

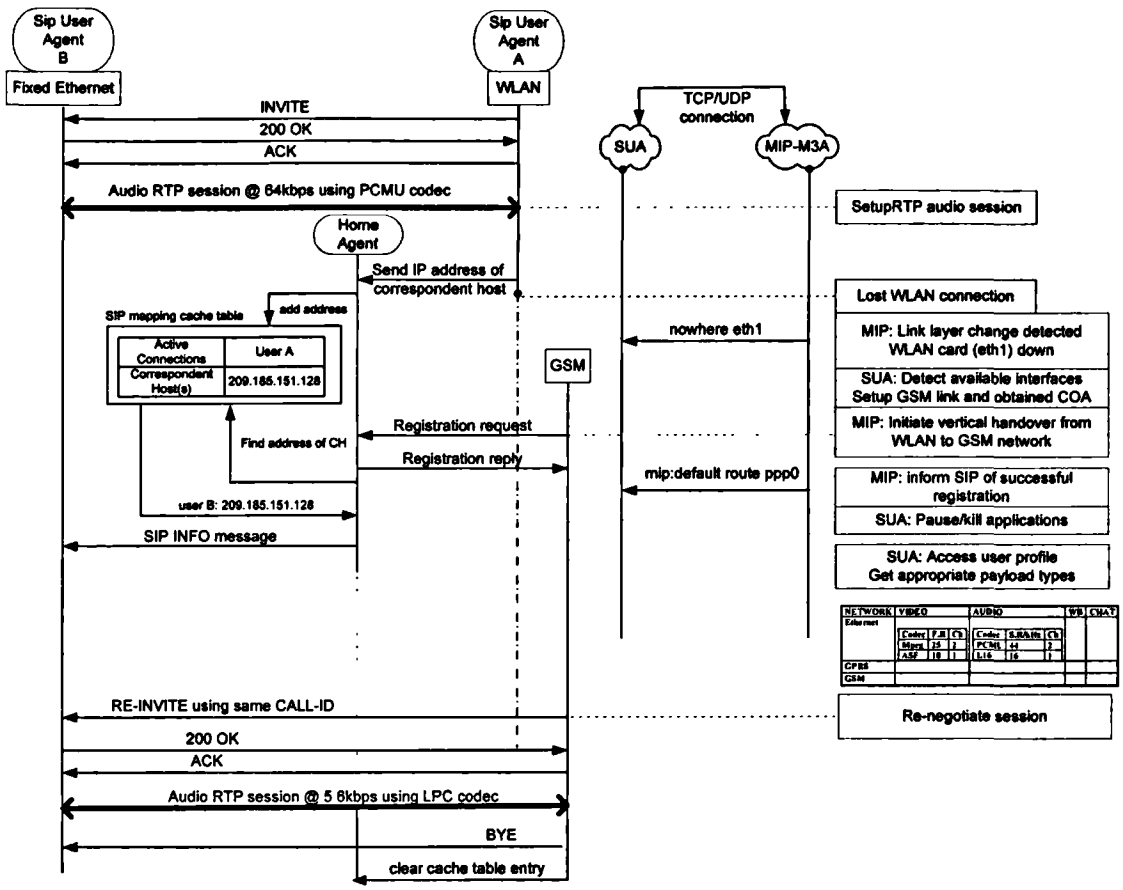


Figure 3-16: Home agent used as a signalling proxy

3.4.3 Results

The experiment described in Section 3.2.3 was once more repeated three times for a range of initial data rates between the users. The results obtained for both upward and downward unplanned vertical handovers are summarised in Table 3-6.

USER A / Data Rates		Interface Setup	Registration + Obtain COA / ≈ Time (seconds)[A]	Link Congestion / ~ Time (seconds) [B]			Session re-negotiation / ~ Time (seconds) [C]			Total Delay / Approximate ~ Time (seconds) [A+B+C]
Initial Session	After Handover	GSM	Upward Vertical Handover							
		Initialization		1	2	3	1	2	3	Range
1.4Mbps	5.6kpbs	≈ 4 sec		8.8	8.1	7.8	5.5	5.2	5.6	15.4 –16.3
512kbps	5.6kpbs	Dialup	≈ 2 sec	6.7	4.7	5.2	5.7	5.2	5.0	11.9 – 14.4
256kbps	5.6kpbs	≈ 4 sec		4.2	4.5	3.1	4.8	5.2	5.7	10.8 – 11.7
128kbps	5.6kpbs	PPP		1.2	1.7	1.5	5.2	5.6	5.2	8.4 – 9.3
64kbps	5.6kpbs	≈ 28 sec		1.5	1.5	1.4	5.9	5.2	5.0	8.4 – 9.4
32kbps	5.6kpbs	Total		0	0.5	0	6.0	5.9	5.9	7.9 – 8.4
13.2kbps	5.6kpbs	≈ 36 sec		0.6	0	0	5.7	5.5	5.8	7.5 – 8.3
		Ethernet	Downward Vertical Handover							
5.6kbps	Any	1.2		0			2.6			3.8

Table 3-6: Summary of results – home agent as a signalling proxy

Figure 3-17 also shows the handover procedure observed by monitoring the downlink of the mobile node. The link congestion is significantly reduced compared to the two previous results. The session re-negotiation is more or less the same and approximately equal to five seconds. Overall, it seems that using the home agent as the signalling proxy approach outperforms the other two modes. This is compared in Figure 3-18. It provides low congestion delays at high bit rates while minimizing the link congestion at lower bit rates. During an unplanned handover at an initial session of 256kbps, the home agent approach results in a link congestion delay of less than five seconds while the UDP end-to-end approach is bigger by a factor of three with an average delay of around fifteen seconds. However, the TCP approach performs the worst, resulting in a huge amount of congestion making it impossible for user A to communicate with B.

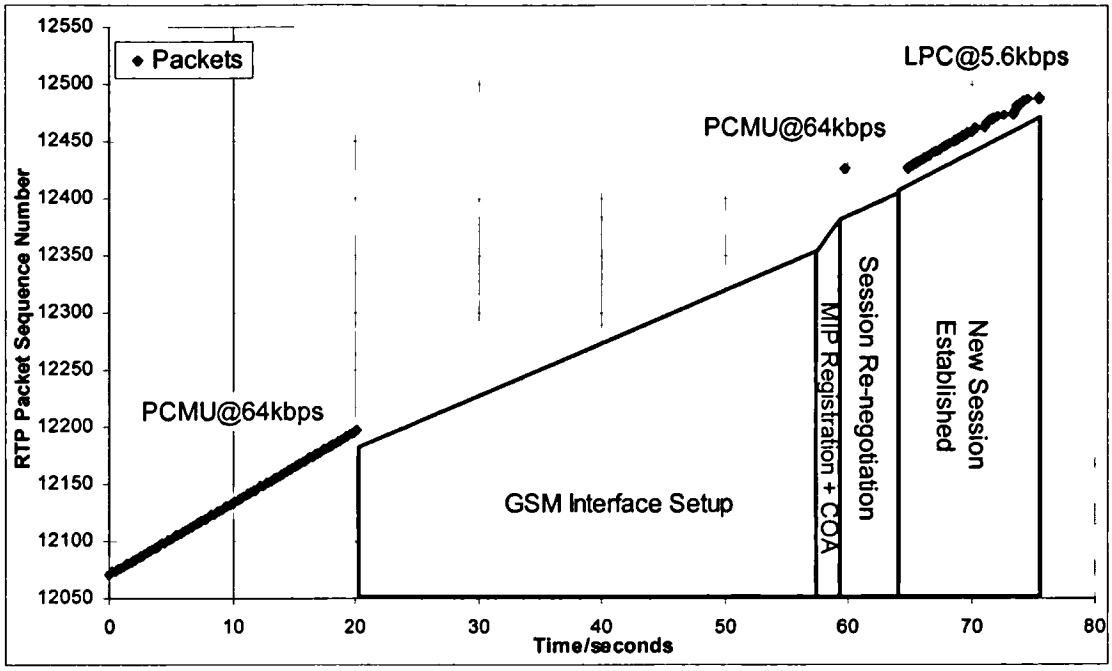


Figure 3-17: Upward vertical handover - Ethernet (pcmu - 64kbps) to GSM (lpc – 5.6kbps)

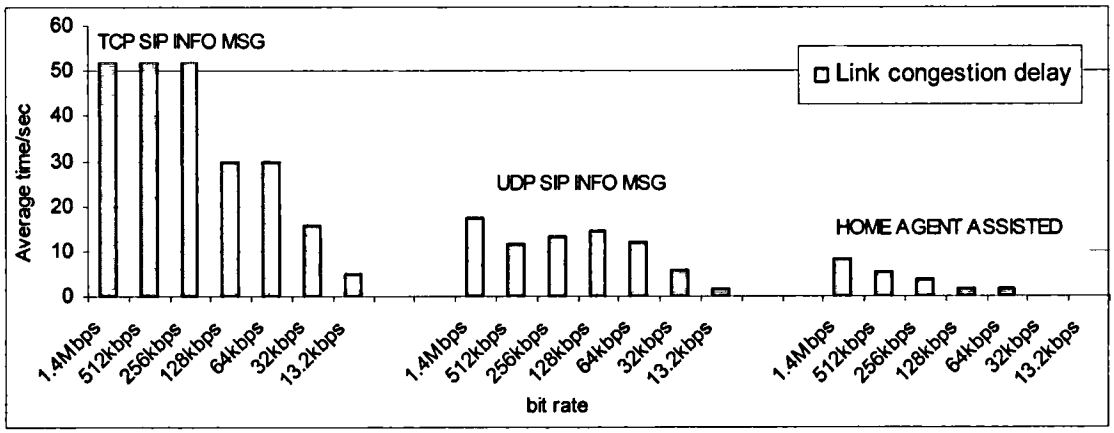


Figure 3-18: Comparison of loose coupling approaches

3.4.4 Downward Vertical Handovers using the Home Agent

The performances of all three signaling methods are the same during a downward vertical handover. The time taken to handover to the Ethernet network is approximately 2.7 seconds and is the same in all signaling modes and all bit rates. This is the total time taken for user A to register with its home agent, re-negotiate the session and establish the new connection with user B. This is significantly better than the upward handover because all communications are speeded up – it is quicker to register with the home agent, and it is easier to signal the change to B.

During downward vertical handovers there is no link congestion since the new network will always be able to support any existing session coming from a lower bandwidth connection.

3.5 Tight Coupling between Mobile IP and SIP

We have so far looked at different ways of improving unplanned handovers using the loose coupling approach between MIP and SIP. The last option described how the home agent was used to assist with the signalling. This eliminated the congestion delay for low bit rates but not for higher rates. This section, however, examines a rather different approach, that of tight coupling or otherwise referred to as integration of the two protocols. This gives higher flexibility avoiding congestion altogether. The two agents, residing in the users terminal, communicate in a bidirectional, synchronised manner exchanging appropriate information during unplanned vertical handovers as shown in Figure 3-19.

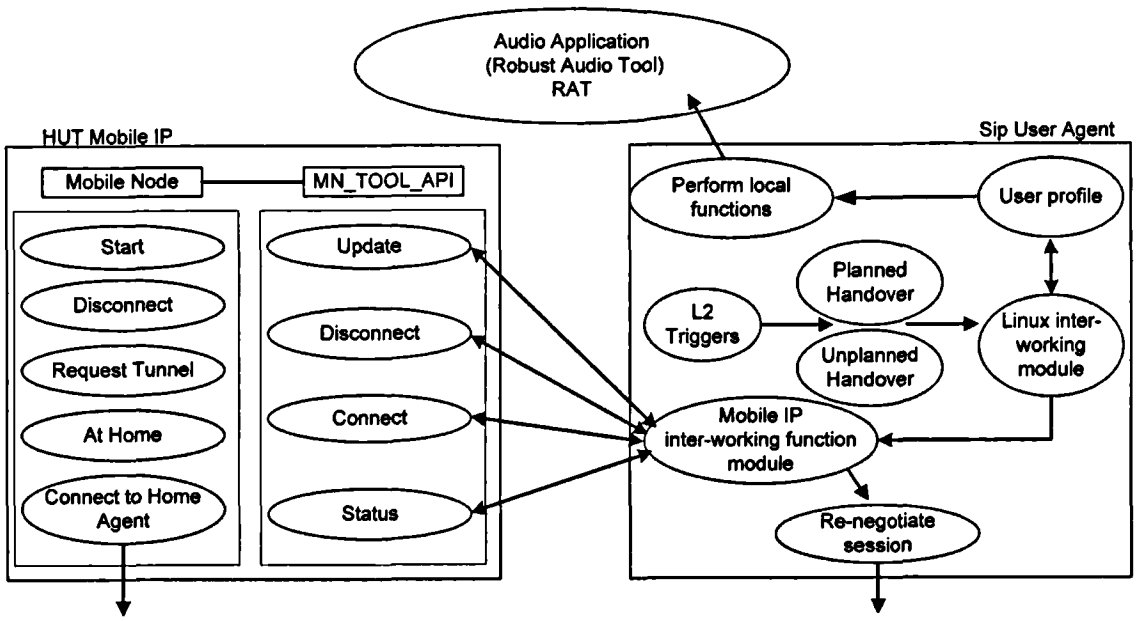


Figure 3-19: Modular architecture of the tight coupling approach

3.5.1 Description of Inter-working Functions

The mobile node is controlled via the mn_tool_api interface using the interworking function within the sip user agent. The mn_tool_api allows the SUA to retrieve status information from the mobile node, connect, disconnect the home agent and perform location updates. The update function is called on a specific interface enabling the

mobile IP agent to scan the configuration and retrieve the IP address of that interface. The new IP address would subsequently be used as the new care of address for the mobile node. The disconnect function tears down the IP-IP tunnel, ending the communication between the home agent and mobile node, while the connect function is used to register the mobile node with the home agent. Finally, the status function provides feedback to the sip user agent as to the state of the mobile node. Status information includes current active IP addresses, indication of successful registrations and the state of IP tunnels.

3.5.2 Experimental Testbed Environment

The following testbed configuration was used to test and evaluate the tight coupling scenario between MIP and SIP. Figure 3-20 depicts our experimental environment which is very similar to the one used for the loose coupling scenario.

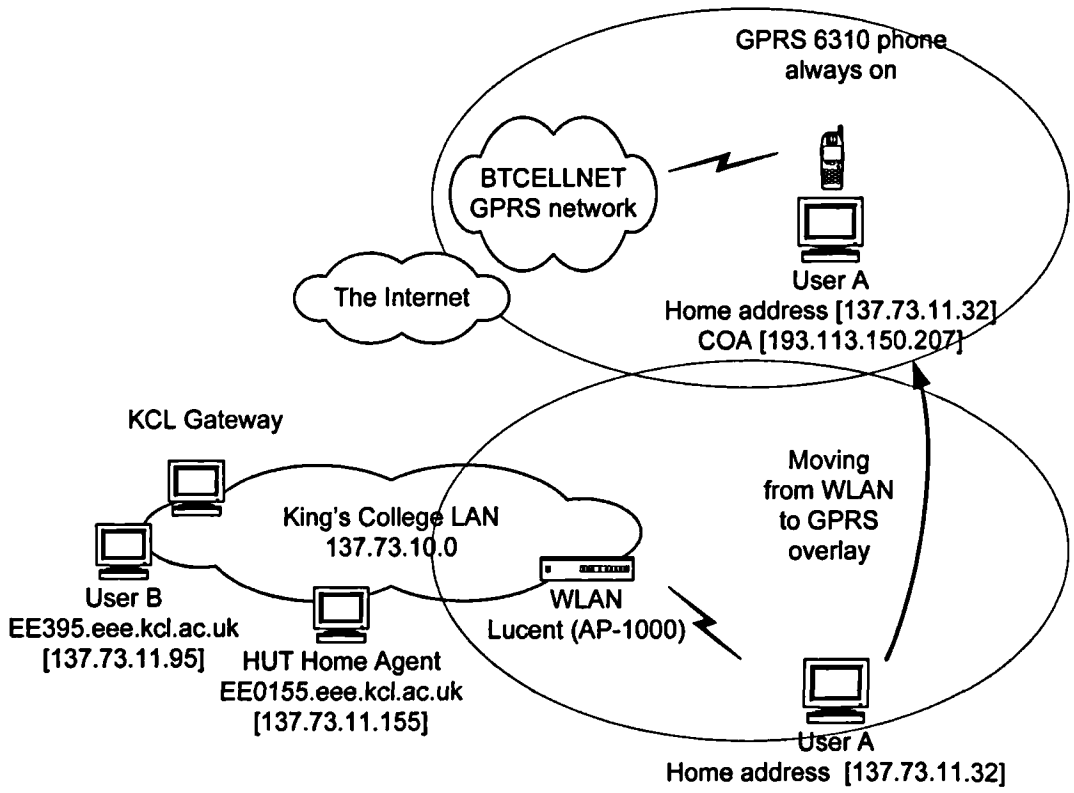


Figure 3-20: Experimental environment for the tight coupling technique

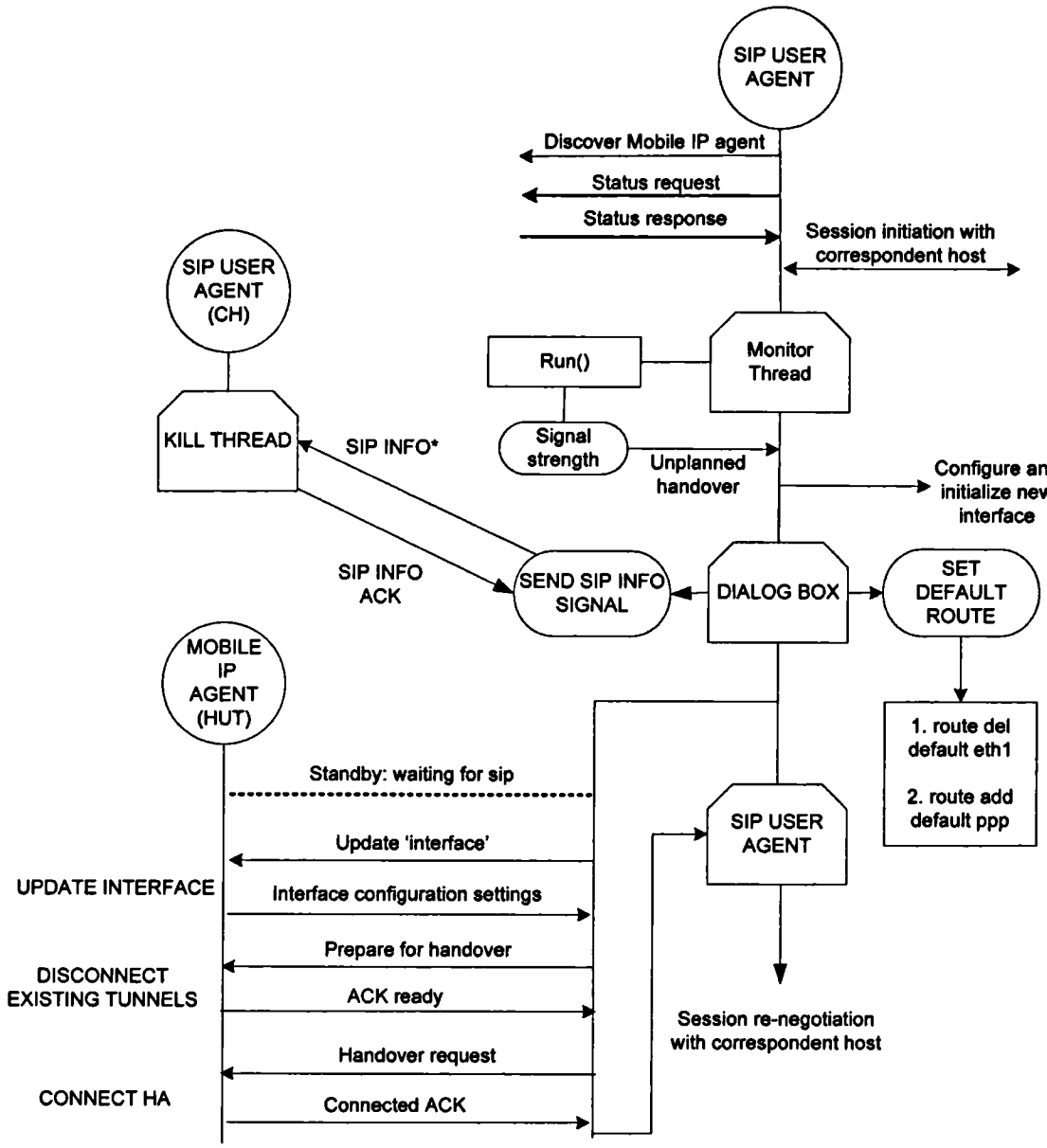
The mobile terminal was upgraded to kernel version 2.4.18 and the GSM interface replaced with a GPRS interface. The sip user agent was also extensively modified to support and interact in a bidirectional manner with HUT mobile IP instead of the M3A version. Table 3-7 gives a summary of the experimental environment.

Summary	Tight coupling technique
Home agent: Using HUT mobile IP with full tunnelling mode, running on Linux Redhat – kernel-2.4.18.	
Correspondent host: Using RAT for audio on Linux Redhat – kernel 2.2.10.Connected to KCL LAN	
Home Network: 137.73.10.0, Netmask 255.255.254.0	
Mobile node: Multihomed terminal using Acer travelmate 710 with WLAN and GPRS interfaces. See Appendix Section 1.1: Configuration and setup of the GPRS connection	

Table 3-7: Summary of experimental environment

3.5.3 Implementation Description

This section gives a technical explanation of the tight coupling mechanism between mobile IP and SIP designed for handling unplanned vertical handovers. All functions and methods are illustrated in Figure 3-21. The monitor thread class is responsible for monitoring and initiating vertical handovers. Once an unplanned handover is initiated the sip user agent detects other available interfaces on the terminal, checks the user profiles, selects, configures and initialises the new network interface. On successful registration, the 'dialogbox class' responds by updating and configuring the kernel routing table with the default route of the new network interface, removing the old one. This is followed by the SIP INFO message sent to the correspondent host. Soon after the ACK is received, the 'dialog box' class informs mobile IP about the new chosen active interface. The mobile agent probes the specified interface, configures itself and returns an update message to the sip user agent containing the new care of address associated with that interface. The sip user agent would also update its variables with the new care of address and send a handover preparation message to mobile IP. This message alerts mobile IP to prepare for a handover. At this stage mobile IP disconnects any active tunnels and replies through a status message. SIP then issues the final handover command causing mobile IP to register the new interface with the home agent. As soon as the registration is complete, mobile IP notifies the sip user agent initiating a session re-negotiation with the correspondent host.



*Implementation notes: 1. The SIP INFO was simplified by implementing it as a simple UDP message.

Figure 3-21: Software implementation – tight coupling approach

3.5.4 Handover Procedure

The synchronisation and the bi-directional communication of the two protocols are the two key features of this approach that put together to eliminate the problem of congestion. The tight coupling handover mechanism is shown in Figure 3-22. During the start-up process, the sip user agent discovers the presence of the mobile IP agent and sends a status request message. A response message from the mobile IP agent would indicate a successful bidirectional communication between the two agents. Unplanned handovers are detected by the link layer (L2) trigger module at the sip user agent. This module runs as a separate thread, constantly monitoring the signal strength of the

terminal's wireless interface(s), initiating both planned and unplanned handovers. In this experiment, the unplanned handover is triggered by turning off the WLAN base station or unplugging the active interface. The rapid decrease in signal strength is detected by the L2 trigger module, initiating an unplanned vertical handover to the GPRS network. Since the GPRS interface is already initialised and configured, the SUA proceeds to update the kernel routing tables changing the default route to the GPRS interface. As a result subsequent signalling would go over the GPRS interface and not through the WLAN. Once the new default route is set, user A sends the first SIP INFO signal message to the correspondent host over the GPRS interface using the care of address as the source address. Unlike the previous approaches, the registration with the home agent is co-ordinated and managed by SIP. Once the SIP INFO ACK is received the sip user agent instructs the mobile IP agent to prepare for the registration procedure with the home agent. During this phase, any active tunnels will be disconnected, advancing the mobile IP agent to the handover ready state. The registration progress is monitored by the sip user agent through the status updates received from mobile IP. Finally, on successful completion, the SUA re-negotiates the current session with the correspondent host.

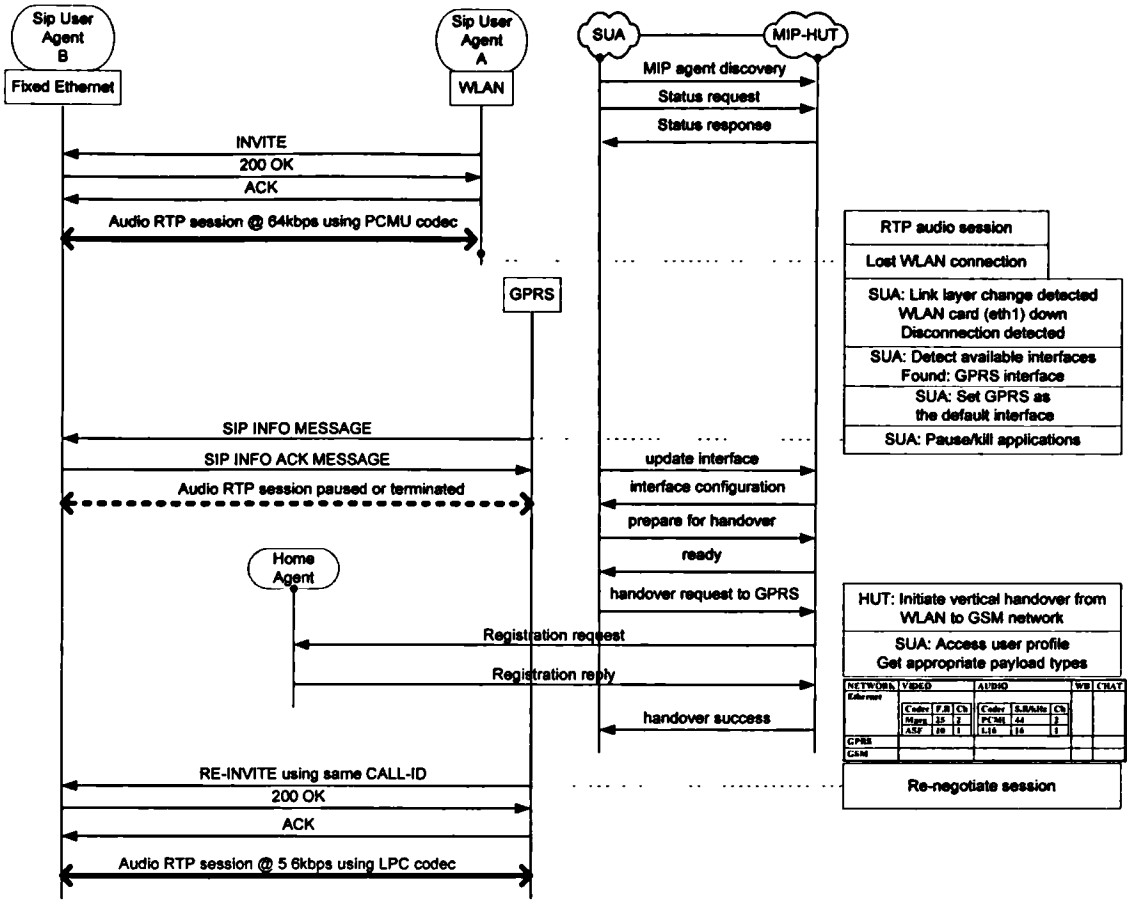


Figure 3-22: Tight coupling signalling mechanism

3.5.5 Results

The following section presents the results of one of the experiments performed with an unplanned vertical handover and based on the tight coupling scenario. The result of the handover is shown in Figure 3-23. Both users are communicating at 64kbps when user A experiences a disconnection at t=13.9sec. The sip agent scans all available interfaces selecting the most appropriate one, based on user profiles. It then updates the Linux kernel routing table by replacing the old Ethernet route with a new default route pointing to the GPRS interface. The detection and routing table update takes a total of two seconds to complete. At t=16sec the SIP INFO message is composed and sent to the correspondent host. Once the acknowledgment is received the sip user agent instructs the mobile IP agent to register with the home agent. During this phase the home agent updates its database to reflect the new GPRS care of address, and attempts to register with the home agent. The registration process lasts three seconds and completes at t=21sec. The successful registration is also relayed at t=21sec to the sip user agent using the mobile IP status report messages. At t=22 the sip user agent initiates a session renegotiation with the correspondent host requesting the session to adapt to a low bitrate

using the LPC codecs at 5kbps. This phase takes 6.5 seconds to complete with the first packet of the modified session received at t=33 over the GPRS network. The next section describes in detail the data received during this experiment.

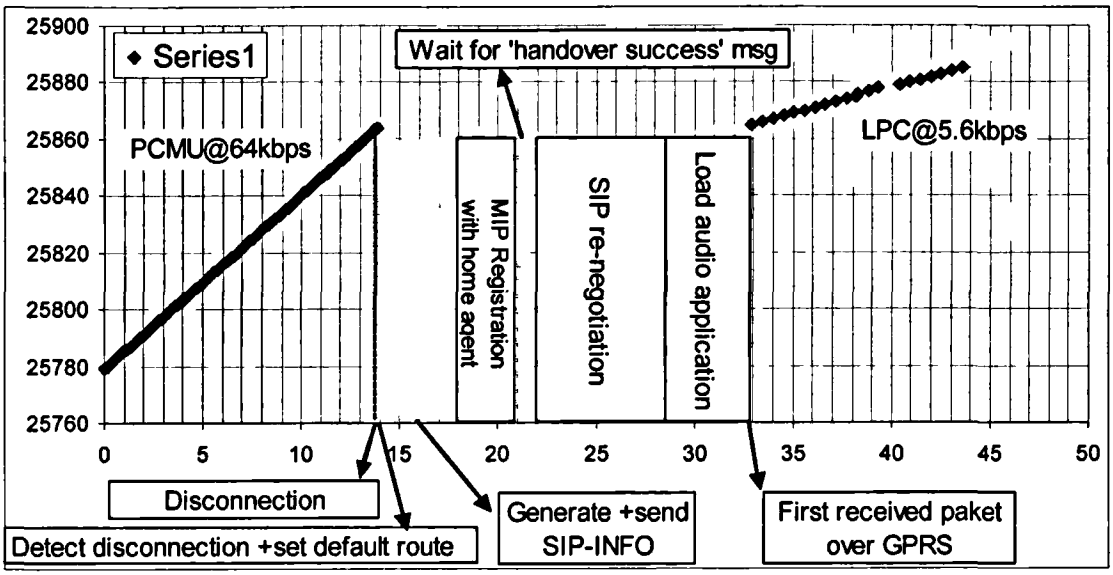


Figure 3-23: Upward unplanned vertical handover – tight coupling
ETH (PCMU - 64kbps) to GSM (LPC - 5.6kbps)

3.5.6 In Depth Description of the Experimental Results

This section presents the results obtained during the unplanned vertical handover experiment presented above. The software tool Tcpdump [13] was used to monitor the signalling exchange and RTP traffic on both the WLAN and the GPRS terminal interfaces as illustrated in Figure 3-24 and Figure 3-25. The sip user agent message exchange was also captured as shown in Figure 3-26. Finally, Figure 3-27 shows the status of the mobile IP agent, before, during and after the handover. The timing information in these results was used to construct the graph presented in Figure 3-23. A detailed explanation is given below using the reference numbers in the figures as a guide.

1. Reference point (1a) shows user A receiving the RTP/UDP audio stream from user B at 64kbps. At this point user A is at his home network using the WLAN interface with an IP address 137.73.11.32. This is shown in (1b)
2. At t= 13:19:29.74 the interface is disconnected causing tcpdump to terminate (2a). The disconnection is detected by the sip user agent 'monitor module',(2b) and the mobile IP agent (2c).

3. The sip user agent updates the Linux kernel routing table setting the gprs interface as the default route (3a). The default interface is successfully registered at t=13:19:31.87 causing mobile IP to send a series of ICMP router solicitation messages trying to discover a foreign agent (3b). However, in our experimental environment, mobile IP was configured to operate using a collocated care of address without the use of foreign agents. Therefore, these ICMP messages get unanswered.
4. Now that the new active route is set, the sip user agent generates and sends the SIP info message using the collocated care of address as the source address (4a). This message is send at t=13:19:34.01 (4b).
5. At this stage the sip user agent signals the mobile IP agent to reconfigure itself to the GPRS interface and prepare for a handover (5a) by disconnecting any active tunnels and re-configuring the following three fields. The local-address and collocated-address fields are both replaced with the new GPRS IP address, while the FA-address field is set to the same IP address as that of the home agent since no foreign agents are used, (5b) and (6c). Once this is complete mobile IP notifies the sip user agent.
6. At t=13:19:3403 the mobile agent receives the request for a handover and proceeds to register with the home agent (6a). The registration lasts approximately three seconds. The signalling messages exchanged are shown in (6b) while the status of the mobile IP agent is also shown in (6c).On successful registration, the sip user agent is notified
7. At t=13:19:38.04 a re-invite message is send to user B. This message contains the same call-id, and home address used in the first invite message. Having the same call-id will indicate at the receiving end (user B) that the new invite refers to the existing session of the user and is not a request for creating a new session. It takes approximately 6.5 seconds for the signalling to complete as shown in (7a-7b).
8. Reference point (8) shows user A receiving the encapsulated RTP/UDP audio packets from user B at 5.6kbps. This handover is completely transparent to user B except for the obvious reduction to the session parameters.

```
13:19:27.781659 137.73.11.95.5036 > EE332.eee.kcl.ac.uk.5036: udp 1292 (DF)
13:19:27.949301 137.73.11.95.5036 > EE332.eee.kcl.ac.uk.5036: udp 1292 (DF)
13:19:28.109285 137.73.11.95.5036 > EE332.eee.kcl.ac.uk.5036: udp 1292 (DF)
13:19:28.273493 137.73.11.95.5036 > EE332.eee.kcl.ac.uk.5036: udp 1292 (DF)
13:19:28.443759 137.73.11.95.5036 > EE332.eee.kcl.ac.uk.5036: udp 1292 (DF)
13:19:28.599220 137.73.11.95.5036 > EE332.eee.kcl.ac.uk.5036: udp 1292 (DF)
13:19:28.759235 137.73.11.95.5036 > EE332.eee.kcl.ac.uk.5036: udp 1292 (DF)
13:19:28.816565 137.73.11.95.5037 > EE332.eee.kcl.ac.uk.5037: udp 60 (DF)
13:19:28.919135 137.73.11.95.5036 > EE332.eee.kcl.ac.uk.5036: udp 1292 (DF)
13:19:29.089133 137.73.11.95.5036 > EE332.eee.kcl.ac.uk.5036: udp 1292 (DF)
13:19:29.249124 137.73.11.95.5036 > EE332.eee.kcl.ac.uk.5036: udp 1292 (DF)
13:19:29.419080 137.73.11.95.5036 > EE332.eee.kcl.ac.uk.5036: udp 1292 (DF)
13:19:29.569060 137.73.11.95.5036 > EE332.eee.kcl.ac.uk.5036: udp 1292 (DF)
13:19:29.739930 137.73.11.95.5036 > EE332.eee.kcl.ac.uk.5036: udp 1292 (DF)
tcpdump: pcap_loop: read: Network is down
[root@ee026 root]#
```

1a

2a

Figure 3-24: Traffic received through wireless LAN interface of the mobile node

```
[root@ee026 root]# ./tcpdump -i ppp0
tcpdump: listening on ppp0
13:19:31.867664 193.113.150.159 > 255.255.255.255: icmp: router solicitation (DF) [ttl 1]
13:19:33.380277 193.113.150.159 > 255.255.255.255: icmp: router solicitation (DF) [ttl 1]
13:19:34.011905 193.113.150.159.32775 > 137.73.11.95.9000: udp 4 (DF)
13:19:34.026993 193.113.150.159.32775 > 137.73.10.155.mobileip-agent: udp 52 (DF)
13:19:35.043319 193.113.150.159.32775 > 137.73.10.155.mobileip-agent: udp 66 (DF)
13:19:35.079976 137.73.10.155.mobileip-agent > 193.113.150.159.32775: udp 46 (DF)
13:19:36.050144 193.113.150.159.32775 > 137.73.10.155.mobileip-agent: udp 66 (DF)
13:19:36.089974 137.73.10.155.mobileip-agent > 193.113.150.159.32775: udp 80 (DF)
13:19:37.139957 137.73.10.155.mobileip-agent > 193.113.150.159.32775: udp 80 (DF)
13:19:38.040576 137.73.11.32.32856 > 137.73.11.96.5060: S 2215530908:2215530908(0) win 5760 <mss
1440,sackOK,timestamp[|tcp]> (DF) (ipip)
13:19:40.149959 137.73.11.96.5060 > 137.73.11.32.32856: S 70918:70918(0) ack 2215530909 win 8640 <mss 1460>
(DF) (ipip)
13:19:40.150088 137.73.11.32.32856 > 137.73.11.96.5060: . ack 1 win 5760 (DF) (ipip)
13:19:40.158776 137.73.11.32.32856 > 137.73.11.96.5060: P 1:348(347) ack 1 win 5760 (DF) (ipip)
13:19:41.289964 137.73.11.96.5060 > 137.73.11.32.32856: . ack 348 win 8293 (DF) (ipip)
13:19:42.879959 137.73.11.96.5060 > 137.73.11.32.32856: P 1:339(338) ack 348 win 8293 (DF) (ipip)
13:19:42.880220 137.73.11.32.32856 > 137.73.11.96.5060: . ack 339 win 6432 (DF) (ipip)
13:19:42.884191 137.73.11.32.32856 > 137.73.11.96.5060: P 348:645(297) ack 339 win 6432 (DF) (ipip)
13:19:42.900326 137.73.11.32.32856 > 137.73.11.96.5060: F 645:645(0) ack 339 win 6432 (DF) (ipip)
13:19:44.039992 137.73.11.96.5060 > 137.73.11.32.32856: . ack 646 win 7996 (DF) (ipip)
13:19:44.559958 137.73.11.96.5060 > 137.73.11.32.32856: F 339:339(0) ack 646 win 7996 (DF) (ipip)
13:19:44.560045 137.73.11.32.32856 > 137.73.11.96.5060: . ack 340 win 6432 (DF) (ipip)
13:19:48.749964 137.73.11.95.5036 > 137.73.11.32.5036: udp 124 (DF) (ipip)
13:19:49.280001 137.73.11.95.5036 > 137.73.11.32.5036: udp 124 (DF) (ipip)
13:19:49.809960 137.73.11.95.5036 > 137.73.11.32.5036: udp 124 (DF) (ipip)
```

3b

4b

6b

7a

8

Figure 3-25: Traffic received through GPRS interface of the mobile node

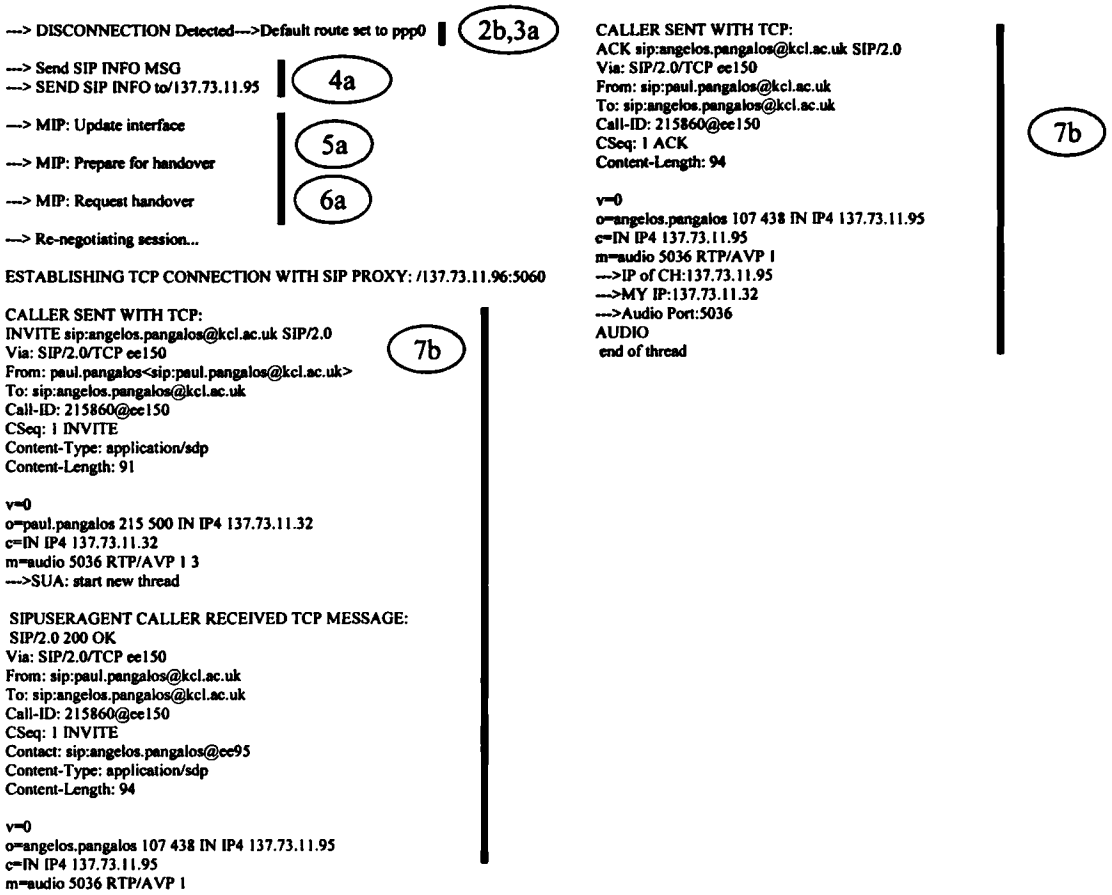


Figure 3-26: SIP signalling exchange at the mobile node

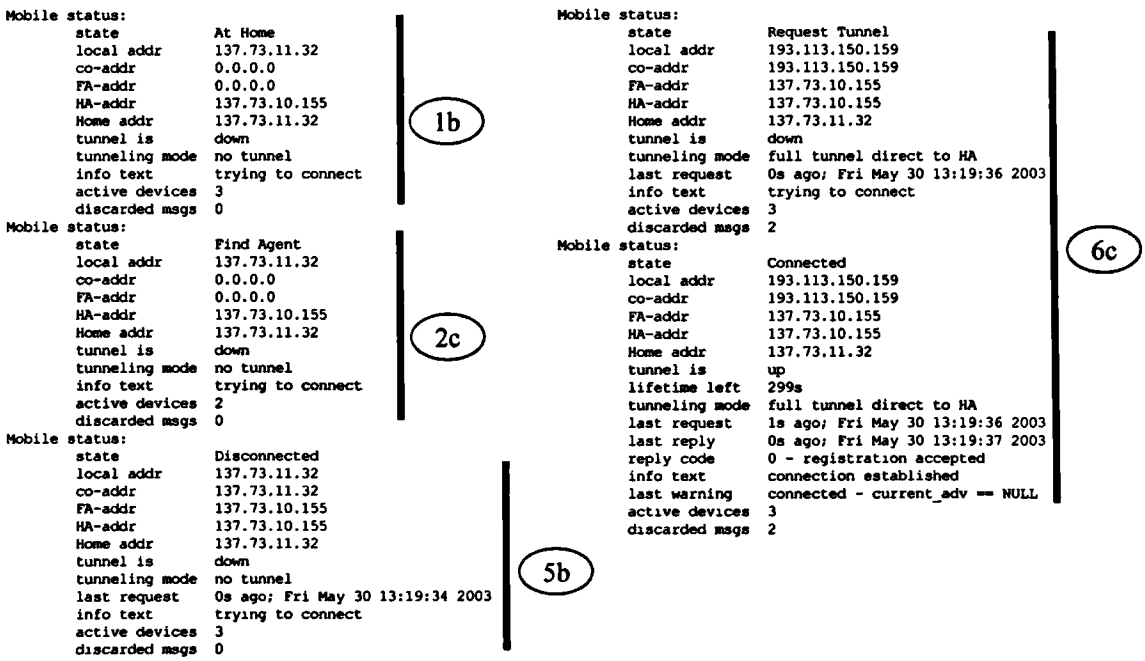


Figure 3-27: Mobile IP agent status updates at mobile node

3.5.7 Summary- Tight Coupling

The tight coupling scenario eliminates congestion completely as the SIP INFO message is sent prior to registering with the home agent. This enables user B to stop transmitting packets that would have otherwise caused link congestion. Table 3-8 gives a breakdown of the delays found in the tight coupling scenario. The GPRS interface does not add to the delay since it's already up and running and therefore does not need to be configured. The mobile IP registration takes about three seconds to complete while detecting disconnection and changing the new default routes result in two further seconds. Finally, twelve seconds are required for sending the SIP INFO message and re-negotiating the session.

