ACA 2574

School of Computing

An Investigation into The Application of Active Networks to Mobile Computing Environments

Kwan-Wu Chin

This thesis is presented as part of the requirements for the award of the Degree of Doctor of Philosophy of the Curtin University of Technology

March, 2000

Abstract

Telecommunication service providers have recently begun to offer ubiquitous access to packetised data. As a result, the Internet is not limited to computers that are physically connected but is also available to users that are equipped with mobile devices. This ubiquitous access fuels the growth and the usage of the Internet even further, and thus the realisation of dynamic Internet. With the realisation of the dynamic Internet, increasing support is needed for Internet protocol (IP) and transmission control protocol (TCP) over wireless/mobile networks.

Two areas of interest in this thesis are unicast and multicast routing in connectionless and connection-oriented networks. To address the problems of routing protocols in mobile computing environments, the active networks (ANs) paradigm is employed. ANs provide an alternative paradigm to solving network problems and comprise programmable network elements that allow enhancement of existing protocols and the execution of active protocols which run for the duration of the communication session.

This thesis investigates the viability and advantages of ANs when applied to routing in mobile communications. Two new AN-based protocols, for IP and asynchronous transfer mode (ATM) networks, that address the problems of multicast routing with mobile group members are outlined. The Internet Engineering Task Force (IETF) mobile IP has been augmented with active programs in order to enhance its operation further. Also, a novel model for rerouting connections in ATM networks is presented.

Results of extensive simulation studies comparing performances of conventional as well as some recently proposed protocols with those of AN-based protocols are presented. The results obtained from these simulation studies show that AN-based protocols have the following benefits: (i) efficient adaptation to mobility, (ii) reduced signaling overheads, (iii) high reuse of allocated network states, (iv) extensibility, (v) network topology independence, and (vi) scalability. The aforementioned points are crucial in mobile environments where states at routers (switches) are frequently updated due to mobility. It was shown that ANs provide the most benefits to protocols that maintain states within the network, for example connection-oriented and multicast protocols. AN-based protocols enable fast and efficient update of the states maintained at the routers/switches without incurring excessive signaling overheads. Moreover, part of a connection or multicast tree can be updated iteratively with the use of ANs, resulting only in modifications to routers (switches) that are affected by host migration. A model for deploying active programs that is coupled with the the protocol operation is also demonstrated. Implementation of such a model eliminates the need for strategic positioning of active services.

Preface

The work presented in this thesis has been published in conferences, journals, and technical reports. The thesis work has resulted in the following publications.

Chapter 4:

Enhancing Mobile IP Routing Using Active Routers. 8th International Conference on High Performance Computing (HiPC'99) Calcutta, India Dec 17-20, 1999.

Enhancements to Mobile IP with Active Networks. The 2nd International Working Conference on Active Networks (IWAN'2000) Tokyo, Japan. Oct. 16th-18th, 2000.

Enhancements to Mobile IP Using Active Networks. Technical Report No. 5, School of Computing, Curtin University of Technology, Bentley, Western Australia, December, 1999.

Chapter 5:

A Model for Enhancing Connection Rerouting Using Active Networks. The Second ACM International Workshop on Modeling and Simulation of Wireless and Mobile Systems (MSWiM'99). Seattle, Washington, August 15-19 1999, USA. *Also as Technical Report No. 1, January, 1999*

A Model for Connection Rerouting in Mobile Networks ACM/Baltzer MONET Special Issue on Modeling and Simulation of Wireless Networks, 2000. *To Appear*

Chapter 6:

AMTree: An Active Approach to Multicasting In Mobile Networks IEEE 8th International Conference on Computer Communications and Networks (ICCCN'99). Boston, Massachusetts,USA. Oct 11-13th 1999.

AMTree: An Active Approach to Multicasting In Mobile Networks ACM/Baltzer MONET, Special Issues on Wireless Multicast and Routing, 2000. *To Appear*

Chapter 7:

MCoRe: An Adaptive Scheme for Rerouting Multicast Connections in Mobile ATM Networks. Elsevier Journal of Computer Communications, 2000. *Being Reviewed*

Acknowledgements

The quest for a doctorate in computing have been a long one indeed! There are a number of people who has been a major influence in my life for the past three years. Personally, I believe they have brought out the best in me and also provided the financial and moral support, which have played a significant role in my life.

First of all I would like to thank my honours supervisors, Andrew Marriott and Dr. Brian von Konsky, for encouraging me to continue my study after I completed my honours. They are the people who brought out the best in me and I am eternally grateful. I like to thank Andrew especially, for being a friend and offering me good advice through out these years, and for getting me projects that enable me to cover my living expenses for the past three years and also helping me obtain a scholarship to cover my tuition fee.

I like to thank my PhD supervisor, Dr. Mohan Kumar, for his guidance and being so meticulous about my work. He has certainly raised the quality of my work and provided tonnes of advices on doing research. Also, I am grateful that he is willing to supervise me after my initial supervisor, Dr. Craig Farrell, left to make his millions.