USER A / Data Rates		Interface Setup	MIP Registration / ≈ Time (seconds) [A]	Link Congestion / ≈ Time (seconds) [B]	Detect disconnections + set default route [C]	SIP INFO [D]	Session re-negotiation / ≈ Time (seconds) [E]	Total Delay / Approximate ≈ Time (seconds) [A+B+C+D+E]
Initial Session	After Handover	GPRS	Upward Vertical Handover					
Any	5.6kbps	Pre-connected 0 sec (Not connected 7 seconds)	3 sec	No Link Congestion	(1 sec each) 2 sec total	1 sec	11 Sec	17 sec
		Ethernet	Downward Vertical Handover					
5.6kbps	Any	1.3		No Link Congestion	(1 sec each) 2 sec total	1 sec	3 sec	6 sec

Table 3-8: Breakdown of delays - tight coupling scenario

The table also shows the results obtained during an unplanned downward handover. As expected the results are very similar to the ones recorded in the loose coupling scenario Table 3-4. However, an additional delay of two seconds was recorded during the detection and initiation of handover raising the total delay to six seconds. This increase is due to way the sip user agent monitors the link layer of the terminal.

3.6 Discussion

In this section we discuss the parts that make the overall latency of unplanned vertical handovers.

3.6.1 Unplanned Handover Latency

We define the unplanned vertical handover latency (L) as the time from when the mobile terminal is disconnected from the old network to when it receives the first packet from the new negotiated session over the new overlay network.

The latency required to complete an unplanned vertical handover is broken down into the following components.

Ld is the component of latency during which the mobile terminal detects that a disconnection has occurred and a handover is necessary. This could be to an upper overlay (upward vertical handover) or to a lower overlay (downward vertical handover). In our system, this is largely a function of the polling frequency of the interface. A smaller polling frequency increases Ld while a greater polling frequency decreases Ld. The polling frequency interval is considered to trade-off the terminals processing power with the speed at which disconnections are detected. On one hand, having a smaller polling frequency will save valuable processing power and resources of the terminal, while a faster polling frequency will provide a faster response to detecting disconnections. In our system disconnections are detected within a second for both the loose and tight coupling scenarios.

Lp is the latency for the mobile terminal to power on the interface including any network registration time and changing the default routes in the kernel routing table. This component of latency is variable and depends on the type of interface that is being setup and whether the interface was already active and configured at the time the disconnection is detected. For example a dialup interface (i.e. GSM connection) requires approximately over 30 seconds to be initialised and configured while a GPRS interface requires less than 10 seconds. A GPRS interface can also remain connected and be used whenever is required without having to initialise and configure it. Finally, the routing entries are updated taking approximately a second to complete.

Lha is the latency for the mobile IP agent to register with the home agent. This latency is variable and also depends on the underlying network used. For a GSM network we have measured an average delay of two seconds while, using the GPRS network, this was increased to three seconds.

Lc is the latency caused by congestion after the mobile terminal re-connects to a new overlay. For the loose coupling scenario Lc makes up a significant part in the total delay of the handover. However for the tight coupling scenario Lc is zero. This is also illustrated in Figure 3-28.

Lsipinfo is the component of latency during which the SIP INFO message is send. This is the time taken for the mobile terminal to inform the correspondent host to stop forwarding packets. For the loose coupling scenario Lsipinfo is variable depending on the congestion resulting from the handover. A higher congestion will result in a longer Lsipinfo delaying the signalling. However, in the tight coupling scenario this latency was measured at one second when signalled over the GPRS network.

Lsession is the latency measured during session re-negotiation. This was measured as from three to eleven seconds depending on the following two factors: The access network used and the processing power of the end terminals.

3.7 Summary

This chapter proposed two techniques (loose and tight coupling) to support unplanned vertical handovers with the aim of reducing the delay caused by link congestion. This was demonstrated by designing and implementing a series of mechanisms inter-working between SIP and MIP. The results show that these techniques significantly improve the performance during unplanned handovers. This was done experimentally using two different coupling methods of inter-working between MIP and SIP.

During an unplanned handover packet loss is inevitable. In order to minimise this loss and resume the session gracefully the following steps have to take place. First the terminal needs to discover and connect to another overlay network - a decision that is based on user profiles. Secondly the terminal has to take care of mobility (i.e. make sure the data flow is re-routed to the new overlay) and finally the session is re-negotiated based on the characteristics of the new network. These steps make up the total delay in recovering from these types of handovers.

In order to facilitate the above steps we discovered the need for inter-working between protocols. In particular we have looked at different techniques in which the session initiation protocol and mobile IP inter-work to provide support for unplanned handovers. In its basic form mobile IP communicates unidirectionaly with SIP providing status information. Based on this, SIP is able to re-negotiate the session parameters once the connection is re-established. However, this approach revealed a serious weakness, that of congestion delay. Upon re-connection with the new overlay the terminal receives data that is beyond its network bandwidth capability causing link congestion.

To improve the situation two more methods were proposed and tested. The first transported the SIP INFO signalling over the UDP protocol whereas the second approach used the home agent to assist with the signalling. These approaches dealt with the problem effectively, but unfortunately did not eliminate it. Therefore, a third technique was designed to combine the two protocols. This is also referred to as tight coupling. The results obtained from each technique are summarised and compared in Figure 3-28.

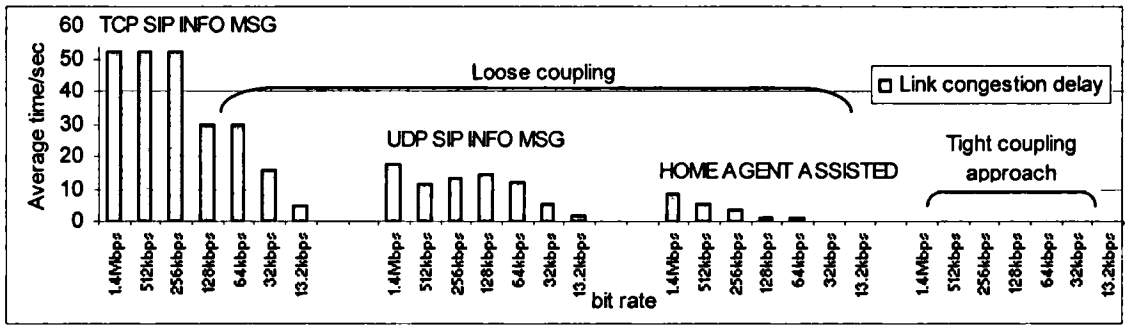


Figure 3-28: Comparing the various techniques

In the loose coupling approach the network layer plays the leading role during the handover. SIP relies on Mobile IP to be notified of events concerning the handover such as detection of disconnections and success of registration messages with the home agent. SIP responds to these messages by setting up new interfaces and re-negotiating sessions. However, in the tight coupling scenario, the application layer is assigned the leading position for managing the handovers. SIP signals MIP to request its services when necessary. MIP does not operate autonomously but is controlled by SIP thus avoiding the congestion problem experienced in the loose coupling scenarios.

The next chapter presents the design, implementation and performance evaluation of a technique designed to handle planned vertical handovers.

3.8 References

[1] C.E. Perkins, " Mobile IP Design Principles and Practise", Prentice-Hall, 1998

[2] M. Handley, H. SchulzrinneE. Schooler, J. Rosenberg, "SIP: Session Initiation Protocol", Request for Comments 2543, Internet Engineering Task Force, March 1999

[3] J Kistler, M. Satyanarayanan, "Disconnected Operation in the Coda Filesystem.", in Proc. ACM Transactions on Computer Systems, pages 6(1):1-25, February 1992

[4] Mummert, L., Ebling, M. Satyanarayanan, "Exploiting weak connectivity for mobile file access" In Proc. of the Fifteenth Symposium on Operating System Principles, Copper Mountain, Colorado, December 1995

[5] Alex C. Snoeren, Hari Balakrishnan, and M. Frans Kaashoek, "Reconsidering Internet Mobility", in Proc. 8th Workshop on hot topics in operating systems pp. 41–46.

[6] Caroline Lebre, Richard Titmuss, Pete Smyth, "Handover between Heterogeneous IP Networks for Mobile Multimedia Services", Exploiting Mobility Conference, 23 September 1998, BT Labs Ipswich

[7] Vicky Hardman, Angela Sasse, Mark Handley and Anna Watson, Reliable Audio for Use over the Internet, in Proceedings of INET'95, June 1995, Honolulu, Hawaii.

[8] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Application, Request for Comments 1889, Internet Engineering Task Force, January 1996.

[9] Rtptools found at http://www.cs.columbia.edu/IRT/software/rtptools/

[10] S. Donovan, "The SIP INFO Method", Request for Comments 2976, Internet Engineering Task Force, October 2000

[11] Paul Anthony Pangalos, Konstantinos Boukis, Dave Wisely, Louise Burness, Prof. A. H. Aghvami, "Experimental Evaluation of Signalling Methods for SIP Based Real Time Application Adaptation During Unplanned Vertical Handovers", in Proc. IST 2001, Spain.

[12] Paul A. Pangalos, Konstantinos Boukis, Louise Burness, Alan Brookland, Caroline Beauchamps, A. H. Aghvami, "End-to-End SIP Based Real Time Application Adaptation During Unplanned Vertical Handovers", Proc Globecom 2001.

[13] Tcpdump found at http://www.tcpdump.org/

You can do anything God assigns you to do, for with the assignment comes the ability
"I can do all things through Christ who strengthens me"
(Philippians 4:13)

Chapter 4

Inter-working of MIP and SIP for Planned Vertical Handovers

4.1 Introduction

In the previous chapter, we addressed different signalling mechanisms used to assist the mobile node to recover during unplanned vertical handovers considering both the network and application layers. In this chapter, we describe the design and implementation of a technique for handling planned vertical handovers. This is achieved by enabling mobile IP and SIP to communicate in a bidirectional manner as well as interacting with the local link layers and operating system of the terminal. This mechanism employs link layer triggers to assist the mobile node in predicting imminent handovers, and enabling it to re-negotiate and establish a new packet flow with its correspondent host prior to the handover [1].

In this chapter we present investigative results of the planned vertical handovers in an attempt to answer the following questions

- How does the signalling mechanism impact the user's session?
- How does the tight coupling between the agents influence the handover time?
- What are the design implementation issues for the proposed signalling method?

The observations made here concentrate on the downlink of the mobile node also referred to as user A.

4.2 Experimental Environment

In this experiment, the mobile node, user A, is equipped with an enhanced sip user agent and has connections to different access points belonging to two different domains. Each interface has a separate IP address belonging to each of the domains as shown in Figure 4-1. The sip proxy server is connected to the home network of the user and is shared between the domains. SIP signalling sets up the multi-media session between user A and user B. In this experiment the mobile node would move from one access domain to the other based on the signal strength measurements of the wireless LAN base station. In some cases however, even if the mobile node is not moving (changing its location) it may alternate from one access interface to another access interface triggered by some local event thus creating a scenario for mobility. In either case, a handover would involve signalling for session re-negotiation, re-routing based on mobile IP and possible re-registration with the SIP proxy server if required.

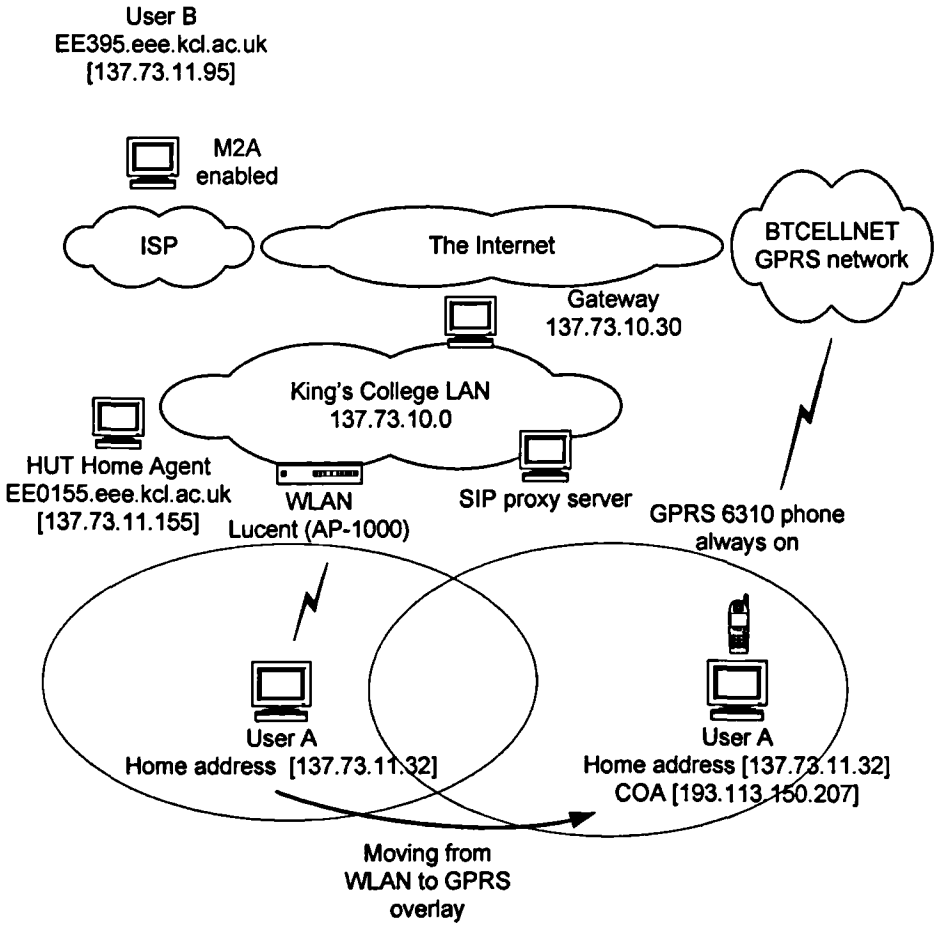


Figure 4-1: Experimental environment

The mobile node terminal is connected to a Nokia 6310 GPRS phone through a serial PPP (point-to-point) link to act as a GPRS terminal. BTcellnets UK's GPRS network was used as the infrastructure. The testbed also consists of another machine: a sip proxy server, which is located in the home network (King's College LAN) serving the mobile node. The sip proxy runs a location database interfaced to Oracle 8.0 over Windows NT and is responsible for locating and relaying SIP signalling messages between users. Furthermore, each end host is equipped with an application layer enhancement designed specifically for our research. This is a java based SIP enabled application capable of interacting with lower layers and responding to vertical handovers. Some of its future uses include, sip user agent interaction, seamless codec change and data re-routing. This application, also referred to as mobile multimedia application, is described in more detail in Chapter 6.

4.3 Implementation Mechanism

This section gives a technical explanation of the tight coupling mechanism implemented between mobile IP and the sip user agent. The software modules and their interactions are shown in Figure 4-2. The sip user agent discovers the presence of the mobile IP agent at the initialisation stage. The discovery takes place with a series of short message exchanges between them (step1). The monitor thread's main function is to monitor the signal strength of the WLAN initiating handover when the signal drops below a certain predefined threshold (step2). Once a handover is initialised the dialog box class is called to set up the required interface and select the appropriate codecs to be used in the new session description (step3).

At this stage the operation of the program is split into two parts, one for upward and the other for downward vertical handovers. During upward handovers, the sip user agent first re-negotiates the session and then instructs the mobile IP agent to register with the home agent (step 4a). However, during a downward handover the reverse method is used. First mobile IP is instructed to handover and update the data flow and once the registration is successful the sip user agent re-negotiates the session over the new overlay network (step4b).

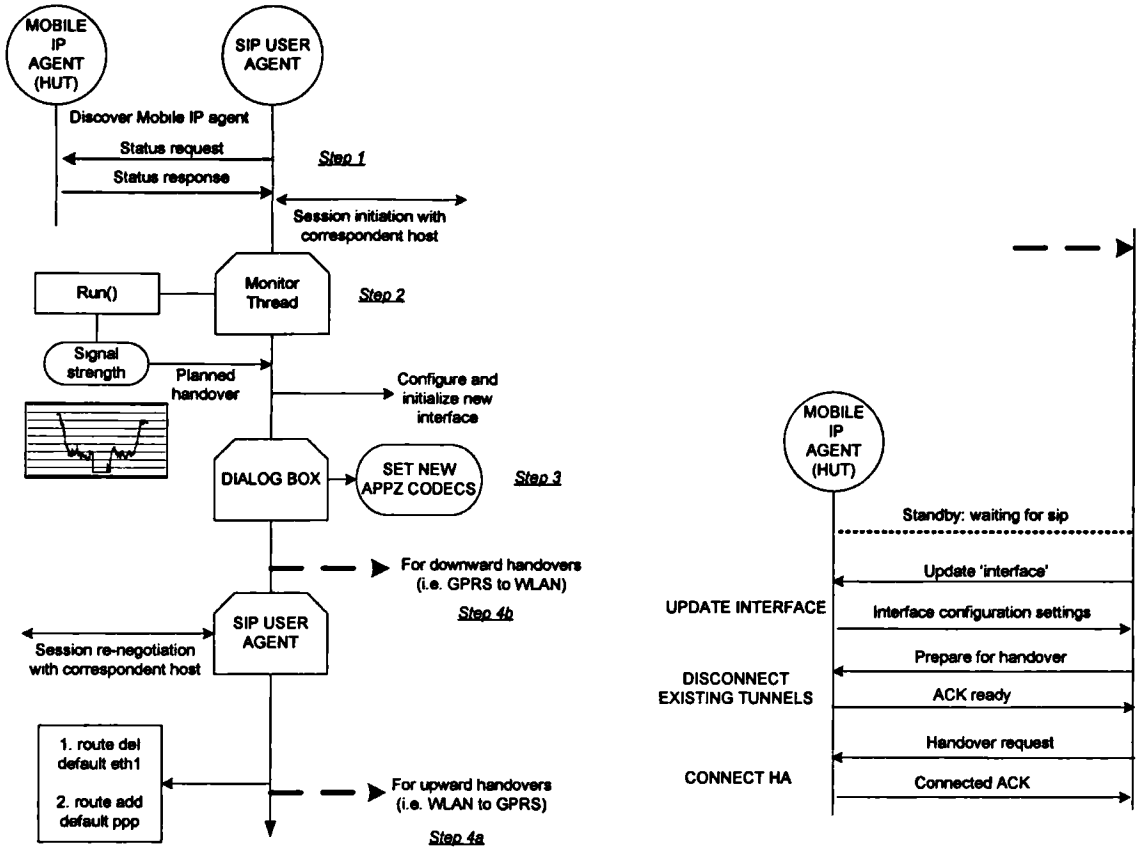


Figure 4-2: Software implementation for planned vertical handovers

4.4 Upward Vertical Handovers

The following is a description of the experiment used to verify heterogeneous mobility for planned upward vertical handovers using tight coupling between mobile IP and the SIP user agent. User A has a connection to two different IP addresses, the WLAN home network and the GPRS overlay network. The correspondent host, user B is in a separate domain across the internet connected to 'freeinternet' Internet service provider. User A initiates an RTP audio session with B, using the java enabled mobile multimedia application (M2A). During the session user A begins to move away from the WLAN coverage area. Before the signal is lost an upward vertical handover is initiated, the session re-negotiated and a new packet flow established. At this point, user A experiences a partially seamless handover, still part of the same audio session while user B still thinks it is sending traffic to the WLAN interface. The signalling method is broken up into three phases

- User registration and session initiation
- Handover phase
- Handover completion

Below is a description of each phase.

4.4.1 Registration and Session Initiation

Both users initially register with the sip proxy server by sending their active interface IP address in the Contact header of the registration message. Since user A has two active interfaces the sip user agent would have to choose the right one to register with the sip proxy server. Further signalling between the two users would go through the sip proxy server. The sip user agent of A would then attempt to discover the presence of a mobility agent, such as mobile IP, by sending a request for a status report within its own terminal. A successful response would indicate that mobile IP is configured and is available to be managed by SIP. User A is now ready to initiate a session with B as described in [2]. The protocol exchanges for the invite request are shown in Figure 4-3. The invitation consists of two requests; an INVITE message followed by an ACK. User A asks B to join a two-party conversation by sending an INVITE message. After user B agrees to participate in the audio call, user A confirms that it has received that response by sending an ACK message request. The INVITE request contains a session description written in SDP format [3] that provides user B with enough information to join the session. It contains the media types and formats that user A is willing to use and also contains the IP address of the WLAN interface to which the media data is to be sent. User B responds to the invitation by returning a similar description listing the media it wishes to use. All message exchanges are passed through the SIP proxy server which is not shown in the diagram for clarity reasons. All requests and responses have the same Call-ID.

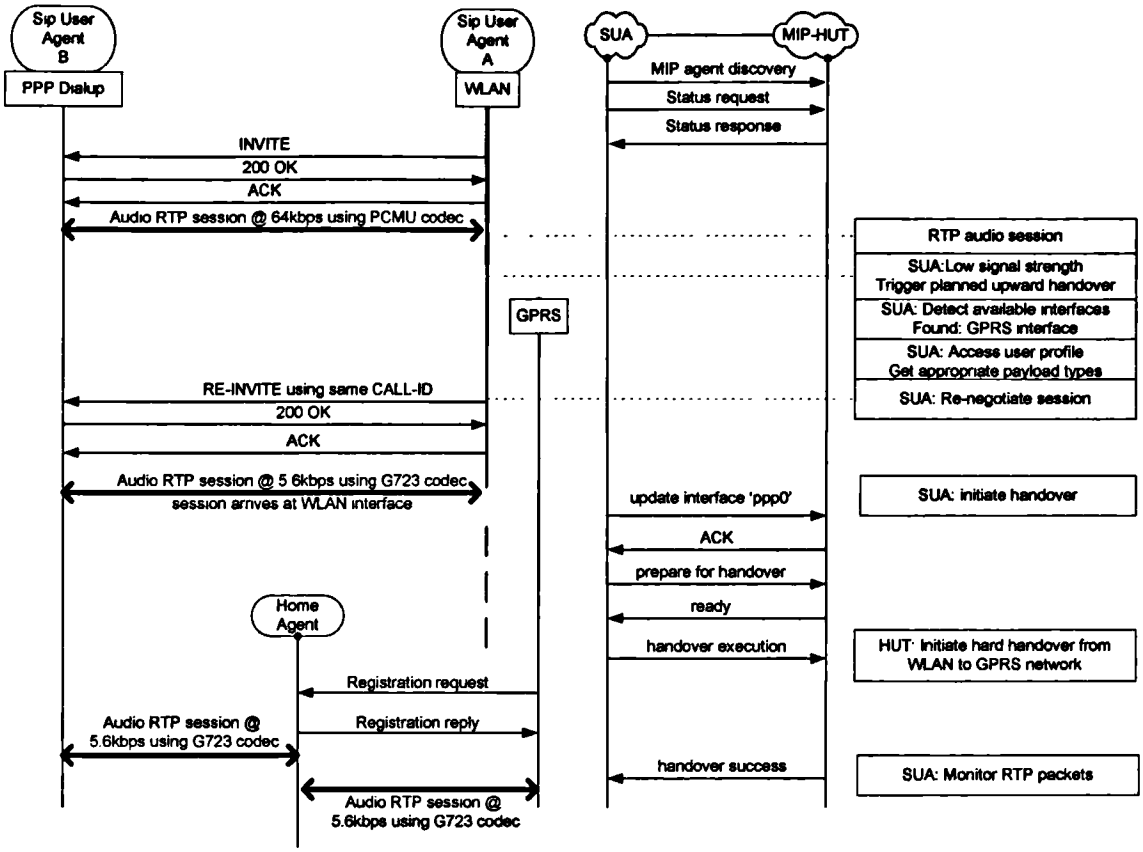


Figure 4-3: Planned handover signalling method

4.4.2 Handover

The handover phase is divided to three further parts. The handover decision phase, the handover preparation phase and the handover execution phase.

4.4.2.1 Handover Decision

Traditionally IP protocols have made minimal assumptions about the link layer between the interface of the mobile node and the access network when specifying mechanisms for movement detection. Discovering the right time to perform handovers in a wireless network can be difficult. User A should ideally stay connected to the WLAN interface for as long as possible until it is absolutely necessary to handover to a higher overlay GPRS network. Anticipating a handover is essential in achieving fast and smooth handovers. Recent IETF handover proposals [4] assume that link layers can accelerate IP level handover procedures. There are various other handover trigger mechanisms which may derive from specific layer events or policy rules. However, in our scheme a handover is triggered based on the signal strength measurements of the WLAN interface.

The sip user agent monitoring module constantly monitors the signal level of the WLAN interface. At some point the signal strength will fall below a certain threshold, triggering a handover.

4.4.2.2 Handover Preparation

Once a handover is initiated the sip user agent determines the next available interface based on user profiles, and begins the upward re-negotiation procedure. Before the handover takes place the audio session is downgraded making it suitable for the new low bandwidth GPRS overlay thus avoiding any potential congestion problems. User A re-negotiates the session by issuing a RE-INVITE request using the same Call-ID and header field, but a different body to convey the new session description information. The RE-INVITE message contact field contains the old WLAN address of user A since mobility will be handled by mobile IP and not directly by user B. Therefore, user B should keep sending packets to the old address of user A.

4.4.2.3 Handover Execution

Once the handover preparation is complete and the session re-negotiated, the sip user agent is ready for transition to the handover execution phase. The handover begins with an 'update interface ppp0' message send to the mobile IP agent. Mobile IP updates both its care of address and foreign address fields with the GPRS IP address, and responds with a status update message. The next message that mobile IP receives is 'prepare for handover' causing it to disconnect any active tunnels and prepare for a hard handover. The sip user agent finally sends the last message called 'handover execution' causing mobile IP to register the new GPRS overlay network with the home agent establishing the new packet flow using IP encapsulation. This is shown in Figure 4-3.

4.4.3 Handover Completion

When user A moves to the new access network, it would usually be required to de-register the IP address assigned to the inactive WLAN interface. However, this is not compulsory in our case. User A can keep the old address as its default contact and use mobile IP to intercept and forward any further incoming requests to the new interface. In some other cases however, when the distance between the home agent and the user is large, it is desirable for the user to register the collocated care of address (i.e. GPRS network address) with the sip proxy server causing new incoming requests to be

forwarded directly to the new overlay avoiding triangular routing. A policy table can be used as described in [5] for deciding what source address to use (home or care of address), whether it should be tunnelled, or even use a bi-directional tunnel.

4.4.4 Performance Evaluation

Figure 4-4 depicts an upward planned vertical handover from a WLAN to the GPRS network. In this experiment, user A is in a two way RTP audio session with user B using the PCMU codec and transmitting at a bit rate of 64kbps. A handover is initiated when the signal strength of the WLAN drops below a predefined threshold value. The figure shows the time at which the handover is triggered. The SUA selects the new GPRS overlay interface and initiates a planned upward vertical handover. In an upward handover the session is required to be re-negotiated before the data flow is updated in order to avoid congestion in the low bandwidth network. Based on the selected interface the SUA generates a new SIP invite message containing the updated session description suitable for the GPRS interface. Since mobile IP is used for mobility the mobile host does not include its new IP address in the Contact field of the SIP header. The session is re-negotiated through the current active WLAN interface taking just under two seconds to complete. On completion, user B passes the new session description to the M2A application initiating a codec change in which the active codec is adapted to the new one without any packet loss, while maintaining both the RTP session ID and sequence number of the stream. A full description of the M2A application is given in Chapter 6. On arrival of the first downgraded packet the M2A application at user A reconfigures itself to the new codec, creates a new player and starts processing the packets. Another function of the M2A application is to notify the sip user agent that the session has been successfully adapted prompting the SUA to change the routing table by adding the GPRS interface as the default route and to complete the handover using MIP. This is done by a series of exchange messages between the SUA and MIP lasting for a total of 2.5 seconds. Finally mobile IP sends a registration message to the home agent through the GPRS interface. The request is accepted and a tunnel is setup between the home agent and user A. In this way, the packets can reach user A via the home agent after the WLAN connection is lost. The home agent then intercepts and forwards packets addressed to user A through the tunnel immediately after it is established as shown in the graph. The first packet arrives at the GPRS interface at around 6.5 seconds.

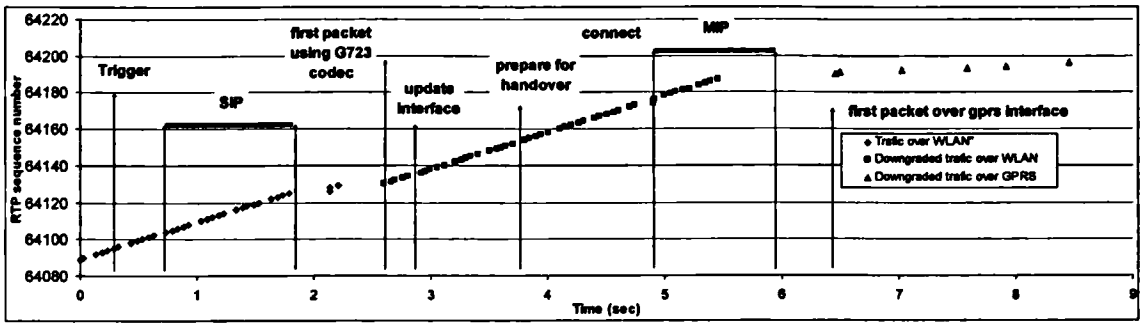


Figure 4-4: Planned vertical handover from WLAN to GPRS

The inter arrival time and packet loss of the RTP traffic has been measured during the handover. The default packetisation interval used within the M2A was 60ms. As shown in Figure 4-5, two regions of inter arrival times are dominant. The first one is during the M2A codec change, with a few packets arriving within 400msec. The second one is observed after the handover with packets arriving between 400ms and 1000ms. The primary reason for this is that the GPRS network, like other wide area wireless networks, exhibits characteristics of low bandwidth, high latencies and significant packet loss. As explained in [6] due to the low bandwidth the GPRS network is almost always the bottleneck. Therefore, packets get queued at the SGSN/GGSN nodes and, when the buffers are full, packet loss occurs. Furthermore, in our experiment we observed a systematic packet loss during the whole experiment as shown in Figure 4-6. The reason is the loss of data caused by the sender's M2A application. Currently there is no solution to this problem but to wait patiently for a fix in the next version of JMF.

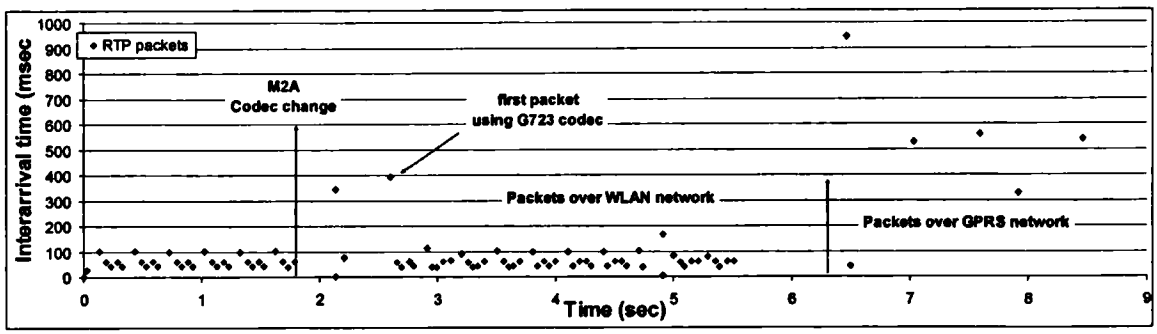


Figure 4-5: RTP inter-arrival time

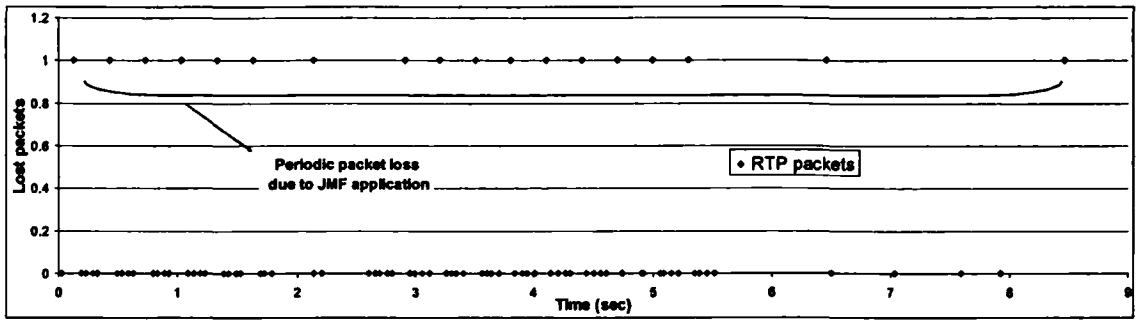


Figure 4-6: RTP packet loss

4.4.5 Results

This section presents the raw results obtained during the planned vertical handover from WLAN to GPRS. Tcpdump [7] was used to monitor the signalling exchange and UDP audio traffic on both the WLAN and the GPRS terminal interfaces as illustrated in Figure 4-7.

1. RTP audio packets received from user A at a rate of 64kbps using the PCMU codec. The mean inter-arrival time is around 60msec.
2. At t=18:12:20:54 the session is re-negotiated lasting just under two seconds.
3. The first packet from the new codec is received a second later through the WLAN interface. The new codec used is the G723 at 5.3kbps.
4. The sip user agent sets the GPRS as the default interface and signals the mobile IP agent to register with the home agent. The registration procedure takes place over the GPRS network lasting just over a second.
5. This is the point at which the tunnel is setup. Packets destined for the WLAN interface are, from now on, intercepted and forwarded to the new GPRS overlay. The last packet received from the WLAN interface has a sequence number 64188.
6. The first packet arrives at the GPRS interface at 18:12:26:37 with a sequence number 64190. Two packets were lost during the hard handover.

```
[root@ee026 root]# ./tcpdump -i eth0
tcpdump: listening on eth0

18:12:20.545332 137.73.11.32.33019 > 137.73.11.96.5060: S 3165411738:3165411738(0) win 5840 <mss
1460,sackOK,timestamp 2783659[|tcp]> (DF)
18:12:20.547438 137.73.11.96.5060 > 137.73.11.32.33019: S 334162:334162(0) ack 3165411739 win 8760 <mss 1460>
(DF)
18:12:20.547503 137.73.11.32.33019 > 137.73.11.96.5060: . ack 1 win 5840 (DF)
18:12:20.556518 137.73.11.32.33019 > 137.73.11.96.5060: P 1:377(376) ack 1 win 5840 (DF)
18:12:20.730273 137.73.11.96.5060 > 137.73.11.32.33019: . ack 377 win 8384 (DF)
18:12:21.638429 137.73.11.96.5060 > 137.73.11.32.33019: P 1:363(362) ack 377 win 8384 (DF)
18:12:21.639976 137.73.11.32.33019 > 137.73.11.96.5060: . ack 363 win 6432 (DF)
18:12:21.645049 137.73.11.32.33019 > 137.73.11.96.5060: P 377:704(327) ack 363 win 6432 (DF)
18:12:21.658795 137.73.11.32.33019 > 137.73.11.96.5060: F 704:704(0) ack 363 win 6432 (DF)
18:12:21.660824 137.73.11.96.5060 > 137.73.11.32.33019: . ack 705 win 8057 (DF)
18:12:21.737223 137.73.11.96.5060 > 137.73.11.32.33019: F 363:363(0) ack 705 win 8057 (DF)
18:12:21.737287 137.73.11.32.33019 > 137.73.11.96.5060: . ack 364 win 6432 (DF)
2015 packets received by filter
0 packets dropped by kernel

[root@ee026 root]# ./tcpdump -i eth0 -T rtp
tcpdump: listening on eth0
18:12:20.996232 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 480 c0  64111 586560
18:12:21.036562 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 480 c0  64112 587040
18:12:21.096447 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 480 c0  64113 587520
18:12:21.136720 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 480 c0  64114 588000
18:12:21.236684 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 480 c0  64116 588480
18:12:21.296630 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 480 c0  64117 588960
18:12:21.336687 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 480 c0  64118 589440
18:12:21.396794 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 480 c0  64119 589920
18:12:21.436947 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 480 c0  64120 590400
18:12:21.537116 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 480 c0  64122 590880
18:12:21.598981 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 480 c0  64123 591360
18:12:21.637596 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 480 c0  64124 591840
18:12:21.699241 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 480 c0  64125 592320
18:12:22.043537 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 480 c0  64126 592800
18:12:22.045380 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 480 c0  64128 593280
18:12:22.120634 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 480 c0  64129 593760
18:12:22.511326 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  64130 594240
18:12:22.569022 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  64131 594720
18:12:22.608456 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  64132 595200
18:12:22.669008 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  64133 595680
18:12:22.709016 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  64134 596160
18:12:22.821553 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  64136 596640
..                                 ..                  ..                  ..
..                                 ..                  ..                  ..
18:12:25.262881 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  64185 616320
18:12:25.301528 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  64186 616800
18:12:25.362660 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  64187 617280
18:12:25.423326 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  64188 617760
2015 packets received by filter
0 packets dropped by kernel

[root@ee026 root]# ./tcpdump -i ppp0 -T rtp
tcpdump: listening on ppp0
18:12:24.768257 193.113.150.5.32785 > 137.73.10.155.mobileip-agent: udp/rtp 54 c34  300 2303265568 (DF)
18:12:25.778338 193.113.150.5.32785 > 137.73.10.155.mobileip-agent: udp/rtp 54 c34  300 2303265568 (DF)
18:12:25.827903 137.73.10.155.mobileip-agent > 193.113.150.5.32785: udp/rtp 68 c0  300 2303265568 (DF)
18:12:26.367901 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  64190 618240 (ipip)
18:12:26.407940 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  64191 618720 (ipip)
18:12:26.937911 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  64192 619200 (ipip)
18:12:27.497906 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  64193 619680 (ipip)
18:12:27.827917 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  64194 620160 (ipip)
18:12:28.367903 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  64196 620640 (ipip)
71 packets received by filter
0 packets dropped by kernel
```


Figure 4-7: Traffic received through the WLAN and GPRS interfaces of the mobile node during the handover

4.5 Downward Vertical Handovers

In the following section, the mechanisms and signalling procedures used for downward vertical handovers are described. The results obtained from the experimental implementation of these mechanisms are also discussed. In this scenario, user A moves from a foreign GPRS network back to his WLAN home network while active in an audio RTP session with user B. The communication between the various protocols and the operating system are also extensively discussed, identifying all the required interactions in order to achieve planned and well executed downward vertical handovers. As with the upward vertical handovers, the signalling is divided into three phases: user registration and session initiation phase, the handover phase and the handover completion phase.

4.5.1 Registration and Session Initiation Phase

In this scenario, the user is located outside the coverage area of his home network. Upon switching on his terminal the sip user agent initiates an interface discovery process. During this phase all interfaces are scanned (step 1 in Figure 4-8.) and the most appropriate one is selected based on the user profiles (step 2). The home interface is always given the highest priority, however; in this case the GPRS interface is selected as the default one since the user is away from home. The SUA then proceeds to register that interface with the home agent and the sip proxy server (step 3-4). On successful registration user A is able to place and receive sip based calls. The first call is placed with user B using the G723 codec at 5.6kbps. All data is sent over the tunnel (step 5) currently setup between the home agent and the mobile terminal.

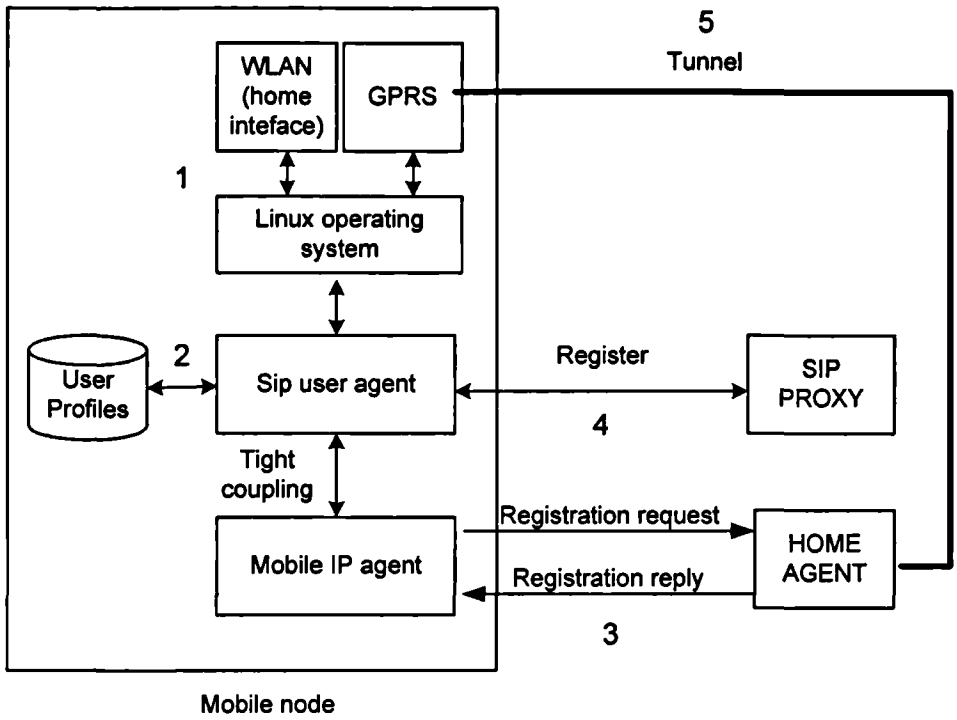


Figure 4-8: Registration and session initiation phase

4.5.2 Handover

The downward handover phase is sub-divided into four parts. The handover decision, handover preparation, handover execution and handover completion phase.

4.5.2.1 Handover Decision Initiation

The sip user agent constantly monitors the signal level of all wireless interfaces and will initiate a handover once a higher overlay network is detected or the user moves outside the coverage of the current access network. A handover will also be triggered if an unplanned handover is detected. Figure 4-10 shows user A entering the coverage area of a WLAN network. This signal change is detected by the local link layer of the terminal and is immediately passed on to higher layers along with the names of the wireless access points detected, also referred to as the ESSID cell identity. The sip user agent compares the ESSID name with a list of other names found in the user profiles and selects the one to use. Once under the home network coverage the terminal will receive a series of home agent advertisements. Under normal circumstances mobile IP will trigger a handover based entirely on these advertisements. However, the tight coupling approach enables the sip user agent to take control of handover initiation, taking a decision that is not only based on agent advertisements but also considers short lived TCP connections. This is discussed in more detail in 4.7.1.

4.5.2.2 Handover Preparation

In the handover preparation phase the routing table is updated, modifying the default route of the terminal as shown in Figure 4-9. The default routing entry specifies the interface that is to be used for all outgoing packets. The first table shows the configuration of the routing table while the user is roaming using the GPRS network. The default route is set to point to the ppp0 (GPRS) interface. It also shows the tunnelling interface configured during the home agent registration. Packets received through the tunnel are decapsulated and passed on to higher layers to be processed by the applications. In order to execute the handover to the WLAN network the routes are modified, replacing the default route with eth1 as shown in the second table. As a result, any further outgoing messages would pass through the WLAN interface. Once the routing table is successfully updated the sip user agent can proceed to the handover execution phase.

```
Kernel IP routing table - default route points to the ppp0 GPRS interface

Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.1        *               255.255.255.255 UH    0      0        0 ppp0
137.73.10.155   *               255.255.255.255 UH    0      0        0 ppp0
127.0.0.0       *               255.0.0.0       U     0      0        0 lo
default         *               0.0.0.0         U     0      0        0 TUNLMNA

Kernel IP routing table - default route points to the eth1 WLAN interface

Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.1        *               255.255.255.255 UH    0      0        0 ppp0
137.73.10.0     *               255.255.254.0   U     0      0        0 eth1
127.0.0.0       *               255.0.0.0       U     0      0        0 lo
default         137.73.10.30    0.0.0.0         UG    0      0        0 eth1
```

Figure 4-9: Updating the kernel routing table

4.5.2.3 Handover Execution

The sip user agent sends an 'update interface eth1' message to the mobile IP agent. In response the mobile IP agent updates its configuration settings and sends a registration request to the home agent executing a hard handover to the WLAN network. During the handover some packet loss might occur primarily depending on the user's session; high bit rate sessions experience more packet loss during a hard handover. Once the handover is complete, the sip user agent is notified with a 'handover success' message. In response the sip user agent re-negotiates the session by issuing a RE-INVITE request

containing the same Call-ID and a new session description upgrading the RTP audio session to a higher bitrate. The complete signalling procedure is shown in Figure 4-10.

4.5.2.4 Handover Completion

As a general rule, the sip proxy server must contain the most recent location of the user and therefore, needs to be updated every time the user moves to another network. Once the user moves to another network, he is required to update the server by sending a re-registration message containing the new contact address. In response the sip proxy will clear the outdated entry associated with the old network and replace it with the new address, making it the default user contact. However, every rule has also got its exceptions. These are described in more detail in 4.7.5.

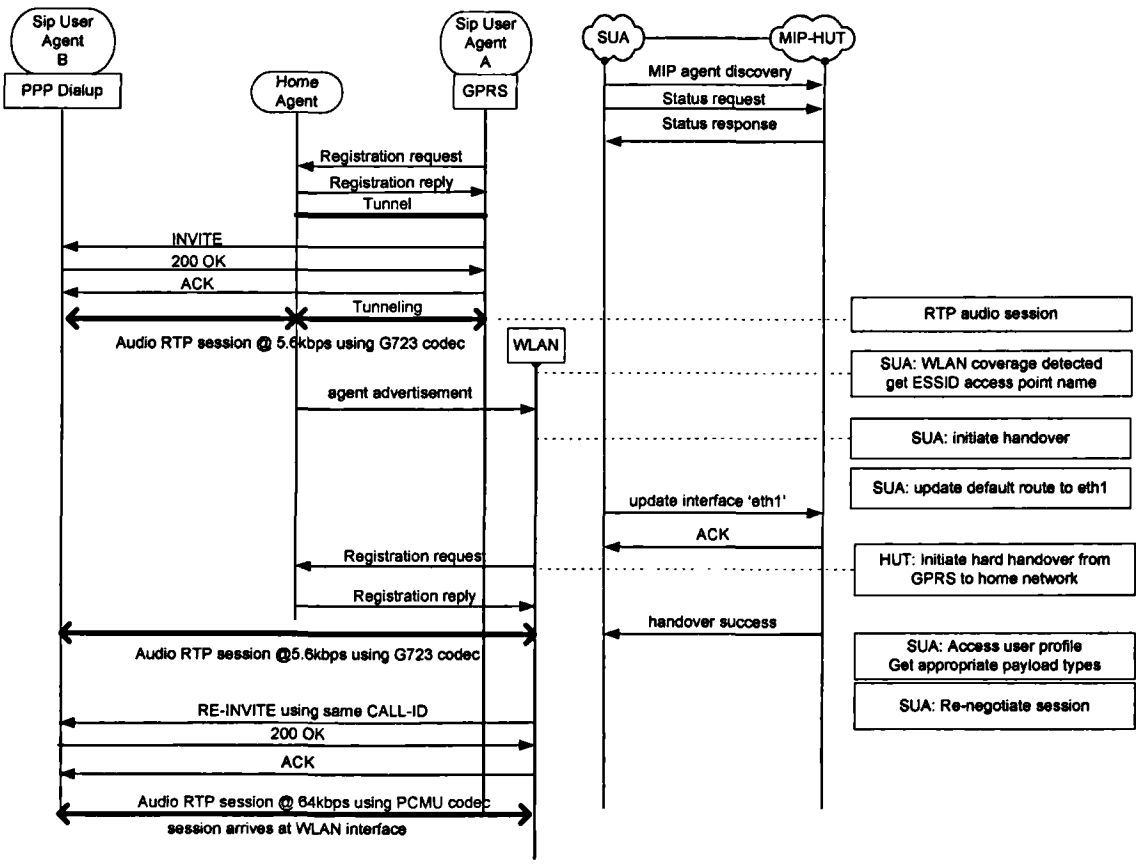


Figure 4-10: Downward vertical handover signalling procedure

4.5.3 Performance Evaluation

In this section, we present and discuss the results of the experiments performed during a downward vertical handover. To measure the performance of the implementation, tcpdump was used to monitor the traffic on both the GPRS and WLAN interfaces. The results were analysed for one of the experiments and are explained below.

4.5.3.1 Discarding Delayed Packets

Figure 4-11 shows the sequence number progression in a connection between the two users. Initially, user A is receiving RTP packets using the G723 codec at 5.6kbps using three frames per packet. We see that the GPRS network cannot sustain this bit rate resulting in user A receiving an average of four packets per second. At t= 4.4 sec user A performs a hard handover to the WLAN overlay network. Packets are now received directly through the home interface of the user while the tunnel to the GPRS care of address is destroyed. However, a significant number of packets are still present within the buffers of the GPRS network and carry on arriving at user A. These packets should not be processed by the application as they will cause a continuous echoing effect thus disturbing the audio session of the user. The reason for this is that each packet received over the GPRS interface has a sequence number that is a few seconds behind the ones received through the WLAN interface. This is also shown on the graph. The first packet received on the new interface has a sequence number 35226 while subsequent packets still being received through the GPRS interface have sequence numbers starting with 35025.

4.5.3.2 Session Re-negotiation and Codec Change

Once the handover is complete, SIP re-negotiates the session over the WLAN interface requesting a session upgrade to the PCMU codec with a bit rate of 64kbps. Since the home network is used for the signalling, the re-negotiation is fast and completes in 1.5 sec after which user B responds by seamlessly switching codecs as shown in the region between 6-6.5 seconds. The first upgraded packet is received at 6.52 seconds, half a second after the session re-negotiation. At t=14.6 the last packet using the G723 codecs arrives from the old overlay.

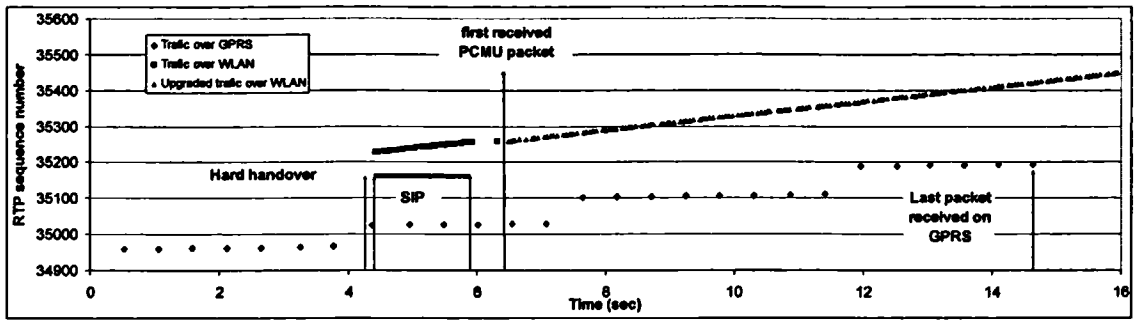


Figure 4-11: Planned vertical handover from GPRS to WLAN

4.5.3.3 Inter-arrival Times

Figure 4-12 shows the inter-arrival times during the handover process. The top half of the graph shows the inter-arrival times of packets received thought the GPRS interface. All packets arrive with 500-600msec separations. This is due to the buffering that occurs within the GPRS network resulting in high latencies and packet loss. The bottom half, however, shows the traffic after the handover with the majority of packets arriving within 65 msec of each other, with the only exception occurring during media adaptation. Two packets are received outside the optimal inter-arrival times, one from the old codec, with a delay of 390msec and one from the new codec with a delay of 175msec. Adapting codecs involves a media adaptation process at the sender application which results in this inter-arrival packet behaviour.

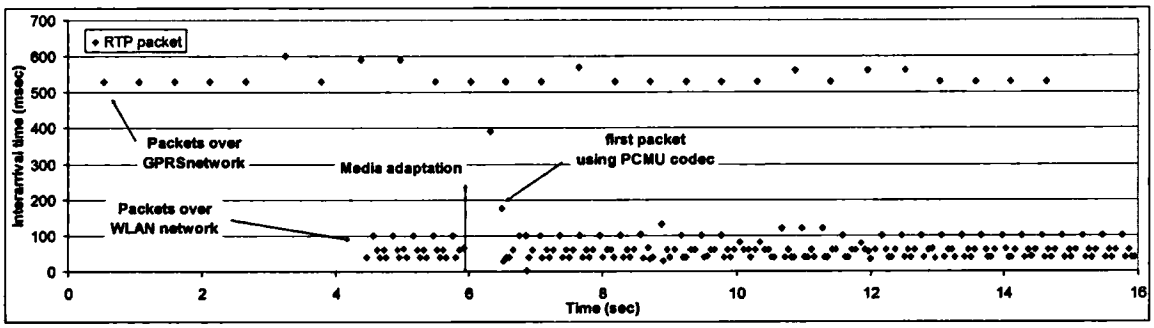


Figure 4-12: inter-arrival time for downward vertical handover

4.5.3.4 Packet Loss

Figure 4-13 shows the packet loss observed during the experiment. The top half of the graph shows three significant points of packet loss experienced at the GPRS network, at t=4.37sec, t= 7.65sec and t=11.95sec. These points correspond to buffer overflows within the GPRS network which result in a significant number of packets being dropped.

After the handover, however, the first packet received over the WLAN interface has a sequence number of 35226 which was well in advance compared to the previous packet received over the GPRS interface labelled 35023. The 203 packet difference observed was not due to the handover mechanism but rather to packets already sent to the GPRS network just before the handover occurred. These are also the packets that continuously arrive at the GPRS interface after the handover. Finally the graph also shows that no packet loss was observed during the media adaptation phase between t=6 – 6.5sec

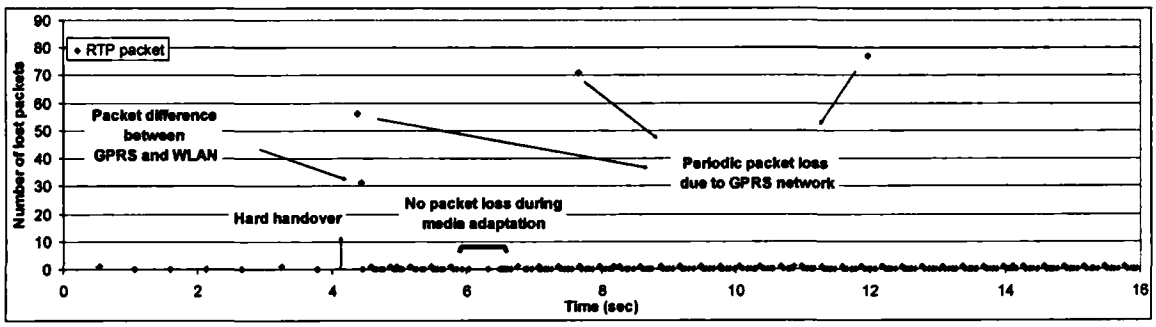


Figure 4-13: Packet loss

4.5.4 Results

This section presents the raw results obtained during the planned downward vertical handover from GPRS to WLAN. Tcpdump was used to monitor the signalling exchange and audio traffic on both terminal interfaces as illustrated in Figure 4-14.

1. Encapsulated RTP audio packets (G723@5.6kbps) received through the GPRS interface.
2. At t=16:38:17.08 the first packet is received over the WLAN interface after a successful handover
3. At t=16:38:17.15 the session is re-negotiated lasting approximately 1.5 seconds.
4. At 16:38:19.16 the first packet is received using the new codecs (PCMU@64kbps)
5. Shows delayed packets still arriving over the GPRS interface.

```
[root@ee026 root]# ./tcpdump -i ppp0 -T rtp port 5036        UDP traffic on GPRS interface   (1)
tcpdump: listening on ppp0
16:38:15.912286 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  34965 620640 (ipip)
16:38:16.442287 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  34966 621120 (ipip)
16:38:17.032296 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  35023 644160 (ipip)
16:38:17.622329 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  35025 644640 (ipip)
16:38:18.152346 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  35026 645120 (ipip)
16:38:18.682289 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  35027 645600 (ipip)
16:38:19.212327 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  35028 646080 (ipip)
...                  ...                  ...                  ...           ...        (5)
16:38:25.702334 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  35190 710880 (ipip)
16:38:26.232346 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  35191 711360 (ipip)
16:38:26.762308 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  35193 711840 (ipip)
16:38:27.292320 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  35194 712320 (ipip)

[root@ee026 root]# ./tcpdump -i eth1 -T rtp port 5036        UDP traffic on WLAN interface   (2)
tcpdump: listening on eth1
16:38:17.081656 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  35226 725280
16:38:17.121724 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  35227 725760
16:38:17.222364 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  35229 726240
16:38:17.281924 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  35230 726720
...                  ...                  ...                  ...           ...
16:38:18.463512 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  35254 736320
16:38:18.523093 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  35255 736800
16:38:18.589339 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  35256 737280
16:38:18.980091 137.73.11.217.6000 > 137.73.11.32.5036: udp/rtp 48 c4  35257 737760   (4)
16:38:19.155821 137.73.11.217.6000 > 137.73.11.32.5036: udp/rt 480 c0  35258 738240
16:38:19.184792 137.73.11.217.6000 > 137.73.11.32.5036: udp/rt 480 c0  35259 738720
16:38:19.224660 137.73.11.217.6000 > 137.73.11.32.5036: udp/rt 480 c0  35260 739200
16:38:19.264547 137.73.11.217.6000 > 137.73.11.32.5036: udp/rt 480 c0  35261 739680
16:38:19.324698 137.73.11.217.6000 > 137.73.11.32.5036: udp/rt 480 c0  35262 740160

[root@ee026 root]# ./tcpdump -i eth1 tcp      TCP traffic on WLAN interface   (3)
16:38:17.145121 137.73.11.32.32955 > 137.73.11.96.5060: S 4168283518:4168283518(0) win 5840
<mss 1460,sackOK,timestamp 1423590[|tcp]> (DF)
16:38:17.147166 137.73.11.96.5060 > 137.73.11.32.32955: S 595187:595187(0) ack 4168283519
win 8760 <mss 1460> (DF)
16:38:17.147260 137.73.11.32.32955 > 137.73.11.96.5060: . ack 1 win 5840 (DF)
16:38:17.150011 137.73.11.32.32955 > 137.73.11.96.5060: P 1:385(384) ack 1 win 5840 (DF)
16:38:17.293750 137.73.11.96.5060 > 137.73.11.32.32955: . ack 385 win 8376 (DF)
16:38:18.199803 137.73.11.96.5060 > 137.73.11.32.32955: P 1:365(364) ack 385 win 8376 (DF)
16:38:18.199959 137.73.11.32.32955 > 137.73.11.96.5060: . ack 365 win 6432 (DF)
16:38:18.465811 137.73.11.32.32955 > 137.73.11.96.5060: P 385:723(338) ack 365 win 6432 (DF)
16:38:18.482807 137.73.11.32.32955 > 137.73.11.96.5060: F 723:723(0) ack 365 win 6432 (DF)
16:38:18.484845 137.73.11.96.5060 > 137.73.11.32.32955: . ack 724 win 8038 (DF)
16:38:18.562297 137.73.11.96.5060 > 137.73.11.32.32955: F 365:365(0) ack 724 win 8038 (DF)
16:38:18.562375 137.73.11.32.32955 > 137.73.11.96.5060: . ack 366 win 6432 (DF)
```

Figure 4-14: Traffic received through the GPRS and WLAN interfaces of the mobile node during the handover

4.6 Handover Latency

We have so far looked at how the signalling mechanism impacts the user's session. The results enabled us to examine the performance of the signalling mechanism for both upward and downward handovers. This section identifies and describes the latency components of the signalling mechanism. The handover latency (L) is defined as the amount of time from when the mobile terminal detects a new overlay network to when it receives the first packet from the newly negotiated session over the new overlay network. The latency is broken down into the following components.

Ld is the component of latency during which the mobile terminal detects a planned handover. Link layer hints are taken into account, accelerating handover initiations. In this system handovers are detected in well under a second when the handover thresholds are broken.

Lp is the time taken for the mobile terminal to discover and select an appropriate interface to handover to. This includes the time taken to power on the interface and register with the network. This component of latency is variable and mostly depends on whether the interfaces are already powered up. The simplest approach to reduce this latency, is to have all the interfaces of the terminal turned on all the time even when they are not in use. However, this will maximise the power drain of the terminal limiting the lifetime of the batteries. For example, as explained in [8] measurements of commercially available wireless interfaces show that a GPRS phone and WaveLAN RF interface together consume approximately 20% of the total power drain of a typical laptop computer. Taking this into account it is essential to manage the network interfaces effectively. In [8] the authors suggest turning off the interfaces that are not being used and turning them on only when geographic or other hints indicate that a handover may be likely.

Lha is the latency for the mobile IP agent to register with the home agent. This latency is variable and also depends on the underlying network used. In the GPRS network we have measured an average delay of one–two seconds.

Lsession is the latency measured during session re-negotiation. This was measured at under three seconds over the GPRS network.

Lm is the latency measured due to media adaptation. The mobile multimedia application (M2A) takes less than one second to adapt the codecs once the SIP re-negotiation is complete.

4.7 Discussion

This section discusses various issues related to planned vertical handover using the tight coupling scenario of MIP and SUA. Each of the issues discussed here is in the context of the upward and downward signalling methods described earlier.

4.7.1 Considering Short Lived TCP Connections

This section discusses the handover initiation technique used for the downward vertical handover scenario described in 4.5.2.1. When user A moves from the foreign GPRS network back to its home network, it will receive a series of home agent advertisements. Under normal circumstances mobile IP will initiate a handover based entirely on these advertisements. However, in our implementation, the sip user agent is in control of all handover initiations, basing its handover decision not just on agent advertisements but also considering short lived TCP connections. If the user is roaming, these type of connections (often created by applications such as e-mail and web-browsing) are directly associated with the terminals care of address and not the home address and therefore cannot survive handovers. During a vertical handover these connections would abruptly terminate unless taken care of. The traditional way to solve the problem would be to associate all connections with the home address passing the responsibility to mobile IP. However, a well known problem with mobile IP is the triangular routing which adds delay to the traffic towards the mobile node, especially when the distance between the mobile node and home agent is large. For delay sensitive traffic, such as web browsing, this is not acceptable. The fact that the packets are tunnelled also means that an overhead caused by the IP to IP encapsulation will be added to each packet. However, in our system we use a rather different approach.

Figure 4-15 presents the results obtained during downward vertical handover, but now the focus is on the mechanism used to handle TCP connections. The handover is initiated at t =1 sec, by monitoring the signal strength of the WLAN access point. As user A moves into the coverage of the WLAN network, the received signal strength increases. Once above a threshold value the vertical handover to the lower overlay

network is initiated. At this point, the sip user agent will begin to monitor all short based TCP connections of the user A. At t=3.80sec the sip user agent detects that all short based TCP connections have finished and proceeds to the handover preparation phase immediately. During this phase the sip user agent updates the IP routing table and sets the WLAN as the new default interface of the terminal without affecting any TCP connections. Further TCP connections established after the route update would be over the WLAN interface and not the GPRS. Under normal circumstances TCP connections are broken when changing the default route. However, by using the sip user agent to monitor these connections and perform route updates once these connections are finished the problem of disconnections is avoided making the handover transparent to the user.

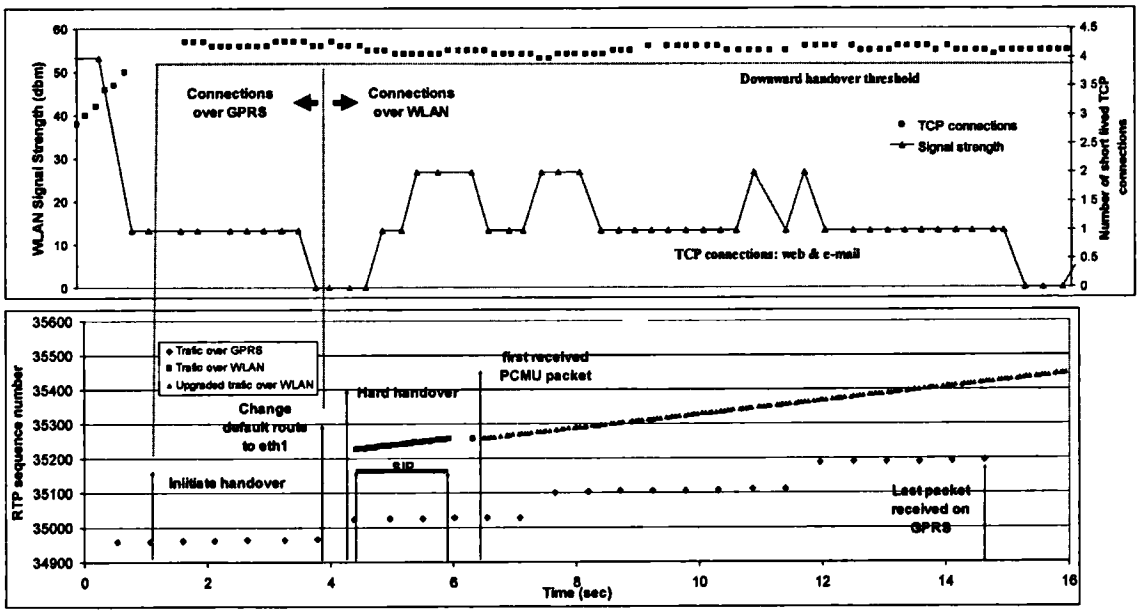


Figure 4-15: Handling short lived TCP connections

4.7.2 Support for Failsafe Handover Mechanisms

The upward handover re-negotiation procedure consists of several intermediate steps. An INVITE message is followed by provisional responses such as (180, 183, etc), a 200 OK and a final Acknowledge message. It may so happen that user A loses its current WLAN network connection before session re-negotiation is complete. In such a scenario Failsafe mechanisms should exists so that the re-negotiation completes through other interfaces that might be available on the multi-homed terminal. Figure 4-16 shows two possible approaches to the problem. The first one solves the problem by detecting disconnections and attempting to re-start the signalling over the new GPRS overlay. The second approach does not solve the problem but avoids it altogether. When a handover

is initiated the sip user agent will replace the default WLAN route with the GPRS one and re-negotiate the session using the new overlay without depending on the old one. This approach, however, suffers from one weakness. Changing the default interface of the terminal prematurely will cause packets transmitted by user A to be rejected by its correspondent host, user B. These packets have, as source address, the care of address of the GPRS network and therefore will not be recognised by the user B. This is only rectified once user A successfully completes the handover with the home agent.

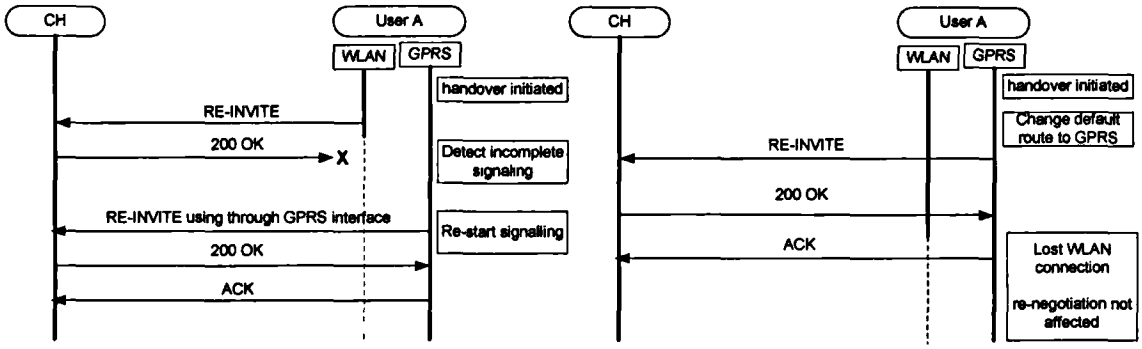


Figure 4-16: Failsafe handover mechanisms

4.7.3 Multiple Registrations

User A has multiple interfaces, which we may think of as "logical interfaces". These logical interfaces may be associated with one or more physical interfaces, and these physical interfaces may be connected to the same or different networks. At any particular instance the mobile terminal may decide to have multiple active IP addresses and, depending upon the destination route matrix or local policy decision, one particular interface is chosen for the transmission or reception of traffic. An example is shown in Figure 4-17. The user may register more than one interface with the sip proxy server by sending a single registration message containing multiple IP addresses within the contact field header. The sip proxy is then responsible to map the various incoming and outgoing session requests to the appropriate interface as shown in the diagram. The mapping can be based on the user profiles such as cost, required bandwidth and application layer quality of service. In this approach most of the intelligence is located in the sip proxy server.

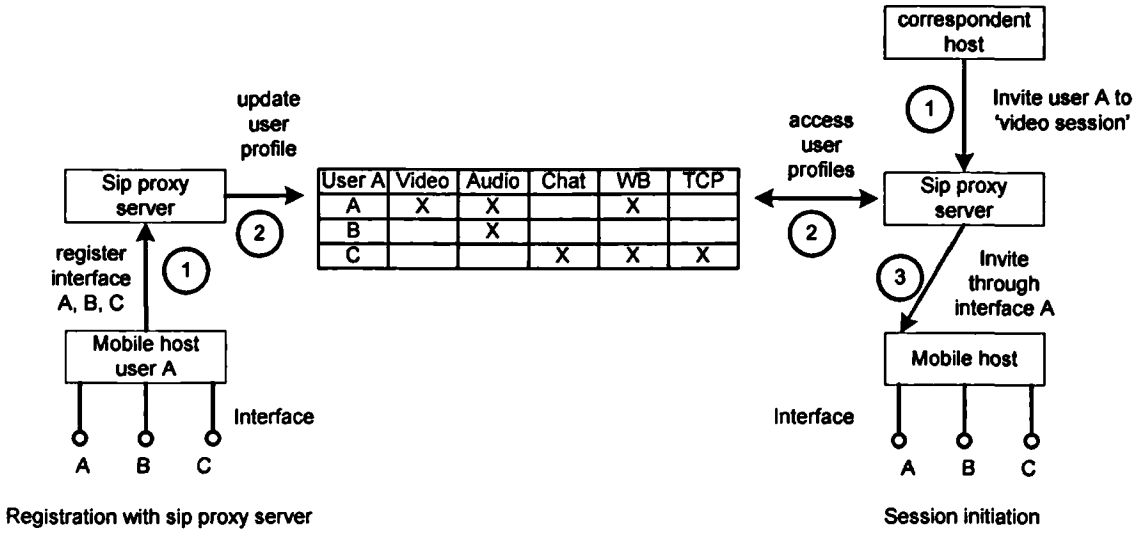


Figure 4-17: Registering multiple interfaces with the SIP proxy

An alternative approach is shown in Figure 4-18. The user registers one of its interfaces with the sip proxy server by sending a registration message containing the chosen default interface. All INVITE requests are then forwarded to that particular interface where they get processed by the terminal. Based on user profiles a response is generated specifying an alternative interface to be used for that session if needed. Figure 4-18 shows the signalling taking place at interface C while the data flow is set up over interface A.

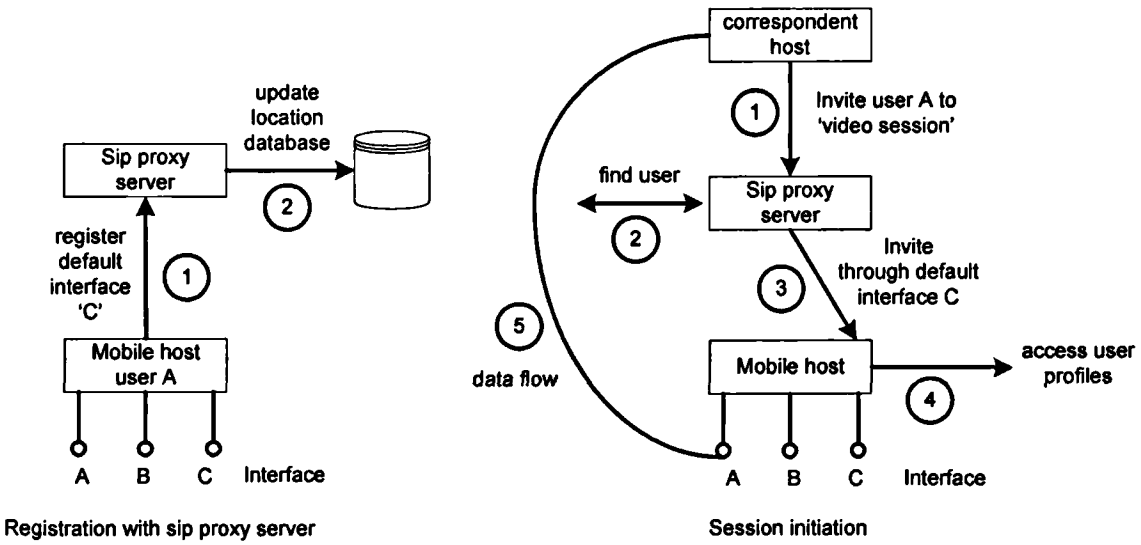


Figure 4-18: Registering one interface with the sip proxy

4.7.4 Signalling Design Considerations

4.7.4.1 Communication Scheme

In this architecture the communication between various entities within the mobile node is built around a flexible mechanism for event discovery and information exchange as illustrated in Figure 4-19. The design issue was whether to base the communication on a polling scheme or a notification scheme. These two approaches can be considered to trade-off simplicity of implementation with simplicity of use and scalability. On one hand, the polling scheme requires a well-defined polling interface specifying the time and frequency of polling, while a notification scheme requires run-time support and a hook to the application. The polling scheme used within the Linux operating system is considered easier to program. These factors led us to favour of the polling scheme architecture for the communication between Mobile IP, the Linux Operating System and the mobile IP agent.

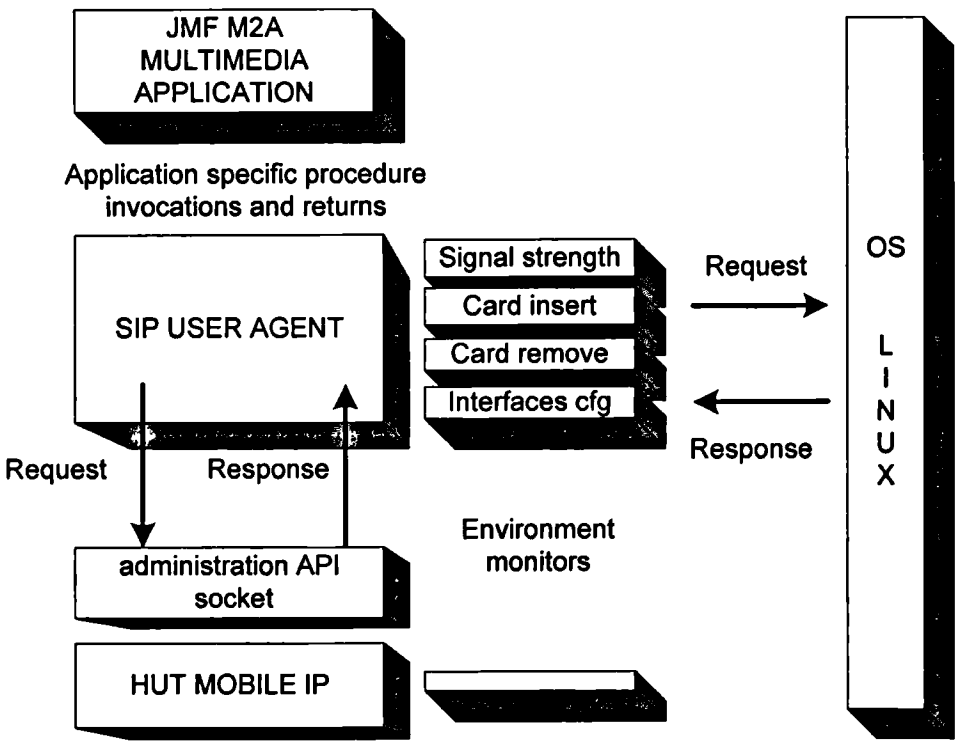


Figure 4-19: Communication scheme architecture

The implementation uses a synchronous, request-reply (sometimes referred to as "call/wait") mechanism which involves blocking the sender until the receiver fulfils its request, that is the sender issues a request and waits for the target's response before

continuing its own processing. The synchronous mode is essential as the required information must be obtained before the program proceeds to the next phase. This polling scheme has its strengths as well as weaknesses that should also be considered. Using a synchronous request-reply mechanism requires the target entity to be always available and functioning correctly. In order to recover from a failure in communication, an implementation is required to provide mechanisms such as error messages, requests timers, and retransmissions.

4.7.4.2 Signalling Between Entities

The signalling between the various entities is best described using the state reference diagram in Figure 4-20. The sip user agent consists of four distinct states namely: the initialisation state, the idle and active states and finally the handover state.

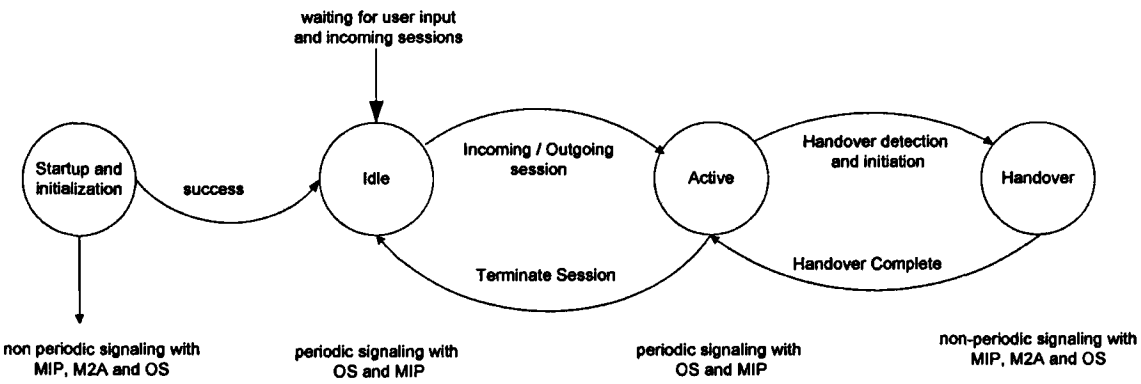


Figure 4-20: State diagram showing signalling between entities

Initialisation state

During the initialisation phase, the current state of the mobile terminal is required in order to configure the relevant parameters, variables and modes of the sip user agent. First the operating system is polled to obtain information such as the available and active interfaces, the signal strength, if wireless, and the IP address of each interface. Secondly, a call is made to the administration API to check if the mobile IP agent is active and ready to co-operate. A successful response indicates that mobility will be handled by mobile IP. If no response is received, the SUA will assume that mobility will be handled by other means. Finally, the sip user agent will also attempt to communicate with the M2A application to enquire about codec support and other enhanced capabilities such as media adaptation and end to end mobility support. Once complete, the sip user agent is ready to register with the sip proxy server.

Idle and active states

In the idle and active states the operating system is periodically polled in order to detect when interfaces are inserted and removed, as well as to obtain their configuration parameters and signal strength measurement, if wireless. The mobile IP agent is also periodically polled, monitoring status information such as tunnel lifetime and handover initiation events prompted by the discovery of home/foreign agent advertisements. By synchronising these two sets of information the sip user agent is then able to trigger prompt handover events thus performing fast and efficient handovers.

Handover state

The handover state requires the sip user agent to interact with the operating system and the M2A application as well as the mobile IP agent, in a synchronised manner. During downward vertical handovers the operating system is contacted first, followed by the application and the mobile IP agent. However, for upward vertical handover the application is the first point of contact, followed by the operating system and the mobile IP agent. Calls made to the operating system carry the interface name that is used to update the default routing entry of the terminal. This is a single call and is always made just before a handover is executed. The interaction between the sip user agent and mobile IP however, comprises a series of calls initiated by the sip user agent aimed at configuring the agent and executing the handover. A single call only is required when the user is returning back to his home network. A single call is also made to the M2A application containing the full SIP message received during the session re-negotiation. In response the application will adapt according to the new specified codecs and respond indicating whether the process was successful. Figure 4-21 summarises the high level signalling exchanges between the various entities within the system.

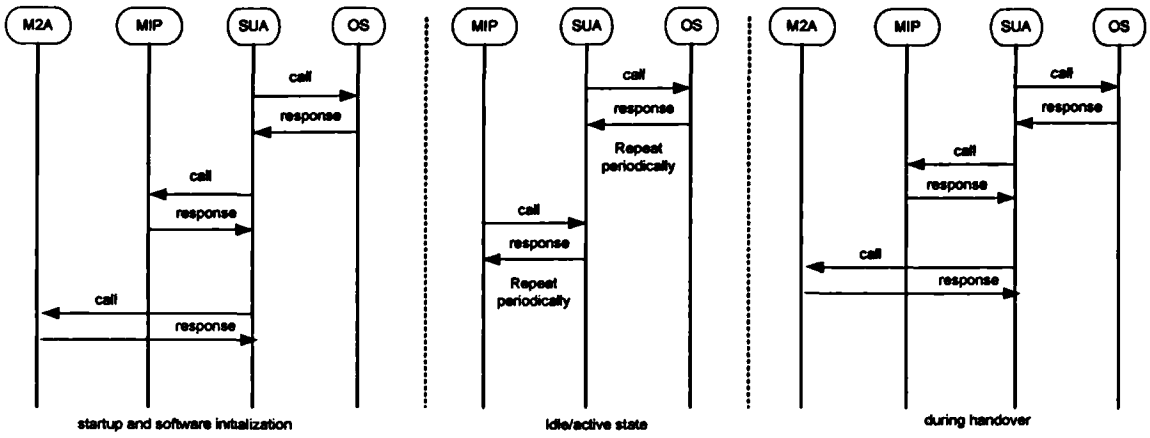


Figure 4-21: Signalling exchanged between the various entities

4.7.5 Updating the Sip Proxy Server

We have already mentioned the need to update the sip proxy server after a handover. However, this is not always a requirement but rather an option. Following a successful upward vertical handover, the user is presented with two options. It can either update the sip proxy server, as shown in Figure 4-22 – option A or avoid doing so as illustrated in option B. These two methods are described in more detail below. In the first phase, the user is initially in his home network and registers interface A with the sip proxy server. Interface A represents his home address.

4.7.5.1 Option A

The next phase shows the user moving to foreign network C. During the handover the home agent is updated followed by a re-registration message send to the sip proxy server. This message contains the new IP address of interface C in the contact field header. In other words, the user instructs the sip proxy that future multimedia calls are to be routed directly to the care of address – interface C. This is a beneficial option as it avoids triangular routing for sessions where delay and jitter is of prime concern. However, due to the nature of this approach, mobility will not be supported for any session established directly with the care of address. In the event of a further handover, any newly established sessions will have to be terminated and re-established using the user's new contact address. The new address could be the home address or another care of address.

4.7.5.2 Option B

On the other hand, option B illustrates the scenario in which the user does not update the sip proxy server after a handover. As a result, the sip proxy server will keep the users home address as the most recent contact of the user. Future calls, would therefore be directed to the home agent and tunnelled to the most recent location of the user. Option B gives the user more flexibility in terms of mobility, however it also bears all the limitations offered by mobile IP such as triangular routing and encapsulation.

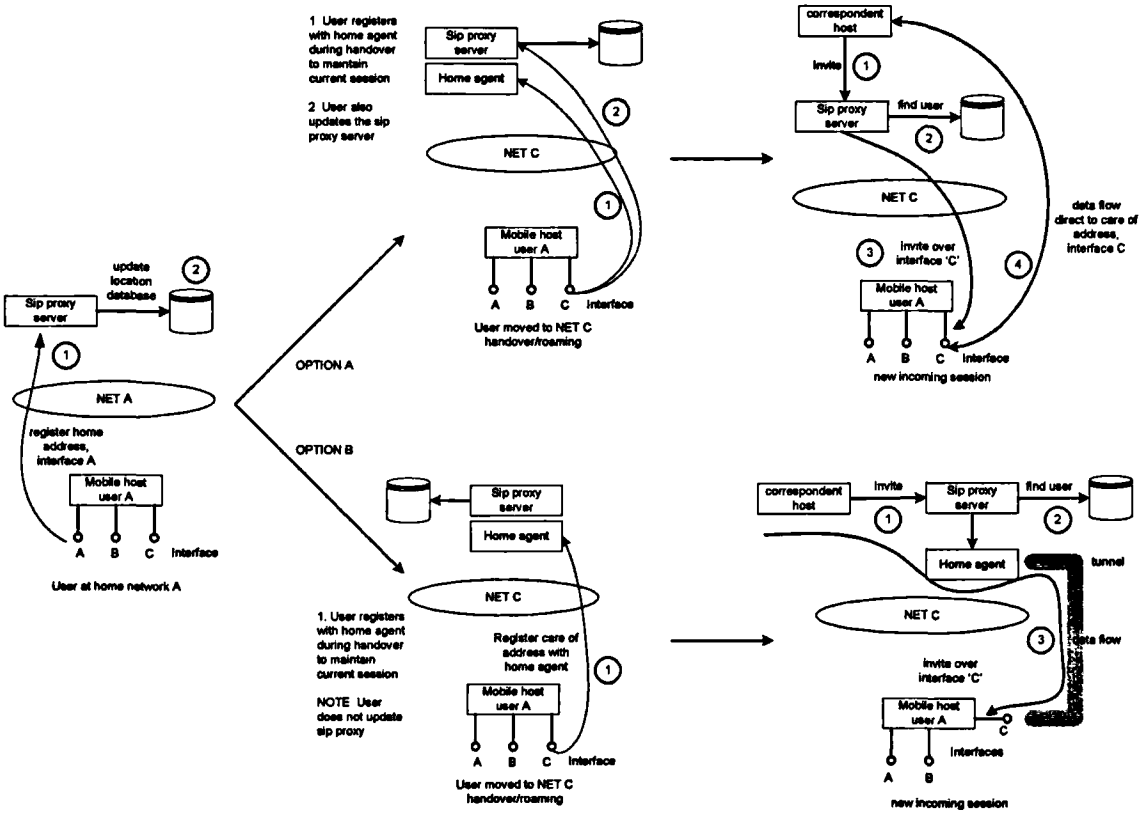


Figure 4-22: Updating the sip proxy server after a handover

4.7.6 Signalling Sequence

We have already looked at both upward and downward types of vertical handover as well as the signalling and inter-working between entities and protocols within the terminal. This section examines the signalling sequence of each handover type. Both upward and downward handovers are very similar in respect to the mechanisms and signalling procedures they use. These were fully described in sections 4.4 and 4.5. However, a key difference between the two mechanisms is the order in which the signalling is carried out. Figure 4-23 shows the signalling sequence. During upward vertical handovers, the multimedia session of user A is required to be re-negotiated

prior the handover. This is an essential requirement as the new overlay network would be of a larger cell size (and lower bandwidth per unit area) and might not support the user's high bandwidth multimedia session. Once the session is successfully downgraded, the user may handover to the new overlay. During a downward handover, however, the procedure is reversed. Since the user would be moving to a smaller cell size (and higher bandwidth per unit area) the session adaptation will take place just after the handover thus avoid overloading the low bandwidth network.

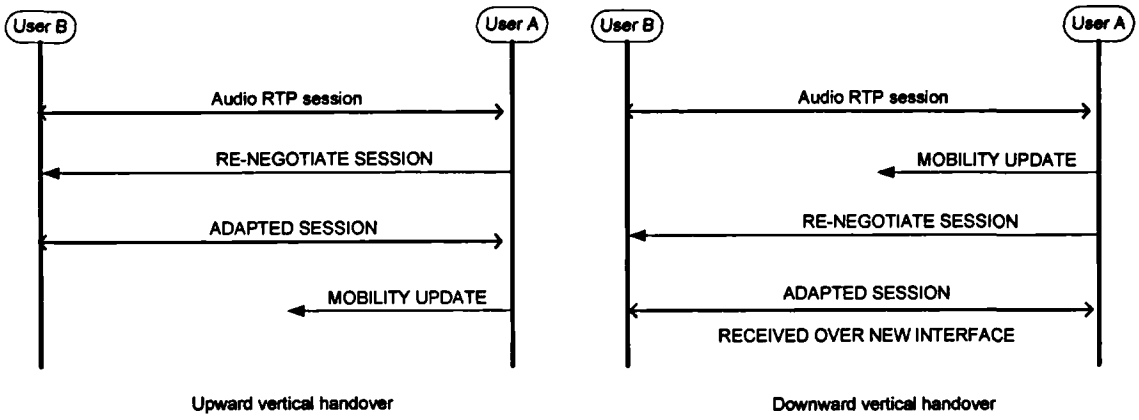


Figure 4-23: Signalling sequence of upward and downward handover mechanisms

4.8 Summary

This chapter, presented the design, implementation and performance evaluation of a technique designed to handle planned vertical handovers. This technique improves vertical handover performance significantly by providing fast and seamless handovers across heterogeneous networks. It achieves this by enabling a bidirectional communication between the link layer, the network layer and the application layer of the terminal.

The session initiation protocol was enhanced to communicate with three entities, the mobile IP agent the M2A application and the operating system. The signalling mechanism was then described showing how these entities interact and relate to each other. These promising solutions were then implemented in our experimental testbed providing performance measurements for both upward and downward vertical handovers. The gathering of packet-level traces helped understand more clearly the reasons for the observed performance as well as identifying the various issues associated with these techniques.

The next chapter presents a detailed design and implementation for a new type of mobility called session mobility. This can also be described as another type of vertical handover in which users move from one network to another by changing devices.

4.9 References

[1] Paul Pangalos and Hamid Aghvami, "IP-Based Vertical Handovers," 4th WWRF workshop in Paris on December 6-7, 2001.

[2] M. Handley, H. Schulzrinne E. Schooler and J. Rosenberg, "SIP: Session Initiation Protocol", Request for Comments 2543, Internet Engineering Task Force, March 1999.

[3] M. Handley and V. Jacobson, "SDP: Session description protocol", Request for Comments 2327, Internet Engineering Task Force, April 1998.

[4] O'Neill, A., Tsirtsis G.and Corson S., "Generalized IP Handoff", Internet-Draft (work in progress), draft-oneill-craps-handoff.txt, August 2000.

[5] Elin Wedlund and Henning Schulzrinne, "Mobility support using SIP", ACM/IEEE International Conference on Wireless and Multimedia (WOWMOM), Aug. 1999, pp. 76-82.

[6] Rajiv Chakravorty and Ian Pratt, "WWW performance over GPRS", IEEE Wireless Communications and Networks Conference 1999 Jaipur , India

[7] Tcpdump found at http://www.tcpdump.org/

[8] Mark Stemm and Randy H. Katz, "Vertical handovers in wireless overlay networks", ACM MONET Special Issue on Mobile Networking in the Internet. 1998.

"The fear of the Lord is the beginning of knowledge"
(Proverbs 1:7)

Chapter 5

Session Mobility

This chapter contains the detailed design and implementation for a new type of mobility called session mobility. We address three fundamental challenges posed by this type of mobility namely: terminal discovery, signalling and mobility support. We start by presenting the terminal description protocol - an application layer protocol - for describing terminals in terms of hardware and software capabilities, followed by a detailed description of the signalling mechanism. We conclude this chapter with the results obtained from the testbed used to evaluate session mobility.

5.1 Introduction

In a typical future environment, users will be able to receive multimedia services via multiple network devices (i.e. mobile phone, desktop phone, public IP terminals, and desktop computer). As people have access to these devices they may desire to switch from one device to another in the middle of a session. There are numerous reason why such a move might be necessary. The users battery might be running out, a better device is found or the user might wish to share a session with a number of other users by transferring the session to a digital projector.

This can also be described as another type of vertical handover in which users move from one network to another by changing devices. This type of mobility is called session mobility and it refers to the user's ability to maintain an active session while

switching between terminals [1]. The operation of maintaining the session across the different networks and devices is called "session mobility" and the ability to switch between the terminals is called "session handover". Previous work has focused on mobility issues related to personal and terminal mobility such as [2], as well as proposing design solutions for service mobility [3] and [4]. Furthermore, previous work has failed to comprehensively address several issues that are required for a complete solution for session mobility. This chapter develops a solution to session mobility by dealing with three fundamental challenges.

1. Terminal discovery and description: Before any session can be handed over, the terminal matching the user's criteria (in terms of hardware and software capabilities) must be located and mapped to an addressable destination.

2. Handover signalling: Once a terminal is located a handover mechanism is required to move the session from one device to the next

3. Mobility support: A mobility mechanism is required to maintain an active session while switching between terminals.

5.2 The terminal Description Protocol (TDP)

The terminal description protocol (TDP) is a lightweight protocol specifically developed as part of the solution to session mobility. Terminals requiring a session handover use the protocol to discover neighbouring terminals that match the user's criteria. A brief overview of the protocol follows.

5.2.1 Protocol Overview

The terminal description protocol (TDP) is an application layer protocol for describing terminals in terms of hardware and software capabilities. A terminal description is expressed in TDP, in a short structured textual description of the terminal interfaces, mobility protocols and application information that is required to decide whether a terminal is likely to be of interest to a user. TDP is an application layer protocol (it does not incorporate a transport protocol), intended to use the session initiation protocol as its transport.

5.2.2 Terminal Descriptions

Terminal descriptions follow the same design principle as [5] and [6] in that they are entirely text based. As explained in [6] the textual form, as opposed to other forms such as binary encoding, is used to enhance portability, to enable a variety of transport protocols to be used (e.g., session initiation protocol) and to allow flexible programming languages, such as java and C, to be used to generate and process session descriptions. A terminal description takes a number of lines of text of the form <key>=<value>. <key> can be a series of characters and <value> contains a structured text string whose format depends on <key> and can be a number of fields delineated by a single space character or free format string. Each description consists of three parts: general terminal information, the terminal's available interfaces and mobility protocols followed by a software description of available applications. An example of such a description is shown in Figure 5-1.

```
t= Mercury <mercury@kcl.ac.uk;123.34.65.3>
o=King's College London
d= Class A<audio;video;whiteboard>
l=strand;east wing;room E2
i1= GPRS
c1= 10 units
i2=WLAN
c2= 20 units
m=Mobile IPv4
m_secret_key= 'abc'
m_algorithms =MD5
a1=M2A:audio/video
a2=web browser/e-mail
a3= Jukebox
```

Figure 5-1: An example of a terminal description.

Terminal name: The 't' field gives the name of the terminal followed by the SIP address. The IP address can also be given enabling direct signalling communication thus avoiding going through a proxy server.

Owner: The 'o' field gives the terminal owner's name. This can be a company or another user.

Description: The 'd' field provides a description of the terminal in terms of classes. For example a Class A terminal might consists of a large colour screen with a high bandwidth connection, while class C might indicate a terminal with a limited display capabilities and a much lower bandwidth availability.

Interface: The'i' field is information about the terminals interfaces. Terminals that are multi-mode (containing more than one interface) may be described by more than one 'i' field per description. Each field describes the type of interface available on the terminal.

Cost: The 'c' field indicates the cost associated with each interface.

Mobility: The 'm' field specifies mobility related information and configuration settings for the terminal. These are specified using multiple instances of the 'm' field.

Applications: The 'a' field specifies available applications on the terminal. Applications are specified using multiple instances of the 'a' field as shown in the example.

5.2.3 Protocol Transactions

The protocol specification is a language independent protocol, which means that it can be implemented in any programming language. It bases its descriptions on capability attributes, which are essentially different ways of describing a terminal. The TDP infrastructure consists of two types of agents: user agents and service agents. The user agents reside on each terminal and have two functions. Registering the terminal with the service agent and discovering other terminals that match the user's criteria. The service agent resides on a separate server, also referred to as terminal description server (TDS), and maintains dynamic information about the available terminals and their descriptions. The TDS is found by using the session initiation protocol. Figure 5-2 illustrates the protocol entities and their relationships.

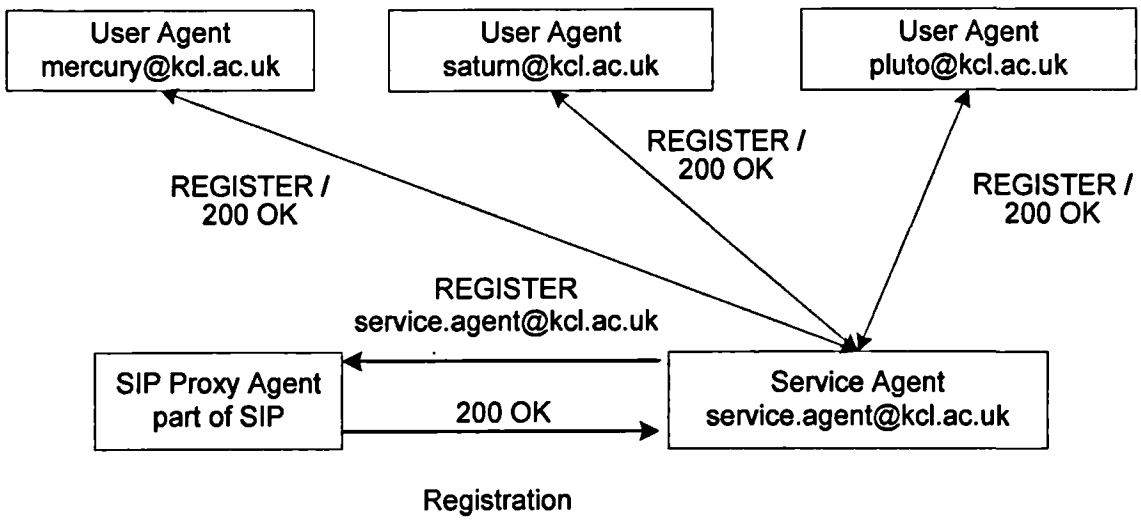


Figure 5-2: Protocol entities and their relationships.

A service agent first registers with the local sip proxy server using an identifier consisting of URI characters that is unique within its network environment (e.g. service.agent@kcl.ac.uk). The register request allows a service agent to let a sip proxy or a redirect server know at which address(es) it can be reached. Once the registration is complete, terminals will register their capabilities with the service agent to fulfil requests for terminal description information. Terminals are only required to discover the location of the local sip proxy server because it contains the whereabouts of the service agent. The discovery process is done by either pre-configuring each terminal or using DHCP [7], [8]. A user agent uses the REGISTER method described in [6] to register the terminal description information with the service agent. An example of a registration message is shown in Figure 5-3. A registration message includes a lifetime (in seconds) that will eventually expire and therefore should be refreshed before the lifetime runs out.

```
REGISTER service.agent@kcl.ac.uk SIP/2.0
Via: SIP/2.0/TCP ee150
From: sip:mercury@kcl.ac.uk
To: sip:service.agent@kcl.ac.uk
Call-ID: 377420@ee150
CSeq: 1 REGISTER
Contact: <sip:mercury@137.73.11.45:4000;transport=tcp>
Expires: 1200
Content-Type: application/tdp
Content-Length:

t= Mercury<mercury@kcl.ac.uk;137.73.11.45>
o=King's College London
d= Class A
l=strand;east wing;room E2
i1= GPRS
c1= 10 units
i2=WLAN:30
c2= 20 units
m=Mobile IPv4
a1=M2A:audio/video
a2=web browser/e-mail
a3= Jukebox
```

Figure 5-3: An example of a user agent registration message.

The 'From' field of the registration message above indicates that the request was initiated by mercury@kcl.ac.uk and addressed to service.agent@kcl.ac.uk. In this case, the terminal description is using TDP, as stated in the Content-Type header. The header is terminated by an empty line and is followed by a message body containing the terminal description. A sample response to the registration is given below. The response states the SIP message is a 200 OK response, which means the registration request, was successful. The Call-ID is taken directly from the original request, along with the remaining fields of the request message. The original sense of 'From' field is preserved.

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP ee150
From: sip:mercury@kcl.ac.uk
To: sip:service.agent@kcl.ac.uk
Call-ID: 377420@ee150
CSeq: 1 REGISTER
Contact: <sip:mercury@137.73.11.45:4000>;expires=THU,26JUN 2003
15:19:03 GMT
Content-Length: -1
```

Figure 5-4: Service agent response to a registration message.

The following describes the operations that a user agent (e.g. user A) will employ to find other available terminals on the site's LAN network. This is shown in Figure 5-5. The user agent obtains the location of the service agent, via the sip proxy server and unicasts a message to it in order to resolve a particular request. User A can either send a message that describes the terminals required that match the user's needs or request a list of all available terminals within the network. This is done using an INFO RX message. The service agent will, in return, unicast a reply to the user agent using an INFO TX message. If the service agent cannot service the request (say it has no information) it returns a response containing no message body. If no body is present in a message, then the Content-Length header field of that message is set to minus one. A successful response, however, will contain a list of terminals that are available including their descriptions. Based on this information the user agent selects an appropriate terminal that matches the user's needs. The response also contains the terminals location (e.g. IP address) that allows user A to send future signalling requests directly to it instead of through the sip proxy server. However, for security reasons the administrator might not want the IP addresses of the terminals to be exposed. In that case, the service

agent would have to be used as the mediator between the two thus hiding the address of the end terminals.

If a terminal becomes unavailable, it will deregister with the service agent by sending a registration request containing a zero lifetime. The service agent will response with an acknowledgment 200OK message as shown in Figure 5-5. step 4.

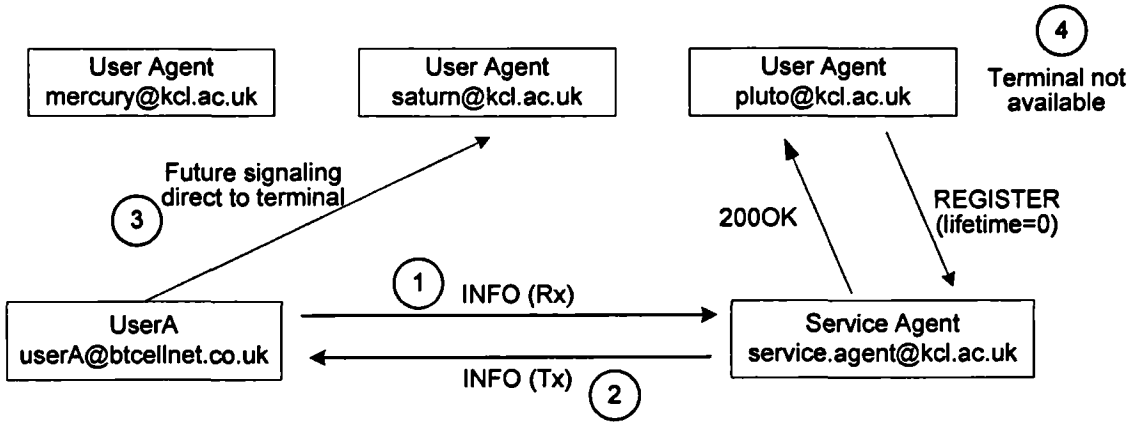


Figure 5-5: Finding other available terminals on the network.

5.2.4 The INFO Method

As explained in [9] the purpose of an INFO message is to carry session control information along the SIP signalling path during a session. This information will generally be carried in message bodies. However, the definition of the message bodies or any new headers for the INFO message is outside the scope of [9]. This section describes the use of the INFO message method for communicating between the user agents and the service agents. The SIP INFO method is used to carry the terminal description requests and responses between the two agents. When the user agent wishes to get information about other terminals on the network, it formulates and issues a SIP request using the new INFO method. An example of the INFO-RX message is shown in Figure 5-6. User A and the service agent belong to two different domains.

```
INFO sip:service.agent@kcl.ac.uk SIP/2.0
Mode: RX
Via: SIP/2.0/TCP proxy.kcl.ac.uk
From: userA<sip:userA@btcellnet.co.uk >
To: sip:service.agent@kcl.ac.uk
Call-ID: 163712@ee396
Contact: <sip:userA@btcellnet.co.uk:7000;transport=tcp>
CSeq: 1 INFO
Content-Type: application/tdp
Content-Length: 50

d= Class A ; Class B
c= < 20 units
l=east wing
m=Mobile IP
a1=audio/video
a2=web-browsing
```

Figure 5-6: An example of an INFO_Rx message

The meaning of some of the key INFO request-header fields is defined as follows. The 'Rx' field specifies that this INFO message is a request for terminal information capabilities. The 'From' field states that the request was initiated by user A and addressed to the service agent ('To' header field). The 'Via' field list the hosts along the full IP path from invitation initiator (the last element of the list) towards the callee. The 'Via' field indicates the path taken by the request message so far. This ensures that replies take the same path as the requests, which assists in firewall traversal and other unusual routing situations. Finally, the 'contact' field indicates that the response should be directed to the address given in the Contact field.

The body of the INFO message uses the Terminal Description Protocol to carry information requesting specific terminal capability. The user could specify any types of capabilities as defined in the Terminal Description Protocol. In the example above, the user indicates interest in Class A and B terminals (i.e. large colour screen with a high bandwidth connection) with a usage cost of less than 20 units, located in the east wing building. Furthermore, the user is only interested in terminals that support audio or video as well as web browsing. Lastly, mobility support is equally important to the user as he/she might be planning to perform a session handover. Therefore, Mobile IP support is also requested. Based on this information the service agent will respond by sending a list of terminals that match the user's criteria. If the body of an INFO_RX message is left empty the service agent will return a list of all available terminals without performing any filtering.

A sample response to the INFO_RX message is given below in Figure 5-7. The first line of the response states that it is an INFO message as well as giving the SIP version number. The 'Mode' field (TX) indicates that it is a response to a previously received INFO_RX request. The body message contains a list of terminals matching the users criteria. Only two terminals are specified in this example. Each terminal is described using the Terminal Description Protocol terminated by an empty line followed by the next terminal description.

```
INFO sip: SIP/2.0
Mode: TX
Via: SIP/2.0/TCP ee396
From: sip:service.agent@kcl.ac.uk
To: sip:userA@btcellnet.co.uk
Call-ID: 182834@ee150
CSeq: 1 INFO
Content-Type: application/tdp
Content-Length: 78

t= Mercury<mercury@kcl.ac.uk;137.73.11.45>
o=King's College London
l=strand;east wing;room E2
d= Class A
i=WLAN
c= 15 units
m=Mobile IPv4
a1=audio;video;web-browsing

t= Saturn <saturn@kcl.ac.uk;137.73.10.46>
o=King's College London
l=strand;east wing;room E5
d= Class B
i=GPRS
c= 12 units
m=Mobile IPv4
a=video;web-browsing
```

Figure 5-7: An example of an INFO_TX message.

5.3 Signalling for Session Handover

Once a terminal is located a handover mechanism is required to move the session from one device to the next. In this section, we present the signalling mechanism that enables a user to move an active session from one device to the next while keeping the session alive. Mobility is also considered in this section as it is a part of the same signalling mechanism. Session handover can be completely user-driven: the user discovers the new terminal, signals it, then hangs up the previous terminal. Automatic session handovers can be enabled when the three fundamental issues of discovery, signalling

and mobility are addressed. So far we have described the terminal description protocol used to describe terminal capabilities and, assisted by SIP, can locate the terminals. In the following section we investigate the basic session handover mechanisms as well as mobility.

5.3.1 Inviting a New Device

A mechanism is required that will allow a user to invite a new terminal in the middle of a session. The REFER method [10] provides such a mechanism by indicating that the recipient should contact a third party using the contact information provided in the request. Furthermore, it allows the party sending the REFER to be notified of the outcome of the referred request. This is illustrated better in the example shown in Figure 5-8.

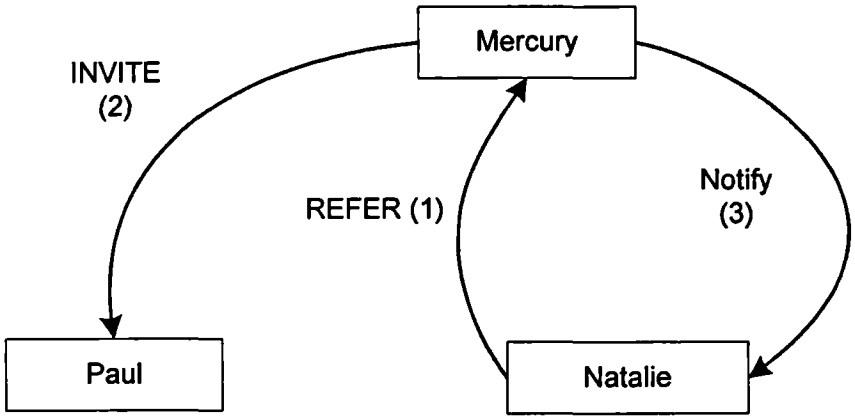


Figure 5-8: Inviting a new device using the REFER method

For instance, if Paul is in a call with Natalie, and Natalie decides to move to another terminal called Mercury, Natalie can instruct her SIP user agent (UA) to send a SIP REFER request to Mercury's UA providing Paul's SIP Contact information. Mercury's UA will attempt to call Paul using that contact. Mercury's UA will then report back to Natalie whether it succeeded in reaching the contact to Paul's UA.

5.4 Mobility Support

However, such a simple mechanism is not sufficient to enable session handovers. The challenge is to switch the current session from one device to another, (and as a result switching from one network to another) without having to restart it. The mechanisms described in the following section, focus specifically on Mobile IP as the underlying

mobility protocol while at the same time we assume that all terminals involved (e.g. Natalie and Mercury) except the correspondent host (e.g. Paul) support mobile IP.

5.4.1 Using Denial of Service

In this section, we propose to use the denial of service concept as the solution to session mobility. As explained in [11], a denial of service can have two forms. A bad guy floods a host with packets, thus preventing the host from processing useful packets or the bad guy interferes with the flow of useful packets to a node. To be more specific, in the case of Mobile IP, denial of service attacks occur when a bad guy manages to sent a bogus registration of a new care of address for a particular mobile node. If such a registration is successful it will give rise to two problems. First, the good guy's mobile node will be disconnected and secondly, all the traffic intended for that node will be re-directed to the bad guy. This is illustrated in Figure 5-9. These two security problems stated above are exactly what is required to solve session mobility. The similar concept is shown in Figure 5-10 by renaming each of the components originally illustrated in Figure 5-9, using names from Figure 5-8.

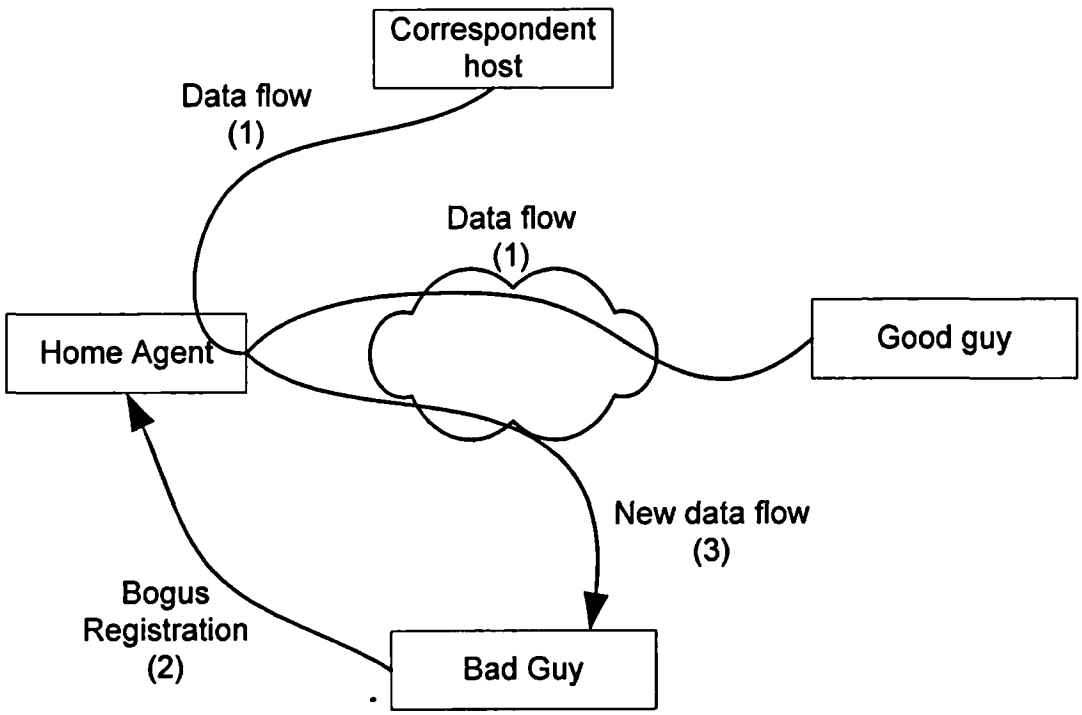


Figure 5-9: An example of denial of service using mobile IP

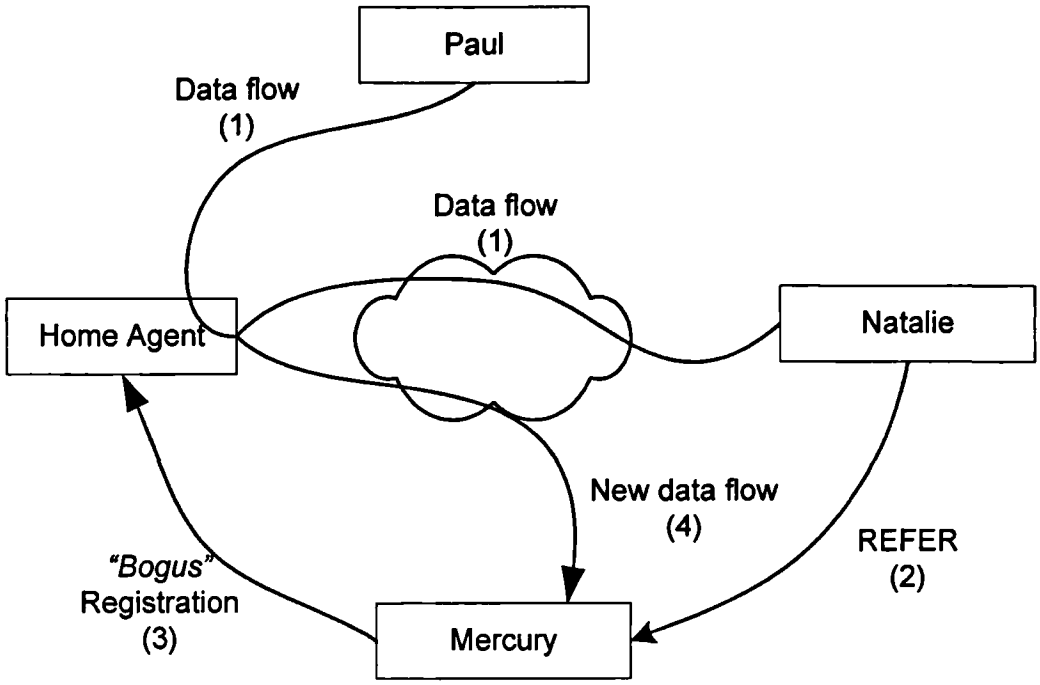


Figure 5-10: Session handover using denial of service

5.4.2 Enabling Session Handovers

The Mobile IP specification [12] has several solutions to prevent such attacks from happening. The first one includes strong authentication on all registration messages that are exchanged during the registration process. However, in order for this to work, the shared secret key between the home agent and the mobile node must not be exposed to anyone else. Secondly, messages should contain either a timestamp or a nonsense value that has previously been agreed between both the home agent and mobile node and is different for each message thus making an attack impossible.

In order to enable session handovers, the above security barriers need to be overcome in a secure and reliable way without posing or creating further security threats. The proposed solution is for the mobile node to provide the new terminal with all the necessary information that is required to perform the 'denial of service'. The following information is required: The mobile node's home address, it's security key with the home agent and the encryption mode that is using. Further configuration settings might also be required by the new terminal primarily depending on the version of mobile IP being used.

5.5 The Complete Signalling Mechanism

We have so far looked at the three fundamental issues to session handovers. In this section, we show how the terminal description protocol, the REFER signalling mechanism and mobility fit together to enable session handovers.

Figure 5-11 depicts the session handover process. Natalie is located within the coverage area of the home network while Paul is a correspondent host across the Internet. Natalie then performs a session handover to a public terminal, called Mercury, located within a foreign network. The handover process has the following steps:

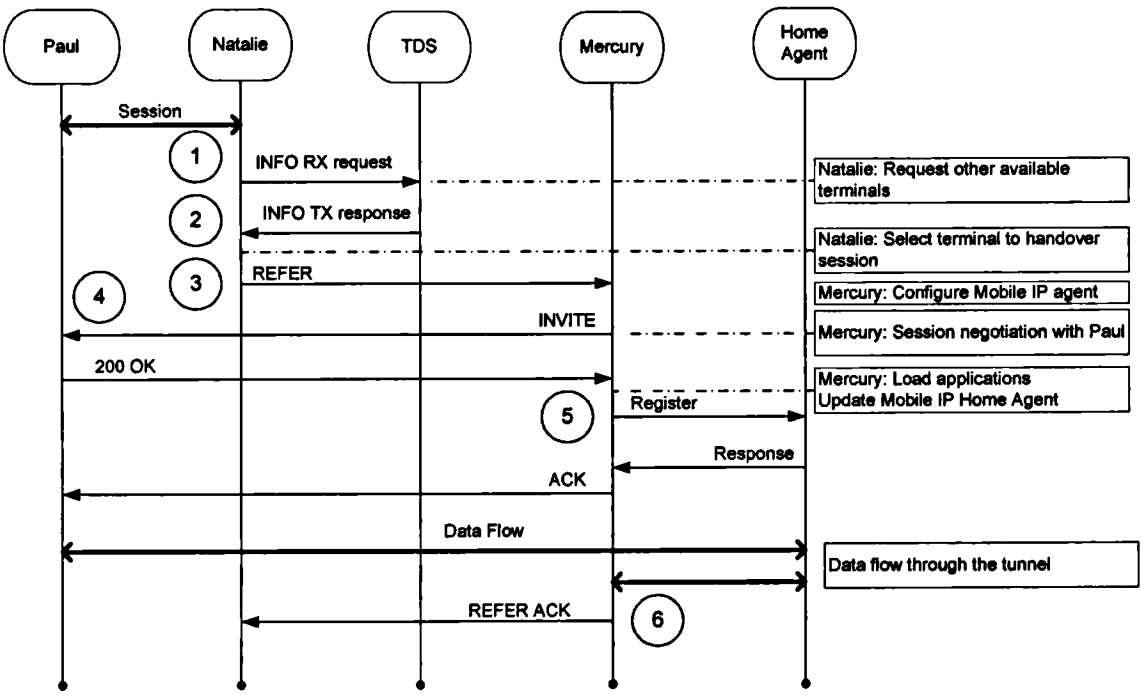


Figure 5-11: Session handover signalling process

1. Natalie is in a session with Paul when she decides to handover the session to another terminal. The user agent of Natalie sends an INFO_RX message to the terminal description server to indicate her intention to request availability of terminals matching specific criteria. The local sip proxy server receives this message and relays it to the appropriate TDS

2. The TDS looks up all the available terminals that match Natalie's criteria and response by sending an INFO_TX message back to Natalie providing the descriptions and locations of the available terminals (terminal IP addresses,

interfaces, mobility support and applications). This is done using the terminal description protocol as described in section 5.2.4

3. On receiving the response, Natalie chooses an appropriate terminal and sends a REFER message to the new terminal, also referred to as Mercury. The REFER message contains detailed information about Natalie's current session with Paul (e.g. Paul's IP address, session codecs and port numbers) as well as mobility related information (e.g. Natalie's home address, it's security key with the home agent and the encryption mode she is using). The REFER message indicates to Mercury that it should contact Paul using the contact information provided in the request.

4. Mercury attempts to establish a session with Paul by sending an INVITE request on behalf of Natalie using the same CALL-ID, but new body and header fields to convey the new information. This INVITE message has a higher sequence number (CSeq filed) than any previous request from Natalie to Paul. Furthermore, the INVITE request has a Contact header containing the home address of Natalie. This is to inform Paul that the session should keep using the home address of Natalie rather than the actual IP address of Mercury. The message body contains the new media types and formats that Mercury supports. Paul accepts the call, and returns a 200 OK message indicating a success.

5. Mercury loads the appropriate applications and configures its mobile IP agent based on the information received by Natalie in the REFER request. Once the configuration is complete, Mercury sends a registration message to the home agent making it believe that Natalie is performing a vertical handover in terms of terminal mobility. When the home agent receives this request it approves it, adds the necessary information to its routing table, sets up the tunnel to Mercury's care of address and sends a registration response back to Mercury. Once the mobile IP registration is complete, Mercury completes the SIP session negotiation by sending an ACK back to Paul.

6. Finally, Mercury sends a REFER ACK response back to Natalie's terminal notifying of the outcome of the referred request indicating that Paul has been contacted and the session successfully transferred.

5.6 Preserving Communication

A key problem was identified by the above signalling mechanism. This section discusses the problem and describes a solution for it.

Once Mercury is registered with the home agent, Natalie's terminal is no longer accessible for any incoming communications. As a result the REFER ACK message shown in step six in Figure 5-11 will fail to reach Natalie. This is because Mercury has lawfully registered a care of address that is associated with the home address of Natalie, thus in effect immobilising the terminal. In order to bypass this difficulty, a solution is proposed by which the terminal is given a pair of IP addresses. The first IP address is called the primary IP and is used as the main contact of the terminal. This is also the IP address that is 'borrowed' during a session handover. When the terminal looses its primary address, the secondary address is there to keep the terminal alive, maintaining the Internet connection. Figure 5-12 illustrates how the secondary IP address helps Natalie's terminal remain contactable during the session handover.

The REFER message is initiated over the secondary interface resulting in the REFER ACK message to be received on that interface. This not only provides a method to complete the session signalling but also enables Natalie's terminal to remain active after the handover. As a result incoming calls can still be received at Natalie's terminal as well as giving Natalie the option of switching back to her primary terminal when required.

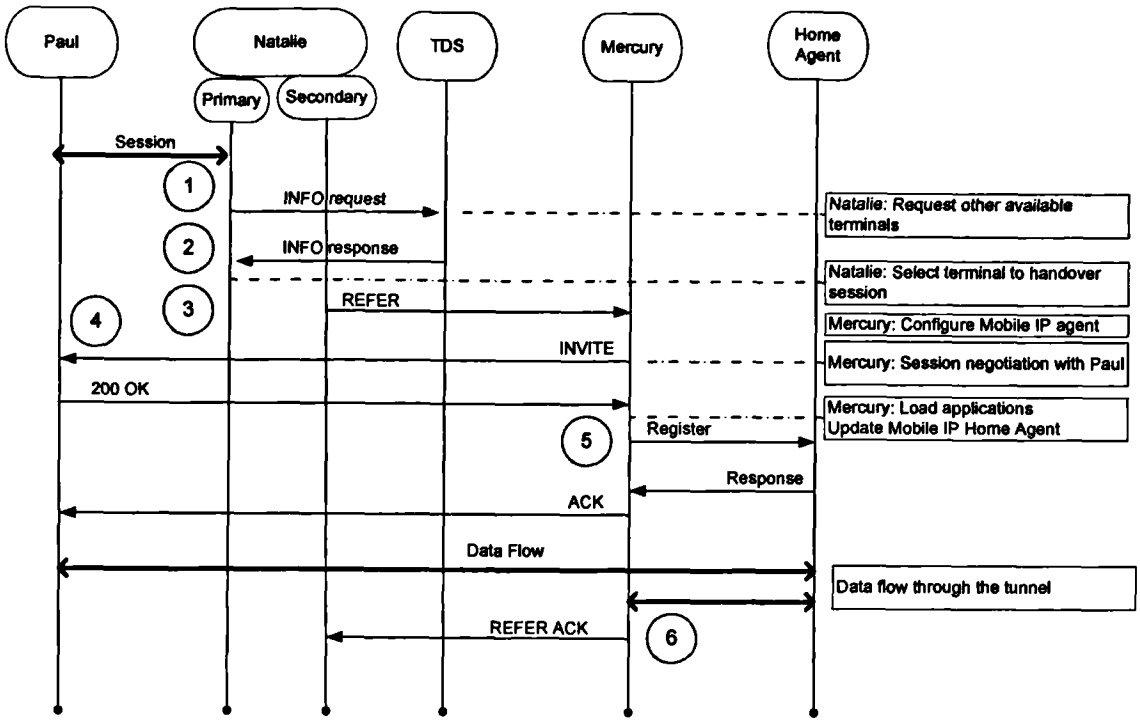


Figure 5-12: Preserving communication after a session handover

5.7 Experimental Results

A test implementation of the session handover mechanism, was made in order to verify its feasibility and to make preliminary measurements. There are several performance measures: signalling delay, packet loss, jitter, etc. Large parts of the session handover architecture have been implemented on a Linux platform using JAVA.

5.7.1 Implementation

The components and their interaction are indicated in Figure 5-13. The figure shows the components of each entity in the system as well as the interactions between them. The sip proxy server accepts registration requests from both users and terminal description servers notifying it at which address(es) they can be reached. The terminal description server accepts, processes and response to terminal description registrations and requests. The home agent is responsible for taking care of mobility of its registered users. Both the mobile node and the public terminal consist of similar components. They each have an audio, a whiteboard and a chat application. The sip user agent is tightly coupled with the mobile IP agent enabling synchronised message exchanges during session handovers. Furthermore the terminal description protocol was implemented and integrated within the sip user agent of both terminals. Finally the following tools were

used to monitor the performance and message exchanges during handovers. rtpdump to monitor received RTP packets, tcpdump to monitor the TCP/UDP signalling and NTPDATE to synchronise the clocks of each terminal. By synchronising the clocks the results obtained can be overlaid to produce graphs that have a common time axis.

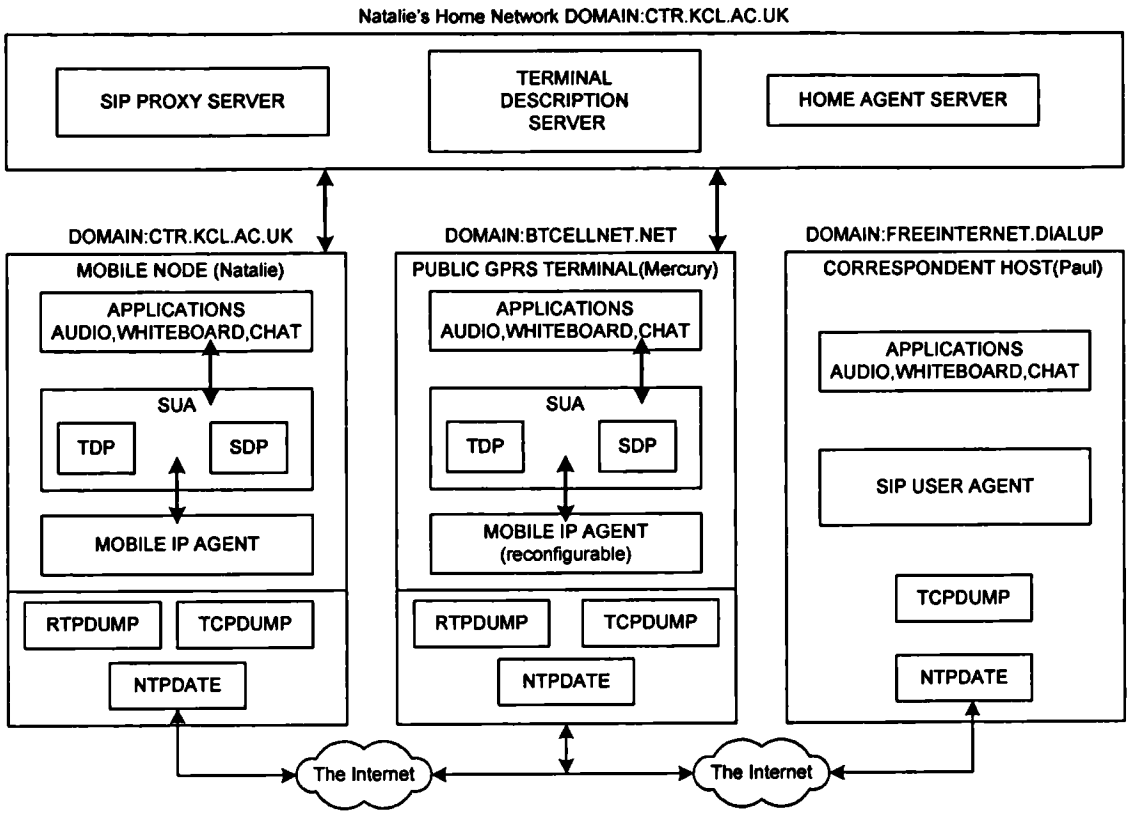


Figure 5-13: Test implementation components

All three servers are connected within the mobile nodes home network, the correspondent host is connected to the free-internet dialup domain while the public terminal to which the session will be transferred is connected to the BTCELLNET domain through a GPRS connection. This is illustrated more clearly in Figure 5-14.

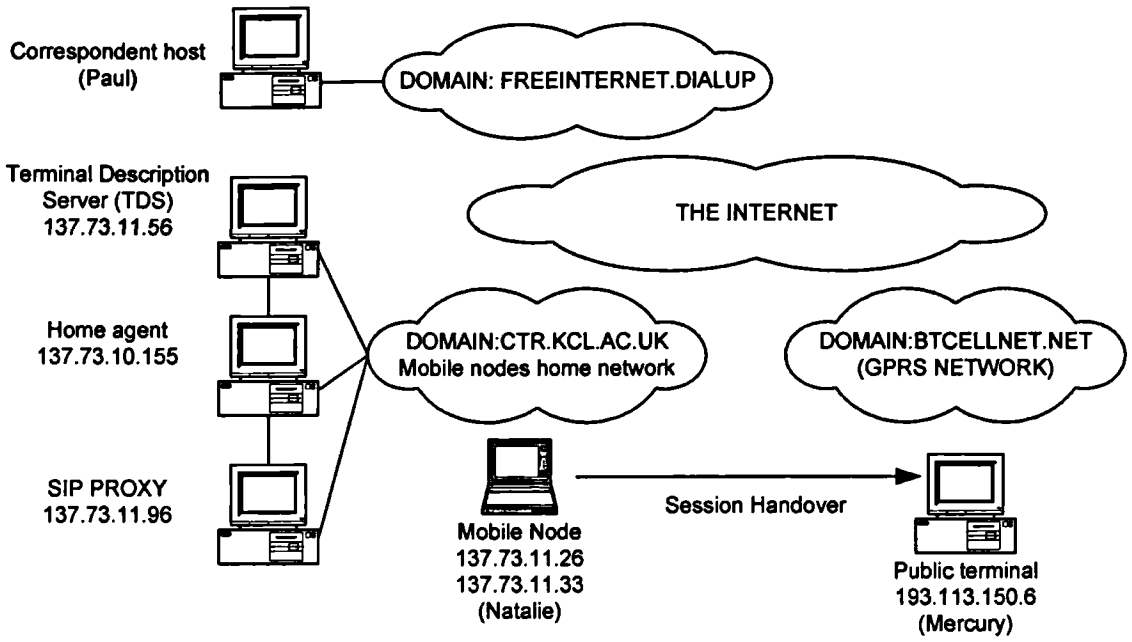


Figure 5-14: Testbed layout

5.7.2 Initial Configuration

In the following section the various components of the terminals are studied in more detail. The signalling messages are also shown and fully described[1].

5.7.2.1 Mobile Node- Registration with the Sip Proxy

The mobile node would first initialise its sip user agent. During this phase the sip user agent detects the two IP addresses of the terminal as shown in Figure 5-15. The primary IP is set as the default route and the secondary is stored to be used during the handover procedure. The terminal consists of only one physical interface but configured with two IP addresses. The second IP address is assigned to a virtual interface which in turn is mapped to the actual physical interface.

[1] There are a few discrepancies between the signalling messages presented in this thesis and some of the ones implemented in the testbed.

```
----------------------------------------
            Loading SUA...
----------------------------------------
Detecting OS...Linux
---> SIP MOBILITY MODE ON
---> SIP USING TCP
---> Found default interface:eth1
---> Active Interface SET to eth1
---> Primary Interface IP Address:137.73.11.26
---> Secondary Interface IP Address:137.73.11.33
---> SUA IP Address set to:137.73.11.26
---> loading config file natalie.cfg
```

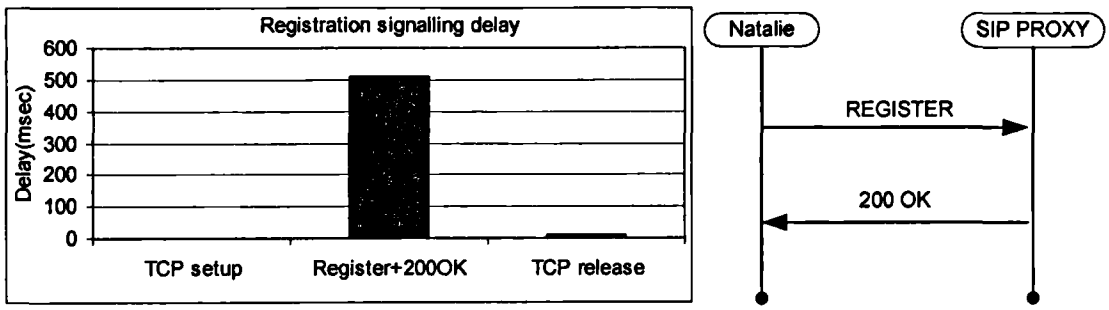
Figure 5-15: Initialisation of the sip user agent on the mobile node

The mobile node, also referred to as Natalie, uses the REGISTER method to register her terminal with the sip proxy server. The actual message exchange is shown in Figure 5-16.

```
OPENING DIALOG BOX
ESTABLISHING TCP CONNECTION WITH: /137.73.11.96:5060
THE ADDRESS IN USE: /137.73.11.26

CALLER SENT WITH TCP:
REGISTER sip:sipproxy@kcl.ac.uk SIP/2.0
Via: SIP/2.0/TCP ee150
From: sip:natalie.pangalos@kcl.ac.uk
To: sip:natalie.pangalos@kcl.ac.uk
Call-ID: 804833@ee026
CSeq: 1 REGISTER
Contact: <sip:natalie.pangalos@137.73.11.26:4000;transport=tcp>
Expires: 1200
Content-Length: -1

--->SUA: start new thread

SIPUSERAGENT CALLER RECEIVED TCP MESSAGE:
SIP/2.0 200 OK
Via: SIP/2.0/TCP ee150
From: sip:natalie.pangalos@kcl.ac.uk
To: sip:natalie.pangalos@kcl.ac.uk
Call-ID: 804833@ee026
CSeq: 1 REGISTER
Contact:  <sip:natalie.pangalos@137.73.11.26:4000>;expires=THU,  10  JUL
2003 12:41:40 GMT
Content-Length: -1

---> 200 OK for register
---> sip:natalie.pangalos@137.73.11.26:4000
---> location: sip:natalie.pangalos@137.73.11.26:4000
```

Figure 5-16: Natalie registering with the sip proxy server

The messages were also monitored by tcpdump measuring the delays associated with each part of the signalling. The signalling delays are shown in Figure 5-17. The connection setup and release (three way TCP handshake) times are 2msec and 10msec respectively. However, SIP registration takes 508msec to complete. This is the time taken for the sip proxy to process the request, add the user to its database and issue a 200 OK message.

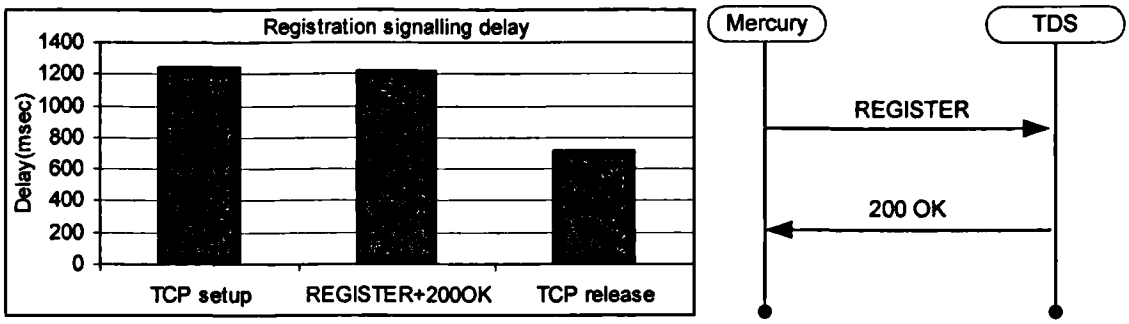



```
13:05:02.713116 ee026.32793 > 137.73.11.96.5060: S 877465654:877465654(0) win 5840
<mss 1460,sackOK,timestamp 358831[|tcp]> (DF)
13:05:02.715209 137.73.11.96.5060 > ee026.32793: S 590420:590420(0) ack 877465655
win 8760 <mss 1460> (DF)
13:05:02.715269 ee026.32793 > 137.73.11.96.5060: . ack 1 win 5840 (DF)
13:05:02.728513 ee026.32793 > 137.73.11.96.5060: P 1:266(265) ack 1 win 5840 (DF)
13:05:02.860212 137.73.11.96.5060 > ee026.32793: . ack 266 win 8495 (DF)
13:05:03.236712 137.73.11.96.5060 > ee026.32793: P 1:259(258) ack 266 win 8495 (DF)
13:05:03.236842 ee026.32793 > 137.73.11.96.5060: . ack 259 win 6432 (DF)
13:05:03.237551 137.73.11.96.5060 > ee026.32793: F 259:259(0) ack 266 win 8495 (DF)
13:05:03.245222 ee026.32793 > 137.73.11.96.5060: F 266:266(0) ack 260 win 6432 (DF)
13:05:03.247183 137.73.11.96.5060 > ee026.32793: . ack 267 win 8495 (DF)
```

Figure 5-17: Natalie's registration messages and signalling delay

5.7.2.2 Mercury - Registration with the TDS

The public terminal referred to as Mercury will also register with the sip proxy server in a similar way. Mercury issues a registration message to the terminal description server describing its hardware and software capabilities. Figure 5-18 shows the delays associated with the registration message. All messages are sent through the GPRS network resulting in higher delays. The TCP connection establishment is measured at just over 1200msec while the release delay is 719msec. In addition the REGISTER message takes 1200 msec to be processed and acknowledged. In this implementation, the terminal description server is located in the mobile nodes home network and shared between all devices. However, multiple servers can be deployed in each network thus reducing these delays as the signals will traverse within the local network.




```
13:07:12.005651 193.113.150.120.1043 > 137.73.11.96.bbs: SWE 1018173741:1018173741(0) v
5840 <mss 1460,sackOK,timestamp 135639 0,nop,wscale 0> (DF)
13:07:13.253794 137.73.11.96.bbs > 193.113.150.120.1043: S 590463:590463(0) ack 101817
win 8760 <mss 1460> (DF)
13:07:13.253960 193.113.150.120.1043 > 137.73.11.96.bbs: . ack 1 win 5840 (DF)
13:07:13.314338 193.113.150.120.1043 > 137.73.11.96.bbs: P 1:252(251) ack 1 win 5840 (I
13:07:14.533795 137.73.11.96.bbs > 193.113.150.120.1043: P 1:215(214) ack 252 win 8509
13:07:14.533965 193.113.150.120.1043 > 137.73.11.96.bbs: . ack 215 win 6432 (DF)
13:07:14.533809 137.73.11.96.bbs > 193.113.150.120.1043: F 215:215(0) ack 252 win 8509
13:07:14.561072 193.113.150.120.1043 > 137.73.11.96.bbs: F 252:252(0) ack 216 win 6432
13:07:15.253797 137.73.11.96.bbs > 193.113.150.120.1043: . ack 253 win 8509 (DF)
```

Figure 5-18: Mercury's registration with the terminal description server

5.7.3 Terminal Discovery

Before a session can be handed over to another terminal, the mobile node (i.e. Natalie) should discover and select the terminal to which she wants the session to be moved to. This is done by sending an INFO_RX message to the terminal description server. The message does not contain a body. As a result the user in return receives a list of all available terminals and selects the most appropriate one to initiate the handover to.

5.7.4 Handover Initiation

Natalie selects the appropriate terminal (i.e. Mercury) and initiates a handover request by sending a REFER message to Mercury (see Figure 5-20). The REFER message contains a set of header fields as well as two message body types. One is based on the terminal description protocol and the other on the session description protocol. This is shown in Figure 5-19. The header field contains the following information: The first line classifies the type of message being sent (i.e. REFER message), while the second line identifies the handover type as being associated with a session transfer (ST). The latter will indicate to Mercury that the user is not inviting another party to join the existing session (as is usually the function of a refer message) but rather requesting a session handover. The 'Refer-to' field indicates the terminal that Mercury should contact in order to re-negotiate the session as well as the sequence number of the last transaction made. The header also contains the same 'Call-ID' value indicating that the

message refers to the existing session of the user. The caller also specifies in the contact field that future responses and messages to her are to be sent directly to the secondary IP address. This is because Mercury will 'borrow' Natalie's home address during the session handover, in effect deactivating her primary IP address.

The session description provides Mercury with information about the current session of Natalie. This includes the media types, ports, and codec format that she is currently using to communicate with Paul. This information will be used by Mercury to generate a new session description body based on Mercury's capabilities as part of the new Invite message sent to Paul. The body of the second message includes a description of Natalie's terminal capabilities in terms of mobility settings such as secret key, home IP address, encryption settings, etc. This information enables Mercury to configure the mobile IP agent enabling it to successfully register with the home agent by borrowing Natalie's identity. The secure transfer of such information is of prime concern. Therefore, it is desirable that the REFER message is encrypted using the encryption field header option. The overall SIP security architecture is described in more detail in [6], section 13. Security mechanisms are outside the scope of this thesis.

```
CALLER SENT WITH TCP:
REFER sip:mercury@btcellnet.net SIP/2.0
Mode: ST
Via: SIP/2.0/TCP ee150
From: natalie.pangalos<sip:natalie.pangalos@kcl.ac.uk>
To: sip:mercury@btcellnet.net:
Refer-to=paul.pangalos@freeinternet.co.uk;Cseq:1 INVITE
Call-ID: 208232@ee026
CSeq: 2 REFER
Contact:sip:natalie.pangalos@kcl.ac.uk:137.73.11.33
Content-Type: application/sdp
Content-Length: 50

v=0
o=natalie.pangalos@kcl.ac.uk 369 656 IN IP4 137.73.11.33
s=RTP Audio Session
c=IN IP4 137.73.11.33
m=audio 5036 RTP/AVP 3 8 9

Content-Type: application/tdp
Content-Length:78

m=Mobile IPv4_hut
m_home_address=137.73.11.26
m_secret_key=paul96251181
m_authentication_algorithm=MD5
m_udp_port=434
```

Figure 5-19: The REFER message

When the REFER request is received subsequently, the following steps are taken (see Figure 5-20).

1. The mobile IP agent configuration file is updated based on the terminal description information received from Natalie.
2. A new Invite message is generated based on the header fields as well as session description information received. The INVITE is addressed to Paul and it contains the new media formats types that Mercury supports.

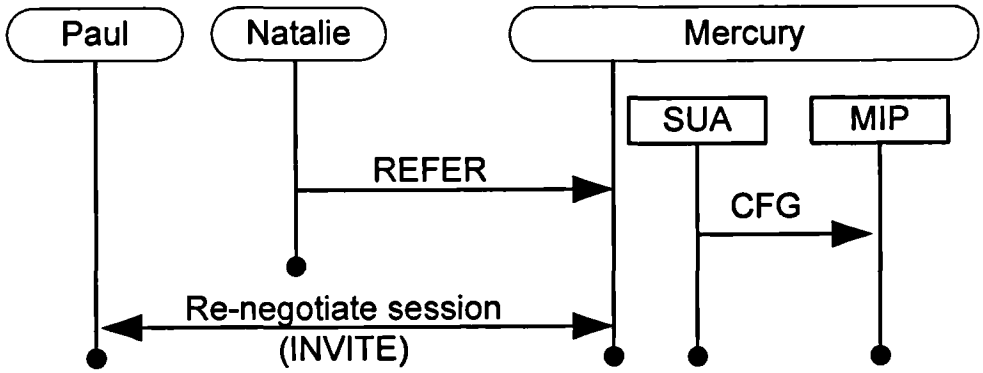


Figure 5-20: The REFER signalling message

5.7.5 Session Re-negotiation

The session re-negotiation observed from Mercury is shown in Figure 5-21. When Paul receives the INVITE it first checks the header fields. Since the received 'Call-ID' already exists, the 'Cseq' contains a higher sequence number and the 'To' and 'From' fields match, the request is identified as a retransmission for a session re-negotiation. The re-invite also carries another header field called 'handover type'. This informs Paul about the handover type that is being performed. In response Paul's SUA will take care of the order in which the adaptation takes place. For example, for upward handovers the session is downgraded and then a 200 OK is sent. However, during downward handovers, the adaptation takes place once an ACK is received from the mobile node. This ensures that no link congestion occurs. The body of the INVITE will also reveal the new codecs required. After Paul has agreed to adapt the session (by sending a 200 OK message) Mercury sends a binding update to the home agent and also confirms that it has received Paul's response by sending an ACK. If for some reason the mobile IP registration fails, Mercury will send a BYE message instead of an ACK.

```
CALLER SENT WITH TCP:
INVITE sip:paul.pangalos@freeinternet.co.uk SIP/2.0
Via: SIP/2.0/TCP
From: natalie.pangalos<sip:natalie.pangalos@kcl.ac.uk>
To: sip:paul.pangalos@freeinternet.co.uk
Call-ID: 341708@btcellnet
Handover_type: upward
CSeq: 3 INVITE
Content-Type: application/sdp
Content-Length: 98

v=0
o=mercury 341 252 IN IP4 193.113.150.120
c=IN IP4 193.113.150.120
m=audio 5036 RTP/AVP 1 8
--->SUA: start new thread

END OF MULTISERVER THREAD.

SIPUSERAGENT CALLER RECEIVED TCP MESSAGE:
SIP/2.0 200 OK
Via: SIP/2.0/TCP
From: sip:natalie.pangalos@kcl.ac.uk
To: sip:paul.pangalos@freeinternet.co.uk
Call-ID: 341708@btcellnet
CSeq: 3 INVITE
Contact: paul.pangalos@freeinternet.co.uk
Content-Type: application/sdp
Content-Length: 94

v=0
o=paul.pangalos@freeinternet.co.uk 369 656 IN IP4 80.225.170.18
c=IN IP4 80.225.170.18
m=audio 5036 RTP/AVP 1

---> Mobility set to: MIP
---> Home agent registration request…

CALLER SENT WITH TCP:
ACK sip:paul.pangalos@freeinternet.co.uk SIP/2.0
Via: SIP/2.0/TCP
From: sip:natalie.pangalos@kcl.ac.uk
To: sip:paul.pangalos@freeinternet.co.uk
Call-ID: 341708@btcellnet
CSeq: 3 ACK
Content-Length: 94

v=0
o=paul.pangalos@freeinternet.co.uk 369 656 IN IP4 80.225.170.18
c=IN IP4 80.225.170.18
m=audio 5036 RTP/AVP 1
```

Figure 5-21: Signalling exchanged between Mercury and Paul

Once the session re-negotiation and the home agent registration are successful, Mercury confirms the session handover request by sending a REFER-ACK to the location named in the Contact header of the previously received REFER message (i.e. 137.73.11.33). Natalie will receive this message through her secondary IP address, and complete the session handover operation. The REFER-ACK is shown in Figure 5-22.

```
MERCURY:
ESTABLISHING TCP CONNECTION WITH: 137.73.11.33/137.73.11.33:4000

CALLER SENT WITH TCP:
REFER sip:natalie@kcl.ac.uk SIP/2.0
Mode: ACK
Via: SIP/2.0/TCP
From:mercury@btcellnet.net <sip:mercury@btcellnet.net >
To: sip:natalie.pangalos@kcl.ac.uk
Refer-to=paul.pangalos@freeinternet.co.uk;Cseq:1 INVITE
Call-ID: 208232@btcellnet
CSeq: 2 REFER
Content-Type: application/sdp
Content-Length: 51

v=0
o=mercury@btcellnet.net 369 656 IN IP4 193.113.150.120
s=RTP Audio Session
c=IN 193.113.150.120
m=audio 5036 RTP/AVP 1
```

Figure 5-22: The REFER ACK message

5.7.6 Discussion on Signalling

This section looks in more detail at the delays associated with signalling messages during a session handover. Figure 5-31 shows the delays of each signalling message associated with Mercury's terminal. The REFER message takes 1.4 sec to complete after which the terminal updates its mobile IP agent configuration file based on the information included in the body of the REFER message. Once complete, the session is re-negotiated by sending a re-invite message through the sip proxy server. This makes one of the longest parts of the delay with a total of 8.76sec. This is the time taken for the messages to arrive over the GPRS network to the sip proxy server (in the home network) and then to the correspondent host, Paul, located at the 'freeinternet' domain. Paul responds by adapting the session and issues a 200 OK message following the reverse path.

This is illustrated in Figure 5-23. This delay can be improved by bypassing the sip proxy server. The REFER message received by Mercury could provide the IP address of Paul using the contact sip header field instead of providing his URI. In this way Mercury can contact Paul directly bypassing the sip proxy server. However, a disadvantage of this approach is that during the handover Paul might also change location (by performing a handover) which means that his old IP contact address will be invalid.

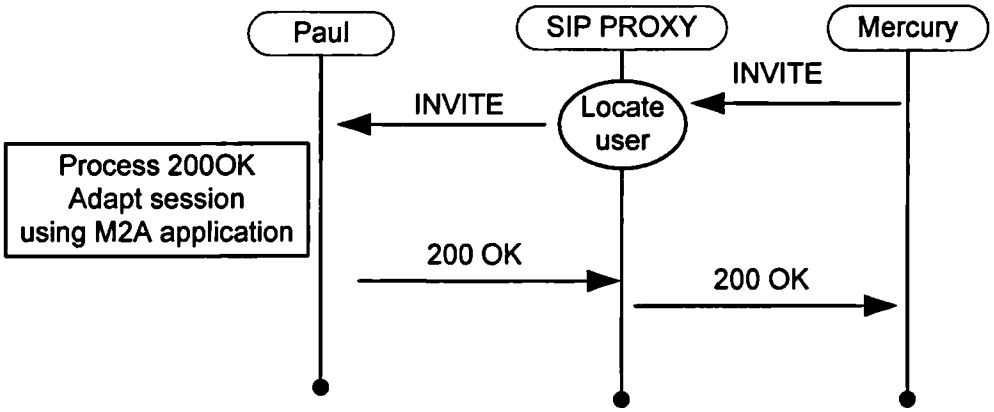


Figure 5-23: Session renegotiation through sip proxy

A 200 OK message indicates to Mercury that a handover can now be initiated. The detailed process is shown in Figure 5-24. The sip user agent initiates the handover, triggering the mobile IP agent to send a registration message to the home agent, successfully registering Mercury. The registration delay was measured at 2.25 seconds. This is the time taken to receive the registration reply once a registration request is sent.

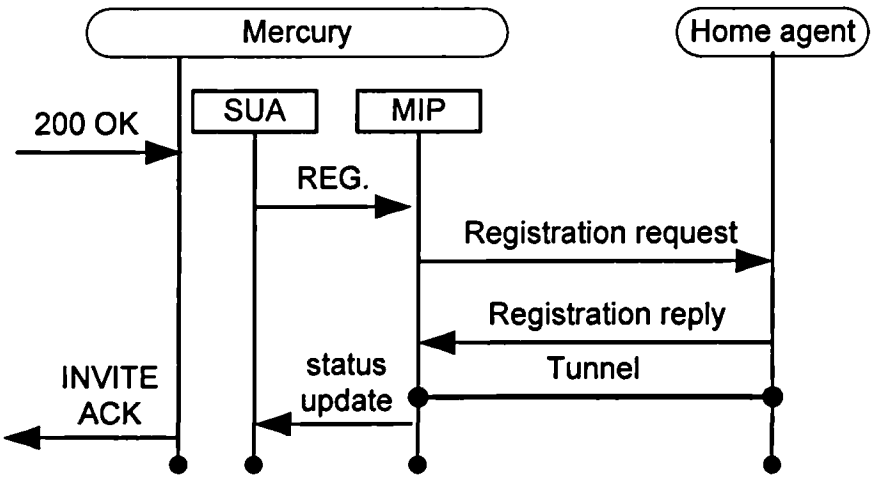


Figure 5-24: Mobile IP registration process

After successful registration, Mobile IP informs the sip user agent of the success prompting it to complete the session re-negotiation by sending the final INVITE ACK message to Paul. If for some reason the mobile IP registration fails, Mercury will send a BYE request instead of an ACK. Special considerations should be taken when sending the ACK message as this is part of an already ongoing TCP connection that was initiated with Mercury before the handover occurred. There are two scenarios that need to be considered and these are discussed below.

5.7.6.1 Mobile IP with Reverse Tunnelling

The first scenario looks at using mobile IP in a reverse tunnelling mode. Figure 5-25 shows the routing table of Mercury. The first table shows the configuration of the host before the handover. The GPRS interface ppp0 is set as the default route of the terminal. However, after the handover the default route is replaced by the tunnel interface of mobile IP. This shows that mobile IP is using a reverse tunnelling mode. That means, packets are received and sent through the home agent using the tunnel. This is not the most efficient way to route packets, but there are several uses of reverse tunnels [13]. In short, reverse tunnels preclude the need for recursive tunneling, they enable mobile nodes away from home to join multicast groups in their home network and they solve the problem of packet TTL expiry since reverse tunnels represent a TTL decrement of one.

```
Kernel IP routing table - before handover
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        *               255.255.255.255 UH    0      0        0 ppp0
default         10.0.0.0        0.0.0.0         UG    0      0        0 ppp0

Kernel IP routing table - after handover ( reverse tunneling)
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        *               255.255.255.255 UH    0      0        0 ppp0
137.73.10.155   10.0.0.0        255.255.255.255 UGH   0      0        0 ppp0
default         *               0.0.0.0         U     0      0        0 TUNLMNA
```

Figure 5-25: Mercury's routing table before and after the handover using reverse tunnelling

As a result of using reverse tunnelling, the INVITE ACK will be encapsulated and sent through the tunnel. This is illustrated in Figure 5-26. This shows the two interfaces configured in the kernel routing table. The first one is referred to as Iface:ppp0 associated with the care of address of the terminal and the iface:tunnel which connects the host with the home agent and is associated with the home address of the terminal.

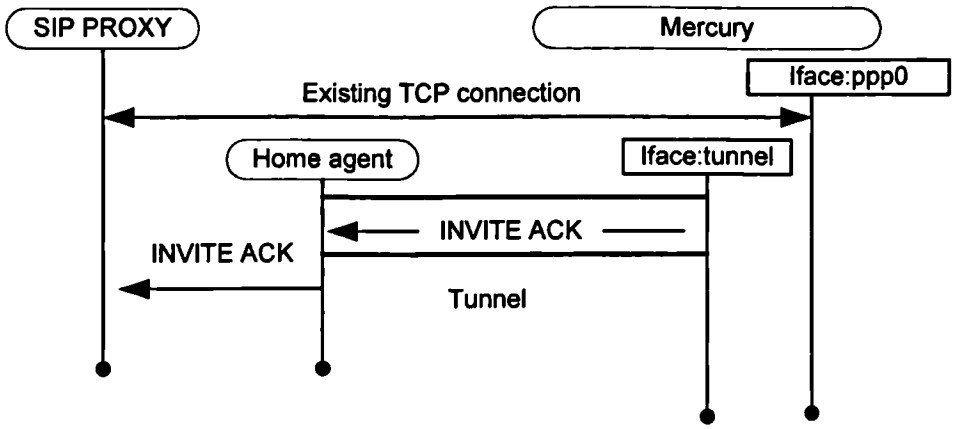


Figure 5-26: Encapsulation of the INVITE ACK message

The actual transmission of the ACK message is shown in Figure 5-27. This message is part of an already existing TCP connection setup between the sip proxy (137.73.11.96) server and mercury's ppp0 interface (193.113.150.22). Since the packet is encapsulated it contains an inner and outer part. The outer packet contains the GPRS co-located care of address (193.113.150.22) as the source address and is addressed to the home agent (137.73.10.155). The inner packet contains the final destination address (137.73.11.96) as the target address and the co-located care of address as the source. Under normal circumstances the inner packet source address should be the home address of the user. However, in order to maintain the original TCP connection that the ACK belongs to, the care of address is used instead. By using the care of address as the source address of the inner packet, the packet becomes vulnerable to ingress filtering. However, in this particular scenario the packet reaches its final destination in 1.18 seconds. This is because the sip proxy server and the home agent are both located within the same administrative domain. If the sip proxy was in another domain the packet will most likely not make it to the destination as the source IP address of a packet transmitted will not correspond to the network prefix from where it emanates.

```
INVITE ACK
18:23:41.886517 193.113.150.22 > 137.73.10.155: 193.113.150.22.1096 >
137.73.11.96.5060: P 383:704(321) ack 363 win 6432 (DF) (ipip)
18:23:41.912451 193.113.150.22 > 137.73.10.155: 193.113.150.22.1096 >
137.73.11.96.5060: F 704:704(0) ack 363 win 6432 (DF) (ipip)
18:23:43.025282 137.73.11.96.5060 > 193.113.150.22.1096: . ack 704 win
8057 (DF)
18:23:43.065290 137.73.11.96.5060 > 193.113.150.22.1096: . ack 705 win
8057 (DF)
```

Figure 5-27: The INVITE ACK message sent through the tunnel interface

5.7.6.2 Mobile IP with Triangular Routing

Mobile IP can also be configured to operate in a triangular routing mode. The routing table configuration is shown in Figure 5-28. In this setup packets destined for the home network (137.73.10.0) are only routed through the tunnel while all other packets are sent directly out of the ppp0 interface. This is more desirable for web browsing as well as e-mail sessions that do not require a bidirectional tunnel. In order to avoid the issue of ingress filtering, the source and destination IP addresses of the packet must be topologically correct. The forward tunnel complies with this, as its endpoints (home agent address and care-of address) are properly assigned addresses for their respective locations. On the other hand, the source IP address of a packet transmitted by the mobile node does not correspond to the network prefix from where it emanates.

```
Kernel IP routing table - before handover
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        *               255.255.255.255 UH    0      0        0 ppp0
default         10.0.0.0        0.0.0.0         UG    0      0        0 ppp0

Kernel IP routing table - sending INVITE ACK (triangular routing)
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        *               255.255.255.255 UH    0      0        0 ppp0
137.73.11.96    10.0.0.0        255.255.255.255 UGH   0      0        0 ppp0
137.73.10.155   10.0.0.0        255.255.255.255 UGH   0      0        0 ppp0
137.73.10.0     *               255.255.254.0   U     0      0        0 TUNLMNA
default         10.0.0.0        0.0.0.0         UG    0      0        0 ppp0
```

Figure 5-28: Mercury's routing table before and after the handover using triangular routing

Since the sip proxy server is located within the home network domain of the user, the INVITE ACK packet will be sent through the tunnel. In order to avoid any ingress filtering issues mentioned above, a host specific route is added to the routing table (shown in Figure 5-28) pointing to the sip proxy server (137.73.11.96). Consequently, the acknowledgment is routed directly to the sip proxy server through the ppp0 interface having a source address the care of address, thus maintaining the TCP connection without being compromised by ingress filtering.

5.7.6.3 The Refer ACK Packet

Once the INVITE ACK is sent, the session re-negotiation and handover procedures are complete. The success has to be also relayed to the originator of the REFER message by sending a REFER ACK back to Natalie's terminal. As mentioned earlier, Natalie has a single Ethernet interface configured with two IP addresses, a primary IP address 137.73.11.26 and a secondary 137.73.11.33. Since the primary IP has been assigned to Mercury, the secondary IP is used as the target address of the REFER ACK message. Special consideration should be taken as to the source IP address of this message. The first half of Figure 5-29 shows a failed attempt to send the REFER ACK using Mobile IP.

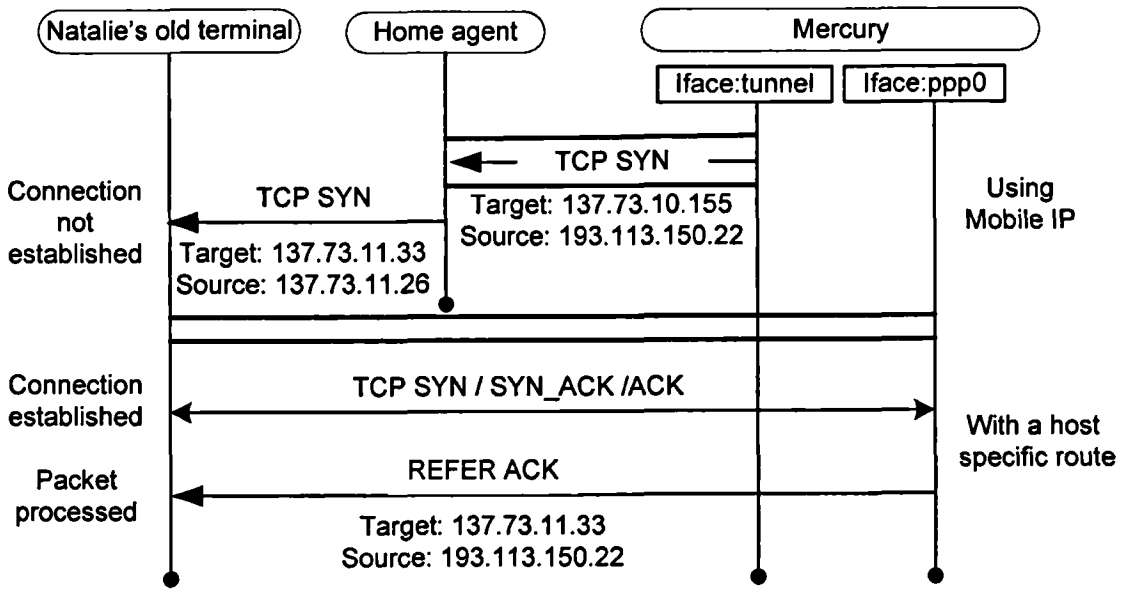


Figure 5-29: Sending the REFER ACK message

In order to initiate a TCP connection, Mercury sends a TCP packet with the SYN flag set. Natalie receives the packet but is unable to respond to Mercury since the source IP address indicated in the packet is also the target IP address of its own terminal. As a result Mercury cannot establish a communication with Natalie. This is also the case when Mobile IP is used in a triangular mode. In order to rectify the problem, a host specific route pointing to Natalie's secondary address is added in the routing table of Mercury as shown in Figure 5-30. This enables Mercury to communicate directly with Natalie' s old terminal by sending packets having a source address the co-located GPRS address. This is shown in the bottom half of Figure 5-29. The REFER ACK takes 2.55 seconds to complete.

```
Kernel IP routing table -
Destination     Gateway         Genmask          Flags Metric Ref    Use Iface
10.0.0.0        *               255.255.255.255 UH    0      0        0 ppp0
137.73.11.33    *               255.255.255.255 UH    0      0        0 ppp0
137.73.10.155   10.0.0.0        255.255.255.255 UGH   0      0        0 ppp0
default         *               0.0.0.0         U     0      0        0 TUNLMNA
```

Figure 5-30: Routing table of Mercury for sending REFER ACK

A summary of each signalling delay is shown in Figure 5-31 as well as the actual message exchange in Figure 5-32.

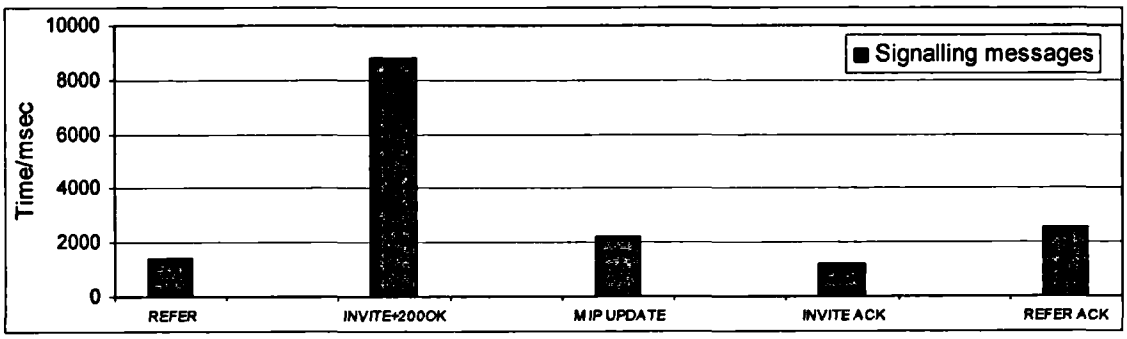


Figure 5-31: Summary of signalling delays at Mercury

REFER

```
18:23:28.215290 137.73.11.26.32841 > 193.113.150.22.4000: S 168966378:168966378(0) win 5840 (DF)
18:23:28.215470 193.113.150.22.4000 > 137.73.11.26.32841: S 165260501:165260501(0) ack 168966379 win
5792 (DF)
18:23:28.845296 137.73.11.26.32841 > 193.113.150.22.4000: . ack 1 win 5840 (DF)
18:23:28.965292 137.73.11.26.32841 > 193.113.150.22.4000: P 1:306(305) ack 1 win 5840 (DF)
18:23:28.965444 193.113.150.22.4000 > 137.73.11.26.32841: . ack 306 win 6432 (DF)
18:23:28.971073 193.113.150.22.4000 > 137.73.11.26.32841: F 1:1(0) ack 306 win 6432 (DF)
18:23:29.615289 137.73.11.26.32841 > 193.113.150.22.4000: F 306:306(0) ack 2 win 5840 (DF)
18:23:29.615432 193.113.150.22.4000 > 137.73.11.26.32841: . ack 307 win 6432 (DF)
```

INVITE

```
18:23:28.973954 193.113.150.22.1096 > 137.73.11.96.5060: SWE 168592330:168592330(0) win 5840 (DF)
18:23:29.645293 137.73.11.96.5060 > 193.113.150.22.1096: S 983804:983804(0) ack 168592331 win 8760
(DF)
18:23:29.645454 193.113.150.22.1096 > 137.73.11.96.5060: . ack 1 win 5840 (DF)
18:23:29.702273 193.113.150.22.1096 > 137.73.11.96.5060: P 1:383(382) ack 1 win 5840 (DF)
18:23:30.715287 137.73.11.96.5060 > 193.113.150.22.1096: . ack 383 win 8378 (DF)
```

200 OK

```
18:23:37.735295 137.73.11.96.5060 > 193.113.150.22.1096: P 1:363(362) ack 383 win 8378 (DF)
18:23:37.735450 193.113.150.22.1096 > 137.73.11.96.5060: . ack 363 win 6432 (DF)
```

MOBILE IP REGISTRATION

```
18:23:38.847554 193.113.150.22.1024 > 137.73.10.155.434: udp/rtp 52 c32  300 2303265562 (DF)
18:23:41.095292 137.73.10.155.434 > 193.113.150.22.1024: udp/rtp 68 c0  300 2303265562 (DF)
```

```
INVITE ACK
18:23:41.886517 193.113.150.22 > 137.73.10.155: 193.113.150.22.1096 > 137.73.11.96.5060: P
383:704(321) ack 363 win 6432 (DF) (ipip)
18:23:41.912451 193.113.150.22 > 137.73.10.155: 193.113.150.22.1096 > 137.73.11.96.5060: F 704:704(0)
ack 363 win 6432 (DF) (ipip)
18:23:43.025282 137.73.11.96.5060 > 193.113.150.22.1096: . ack 704 win 8057 (DF)
18:23:43.065290 137.73.11.96.5060 > 193.113.150.22.1096: . ack 705 win 8057 (DF)

REFER ACK
18:23:46.961348 193.113.150.22.1097 > 137.73.11.33.4000: SWE 198749578:198749578(0) win 5840 <mss
1460,sackOK,timestamp 438783 0,nop,wscale 0> (DF)
18:23:47.975282 137.73.11.33.4000 > 193.113.150.22.1097: S 186785979:186785979(0) ack 198749579 win
5792(DF)
18:23:47.975446 193.113.150.22.1097 > 137.73.11.33.4000: . ack 1 win 5840 (DF)
18:23:48.016883 193.113.150.22.1097 > 137.73.11.33.4000: P 1:313(312) ack 1 win 5840 (DF)
18:23:48.855288 137.73.11.33.4000 > 193.113.150.22.1097: . ack 313 win 6432 (DF)
18:23:48.885288 137.73.11.33.4000 > 193.113.150.22.1097: F 1:1(0) ack 313 win 6432 (DF)
18:23:48.892394 193.113.150.22.1097 > 137.73.11.33.4000: F 313:313(0) ack 2 win 5840 (DF)
18:23:49.515289 137.73.11.33.4000 > 193.113.150.22.1097: . ack 314 win 6432 (DF)
```

Figure 5-32: Actual messages sent and received by Mercury

Figure 5-33 shows a trace of the session handover described above. Packets are monitored at both the old and the new terminal during the handover. The clocks of both terminals are synchronised using the NTPDATE utility. The figure shows:

(i).The instant when the source adapts codecs in response to a session re-negotiation initiated by mercury (indicated as media adaptation).
(ii) The instant when Mercury successfully registers with the home agent thus intercepting packets destined for Natalie (indicated as MIP registration request and response)
(iii).The signalling packets sent during the handover procedure (indicated by the vertical arrows). These are explained in the following.

When the correspondent host (Paul) receives the re-INVITE request from Mercury, the media stream is downgraded based on the requested codecs. In this experiment the codecs are changed from PCMU@64kbps to LPC @ 5.6kbps. The last PCMU packet received is at 1.87sec followed by the new LPC packet 280msec later. Paul responds by sending a 200 OK message which is received by Mercury at 4.63 sec triggering the session handover. The registration with Mobile IP takes 2.38 seconds to complete with the first packet arriving at Mercury at 7.79 sec. During this handover no packet loss is observed. All the messages exchanged during the handover are shown in Figure 5-34.

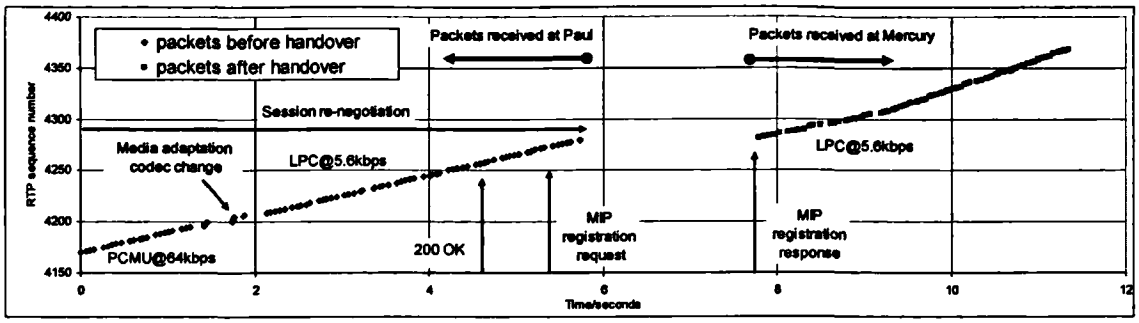


Figure 5-33: Session handover observed at Mercury

```
INVITE
18:16:36.676049 193.113.150.30.1158 > 137.73.11.96.5060: SWE 43708654:43708654(0) win 5840  (DF)
18:16:37.264390 137.73.11.96.5060 > 193.113.150.30.1158: S 724031:724031(0) ack 43708655 win 8760
(DF)
18:16:37.264541 193.113.150.30.1158 > 137.73.11.96.5060: . ack 1 win 5840 (DF)
18:16:37.311625 193.113.150.30.1158 > 137.73.11.96.5060: P 1:374(373) ack 1 win 5840 (DF)
18:16:38.484386 137.73.11.96.5060 > 193.113.150.30.1158: . ack 374 win 8387 (DF)

200OK received
18:16:46.434382 137.73.11.96.5060 > 193.113.150.30.1158: P 1:365(364) ack 374 win 8387 (DF)
18:16:46.434520 193.113.150.30.1158 > 137.73.11.96.5060: . ack 365 win 6432 (DF)

Mobile IP Registration
18:16:47.547139 193.113.150.30.1024 > 137.73.10.155.434: udp/rtp 52 c32  300 2303265562 (DF)
18:16:49.924370 137.73.10.155.434 > 193.113.150.30.1024: udp/rtp 68 c0  300 2303265562 (DF)
18:16:49.994379 137.73.10.155 > 193.113.150.30: 137.73.11.217.6000 > 137.73.11.26.5036: udp/rtp 99 c3
4281 369538 (ipip)
18:16:50.054378 137.73.10.155 > 193.113.150.30: 137.73.11.217.6000 > 137.73.11.26.5036: udp/rtp 99 c3
4282 370017 (ipip)
18:16:50.114369 137.73.10.155 > 193.113.150.30: 137.73.11.217.6000 > 137.73.11.26.5036: udp/rtp 99 c3
4283 370496 (ipip)
18:16:50.174383 137.73.10.155 > 193.113.150.30: 137.73.11.217.6000 > 137.73.11.26.5036: udp/rtp 99 c3
4284 370975 (ipip)
```

Figure 5-34: Message exchange at Mercury during session handover

5.8 Post Handover Signalling

Upon completion of the handover, Natalie's own terminal would no longer receive any new incoming calls. Her local sip proxy will direct any new calls to her primary IP address which is currently being used by Mercury. Furthermore, as the signalling protocol stands, Natalie has no way to transfer her session back to her own terminal if necessary. Therefore, additional signalling is required to resolve the above issue. Post handover signalling refers to the signalling required after the handover procedure is completed. This is illustrated in Figure 5-35. Natalie's terminal will cancel the existing primary interface registration with her local sip proxy server by sending a REGISTER request with an expiration time of zero seconds, while at the same time requesting a registration for the secondary interface (step7). The sip proxy will return a list of registrations in the 200 OK response indicating the result. A further terminal description type of registration message is send to the service agent enabling Natalie to switch back to her old device if required (step8). This registration message also indicates that only Natalie is allowed to handover back to this terminal making this entry inaccessible to other users. On the other hand, Mercury sends a de-registration message to the TDS

server indicating that it is no longer available to other users (step9). The TDS server would mark that terminal BUSY and would no longer consider it in the list of available terminals. Natalie might also decide that future incoming calls should not be directed to her old terminal but rather be routed to the current terminal Mercury. In that case, her old terminal will no longer be required to register the secondary IP address with the sip proxy but Mercury will register instead (step10). Since this entry is temporary (only valid while Natalie is using Mercury) it should be deleted immediately after Natalie has finished working with that terminal.

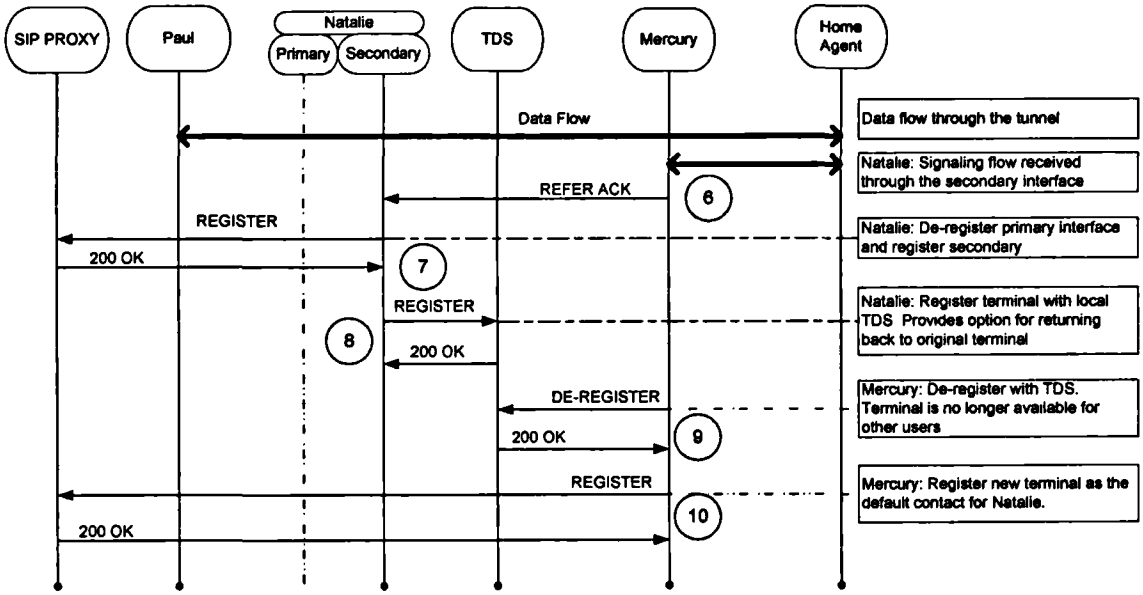


Figure 5-35: Post handover signalling

5.9 Summary

This chapter presented the design and implementation for session mobility to enable users to switch from one device to another in the middle of a session. The terminal description protocol (TDP) was presented used for describing terminals in terms of hardware and software capabilities. The protocol provides short textual descriptions of the terminals interfaces, mobility protocols and application information that are required to decide whether a terminal is likely to be of interest to a user. The mechanism and signalling required for session handovers is also described. This focuses specifically on Mobile IP, proposing a concept based on denial of service as the solution to session mobility. Finally, a test implementation of the session handover mechanism was carried out, in order to verify its feasibility and to make preliminary measurements.

In this chapter the research focused on session handovers between the user's home network and a foreign network. In the future, we intend to investigate handover procedures (i) between foreign networks, (ii) from foreign networks to the user's home network and (iii) between terminals located within the user's home network. There are a number of interesting research issues related to these scenarios. There is a need to look at the signalling between the hosts as well as the different security issues that arise. For example, how does the tunnelling mechanism work when the user is performing session handovers within its own home network.

5.10 References

[1] Helen J. Wang and Randy H. Katz, "Mobility Support in Unified Communication networks", Workshop on Wireless Mobile Multimedia, Rome, Italy, July 2001.

[2] Faramak V et al., "Mobility Management in a SIP Environment Requirements, Functions and Issues." draft-itsumo-sip-mobility-req-02.txt, June 2001.

[3] Bjorn Landfeldt, Tomas Larsson, "SLM, A framework for session layer mobility management", in Proc. IEEE International Conference on Computer Communications and Networks, pages 452--456, Natick, Massachusetts, October 1999.

[4] Anthony D. Joseph, B. R. Badrinath, and Randy H. Katz. The case for service over cascaded networks. In the first ACM/IEEE International Conference on Wireless and Mobile Multimedia (WOWMoM'98), 1998.

[5] M. Handley, V. Jacobson, "SDP: Session description protocol", Request for Comments 2327, Internet Engineering Task Force, April 1998.

[6] J. Rosenberg, H. Schulzrinne, G. Camarillo, "SIP: Session Initiation Protocol", Request for Comments 3261, Internet Engineering Task Force, June 2002.

[7] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", Request for Comments 1533, Internet Engineering Task Force, March 1993.

[8] R. Droms, "Dynamic Host Configuration Protocol, Request for Comments 2131, Internet Engineering Task Force, March 1997.

[9] S. Donovan, "The SIP INFO Method", Request for Comments 2976, Internet Engineering Task Force, October 2000.

[10] R. Sparks, "The Session Initiation Protocol (SIP) Refer Method", Request for Comments 3515, Internet Engineering Task Force, April 2003.

[11] Gloria Tuquerres, Marcos Rogério Salvador and Ron Sprenkelse, "Mobile IP: Security and Application", Technical Report, Telematics Systems and Services, Centre for Telematics and Information Technology, University of Twente, The Netherlands, December 1999.

[12] C.E. Perkins, " Mobile IP Design Principles and Practise", Prentice-Hall, 1998.

[13] G. Montenegro, "Reverse Tunnelling for Mobile IP", Request for Comments 2344, Internet Engineering Task Force, May 1998.

Ability is what you're capable of doing, motivation determines what you do,
attitude determines how well you do it.
"Be transformed by the renewal of your mind"
(Romans12:2)

Chapter 6

The Mobile Multimedia Application

This chapter describes an approach to media adaptation using the java media framework API. The essence of this model is a collaborative partnership between the session initiation protocol (SIP) and the M2A application that allows seamless codec change in response to a vertical handover.

6.1 Introduction

Adaptation is the key to mobility. Only through alertness and prompt reactions can a mobile node adapt in response to a vertical handover. Media adaptation is the process of adjusting the streaming parameters of the application during a change in the transmission characteristics that could be due to a vertical handover. Media adaptation can be performed using a broad variety of mechanisms, including file switching, codec switching, filtering and transcoding. Examples can be found in [1] and [2]. From the horizontal point of view, adaptation can be performed in an end-to-end approach (in the end systems) or using a proxy server known as a transcoder. From the vertical point of view, adaptation can be performed by the application reconfiguring itself internally to maintain a useful service. Adaptation could be performed at three different levels as described below.

6.1.1 Receiver Based Adaptation

One solution is to have the receiver adapt according to its current network connection. For example, if the mobile node moves to a network of a lower bandwidth, then the application itself will intercept, process and filter all the packets according to the user profiles. An example is shown in Figure 6-1. The mobile node has three different network connections. During a vertical handover, from a WLAN to a GPRS network the receiver application would remove the video data stream and compress the audio stream. Unfortunately this approach has one disadvantage. Transferring data, in this case the video stream or a high bit rate audio stream, in a format that is beyond the capability of the receiving end (the host that moved to the GPRS network) is a waste of bandwidth as well as distort/impair the perceived audio/video QoS.

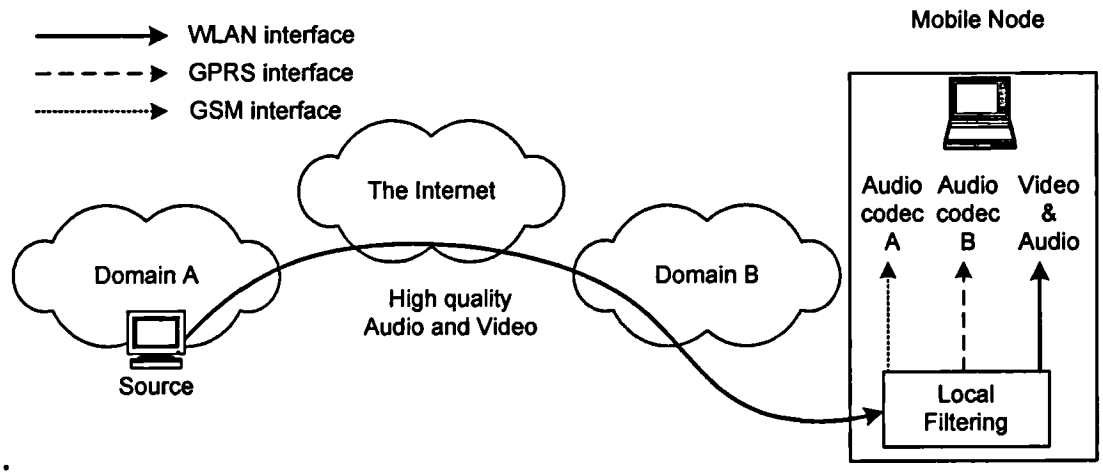


Figure 6-1: The receiver processes and filters all the packets according to the terminals network connections and demands

6.1.2 Proxy Based Adaptation

Alternatively, a proxy–based solution could be adopted using transcoders [3]. The transcoder has an updated profile of its registered hosts and provides filtering decisions based on the mobile nodes network connection. A transcoder could also inter-work with a SIP proxy or a home agent server to provide the user with smart routing and proxy filtering decisions as shown in Figure 6-2. Once a mobile node hands over to a different type of network, the routes are updated dynamically by mobile IP or SIP; packets are intercepted, processed and passed to the transcoder to be filtered. One disadvantage of using a proxy–based approach is the possibility of overloading the proxy server, resulting in higher delay, high jitter or even packet loss. Furthermore, this approach is also wasteful of network resources, as all data must be transmitted to the central server before filtering can take place, thus making route optimisation not possible.

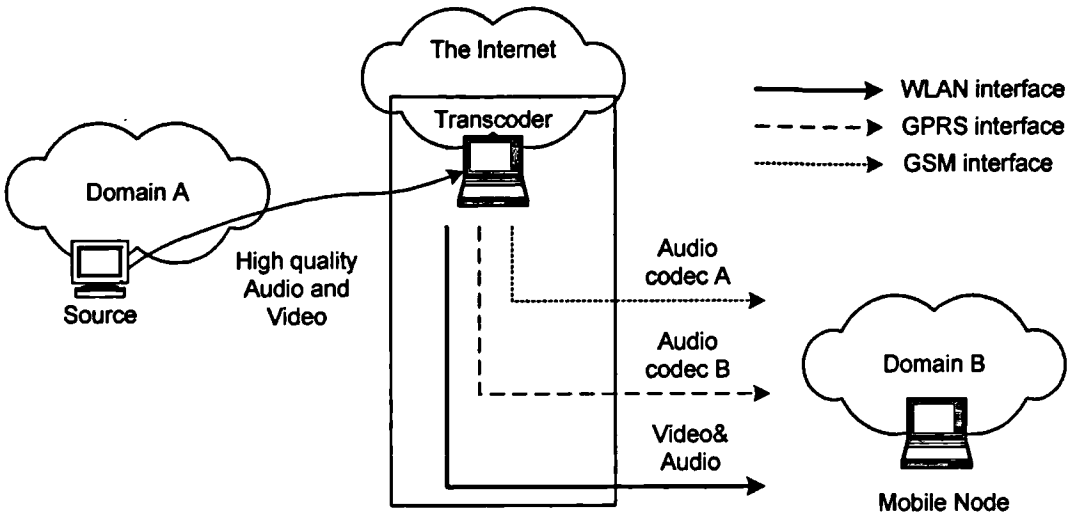


Figure 6-2: The proxy intercepts, processes and filters all the packets according to the receiver's network connection and demands.

6.1.3 Sender Based Adaptation

This approach provides an efficient solution, where the sender adapts its application in a dynamic manner preserving the illusion of seamlessness for the mobile node. The approach could be based on the session initiation protocol (SIP). SIP could be used with or without mobile IP to provide an end–to–end solution for application adaptation in a mobile environment. Sender based application–aware adaptation is a smart, bandwidth and processing–efficient way to adapt when the need arises. A sender based adaptation is shown in Figure 6-3.

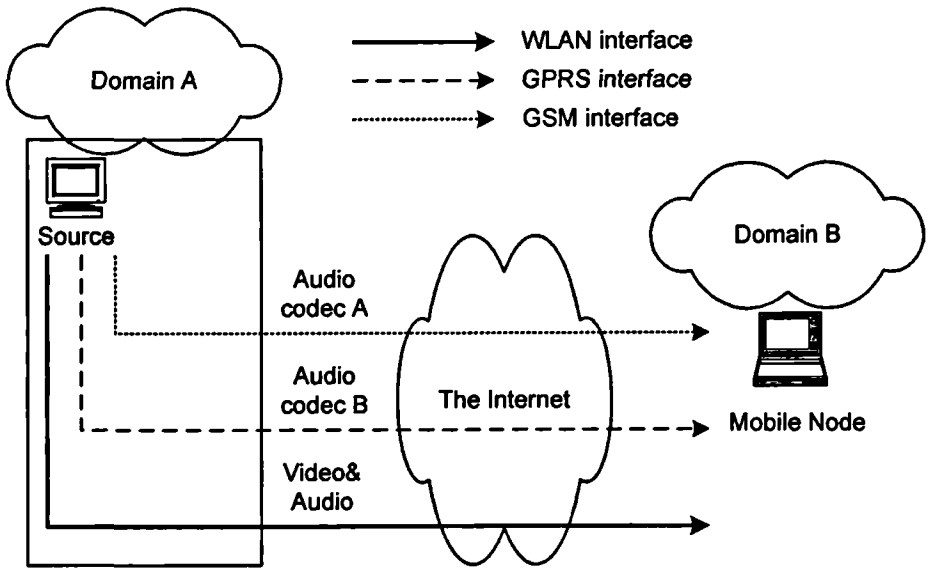


Figure 6-3: The source processes and filters all the packets according to the receiver's network connection and demands.

Mobile terminals in a heterogeneous environment can experience highly variable network performance during vertical handovers. Providing continuous connectivity in such an environment is a challenging task. This chapter describes an approach to media adaptation based on the sender adaptation model described above and implemented using the java media framework API. One of the central problems with current applications run on mobile terminals is the lack of adaptability during variability in network performance. For instance, a mobile node can experience rapid and large-scale changes in bandwidth availability after a vertical handover. Applications which execute in such environments have to be able to adapt to these changes in a dynamic manner in order to preserve the illusion of seamlessness for the end-user as far as possible.

In order to avoid proprietary solutions, and the re-invention of the 'adaptive wheel', a solution is proposed and developed [4] and [5] , which supports media adaptation by using existing dedicated APIs based on the Java Media Framework (JMF) [6]. The main goal is to optimise the performance according to the user's network connection, terminal capabilities and preferences in response to a vertical handover. The capability to adapt the behaviour of applications is also of further benefit for the network itself, since the resource usage can be dynamically optimised to a degree not possible with static configurations.

6.2 The Java Media Framework

In this section, we discuss and give an overview of the Java Media Framework (JMF). Java is a powerful computer language first developed by Sun Microsystems. Applications written in Java language are platform independent, and therefore can be run on any machine that has a Java Virtual Machine (JVM) installed. The most recent addition to the Java family is the Micro Edition (J2ME) which is designed for devices with limited memory, display and processing power capabilities (e.g. mobile phones and personal digital assistants). However, J2ME is not considered here since it does not support the real time protocol which this research is based on.

JMF is a set of multimedia Java Programming APIs that enables audio, video and other time-based media to be added to Java applications and applets. It is an optional package which can capture, playback, stream and transcode multiple media formats extending the multimedia capabilities of the java platform providing developers with a powerful

toolkit to develop scalable, cross-platform technology. Furthermore, the JMF API provides the basic Real-time Transport Protocol (RTP) functionality for the transport of real-time data, including audio and video. It can be used for media-on-demand as well as interactive services such as Internet telephony.

JMF provides the following RTP features:

- Open an RTP Session: enables end-to-end services for playback and transmission of real-time data streaming over a network. It provides a single RTP session ID that can be supplied to multiple hosts (destinations) over the IP network.
- Receive and send RTP streams: send and receive audio and video streams transmitted using RTP packets over the UDP transport protocol.
- RTP payloads mapping: Presents and transmits supported RTP codec (formats) mapping information. The JMF API however, provides limited RTP codec support.
- RTP buffering: The buffer length and threshold values of both incoming and outgoing streams can be controlled.
- RTP Session Manager: The RTPManager is used to coordinate the RTP session properties. It keeps track of the session participants and streams that are being transmitted.
- Provide stream statistics by utilising RTCP reports: Access, send and receive stream statistics by monitoring the RTP session using the RTP control protocol (RTCP) reports.

These are some of the main RTP related features that the JMF API supports. However, there are also some shortcomings about this API worth mentioning. The API does not directly provide methods to enable codec change during an ongoing session. Once a codec has been chosen and the processor realised, there is no way to change the codec unless the processor is destroyed and another one is created using a different codec. This results in disturbing the users session as well as experiencing a large delay in setting up and configuring the next processor. Furthermore, we have observed that some JMF codecs generate a periodic packet loss. This is purely an implementation issue that hopefully will be corrected in a future JMF release.

6.3 Requirements of M2A

The key objective of the M2A application is to inter-work with the session initiation protocol to provide seamless codec change during an ongoing session in response to a vertical handover. A seamless handover, as defined in [7], is one in which there is no change in service capability, security, or quality. In practice, some service degradation is expected. An implementation of M2A must meet the following requirements:

- Operate on SIP messages. When an INVITE request is received the application must create an appropriate transmitter using the required audio and video codecs
- Respond to session re-negotiations for both upward and downward vertical handovers by seamlessly changing codecs, maintaining the RTP session parameters. (i.e. session ID and sequence number)
- Handle mobility i.e. change of IP address: When the correspondent host performs a vertical handover the mobile node might be required to maintain the session by re-routing the packets to the new IP address.

6.4 Design and Implementation

This section presents the design stages of the first attempt to implement the M2A application. This section focuses on the implementation of the media adaptation part of the application.

Figure 6-4 illustrates the design approach. A flow chart is used to represent the details of the design algorithm. This provides a complete, unambiguous and effective procedure to clearly describe the approach. The two main building blocks of the design are described below.

6.4.1 High Level Description

This section provides the reader with a high level description of the flowchart without going into details. The processor module is responsible for the configuration, processing and controlling of the media data. The processor supports a programmatic interface that

enables control over the media data processing and access to output data streams. Media data processing includes media demultiplexing, data transcoding and media multiplexing. A processor has several states during its lifetime. Unrealised, Configuring, Configured, Realising, Realised, Prefetching, Prefetched, and Started. Once the Processor reaches the Realised state, reprogramming of the Processor is not possible. This is because reprogramming requires reconstruction of its internal functions, a process that is not supported by the JMF API. The constructTrack object is provided by the Processor module to query, control and manipulate the data of individual media tracks. The multiplexing performed by a Processor is controlled by specifying the output codec required using the 'setFormat' method. Finally when the processor is realised the output data streams are obtained through its DataSource. This function provides a gateway for the output data to be read. The main part of the transmitter module is the RTPManager. This is responsible for creating, maintaining and closing an RTP session. The addSession method controls the initialisation of the session using the IP address and port number of the end point of the session. Finally the sendStream is the object representing a sending stream within an RTP session. The 'processor.start' function is used to begin transmission over the network using the new configured RTP stream. During a codec change event, a new processor is created using a clone of the data source and is configured using the new required codec. Once the construct track and stream functions are completed for the new processor, a new data output source is created and directly passed into the sendStream object of the transmitter module. As a result, the output RTP stream is switched to the new codec format. The RTPManager and AddSession objects are not modified during this process. Since the RTPManager is responsible for maintenance of the current stream, it should not be modified in any way in order to maintain the RTP stream parameters (such as Sequence number and ID numbers) during the codec change. Furthermore, the AddSession object is responsible for the IP address and port number parameters of the RTP stream. This is only modified in response to an IP address change request.

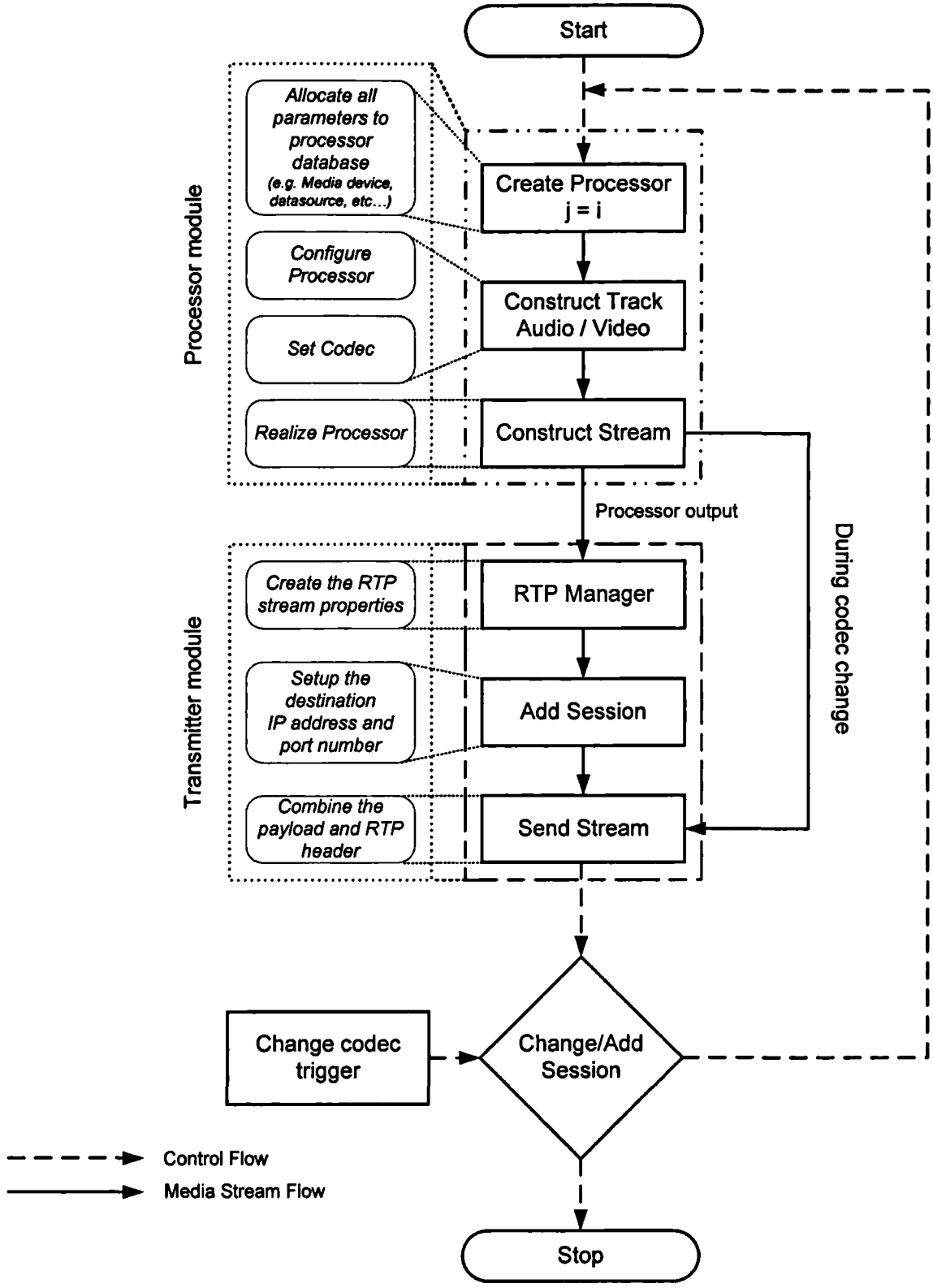


Figure 6-4: Real time media adaptation using JMF – design I

6.4.2 Low Level Description

This section describes in more detail the operations specified in the flowchart. The pseudo-code associated with each step is given in Appendix A section A.4

1. First create a Processor for the given capture device
2. Configure the Processor with the specified codec
3. Set the output content descriptor (ContentDescriptor.RAW) on the Processor
4. Find the matching track format, and set the track.
5. Realise the Processor with the specified track format
6. Obtain the output DataSource from the output of the Processor and use it for the input of the CreateSendStream to create the RTP transmitting data source that will be used by the RTPManager object.
7. Create RTPManager for generating the RTP packets
8. Use the initialise method to set the local IP address and port
9. Use the addTarget method to set the destination IP address and port
10. Use SendStream to send the RTP packets to the destination IP address using the specified codec and specific output data source for that processor
11. Codec change follows steps 1-6 and 10, bypassing steps 7-9.

6.4.3 Performance Evaluation

The performance of this design is evaluated by measuring RTP packet properties such as packet inter-arrival times, sequence numbers and packet loss during codec change. Figure 6-5 depicts the experimental environment.

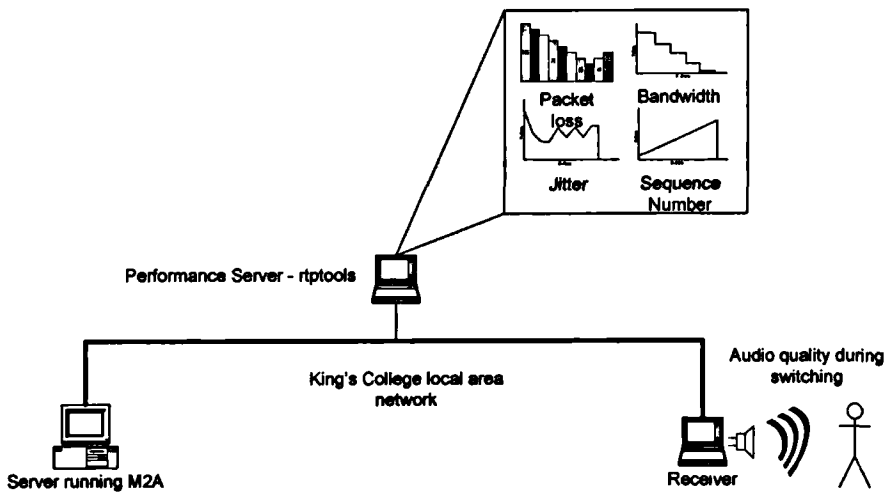


Figure 6-5: JMF media adaptation experimental environment

It consists of a server running the M2A application, a correspondent host acting as the receiver and a further performance server used to monitor the traffic. In our experiments, the switching occurs between three codecs, PCMU, DVI and GSM with the codec change trigger interval set to 2.5 seconds. The IP packet of the PCMU codec has a length of 534 bytes consisting of 20 bytes of IP header, 14 bytes of IP options, 8 bytes of UDP header and 492 bytes of RTP message. The 492 bytes of RTP message consist of 480 bytes of data following the 12 bytes of RTP header. This is because the PCMU codec, produces 160 bytes of compressed frame at a time. This codec sends three frames per packet. This information is summarised in Table 6-1 along with the information of the other codecs too.

Encoding	Sample Rates	Bits per Sample	RTP packet size	Frames per packet	Transmission Bit Rate	Set Duration
PCMU/RTP	8000	8	492	3	~64 Kbps	2.5
DVI/ RTP	8000	8	256	3	~32 Kbps	2.5
GSM/ RTP	8000	8	111	3	~13.3 Kbps	2.5

Table 6-1: RTP configuration of design attempt I

The inter-arrival time was measured during the two codec change events. The first codec change is triggered automatically after 2.5 seconds of running the first codec and the second trigger is scheduled 2.5 seconds after the first one starts operating. Figure 6-6 shows the packet inter-arrival times of one of the results. As shown in the figure, the inter-arrival times, 169msec and 500msec are the two peaks in the overall switching delays. These are the points during which the application switches between the codecs. This delay is also shown in Figure 6-7. As shown in the figure there is a significant time between the instance when the codec change is triggered and the time the first packet of the new codec arrives. This delay is primarily caused by the codec change mechanism described above. The first switch (PCMU to DVI) takes 1.1 seconds to complete while the (DVI to GSM) takes 1.5 seconds which both are too long. The graph also shows that the sequence number of the RTP stream is not incremented appropriately. Each time the codec is changed the sequence number is reset to a random value. As a result, packet loss cannot be measured. This was an unexpected 'feature', that was later identified as being caused by connecting the output of the processor directly to the input of the RTPManager.

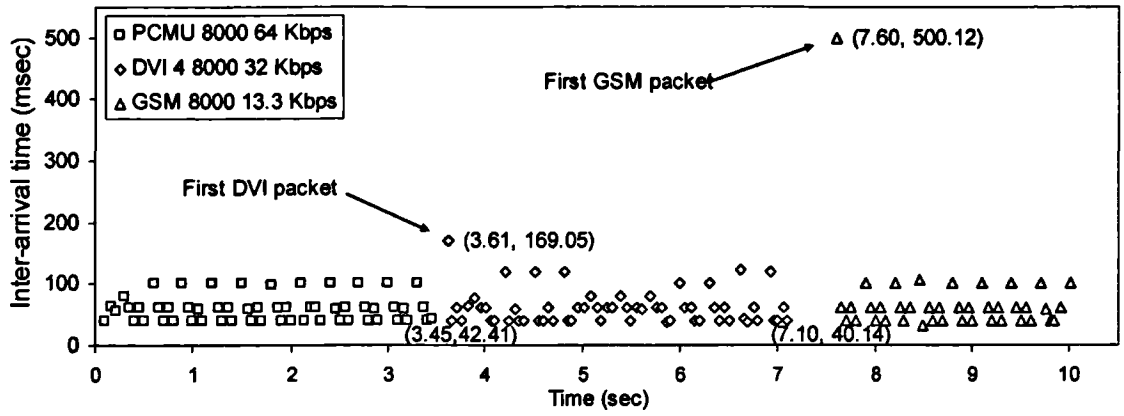


Figure 6-6: Inter-arrival times during codec change

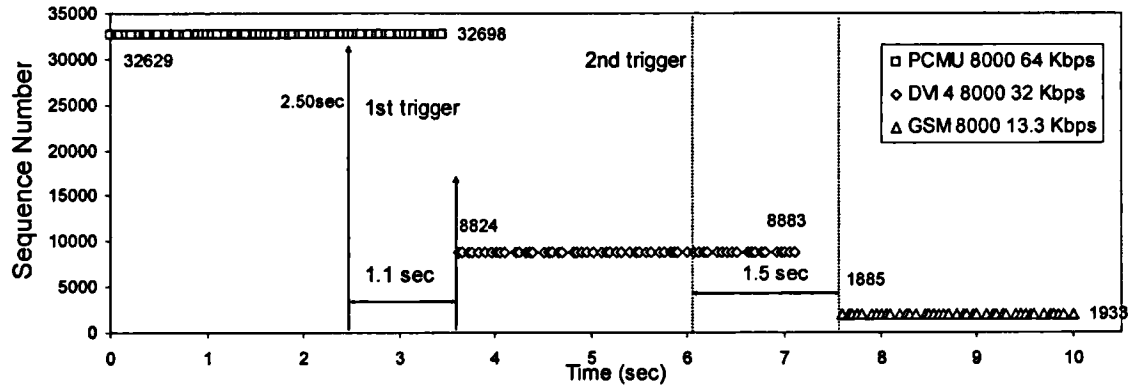


Figure 6-7: RTP sequence numbers during codec change

6.4.4 Summary

The key property of an adaptive application is the speed at which it responds to a codec change request. In other words, there should be minimum time lag between the request for codec change and the application response. Results confirm that this application does not respond fast enough to requests. In both cases, there is a delay of over a second before the codecs are changed adding unnecessary delays to the overall adaptive procedure. The application also fails to provide seamless codec change by resetting the RTP stream sequence number randomly. As a result some applications will reject new packets that have a lower sequence number than the ones received before, causing severe interruption to the user's session. Unfortunately, these shortcomings of the application cannot be improved without making major design changes to the source code. As mentioned before, JMF does not support real time codec change and this makes it a difficult task to implement such a function. The main reason for these problems lies in the architecture of the design, the way in which the processor module

interacts with the transmitter module. The second attempt presents a rather different approach. Instead of connecting the processor and transmitter modules directly together, a Switchable data source module is used to link them up.

6.2 Design and Implementation – second attempt

This section presents the second attempt to improve the design of the M2A application. The main objective of the new design is to accelerate the response time of the application, to reduce delay variations and also maintain the sequence number of the RTP stream during codec change.

6.4.5 High Level Description

The application is re-designed as illustrated in the flowchart in Figure 6-8. During initialisation a processor is created and configured for a given input media, such as a capture device or a given MP3 file or any other source supported by JMF. The processor uses the ConstructTrack method to query the data of the selected media, matching the tracks to the appropriate format in order to perform the necessary transcoding. Once complete, the processor output is obtained and converted to a PushBufferDataSource. This is a data source type that manages data in the form of push streams. A Switch Data Source module is then created to accept the output of the PushBufferDataSource from the Processor thus generating a continuous stream of data. In the final stages the RTPManager object is created for constructing, maintaining and closing the RTP session. This includes the configuration of the media stream including the destination port and IP addresses. Finally the SendStream method is used to create a sending stream within the RTP session by using the output datasource from the Switch Data Source module. This call is only made once. The Switch Data Source defines a module for processing a buffer object and controlling the switching of processors. A switch data source object is a media-data container that carries media data from the output of processor to the RTP manager. A buffer object maintains information such as the time stamp, length, and format of the data it carries, as well as any header information that might be required to process the media data. During a codec change event a new processor database is created. This database contains the name of the new processor, the media locator, the tracks control information, the codec format and a new PushBufferDataSource. The new processor is configured and the tracks constructed based on the new codec format. Finally the new processor is set to output its stream into

the Switch Data Source module. This is immediately reflected into the SendStream function causing a smooth codec change.

6.4.6 Low Level Description

The actual algorithm for creating M2A is summarised below. The pseudo-code associated with each step is given in Appendix A section A.5

1. Create a Processor for the given capture device
2. Configure the Processor with the specified codec
3. Set the output content descriptor (ContentDescriptor.RAW) on the Processor
4. Find the matching track format, and set the track.
5. Realise the Processor with the specified track format
6. Obtain the output DataSource from the Processor, convert it to a PushBufferDataSource and use it for the input of the SwitchDataSource
7. Create an RTPManager
8. Use initialise to set the local IP address and port
9. Use addTarget to set the destination IP address and port
10. Use SendStream to send the RTP packets to the destination IP address using the output from the SwitchableDataSource.
11. Codec change follows steps 1-6.

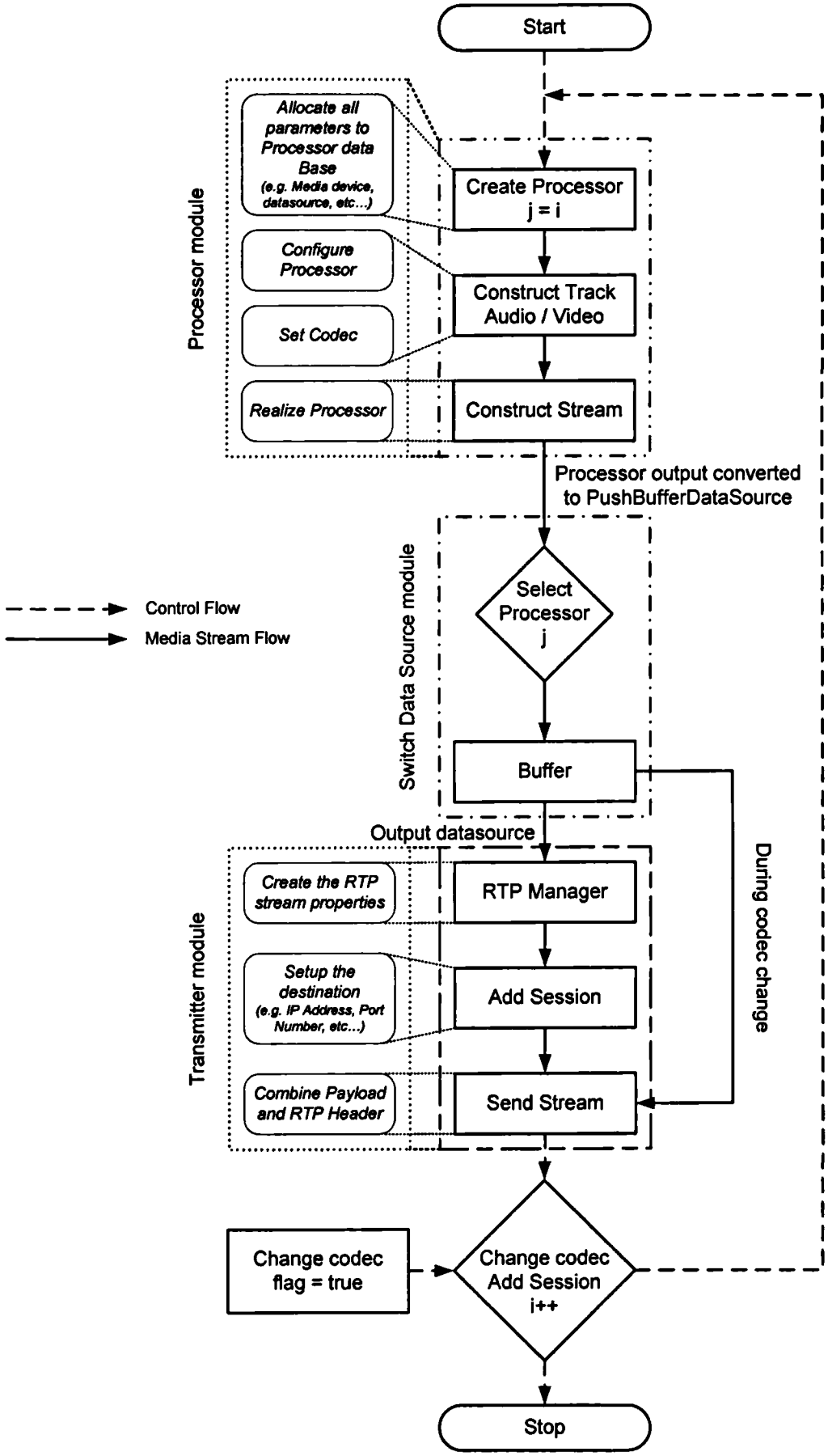


Figure 6-8: Real time media adaptation using JMF – design II

6.4.7 Performance Evaluation

The performance of the second design attempt is measured using the same experimental environment described in the Section 6.4.3. This evaluation is focused on two aspects of the new design. How fast the application responds to media adaptation triggers in terms of delay and jitter and whether the sequence number is maintained during the lifetime of the experiment. Figure 6-9 shows the packet inter-arrival times in the new design. The inter-arrival times are quite similar to those for the older design. However there is a significant improvement during codec change. The two peaks for inter-arrival times are 183msec and 220 msec for each codec change compared to 169msec and 500msec using the first design attempt. During the second codec change from DVI to GSM the inter-arrival time has been reduced considerably. Measuring the inter-arrival times only provides the time difference between the last packet received from the old codec and the first packet received using the new codec. Figure 6-10 however, shows the full response of the application to a codec trigger event. Codec trigger events are activated on fixed time intervals of 2.5 seconds. The first codec change is triggered at 2.5sec resulting in the first packet arriving at 2.63 sec and the seconds trigger at 5.13sec with the first GSM packet arriving at 5.45sec.

This clearly shows that the second design attempt responds promptly to codec event triggers with an average response time of less than half a second. The graph also shows that the RTP sequence number is kept intact during the media adaptation. Finally, Figure 6-11 shows that no packets are lost during the switching process. More experimental results were performed confirming the above findings.

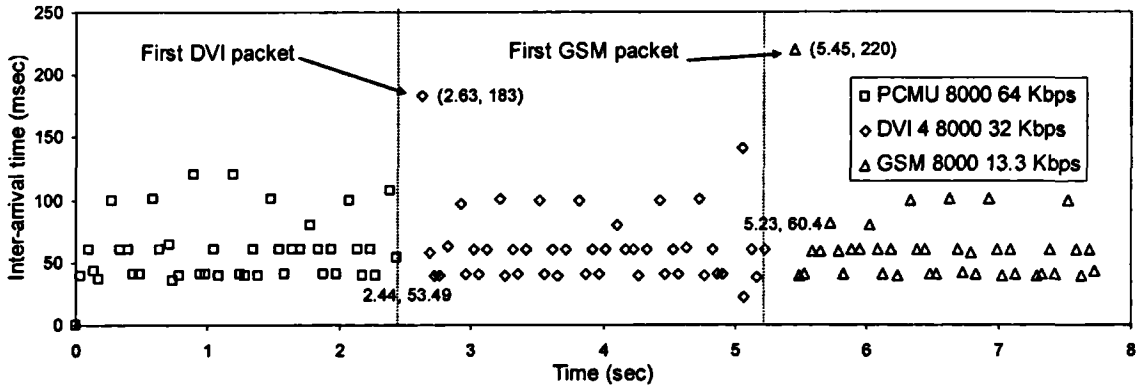


Figure 6-9: Inter-arrival times during codec change

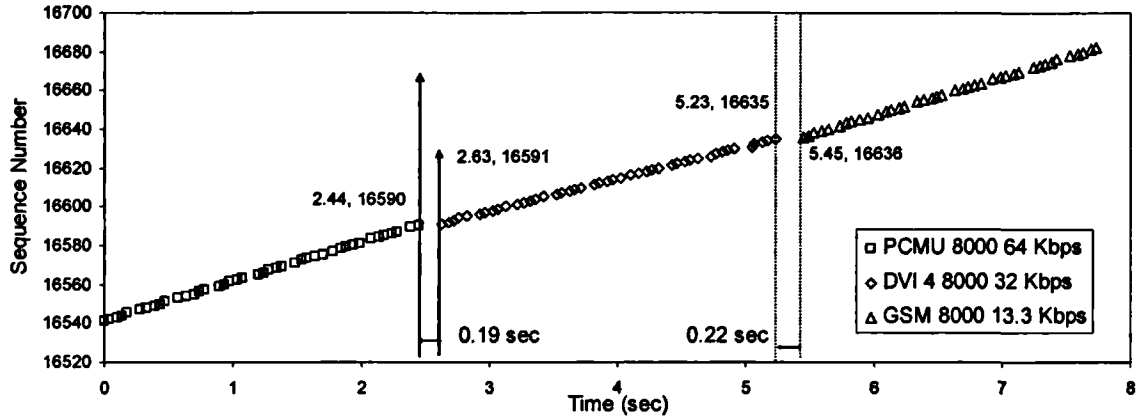


Figure 6-10: RTP sequence numbers during codec change

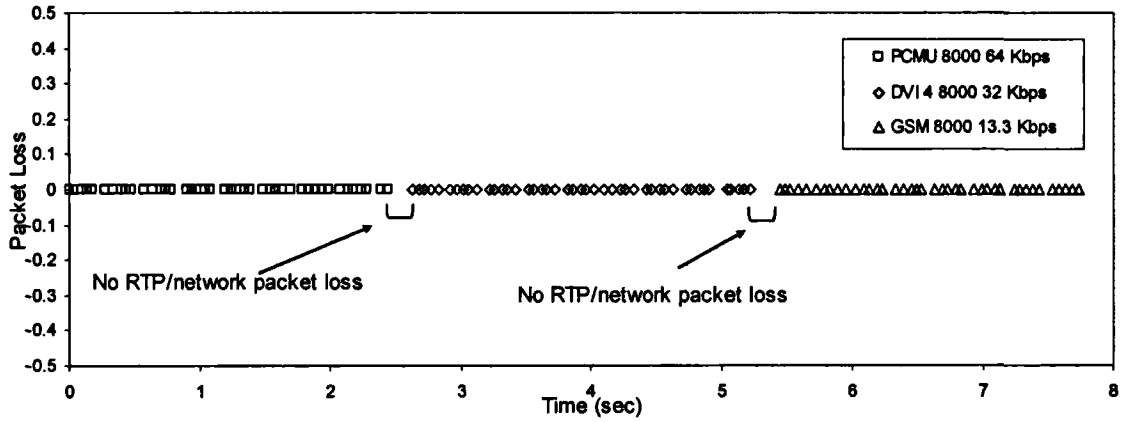


Figure 6-11: Packet loss during codec change

6.4.8 Summary

The second design attempt to media adaptation confirms that it is possible to achieve seamless real-time codec change using the Java Media Framework. This was made possible by using the switch data source object between the output of the processor and the input to the RTPManager. The results showed the application responds promptly to trigger events and adapts the session without any packet loss.

So far we have presented the implementation of a Java based application, designed to perform seamless real-time RTP codec change. This was a major part in the main objective of the M2A application. The next section examines the inter-working between the session initiation protocol and the M2A application in order to completely fulfil the key objective of seamlessly changing codecs.

6.3 Inter-working of M2A and SIP

This section focuses on the tight inter-working between the SIP user agent and the M2A application. In order to support this inter-working and minimise the signalling between them, the M2A application was built as part of the SUA. The M2A application has three states: Initialisation state, idle state and active state. The transitions between the states are illustrated in Figure 6-12 and explained in more detail below.

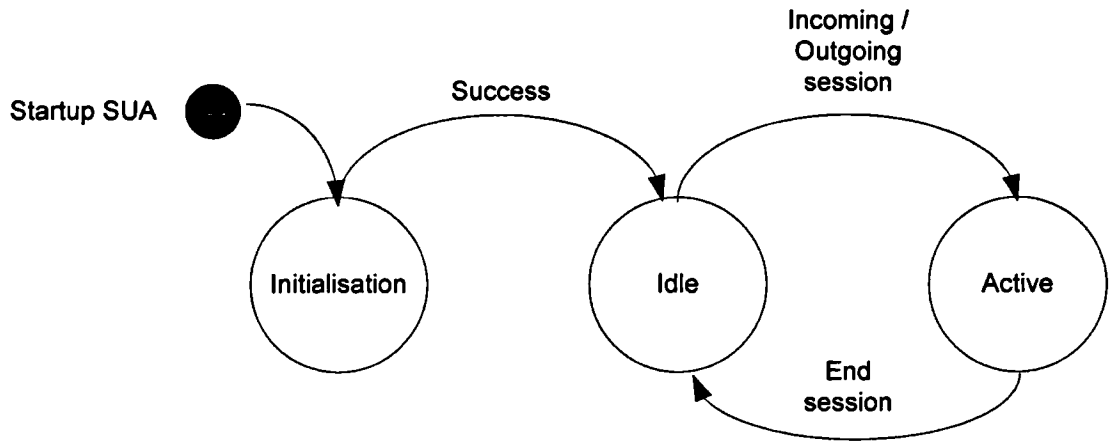


Figure 6-12: State diagram for the M2A application

6.3.1 Initialisation State

During the initialisation phase the SUA initiates a call to the M2A application to enquire about codec and available application objects. In response, the M2A will respond by sending the object types it supports (i.e. text, whiteboard, audio or video) as well as the codec payload numbers supported. The M2A currently supports only audio with the full range of JMF codecs. It is expected that future extensions will include support for video as well as text and whiteboard objects.

The terminal hardware capabilities should also be considered. The application might support a variety of codecs but the terminal might only be able to utilise some of them. Consider a hypothetical scenario is which a user has two devices each having the M2A application installed. One device is his PDA and the other his laptop computer. The terminals vary in terms of connectivity, screen size, processor power, and sound support. In this scenario the M2A application would take this into account configuring the SUA with the appropriate codecs. The interactions between the SUA and M2A are shown in Figure 6-13, steps 1, 2 and 4. Step 3 might be considered as a future extension to the M2A application.

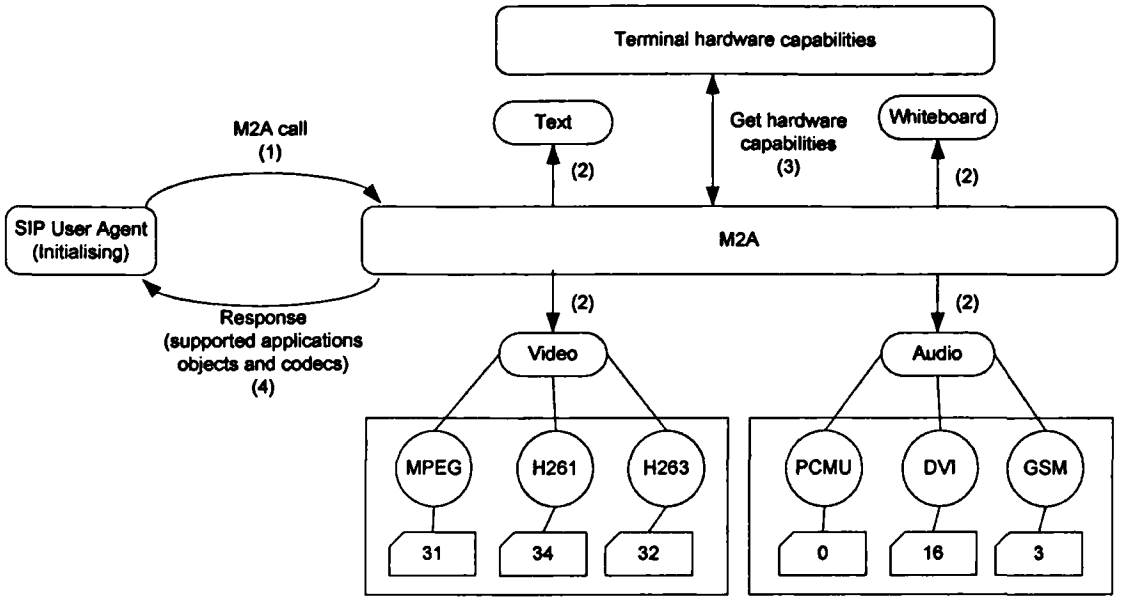


Figure 6-13: SUA to M2A call during initialisation phase

6.3.2 Idle State

During the idle state the M2A application is ready to respond to session requests. Figure 6-14 shows the interactions between the SUA and the M2A application during such a request. When an INVITE is received, the mobile node processes it and replies with a 200 OK message. A sample response to the invitation message is also shown in the figure. The SUA is then responsible to pass this message to the M2A application which in turn will extract the information needed and load the appropriate application objects. The required message fields are indicated in the diagram. The Call-ID is taken directly from the original request and is used to keep track of the mappings between the various application objects and the session to which they belong. As a result each object can be located and treated separately in response to a session re-negotiation request. Other significant fields required are the name of the caller, the IP address, and finally the payload and port numbers. A successful M2A configuration is one in which the SIP message is successfully received, understood, and accepted. This is indicated by an OK message send back to the SUA. If for any reason the M2A configuration is not successful the SUA will be notified and in response will terminate the call by sending a BYE message to the correspondent host.

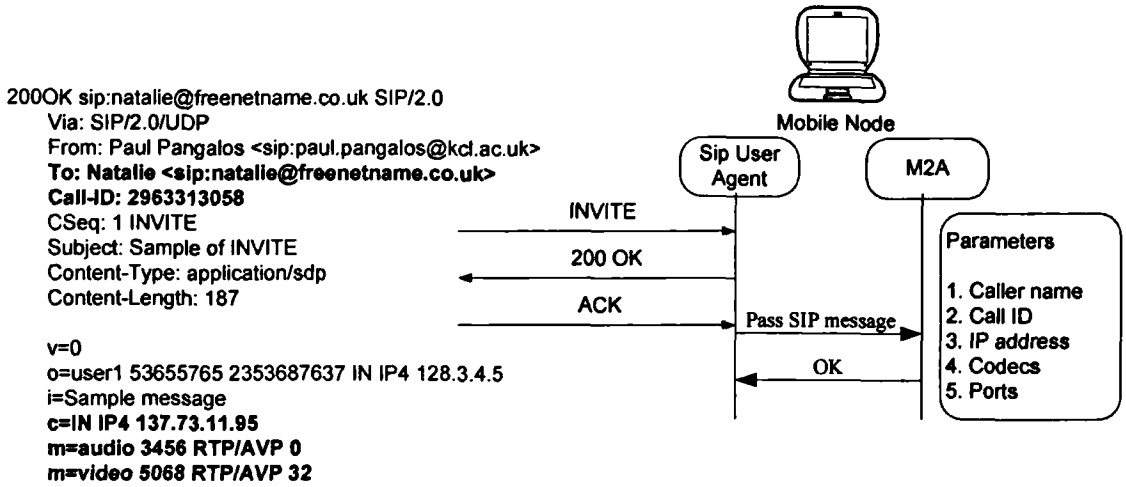


Figure 6-14: SUA-M2A interaction during an incoming call request

6.4.9 Active State

The active state indicates that the user is part of one or more multimedia sessions. During this state the SUA listens for new incoming messages. Figure 6-15 shows the mobile node receiving a new INVITE request. The SUA first checks the header fields. Since the received CALL-ID already exists, the Cseq contains a higher sequence number and the To and From fields match, the request is identified as a retransmission for a session re-negotiation. The INVITE message also carries another header field called 'handover type'. This informs the M2A application about the handover type that is being performed by the correspondent host. In response the mobile node's M2A application will set the operating mode, i.e. the order in which the media adaptation and mobility will take place. The 'handover_type' field provides the necessary information to ensure that no link congestion occurs at the correspondent host side during the session re-negotiation procedure. Once the operating mode is set the CALL-ID field is used to identify the application objects associated with that particular session. Media servers are the classic example of having hundreds of application objects each serving a different user. In that case mapping the CALL-ID to each application object(s) is essential.

As soon as the relevant application objects are identified, the upward vertical handover mode will cause the M2A to adapt the session first and then change the target IP address whereas for downward vertical handovers the target IP will change first followed by the session change. The M2A then relays the success to the SUA which in turn sends a 200 OK message to the correspondent host. This is illustrated in Figure 6-15.

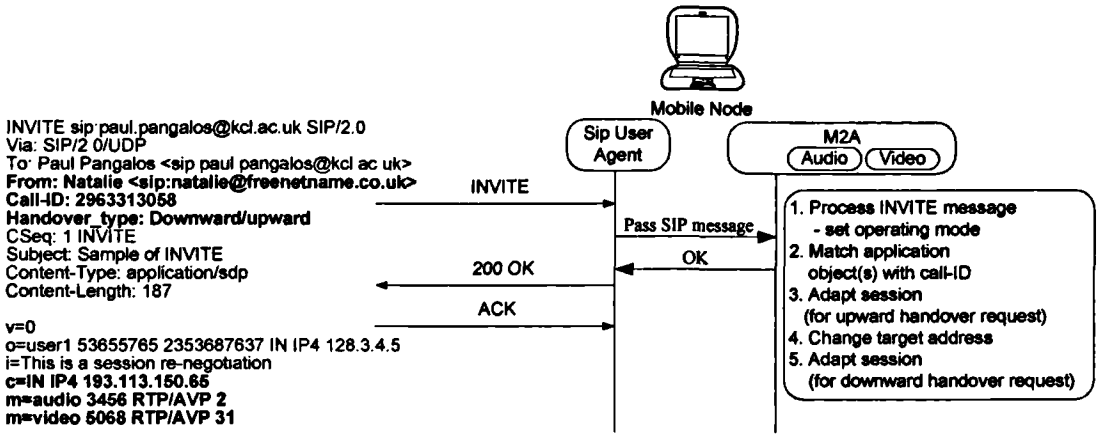


Figure 6-15: SUA –M2A interactions during session renegotiation

6.4 Summary

The need for adaptation in a heterogeneous environment is now widely accepted. This chapter put forth the view that application-aware adaptation is an effective approach to addressing this need. The essence of this approach is a collaborative relationship between the application and the session initiation protocol. The M2A application was developed to support real-time media adaptation using JMF. The successful implementation resulted from a two-stage process. The first implementation design followed a rather plain design procedure to enable codec change during an ongoing session. However, the experimental evaluation identified two problems with the design. The sequence number of the RTP stream was reset during the adaptation process and the application did not respond fast enough to the codec change trigger event. This lead to a more advance design that resolved the above issues. Furthermore, a description of the inter-working between the M2A and SIP was also presented, identifying and describing their interactions.

6.5 References

[1] Brian D. Noble, M. Satyanarayanan, "Agile Application-Aware Adaptation for Mobility", In Proceedings of the sixteenth ACM symposium on Operating systems principles, pages 276-287. ACM Press, 1997.

[2] Benjamin Atkin, Kenneth P. Birman, "Network-aware application adaptation for mobile hosts", ICDCS 2003 Doctoral Symposium, Providence, Rhode Island, May 2003.

[3] Caroline Lebre, Richard Titmuss, Pete Smyth, "Handover between heterogeneous IP networks for mobile multimedia services", Acts Mobile Summit, June 1999

[4] Paul Pangalos, Wing Shun Wong, Hamid Aghvami "Real-time Transport Protocol in Java Media Framework, application adaptation performance over heterogeneous networks," WWRF8, Beijing, China, 7-8 April 2003

[5] Paul Pangalos, Wing Shun Wong, Hamid Aghvami, "Java Based Adaptable Multimedia Application (M2A)," SOFTCOM 2003, Ancona, Venice (Italy), Split, Dubrovnik (Croatia), October 2003

[6] http://java.sun.com/products/java-media/jmf/

[7] J. Manner, M. Kojo, "Mobility Related Terminology", Internet Engineering Task Force, Internet draft, draft-ietf-seamoby-mobility-terminology-06.txt, April 2003.

"The fear of the Lord teaches a man wisdom
and humility comes before honour"
(Proverbs 4:5)

Chapter 7

Conclusions and Future Work

We conclude this dissertation with a summary of the contributions and suggested future work.

7.1 Summary

This thesis analysed problems introduced by vertical handovers in heterogeneous environments and designed implemented and evaluated solutions to them that resulted in significant improvements in vertical handovers. Because of the network and device heterogeneity, and the enormous variation in their characteristics and protocols there is no single solution to the problems. We recognised this and developed a suite of solutions and techniques to address a range of scenarios, that together comprehensively solve these problems. These solutions aim to provide advanced techniques of inter-working between various protocols assisted by higher layers as well as enhancements to applications at the sender and receiver nodes. We believe these solutions, will accelerate the widespread development of vertical handovers and make them an integral part of the Internet infrastructure of the future. This thesis identifies two fundamental challenges to mobility

- Heterogeneity: The Internet today consists of a large diversity of heterogeneous networks. The networks have very different and varying capabilities such as bandwidth, interface speed, edge-to-edge latency and connection types. The introduction of laptops and palmtops have also increased and so have the operating systems, providing a large diversity in the end system capabilities.

This diversity combined with the network heterogeneity results in an unacceptable level of incompatibility between both the networks and terminals. As a result interpolation of protocols, services, and technologies in this kind of environment remains a challenging task.

- Vertical handovers: It is fairly common today for users to have access to multiple Internet Service Providers (ISPs) and alternate between them. Some of the ISPs offer free low speed dialup connections (i.e. GSM, GPRS) while others offer high bandwidth, always on online broadband connections such as DSL and cable. However, in these cases users tend to only use one connection at a time. With the introduction of overlay networks and constant reduction of prices, this is expected to change. Users may wish to handover or use a combination of several networks, each of which is optimised for some particular service and also depending on factors such as time of day and current traffic load, in order to optimise cost versus quality. Vertical handovers is the link that would bridge these heterogeneous access technologies and enable users to switch efficiently between them.

This dissertation analysed the problems introduced by the above challenges, and solved them using a collaborative relationship between applications, the operating system, and mobility protocols. This was done through a practical realisation in a test-bed consisting of different wireless mediums and associated networks. The test-bed architecture is based on an open coupling scenario and was best introduced in terms of the three essential requirements that it was designed to follow.

- Reuse and extend existing globally accepted IETF protocols whenever possible, by extending and enhancing existing features to achieve new functionalities. Reusing existing protocols also leads to a direct cost reduction in innovation processes by limiting the 'reinvention of the wheel' phenomenon.
- Reuse existing network entities within the network creating a practically deployable architecture.
- Handover support between heterogeneous networks, thus providing maximum flexibility in network configuration for operators. This flexibility allows operators to offer innovative user services anytime, anywhere while users are on the move.

The complete architecture models as well as the building blocks were fully described in Chapter 1. The architecture shows an approach behind a vision to enable the efficient use of multiple heterogeneous networks focusing on vertical handovers. It is based on the end-to-end intelligence model with all the intelligence placed in the end terminals and the independent ISP core network. The vision of this architecture is that it inter-operates across various heterogeneous wireless and fixed networks providing the user with the same benefits as if the inter-working was managed within one network. The primary technical objectives to enable the above vision are explained in the following sections:

7.1.1 Handling of Unplanned Vertical Handovers

Frequent disconnections are an important characteristic of heterogeneous environments and should not be considered as a failure but rather as a normal event. The primary characteristic of an unplanned handover is that it happens suddenly and is not predicted, causing severe disruption to the IP layer. As a result, no signalling can take place prior to the handover; therefore, the user experiences an interruption to its session. A disconnection can be voluntary when a user unplugs a cable or a network interface or involuntary due to physical wireless communication breakdowns such as an uncovered area or when the user has moved out of the reach of a base station. Chapter 3 proposes two techniques in which MIP and SIP could inter-work to handle unplanned vertical handovers for real time applications. In the first technique mobile IP communicates unidirectionaly with SIP providing status information. This is also referred to as loose coupling. Based on this approach, SIP re-negotiates the session parameters once a new network connection is re-established. However, this method revealed a serious weakness, that of congestion delay. Once re-connected with the new overlay, the terminal receives data that is beyond its network bandwidth capability causing link congestion. The rest of chapter 3 is concerned with ways to improve the handover technique so that the link congestion delay is reduced. The first improvement made use of the UDP protocol for part of the signalling whereas the second improvement used the home agent to assist with the handover. These improvements dealt with the problem effectively, but did not eliminate it. Therefore, a second technique was designed in which the two protocols inter-worked in a bidirectional manner eliminating the congestion problem completely. This approach is also referred to as tight coupling. Table 7-1 gives a summary of the contributions stated in this chapter.

Chapter 3: high level summary of contributions
1. Designing of signalling mechanism and interactions between the protocols for both loose and tight coupling techniques (i.e. three optionsfor the loose coupling approach and one for the tight coupling approach)
2. Defining a new SIP message called SIP INFO used for congestion notification avoidance.
3. Implementation of the signalling mechanism by extending the basic sip user agent, home agent and mobile IP agent modules.
4. Setting up an experimental testbed for the performance and evaluation of the signalling mechanisms.
5. Performance evaluation and comparison between the different techniques.

Table 7-1: Contributions summary of chapter 3

7.1.2 Handling of Planned Vertical Handovers

From a high level perspective planned vertical handovers should provide the following. Firstly, support optimum triggering and network access selection. Discovering the right time to perform a handover is a key issue. Only though prompt reactions can a handover be initiated and performed efficiently. A handover can be triggered by the network (i.e. load balancing scenario) or it could also be user initiated (i.e. user requesting a handover to a specific network). In order to support vertical handovers there is a need to exchange information in the form of signalling. This signalling may represent an important overhead in terms of bandwidth and processing requirements, having a significant impact on the performance of handovers. Therefore, it is important to minimise the signalling overhead as much as possible in order to improve and optimise the handover performance.

Chapter 4 presented the design, implementation and performance evaluation of a technique designed to handle planned vertical handovers. This technique improves vertical handover performance significantly by providing fast and seamless handovers across heterogeneous networks. It achieves this by enabling bidirectional communication between the link layer, the network layer the application layer and the operating system of the terminal. The session initiation protocol was enhanced to communicate with three entities, the mobile IP agent, the M2A application and the operating system. The signalling mechanism is described showing how these entities interact and relate to each other. These promising solutions were then implemented in a testbed providing performance measurements for both upward and downward vertical

handovers. The gathering of packet-level traces helped to understand more clearly how the signalling mechanism impacts the user's session and how the tight coupling between agents influence the handover times. The chapter concludes by a discussion on the design implementation issues for the proposed signalling methods. Table 7-2 summarises the contributions of this chapter.

Chapter 4: high level summary of contributions
1. Designing of the signalling mechanism and interactions between the Mobile IP agent and the SIP User Agent for both upward and downward vertical handovers
2. Implementation of these techniques by extending the sip user agent further including interactions with the M2A application, the operating system and the link layers of the terminal.
3. Setting up an experimental testbed for the performance evaluation of upward and downward vertical handovers.
4. Design, implementation and performance evaluation of a mechanism for handling short TCP connections during planned vertical handovers
5. Discussion and solutions on issues relating to the tight coupling mechanisms presented.

Table 7-2: Contributions summary of chapter 4

7.1.3 Support for Session Mobility

In a typical future environment, users will be able to receive multimedia services via multiple network devices (i.e. mobile phone, desktop phone, public IP terminals, and desktop computer). As people have access to these devices they may desire to switch from one device to another in the middle of a session. There are numerous reason why such a move might be necessary. The users battery might be running out, a better device is found or the user might wish to share a session with a number of other users by transferring the session to a digital projector. This can also be described as another type of vertical handover in which users move from one network to another by changing devices. This type of mobility is called session mobility and it refers to the user's ability to maintain an active session while switching between terminals. The operation of maintaining the session across the different networks and devices is called session mobility and the ability of switching between the terminals is called session handover. Chapter 5 develops a solution to session mobility by dealing with three fundamental challenges. The first one is terminal discovery and description: before any session can be handed over, the terminal matching the users criteria (in terms of hardware and

software capabilities) must be located and mapped to an addressable destination. The second one is handover signalling: once a terminal is located, a handover mechanism is required to move the session from one device to the next. The third one is mobility support: a mobility mechanism is required to maintain an active session while switching between terminals. The terminal description protocol (TDP) was designed for describing terminals in terms of hardware and software capabilities. The protocol provides short textual descriptions of the terminals interfaces, mobility protocols and application information that are required to decide whether a terminal is likely to be of interest to a user. Furthermore, Chapter 5 described the mechanism and signalling required for session handovers. This focused specifically on Mobile IP proposing a concept based on the denial of service as the solution to session mobility. Finally, a test implementation of the session handover mechanism was made, in order to verify its feasibility and to make preliminary measurements. Table 7-3 summarises the contributions of this chapter.

Chapter 5: high level summary of contributions
1. Design and implementation of the terminal description protocol (TDP) created as part of the solution to session mobility for describing terminals in terms of hardware and software capabilities.
2. Design and implementation of the session handover mechanism based on the denial of service concept.
3. Setting up an experimental testbed for the performance evaluation of session handovers.
4. Performance evaluation of the session handover mechanism.

Table 7-3: Contributions summary of chapter 5

7.1.4 Reliable and Transparent Handovers

Another important aspect of vertical handovers is the need to provide reliable and transparent handovers. As users move between network boundaries, the handover mechanism should enable a transition which is transparent to the application layer with the only visible change to the user being due to the limitations of the new network interface. In order to incorporate adaptiveness in an application, it is clearly essential for the system to detect and inform the application about the changes in the user's environment. Adaptation is the key to mobility. Only through alertness and prompt reactions can a mobile node adapt in response to a vertical handover. Media adaptation is the process of adjusting the streaming parameters of the application during a change

in the transmission characteristics that could be due to a vertical handover. From the horizontal point of view, adaptation can be performed in an end-to-end approach (in the end systems) or using a proxy server known as a transcoder. From the vertical point of view, adaptation can be performed by the application reconfiguring itself internally to maintain a useful service.

The need for adaptation in a heterogeneous environment is now widely accepted. Chapter 6 put forward the view that application-aware adaptation is an effective approach to addressing this need. The essence of this approach is a collaborative partnership between the application and the session initiation protocol. The M2A application was developed to support real-time media adaptation using JMF. The successful implementation resulted from a two stage process. The first implementation design followed a rather plain design procedure to enable codec change during an ongoing session. However, the evaluation identified two problems with the design. The sequence number of the RTP stream was reset during the adaptation process and the application did not respond fast enough to the codec change trigger event. This led to a more advance design that resolved the above issues. Furthermore, a description of the inter-working between the M2A and SIP was also presented, identifying and describing their interactions. Table 7-4 summarises the contributions of this chapter.

Chapter 6 high level summary of contributions
1. Design and implementation of an adaptable aware application using the Java Media Framework. JMF methods and techniques were applied in novel ways to achieve seamless codec change. This was a three stage process:
2. In the first stage, problems affecting seamless codec change were investigated using the first prototype design. The second phase identified a new way of programming the application to achieve the desired performance. In the final stage the second design of the application was implemented.
3. Performance evaluation of the M2A application in terms of packet loss, jitter and RTP sequence numbers during a series of codec changes.
4. Presenting the mobile environment changes to the M2A application using the Session Initiation Protocol. The complete tight inter-working signalling mechanism was given.
5. Design and implementation of the signalling mechanism enabling tight inter-working between the SIP user agent and the M2A application.

Table 7-4: Contributions summary of chapter 6

7.2 Availability

All the protocols and software implementations described in this dissertation are available on request. Furthermore, testbed set-ups of the following inter-working platforms are also available.

- Unplanned vertical handovers – tight coupling approach
- Planned vertical handovers
- Session mobility
- The Mobile Multimedia Application (M2A) – seamless codec change

7.3 Future Directions

There are several interesting directions for future work based on the work described in this dissertation. Some of these are direct extensions of this work, while some others are motivated by the more general problem of vertical handovers in heterogeneous environments. There are short and long-term tasks ahead of this research. In the short term, the work presented in Chapter 6 can be expanded to incorporate adaptation for other types of objects other than audio. Furthermore, terminal capability support can also be added to the M2A application as described in section 6.4.1. The research in Chapter 5 focused on session handovers between the user's home network and a foreign network. This work can also be expanded to investigate handover procedures between (i) foreign networks, (ii) foreign network to the user's home network and (iii) between terminals located within the users home network. There are a number of interesting research issues related to these scenarios. There is a need to look at the signalling between the hosts as well as the different security issues that arise (i.e. how does the tunnelling mechanism work when the user is performing session handovers within its own home network). Finally, the research performed in Chapters 3 and Chapter 4, can be expanded to explore a purely SIP based solution that uses the sip user agent to handle vertical handovers without requiring Mobile IP support. Some preliminary results of this research are presented in Appendix B section 1.1, 1.2 and 1.3.

The long term plans are more speculative. Currently, the following areas are under examination.

7.3.1 Automatic Detection of Mobility Protocols

Throughout this work, we have used Mobile IP as the solution to mobility by providing users with a fixed home address and tunnelling packets from the home agent to the mobile user. While Mobile IP is expected to be the future solution to mobility, it will not be the only one. Other solutions have also been proposed such as MSOCKS [1], TCPMIGRATE [2], ROCKS [3] and SLM [4]. Some of these solutions might well be implemented in future terminals and networks. Therefore, they should be taken into account when a user selects the mobility mechanism to use. Furthermore, Mobile IP might not always be an option for the user. This can be because some foreign networks might not have any security associations with the users home network, or even the users home network might not support Mobile IP. In that case an alternative solution will be required. Chapter 5 proposed the terminal description protocol for describing terminals in terms of software and hardware capabilities. This protocol could be extended further not only for use during session mobility but for terminal mobility too. The TDP protocol could be part of initial session negotiation between users with the goal of selecting an active mobility protocol that will be used during the session. More than one mobility protocol could also be selected if required and alternate between them based on the session requirements. For example, for delay sensitive applications an end-to-end mobility protocol is more suitable while for TCP based connections Mobile IP is used instead. The selection of the mobility protocol could also be based on the distance between the user's home network and the user. When that distance is large an end to end approach to mobility is preferable, avoiding triangular routing which will result in large delays.

7.3.2 Multimedia Conversion

With heterogeneous environments providing a varied quality of service it is desirable to have media conversion and caching in the network. Format conversion and stripping can decrease the download time over a lower bandwidth network. Taken a step further this format conversion should also be applied as part of the handover process. This is currently an area of active research (e.g., [5]). It is also possible that some networks might not be suitable for downloading specific types of traffic classes at a reasonable quality while other networks might be more suitable. In this case the network can cache the data and send it to the user when a suitable network is made available. While we expect our techniques from Chapter 6 to apply to these networks, there are other important problems to overcome, such as issues of inter-working between transcoders, caching servers, mobility protocols and the user.

7.3.3 SIP over WAP

As the mobile market continues to grow, the range of value-added services available to mobile users is expanding at an astonishing rate. Heading these up is the current global adoption of Wireless Application Protocol (WAP) services. WAP is designed to tackle the limitations of the mobile (cellular) network, in order to allow mobile devices to access internet-based services such as email, simple web access, stock alerts, etc. The WAP protocol is outside the scope of call signalling and audio transport. Therefore there is no direct mapping between the WAP protocol and SIP. Future research aims to provide a basic platform in which SIP-enabled IP terminals can interact with WAP enabled mobile phones. The Enhanced Sip User Agent developed in the current research could be expanded to take into account the WAP and SIP protocols providing a mapping between them from the signalling point of view. Content delivery between mobile IP terminals and mobile phones will be outside the scope of this research. The overall testbed could also be expanded to form a WAP-SIP platform that will form the skeleton for expanding this research activity. This is already part of active research presented in [6].

7.3.4 Multi-homing using SIP

The process of achieving simultaneous connectivity to different wireless networks is considered to be in the realm of multi-homing. It shares a variety of common issues with mobility. In fact, in heterogeneous wireless environments, exploring the

possibilities offered by multi-homing is considered highly beneficial because it allows for flexible and adaptable management of connections by mobile devices. Mobile devices should possess enough intelligence to adequately use any new wireless access which becomes available. From a practical point of view, this means that any new wireless connection available to a mobile device should not always be the trigger for an immediate and definite vertical handover from the old wireless access points. It should be noted that the intention here is not to analyse every vertical handover trigger (as it would be too complex and probably not efficient to consider such an option especially for highly mobile devices) but to offer a possibility of utilising different wireless accesses simultaneously during the overlap periods (also, one may consider other non-mobility bound scenarios where this could be useful for simultaneous access to hot-spot and macro coverage wireless accesses). The focus of this investigation of multi-homing mechanisms is placed at mobile terminals with multiple network interfaces with different addresses bound to each interface[1]. The Session Initiation Protocol could once again be the focus of this research for managing connectivity of multi-homed mobile terminals for achieving load splitting and balancing between the wireless accesses, *i.e.* mobile terminals are treating different interfaces independently but as a unit to split and balance single sessions between the interfaces according to the connection preferences.

[1] This is a particular instance of multi-homing. We are not considering the situation with multiple addresses for a single interface of the context of site multi-homing of the IETF's "multi6" working group.

7.4 References

[1] David A. Maltz, Pravin Bhagwat, "MSOCKS: An Architecture for Transport Layer Mobility", in proc. of infocom 98, page 1037, San Francisco, California.

[2] Alex C. Snoeren, Hari Balakrishnan, "An End-to-End Approach to Host Mobility, Proc. 6th ACM MobiCom, August 2000.

[3] http://www.cs.wisc.edu/~zandy/rocks/

[4] Bjorn Landfeldt, Tomas Larsson, "SLM, A framework for session layer mobility management", in Proc. IEEE International Conference on Computer Communications and Networks, pages 452--456, Natick, Massachusetts, October 1999.

[5] Caroline Lebre, Richard Titmuss, Pete Smyth, "Handover between heterogeneous IP networks for mobile multimedia services", Acts Mobile Summit, June 1999

[6] Paul Pangalos, Hamid Aghvami, "Transversal issues on cooperation between different Access technologies," 7th WWRF Meeting in Eindhoven, The Netherlands, 3rd - 4th December 2002.

"I used to think God's gifts were on shelves one above the other,
and the taller we grew the easier they were to reach.
Then I discovered they're on shelves one beneath the other,
and the lower we stoop the more we get." - F. B Meyer

Appendix A

A.1 Configuration and Setup of the GPRS Connection

The following scripts configure and setup a GPRS connection on a Linux terminal running kernel 2.4.18. This section describes the procedure used to setup and use the Nokia 6310 phone (BTCellnet) over an infra red connection.

Preliminary configurations: The pppd and irda packages need to be installed and configured. Full documentation can be found in their help pages and relevant howto's documents.

Files required:

Irda	// Initialise irda connection on laptop
gprs	// Contains pppd configurations
gprs-connect-chat	// GPRS connect AT command script
gprs-disconnect-chat	// GPRS disconnect AT command script

To initiate the GPRS connection: 1. Execute the 'irda' script to load all the infra red modules and initialise the irda port. 2. Execute the 'gprs' script. This will configure the infra red port and setup the GPRS connection through the BTCellnet network. The content of each script is shown below.

```
# File: gprs
# This file holds the IrDA pppd options for GPRS phones
# To give some debug info
debug

# Serial device to which terminal is connected;
# with serial port (COM1 in Windows) use /dev/ttyS0
# and with IrDA use /dev/ircomm0.
/dev/ircomm # IrDA
#/dev/ttyS0 # serial cable


# Serial port line speed
57600

# Hardware flow control needs to be used with serial cable.
# With IrDA it should be disabled with nocrtscts option.
#crtscts  # serial cable
nocrtscts # IrDA

# To keep pppd on the terminal
nodetach

# Connect script
connect /home/root/nokia_6310/gprs-connect-chat

# IP addresses:
# - accept peers idea of our local address and set address peer
as 10.0.0.1
# (any address would do, since IPCP gives 0.0.0.0 to it)
# - if you use the 10. network at home or something and pppd
rejects it,
# change the address to something else
:10.0.0.1

# pppd must not propose any IP address to the peer!
noipdefault

# Accept peers idea of our local address
ipcp-accept-local

# Ignore carrier detect signal from the modem
local

# No ppp compression
novj
novjccomp

# Disconnect script
disconnect /gprs-disconnect-chat

# Add default route
defaultroute
```

```
# File: irda
# The irda script:
killall irattach
#killall irmanager
sleep 1
rm -f /dev/ircomm
#mknod /dev/ircomm c 60 64
mknod /dev/ircomm c 161 0
rmmod ircomm-tty
rmmod ircomm
rmmod irtty
rmmod irda

insmod irda
insmod irtty
insmod ircomm
insmod ircomm-tty
irattach /dev/ttyS1 -s 1
#irattach /dev/ttyS1 -d tekram
#irmanager -d 1
sleep 5
cat /proc/net/irda/discovery
```

```
# File: chat-gprs-connect
# The chat script:
exec chat                                                       \
        TIMEOUT         5                                       \
        ECHO            ON                                      \
        ABORT           '\nBUSY\r'                              \
        ABORT           '\nERROR\r'                             \
        ABORT           '\nNO ANSWER\r'                         \
        ABORT           '\nNO CARRIER\r'                        \
        ABORT           '\nNO DIALTONE\r'                       \
        ABORT           '\nRINGING\r\n\r\nRINGING\r'            \
        ''              \rAT                                    \
        TIMEOUT         12                                      \
        SAY             "Press CTRL-C to close the connection at any stage!"    \
        SAY             "\ndefining PDP context...\n"           \
        OK              ATE1                                    \
        OK              'AT+CGDCONT=1,"IP","ADF.BTCELLNET.NET","",0,0'          \
        OK              ATD*99***1#                             \
        TIMEOUT         22                                      \
        SAY             "\nwaiting for connect...\n"            \
        CONNECT                 ""                              \
        SAY             "\nConnected." \
        SAY             "\nIf the following ppp negotiations fail,\n"  \
        SAY             "try restarting the phone.\n
```

```
# File: chat-gprs-disconnect
# The disconnect script:
# send break
exec /usr/sbin/chat -V -s -S            \
ABORT           "BUSY"          \
ABORT           "ERROR"         \
ABORT           "NO DIALTONE"   \
SAY             "\nSending break to the modem\n" \
""              "\K"            \
""              "+++ATH"        \
SAY             "\nPDP context detached\n"
```

Figure A-1: GPRS connection scripts

A.2 Implementation and Design Notes for the Sip User Agent

This section gives a detailed description of the key functionalities of the sip user agent. The main thread of the flowchart describes the program flow information indicating clearly the active class at any given point. Methods associated with each class are shown branching off the main classes. Calls to other classes are also noted and represented using boxed type shapes.

A.2.1 Initialisation of the sip user agent

The flowchart in Figure A-2 shows the classes and methods called during the initialisation process of the sip user agent (SUA).

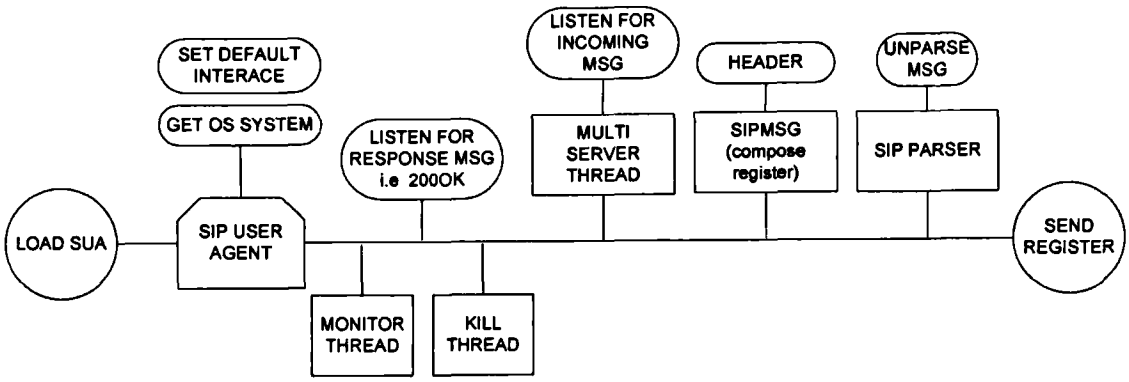


Figure A-2: SUA initialisation procedure

Description of the initialisation procedure

The SUA is programmed using Java, enabling support for both Windows and Linux based operating systems (OS). The GUI interface is shown in Figure A-3. During the initialisation phase, the 'main' method is called which is located in the SipUserAgent class. The program can be started from an MSDOS command prompt or a Linux terminal window. The OS is detected setting the appropriate flag. This is required since some methods were designed specifically for Linux operating system while others for windows based systems. All active interfaces are scanned and the appropriate one selected and set as the active SUA interface used for the signalling. If more than one interface is active, the decision is based on the user preferences. The monitor thread is used for triggering and initiating vertical handovers. This is done by monitoring the signal strength of all active interfaces. The kill thread listens for incoming SIP INFO messages. When such a message is received all applications associated with the current session are terminated or paused. Kill thread is only used during the inter-working of mobile IP and SIP mode. Once both of the above threads are running the sip user agent is ready to receive and sent sip messages. All incoming messages (i.e. INVITE) are received at the multi server thread class while all the response messages (i.e. 200 OK) are received in the sip user agent class. A registration message is composed and forwarded to the sip proxy updating the server's location database. The server responds with a 200 OK. The message exchange is shown in Figure A-4

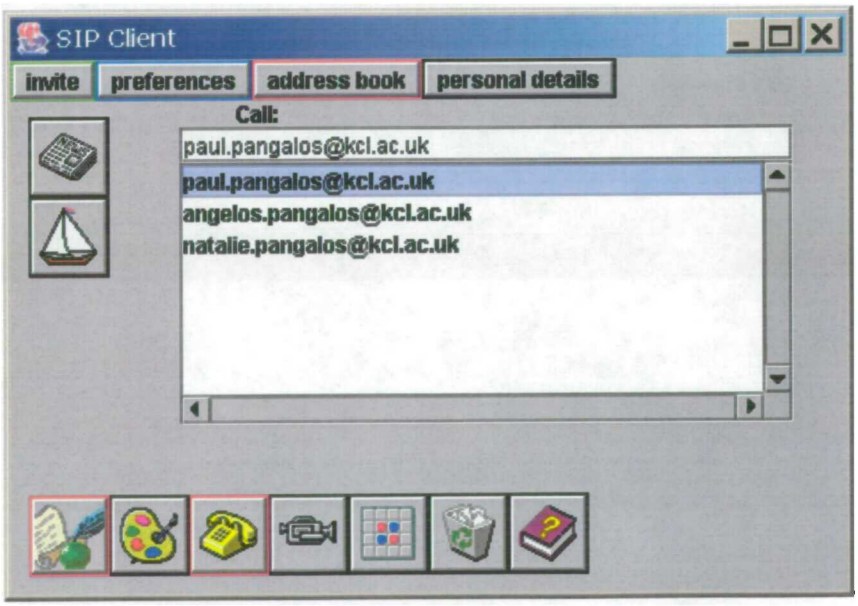


Figure A-3: GUI interface of the SUA

```
[root@ee026 SUA 09052003]# /usr/local/j2sdk1.4.0_01/bin/java SipUserAgent 7.73.11.96 siptcp
----------------------------------------
        Loading SUA...
----------------------------------------

Detecting OS...Linux
---> SIP MOBILITY MODE ON
---> SIP USING TCP
---> Found default interface:eth1
---> Active Interface SET to eth1
---> Primary Interface IP Address:137.73.11.32
---> SUA IP Address set to:137.73.11.32
loading config file paul.cfg

OPENING DIALOG BOX
ESTABLISHING TCP CONNECTION WITH: /137.73.11.96:5060
THE ADDRESS IN USE: /137.73.11.32
CALLER SENT REGISTRATION MSG WITH TCP:

REGISTER sip:kcl.ac.uk SIP/2.0
Via: SIP/2.0/TCP ee150
From: sip:paul.pangalos@kcl.ac.uk
To: sip:paul.pangalos@kcl.ac.uk
Call-ID: 916253@ee150
CSeq: 1 REGISTER
Contact: <sip:paul.pangalos@137.73.11.32:4000;transport=tcp>
Expires: 1200
Content-Length: -1


 SIPUSERAGENT CALLER RECEIVED TCP MESSAGE:
 SIP/2.0 200 OK
Via: SIP/2.0/TCP ee150
From: sip:paul.pangalos@kcl.ac.uk
To: sip:paul.pangalos@kcl.ac.uk
Call-ID: 916253@ee150
CSeq: 1 REGISTER
Contact: <sip:paul.pangalos@137.73.11.32:4000>;expires=TUE, 13 MAY 2003 17:07:04 GMT
Content-Length: -1
```

Figure A-4: The initialisation and registration phase of the SUA

A.2.2 Session Negotiation

The following three diagrams describe a session negotiation procedure setup between two sip user agents. Figure A-5 shows the process of sending an Invite message.

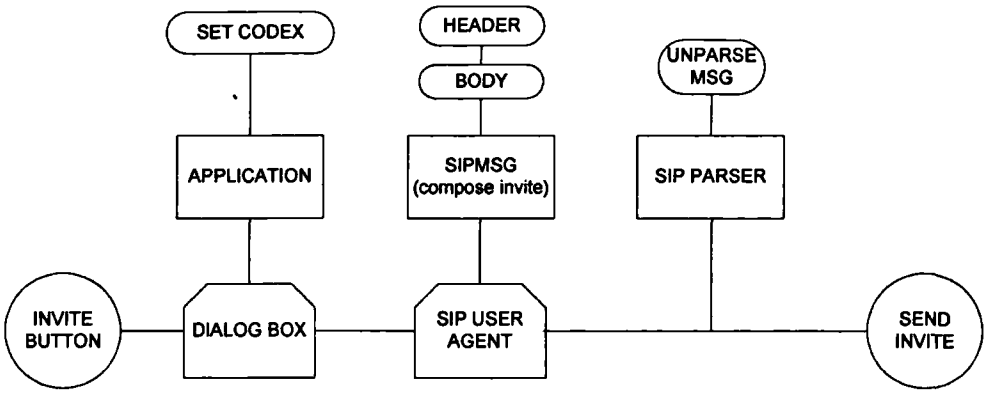


Figure A-5: Caller sending an Invite message

Using the graphical user interface, the user selects the required session (audio, whiteboard, video or chat) types, the sip address of the correspondent host and selects the 'invite' button. The application codecs are set within the dialog box and then passed to the sip user agent class. The sip message class is called to initialise and set all variables that make up the header and body parts of the invite message. Finally, it is the sip parser's job to compose the invite message and convert it to a string ready to be sent off to the sip proxy server.

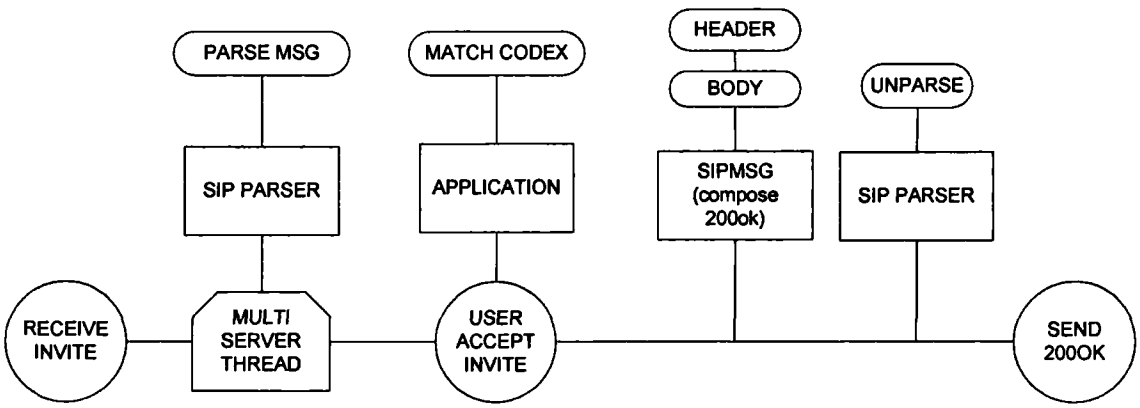


Figure A-6: Callee receiving the invite and responds with a 200OK

On reception of the invite message the callee will follow a reverse procedure to translate, examine and respond to the message Figure A-6. The message is received by the multi server thread and passed to the sip parser were it gets parsed back into the local variables. Once the user accepts the call, the codecs of the caller will be matched to those of the callee keeping the ones in common to both users. The same procedure is

then followed to generate the corresponding 200 OK message. All variables are set by the 'sipmsg' class while the sip parser composes and sends the message. Figure A-7 shows the corresponding message exchange.

```
CALLER SENT WITH TCP:
INVITE sip:angelos.pangalos@kcl.ac.uk SIP/2.0
Via: SIP/2.0/TCP ee150
From: paul.pangalos<sip:paul.pangalos@kcl.ac.uk>
To: sip:angelos.pangalos@kcl.ac.uk
Call-ID: 30649@ee150
CSeq: 1 INVITE
Content-Type: application/sdp
Content-Length: 90

v=0
o=paul.pangalos 30 607 IN IP4 137.73.11.32
c=IN IP4 137.73.11.32
m=audio 5036 RTP/AVP 1 3
---> Orininator Flag SET
--->SUA: start new thread
```

```
SIPUSERAGENT CALLER RECEIVED TCP MESSAGE:
 SIP/2.0 200 OK
Via: SIP/2.0/TCP ee150
From: sip:paul.pangalos@kcl.ac.uk
To: sip:angelos.pangalos@kcl.ac.uk
Call-ID: 30649@ee150
CSeq: 1 INVITE
Contact: sip:angelos.pangalos@ee95
Content-Type: application/sdp
Content-Length: 94

v=0
o=angelos.pangalos 899 626 IN IP4 137.73.11.153
c=IN IP4 137.73.11.153
m=audio 5036 RTP/AVP 1
```

Figure A-7: The Invite followed by the 200 OK during a session negotiation

The final leg of the negotiation procedure is initiated when a 200OK message is received by the caller. This is shown in Figure A-8. This indicates that the callee has agreed and is ready to join the session. The 200OK message is received by the sip user agent class and processed by the sip parser, separating the header and body fields. The caller responds by setting all the variables used by the sip parser and generates the ACK message. Before this message is send out, the caller starts its applications. Once the ACK is received the callee will also start up the required applications initiating a multimedia session between the two users. The message format of an ACK message is shown in Figure A-9.

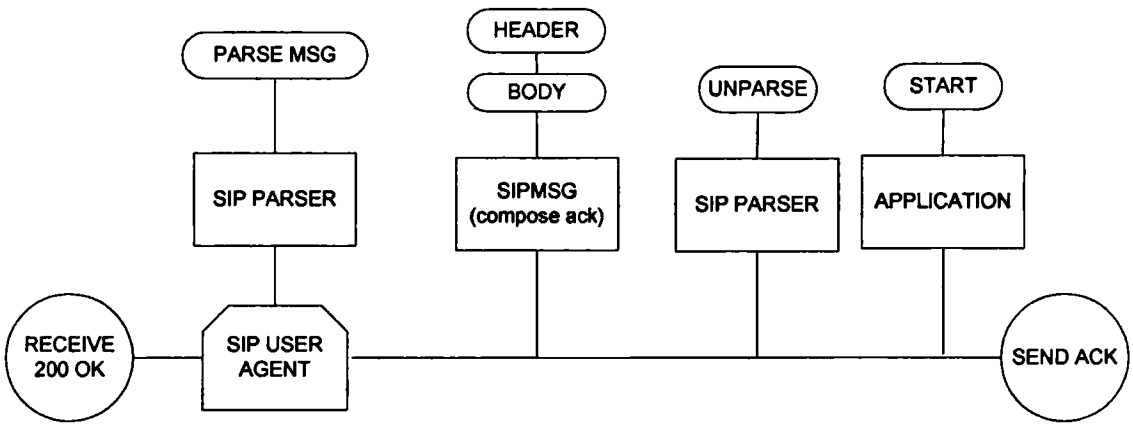


Figure A-8: Caller receiving 200OK and generates an acknowledgment.

```
CALLER SENT WITH TCP:
ACK sip:angelos.pangalos@kcl.ac.uk SIP/2.0
Via: SIP/2.0/TCP ee150
From: sip:paul.pangalos@kcl.ac.uk
To: sip:angelos.pangalos@kcl.ac.uk
Call-ID: 30649@ee150
CSeq: 1 ACK
Content-Length: 94

v=0
o=angelos.pangalos 899 626 IN IP4 137.73.11.153
c=IN IP4 137.73.11.153
m=audio 5036 RTP/AVP 1
```

Figure A-9: The ACK message

A.2.3 Terminating the Sip User Agent

The reverse procedure is followed in Figure A-10 for shutting down the agent. All the methods called are located in the sip user agent class. Firstly, all the current active threads are closed. That includes both the kill, and monitor thread. The elements of a de-registration sip message are then created and the full message composed within the sip parser Figure A-11. This message is sent to the sip proxy server to clear the sip entry for that mobile node.

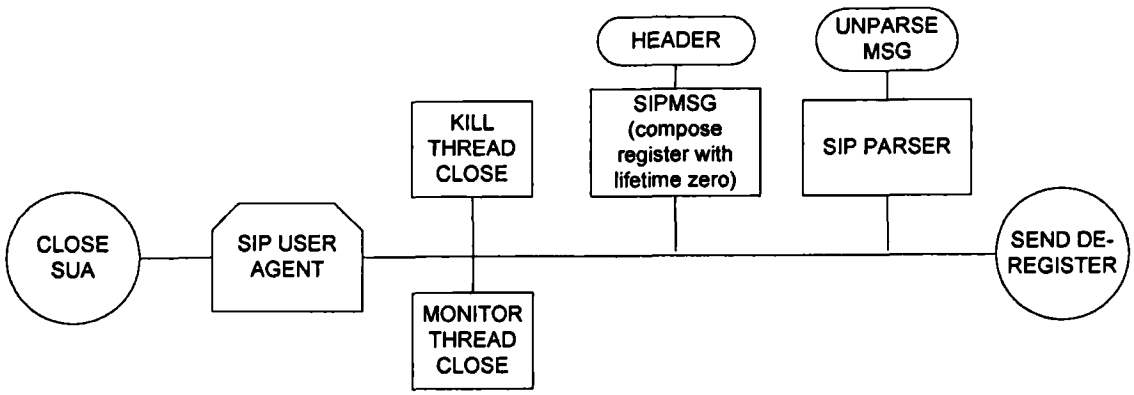


Figure A-10: Terminating the sip user agent.

```
CALLER SENT WITH TCP:
REGISTER sip:kcl.ac.uk SIP/2.0
Via: SIP/2.0/TCP ee150
From: sip:paul.pangalos@kcl.ac.uk
To: sip:paul.pangalos@kcl.ac.uk
Call-ID: 916253@ee150
CSeq: 2 REGISTER
Contact: <sip:paul.pangalos@137.73.11.32:4000;transport=tcp>
Expires: 1200
Content-Length: -1
```

Figure A-11: A deregistration message

A.3 Technical Description of the Enhanced Sip User Agent

Application.java

This class handles application related information. It stores configuration information for each type of application (voice, video, chat, whiteboard) such as port and relevant codec payload numbers. It is also responsible for loading and maintaining the relevant applications in terms of mobility (IP address change) and codec change (media adaptation).

isChat(), isRat(), isFphone(), isRvplayer(), isWhiteboard()
Methods used for setting and configuring various types of applications. Parameters include name of application, port numbers, and payload codec information.

matchCodecs()
Match codecs with the ones of the correspondent host, given in the Invite msg, and keep the ones in common. It supports both audio and video codecs.

setIptables()
Used to interface and configure the iptables in the linux OS. This requires the configuration of three chains named: POSTROUTING, PREROUTING, and OUTPUT chains. This method is used in the end-to-end mobility support mode of the sip user agent.

setPortfwd()
This method inter-works with the setIptables to handle mobility in the end-to-end mode of the sip user agent. It provides mobility support for vertical handover for normal applications, that is ones that are not capable of handling IP address change.

start()

Once a 200OK message is received from the correspondent host, this method loads the applications required.

sendIPtohomeagent()

Send the correspondent host IP address to the home agent. This is only required when the sip user agent is operated in the loose coupling mode.

tcp_Conn()

Wait for short based TCP connections (i.e. web based, e-mail) to close before the sip user agent hands over to another interface. This is initiated only during planned handovers.

ProcessInfo.java

Class for processing session descriptions and displaying the received information to the user. Based on this information the user is able to select a terminal to handover his session. Part of this class is also used to extend the sip proxy server to handle SIP INFO messages. This includes processing, storing and retrieving session descriptions.

SipInfoParser.java

Consists of two basic methods: Unparse() and Parse().

Parser():

Parsers an incoming INFO message into component parts and if successful it returns the Sip Message otherwise it returns an empty message.

Unparse():

Transforms a SIP INFO message into a sendable string.

This class has a similar structure to the SipParser.java class

SipParser.java

Parsers and unparsers SIP messages. Functionality similar to the SipInfoParser.

Config.java

The aim of this class is to load the configuration file for the sip user agent host. The configuration file has the following format. The username and passwords are used to authenticate the user while the other variables are parsed into various variables in the main program.

```
user{
    UserInfo paul.pangalos;
    FullName Paul.Pangalos;
    Domain kcl.ac.uk;
    Password kgcf2292;
    hostname ee150;
}

preferences {
    AudioPayload 1.3;
    VideoPayload 31.33;
    bitrate 10M;
    media Ethernet;
    class 2;
}
```

User {}: Includes the users full name, domain and authentication information.

Preferences {}: Stores the users preferences such as preferred audio and video codecs, as well as the bitrate and media type of users current internet connection.

ConfigurationFile.java

Stores all necessary variables required by SIP to inter-work with Mobile IP in both loose and tight coupling modes.

Each of the variables is briefly described below.

```
//This boolean variable defines if the node is a mobile node
public static boolean isMobileNode = true;

//This String gives the IP of the home agent
public static String homeAgentIP = "137.73.11.95";

//This integer has the value of the port in which the home agent
listens for messages from mobile nodes
public static int HomeAgentKillPort = 9009;

//This integer has the value for the port to which kill messages are sent
public static int killPort = 9000;

//This integer has the value of the port in which mobile IP and SIP communicate
public static int listeningPort = 9478;
```

TrigerThread.java

This is a thread started by the sip users agent. It is used to listen for messages coming from mobile IP in the loose coupling mode. It was initially implemented to run in TCP mode but later changed to UDP. The corresponding thread is loaded in mobile IP, and is the one that performs the call for initiation of the connection. However if the sip user agent is not already running when mobile IP is loaded, a run time error will occur and the thread will terminate. This leads to a state where the two protocols can not communicate with each other. To eliminate such problems the UDP protocol was used instead.

HomeAgentKillThread.java

A thread opened at the home agent to perform two tasks: First, it listens for UDP messages from the mobile node containing the IP address of its correspondent host. Second, it sends the correspondent host a kill INFO message when the mobile node experiences a disconnection.

Run()

Main method, listening for incoming messages. A hashtable is used to keep track of all IP addresses received from all mobile hosts.

SendKillToCN()

Sends a kill message to the correspondent host on behalf of the mobile node.

DialogBox.java

This class implements the graphic user interface (GUI) of the sip user agent. It consists of a text box and various buttons. The constructor of this class is used to create the dialog box interface with all the action listeners. Figure A-3 shows the GUI Interface. There are four settings that are used to set the type of session required. They are the chat, whiteboard, voice, video and game buttons. The invite key, is used to generate an invite message based on the callee address and the session setting parameters. Finally, the two buttons labelled 'txinfo' and 'rxinfo' are used during session mobility to send and receive terminal based information to and from the terminal description server.

actionPerformed()

It listens for all button interactions within the dialog box and initiates the appropriate actions/flags.

trigerInvite()

This method is used for both proxy and end-to-end based operation of SIP. In the proxy mode, trigerInvite() acts as the bridge between the sip user agent and mobile IP. It accepts incoming messages from mobile IP, enabling the inter-working between the two agents for both the loose and tight coupling architectures. Furthermore, this method is also called from the monitorThread.java during a pure end-to-end SIP operation mode to initiate a session re-negotiation during both upward and downward handovers. All calls to this method carry a string referred to as 'networkType'.

MonitorThread.java

The function of the MonitorThread class is to detect and initiate both planned and unplanned handovers. This is done by monitoring the application and link layers of the terminal. The application layer mechanisms monitor specific audio/video port(s) listening for real time control protocol (RTCP) packets. These packets contain jitter and packet loss information about the RTP stream assisting sip in handover decisions. However, handover decisions are not solely based on one mechanism. Another mechanism employed is monitoring of the link layers by obtaining regular signal strength measurements for each wireless interface in the terminal. A rapid decrease in signal strength (interface unplugged) would initiate an unplanned handover while a more smooth degradation (user moving outside a coverage area) would trigger a planned handover.

Info.java

This is the core class used to generate the terminal description information used in the session description protocol. It collects terminal information parameters such as interface configuration, cost, the terminal name, hostname, signal strength and passes this information to the INFO parser to be composed in a complete SIP INFO message.

JMFApi.java

This class contains the core functions for the Mobile Multimedia Application (M2A). Furthermore, it receives processes and response to SIP messages by creating appropriate audio players for each session. This class also contains functions required to adapt an ongoing session in response to users request or a vertical handover.

MobileIpTriger.java

This class acts as a bridge between the M3A mobile IP application and the Sip User Agent. It is used to transfer signalling information in a Unidirectional manner between Mobile IP and the Sip User Agent.

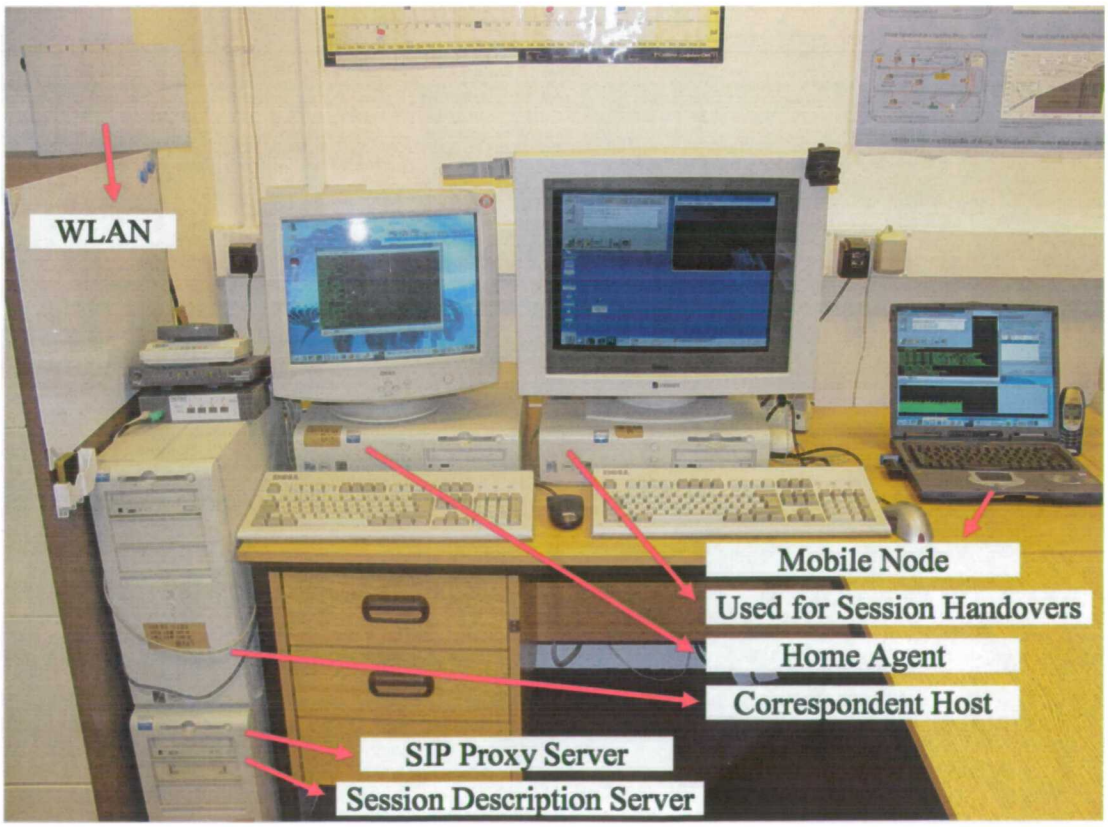
SipMsg.java, MsgBody.java and MsgHeader.java

Classes responsible for creating the SIP message body and header fields. These classes were expanded to carry new types of fields as well as creating SIP INFO messages.

MultiServerThread.java

All incoming messages are received and processed here. This class was also extended to support vertical handovers as well as receiving and processing SIP INFO type of messages. Some of the key output methods within this class include, processing of ACK messages, composing 200 OK messages and processing INFO ACK messages. Finally, session handover functionality was also included found in the TransferSession() method.

A.4 Testbed Photo



A.5 Low Level Description of M2A – Design 1

This section describes in more detail the operation of the flowchart using phedocode.

1. First create a Processor for the given capture device

```
locator = new MediaLocator("dsound://");
DataSource = javax.media.Manager.createDataSource(locator);
processor = javax.media.Manager.createProcessor(DataSource);
```

2. Configure the Processor with the specified codec

```
private AudioFormat GSMFormat = new AudioFormat(AudioFormat.GSM_RTP,
8000, 8, 1);
private AudioFormat ULAWFormat = new AudioFormat(AudioFormat.ULAW_RTP,
8000, 8, 1);
private AudioFormat DVI = new AudioFormat(AudioFormat.DVI_RTP, 8000, 4,
1);
private Processor processorULAW = null, processorDVI = null,
processorGSM = null;
```

3. Set the output content descriptor (ContentDescriptor.RAW) on the Processor

```
ContentDescriptor cd = new ContentDescriptor(ContentDescriptor.RAW);
```

4. Find the matching track format, and set the track.
5. Realize the Processor with the specified track format

```
processorULAW.configure();
waitForState(processorULAW, Processor.Configured);
TrackControl track[] = processorULAW.getTrackControls();
for (int j = 0; j < supported.length; j++) {
if (supported[j].matches(ULAWFormat)) {
chosenFormat = supported[j];
break;
tracks[i].setFormat(chosenFormat);
waitForState(processor, Controller.Realized); }
```

6. Obtain the output DataSource from the output of the Processor and use it for the input of the CreateSendStream to create the RTP transmitting data source that will be used by the RTPManager object.

```
dataOutput = processor.getDataOutput();
```

7. Create RTPManager for generating the RTP packer properties

```
private RTPManager rtpManager;
rtpManager = RTPManager.newInstance();
```

8. Use the initialize method to set the local IP address and port

```
localAddress = new SessionAddress(
InetAddress.getLocalHost(),localport);
rtpManager.initialize( localAddr);
```

9. Use the addTarget method to set the destination IP address and port

```
destinationAddress = new SessionAddress( ipAddress, destport);
rtpManager.addTarget(destinationAddress);
```

10. Use SendStream to send the RTP packets to the destination IP address using the specified codec

```
sendStream = rtpManager.createSendStream(dataOutput, 0);
sendStream.start();
```

A.6 Low Level Description of M2A – Design 2

1. Create a Processor for the given capture device

```
locator = new MediaLocator("dsound://");
origDataSources[i] = Manager.createDataSource(locator[i]);
cloneDataSource =
javax.media.Manager.createCloneableDataSource(origDataSources[i]);
processor_info[i].processor = Manager.createProcessor(cloneDataSource);
```

2. Configure the Processor with the specified codec
3. Set the output content descriptor (ContentDescriptor.RAW) on the Processor
4. Find the matching track format, and set the track.
5. Realise the Processor with the specified track format

```
TrackControl track[];
ContentDescriptor cd = new ContentDescriptor(ContentDescriptor.RAW);
waitForState(
processor_info[i].processor,processor_info[i].processor.Configured)) {
track = processor_info[i].processor.getTrackControls();
processor_info[i].processor.tracks = tracks;
processor_info[i].processor.setContentDescriptor(cd);
for (int j = 0; j < supported.length; j++) {
if (supported[j].matches(processor_info[i].processor.format)) {
chosenFormat = supported[j];
break;}}
processor_info[i].processor.tracks [j].setFormat(chosenFormat);
waitForState(processor_info[i].processor,
processor_info[i].processor.Realized);
processor_info[i].pbdatasource=(PushBufferDataSource)processor_info[i].p
rocessor.getataOutput();
processor_info[i].pbstreams = (PushBufferStream
[])processor_info[i].pbdatasource.getStreams();
```

6. Obtain the output DataSource from the Processor and use it for the input of the SwitchDataSource to create a buffer object frame by frame

```
class SwitchDataSource extends PushBufferDataSource {
       Stream streams[];
       public SwitchDataSource(PushBufferDataSource pbds) {
              streams = new Stream[];
              streams[] = new Stream(pbds);                    }
       public void connect() throws java.io.IOException {}
       public PushBufferStream [] getStreams() {
              return streams;}
       public void start() throws java.io.IOException {}
       public Object getControl(String name) {}
       public String getContentType() {
              return ContentDescriptor.RAW;}}

class Stream implements PushBufferStream, BufferTransferHandler {
       SwitchDataSource ds;
       PushBufferDataSource pbds;
       PushBufferStream pbs;
       BufferTransferHandler bth;
       public GlueStream(PushBufferDataSource pbds) {this.pbds = pbds;
       }
```

```
        public void setStream() {
                if (pbs != null)
                pbs.setTransferHandler(null);
                …
                pbs.setTransferHandler(this);
        }
        public void read(Buffer buffer) throws IOException {
                pbs.read(buffer);
                …
                if (buffer.getFormat() instanceof AudioFormat) {
                        AudioFormat af = (AudioFormat)buffer.getFormat();
                }
                if (sessionflag){
                        buffer.setEOM(true);
                        if (handleEOM()) {
                                buffer.setEOM(false);
                                buffer.setDiscard(true);
                                sessionflag =false;
                                }
                        }
        }
        synchronized boolean handleEOM(){
        setStream();
        }
        public void setTransferHandler(BufferTransferHandler bth) {
                this.bth = bth;
        }
        public synchronized void transferData(PushBufferStream pbs) {
                if (bth != null)
                        bth.transferData(this);
        }
} // class Stream
```

7. Create RTPManager for generating the RTP packer properties

```
private RTPManager rtpManager;
rtpManager = RTPManager.newInstance();
```

8. Use initialize to set the local IP address and port

```
localAddress = new SessionAddress(InetAddress.getLocalHost(),localport);
rtpManager.initialize( localAddr);
```

9. Use addTarget to set the destination IP address and port

```
destinationAddress = new SessionAddress( ipAddress, destport);
rtpManager.addTarget(destinationAddress);
```

10. Use SendStream to send the RTP packets to the destination IP address

```
sendStream = rtpManager.createSendStream(datasource, 0);
sendStream.start();
```

"The proper function of man is to live - not just exist. Therefore
I shall not waste my days by trying to prolong them.
I shall use my time!"- Jack London

Appendix B

This appendix presents the preliminary results for the further work described in section 7.3. Figure B-1 illustrates the results obtained from a pure SIP based signalling approach during an upward vertical handover. Figure B-2 illustrates the same signalling technique used for downward handovers. Each of these examples are then summarised in terms of the signalling delays associated with the handovers. Figure B-3 illustrates this. Finally Figure B-4, shows results obtained during the initial stages of research performed based on multihoming scenarios using the session initiation protocol.

B.1 Upward Vertical Handover using a Pure SIP Approach

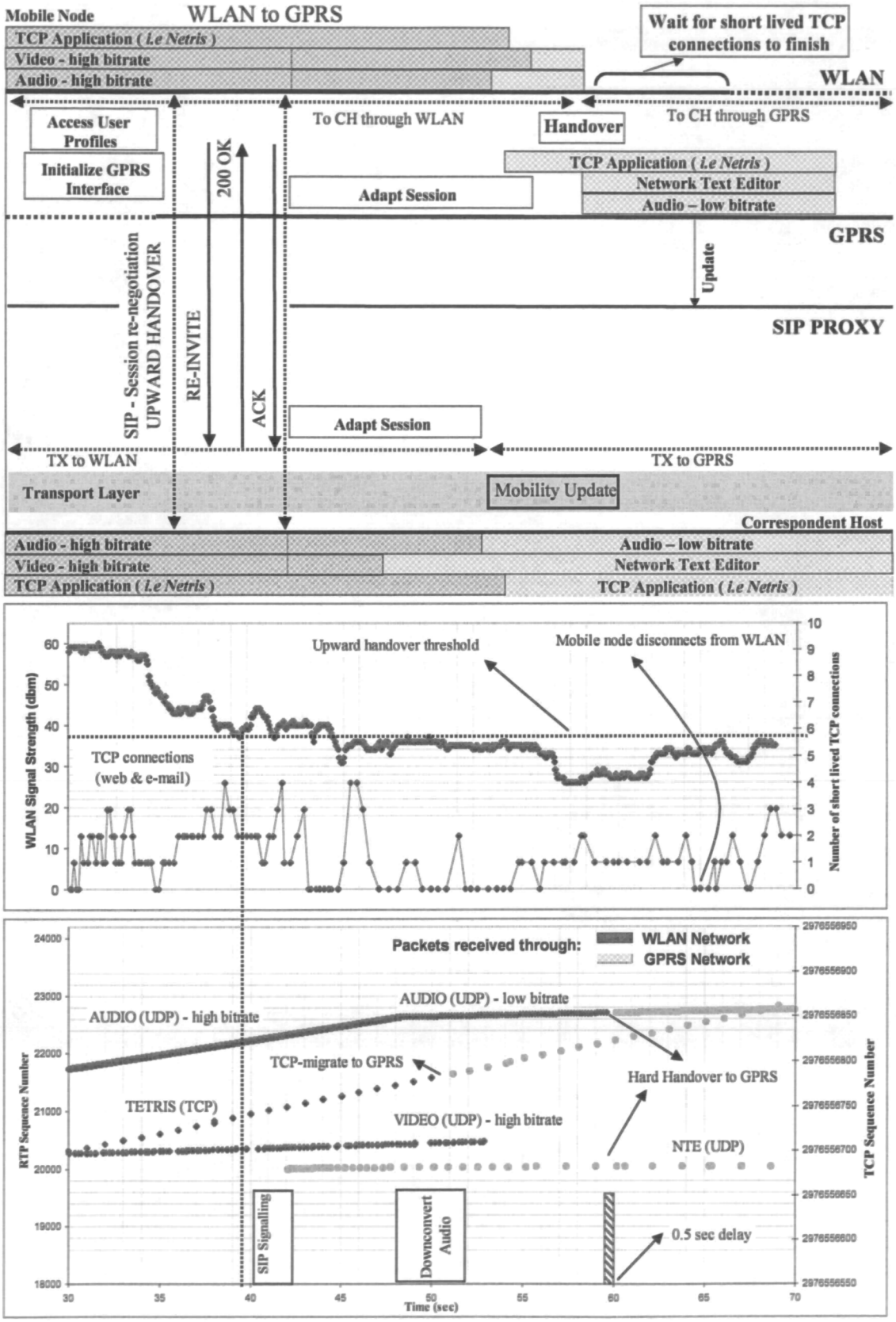


Figure B-1: Upward vertical handover (WLAN to GPRS network)

B.2 Downward Vertical Handover using a Pure SIP Approach

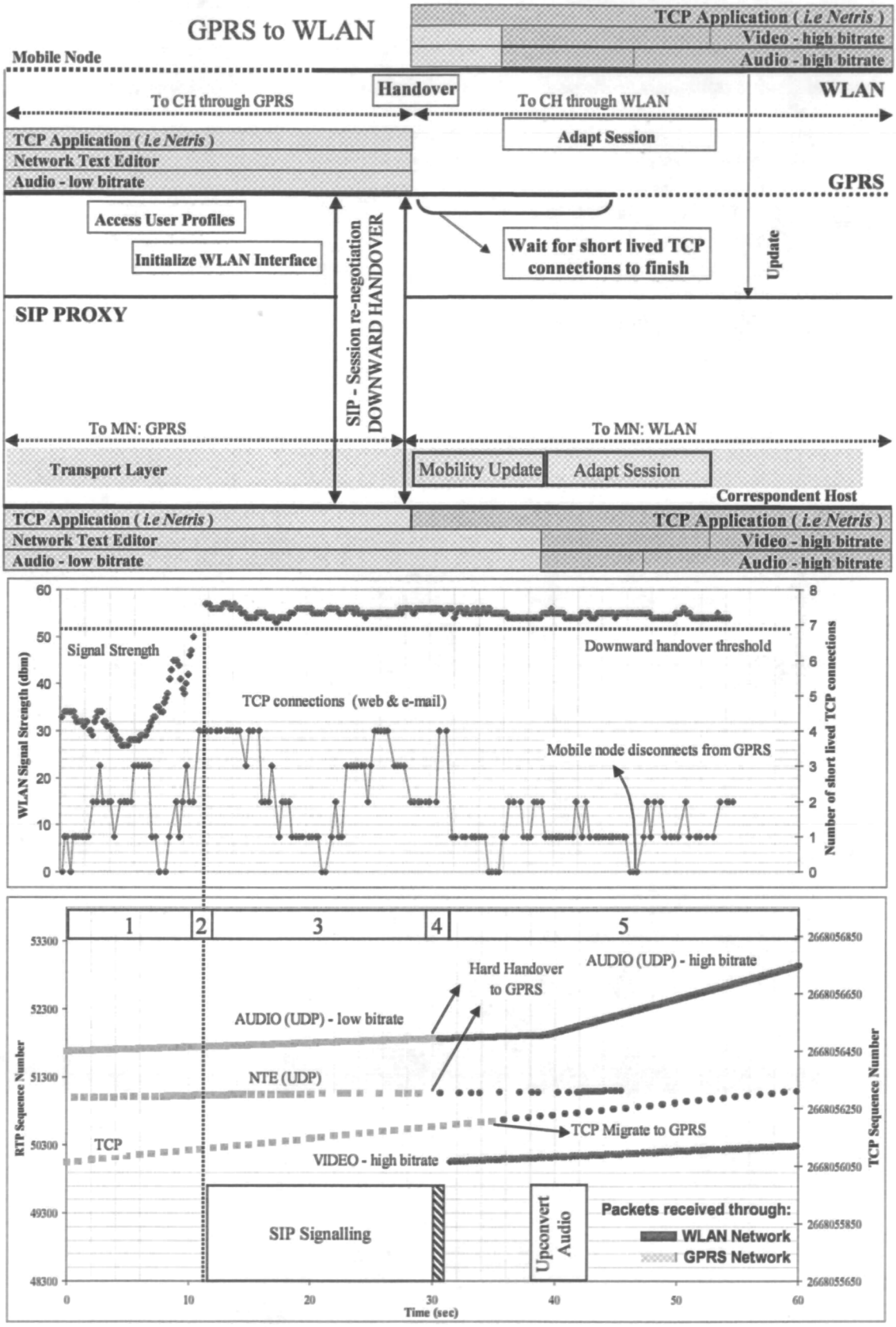


Figure B-2: Downward vertical handover (GPRS to WLAN network)

B.3 Signalling Transmission Delays using a Pure SIP Approach

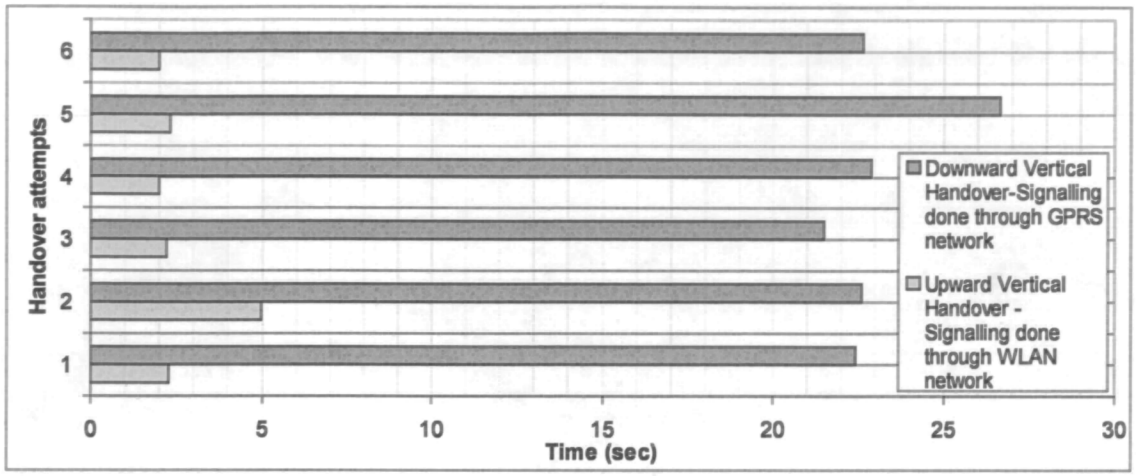


Figure B-3: Signalling transmission delays during vertical handovers

B.4 Multihoming Scenario for Real Time Applications

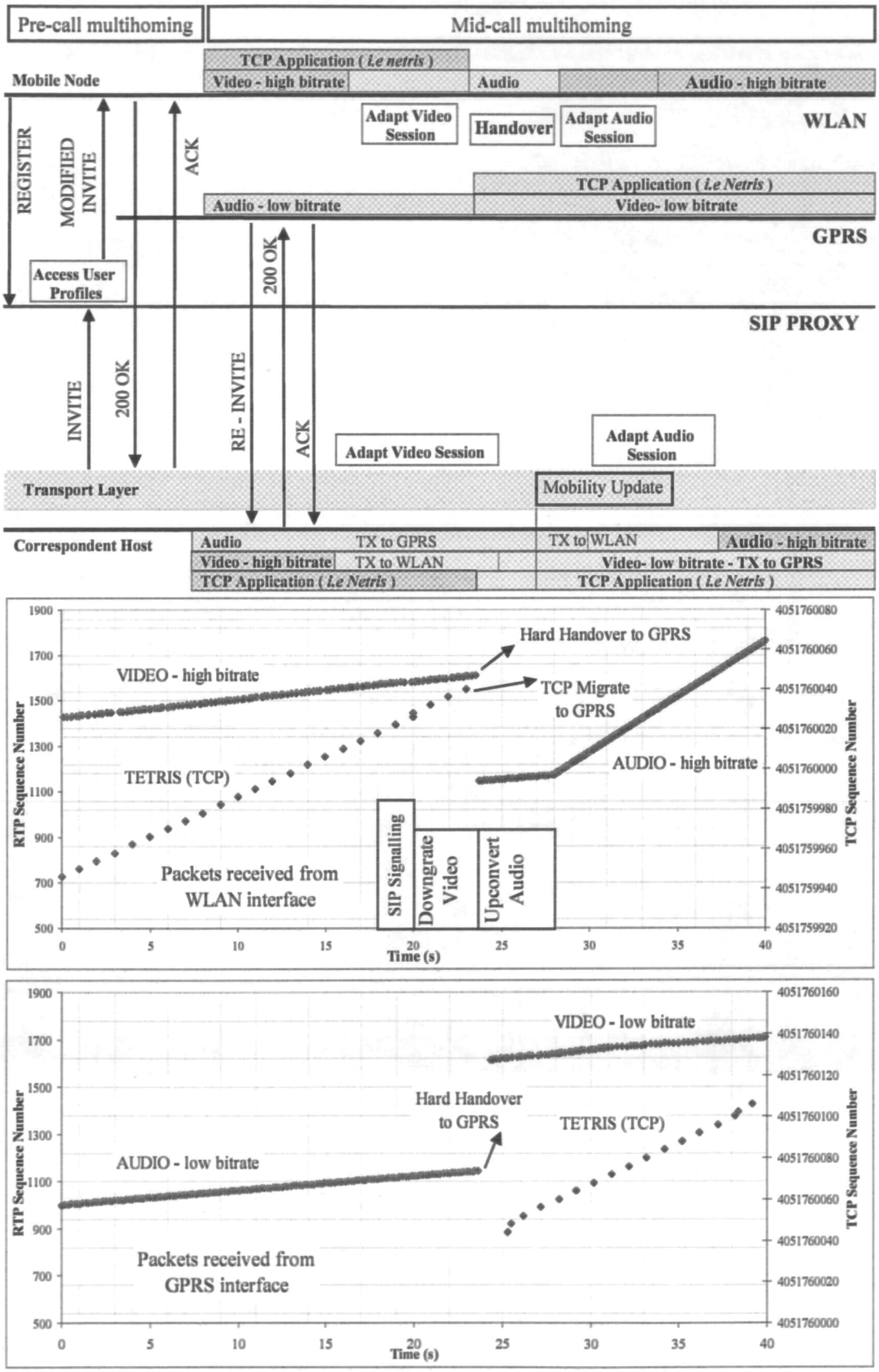


Figure B-4: Multihoming scenario for real time applications

Publications

Book Contributions

Prof. A. H. Aghvami, Paul Pangalos, Dr. Babak Jafarian, "Mobility Management for an Integrated Network Environment," to be published end of 2003.

Conferences:

1. Nima Sattari, Paul A Pangalos, Hamid Aghvami, "Handovers between UMTS and WLAN" VTC2004, Italy Milan, May 19-21, 2004.

2. Paul A Pangalos, Hamid Aghvami, "The Terminal Description Protocol (TDP)", WWRF8bis Beijing, China, 26-27 February 2004.

3. Paul A Pangalos, Hamid Aghvami, "A Vision for Efficient Heterogeneous Internet Connectivity for Mobile Devices and Networks",WWRF10, in New York, USA, 27 – 28 October, 2003.

4. Paul A Pangalos, Daniel Morris, Nima Sattari, Hamid Aghvami, Kar Ann Chew, Rahim Tafazolli, "Inter-working of Broadcast, WLAN and Cellular networks", WWRF9, Zürich, Switzerland, 1-2 July 2003.

5. Paul A Pangalos, Wing Shun Wong, Hamid Aghvami "Real-time Transport Protocol in Java Media Framework, application adaptation performance over heterogeneous networks", WWRF8, Beijing, China, 7-8 April 2003.

6. Paul A Pangalos, Wing Shun Wong, Hamid Aghvami, "Java Based Adaptable Multimedia Application (M2A)", SOFTCOM 2003, Ancona, Venice (Italy), Split, Dubrovnik (Croatia), October 2003.

7. Paul A Pangalos, Hamid Aghvami, "Transversal issues on cooperation between different Access technologies," 7th WWRF Meeting in Eindhoven, The Netherlands, 3rd - 4th December 2002.

8. Hamid Aghvami, Paul A. Pangalos, "Towards 4G", 2nd WWRF workshop, May 10-11, 2001 in Helsinki.

9. Paul A Pangalos, Hamid Aghvami, "IP-Based Vertical Handovers", 4th WWRF workshop in Paris on December 6-7, 2001.

10. Paul A. Pangalos, Konstantinos Boukis, Louise Burness, Alan Brookland, Caroline Beauchamps, A. H. Aghvami, "End-to-End SIP Based Real Time Application Adaptation During Unplanned Vertical Handovers", Proc. GLOBECOM 2001.

11. Paul A Pangalos, Konstantinos Boukis, Dave Wisely, Louise Burness, Prof. A. H. Aghvami, "Experimental Evaluation of Signaling Methods for SIP Based Real Time Application Adaptation During Unplanned Vertical Handovers", Proc. IST 2001.

Other Publications

J.F. Whidborne, P. Pangalos, Y.H. Zweiri and S.J. King, "A graphical user interface for computer-aided robust control system design," Proc. Engineering Design Conf. 2002, pp. 383-392, London, UK, July 2002.