I like to thank the Sabah state government for funding my bachelor degree, without their support I would not have gotten this far. Also, I like to thank the school of computing for giving me a PhD scholarship.

I like to thank Dr. Upkar Varshney for agreeing to review my paper which have improved my work considerably. Also to all anonymous reviewers at conferences and journals who have taken the time to review my work and provided constructive/destructive criticisms and positive feedbacks which have certainly raised the standard of my work.

Friends that I like to thank include Allan Loh, Stuart Campbell, Brad Williamson, William Lau and Lim Kok Yong. I like to thank William Lau especially for going through some of the ideas in this thesis. All those countless disagreements and brain storming sessions have proven to be very beneficial.

Another person that I am eternally grateful to, is my sister shui han. She has been so supportive and has taken care of me for the past few years. She has made this journey so much easier. Without her constant support, this thesis would have taken much longer to complete. Finally I would like to thank my parents, who constantly believe that I can accomplish something significant.

Finally, to my soul-mate, Rita Chan, who has been so supportive all these years. Without her love and understanding, this thesis would never have started or ended.

Thank You!

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AAL	ATM Adaptation Layer
ABR	Available Bit Rate
ACK	Acknowledgement
ACR	Active Connection Rerouting
ADC	Active Discovery Capsule
ADS	Active Delivery Scheme
AMTree	Active Multicast Tree
ANEP	Active Network Encapsulation Protocol
ANs	Active Networks
APM	Active Program Manager
AR	Active Router
ARP	Address Resolution Protocol
AS	Active Switch
ASN.1	Abstract Syntax Notation Number one
ATM	Asynchronous Transfer Mode
BS	Base Station
BSC	Base Station Controller
BTS	Base Transceiver Station
CBT	Core Base Tree
CBR	Constant Bit Rate
CH	Corresponding Host
CPU	Central Processing Unit
CX	Crossover Switch
DNS	Domain Name Service
DTL	Designated Transit List
DVMRP	Distance Vector Multicast Routing Protocol
EDGE	Enhanced Data Rates for GSM Evolution
HA	Home Agent
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IP	Internet Protocol

FA	Foreign Agent
FH	Fixed Host
GFA	Gateway Foreign Agent
GSM	Global System for Mobile Communication
IMHP	Internet Mobile Host Protocol
LAN	Local Area Network
LD	Logical Domain
LIJ	Leave Initiated Join
LSR	Loose Source Routing
MAC	Media Access Control
MARS	Multicast Address Resolution Server
MBONE	Multicast Backbone
MCS	Multicast Server
MH	Mobile Host
MHA	Multicast Home Agent
MHRP	Mobile Host Routing Protocol
MSC	Mobile Switching Centre
MSS	Mobile Support Station
MTM	Multipoint to Multipoint
NACKS	Negative Acknowledgments
OSPF	Open Shortest Path Forwarding
PC	Personal Computer
PDA	Personal Digital Assistance
PIM	Protocol Independent Multicast
PLAN	Packet Language for Active Network
PPK	Prior Path Knowledge
PNNI	Private Network to Network Interface
PR	Proxy Root
PPF	Packet Processing Filter
PTM	Point to Multipoint
QoS	Quality of Service
RIP	Routing Internet Protocol
RM	Resource Management
RP	Rendezvous Point
RTT	Round Trip Time
SNMP	Simple Network Management Protocol
SRT	Source Rooted Tree
TCP	Transmission Control Protocol
TCL	Tool Command Language
TTL	Time to live
WAP	Wireless Application Protocol
UbiComp	Ubiquitous Computing
UDP	User Datagram Protocol

UMTS	Universal Mobile Telecommunication System
UNI	User Network Interface
VCs	Virtual Circuits
VCT	Virtual Connection Tree
VCNs	Virtual Circuit Numbers
VP	Virtual Path
WAN	Wide Area Network

Chapter 1

Introduction

In recent years we have seen a phenomenal growth of the Internet, especially since the advent of the World Wide Web (WWW). As mobile communication technologies mature, Internet usage is set to become more pervasive with mobile users being able to access the Internet anytime anywhere in the world. Mobile users are likely to remain connected for long periods and access services continuously. Telecommunication providers are beginning to deploy third generation mobile communication systems which aim to provide multimedia capabilities to mobile terminals. Some of the desired objectives of third generation networks include, high quality voice for indoor and pedestrian mode, enhanced packet and circuit wireless data, higher quality coverage, and support for high-speed packet data access [1]. Data services offered by telecommunication networks play a crucial role in gauging the success of providing Internet access to mobile users. To improve mobile services, a data access protocol called enhanced data rates for GSM Evolution (EDGE) has been proposed, providing data rates up to 384 kbits/s, starting from Year 2002, the universal mobile telecommunication systems (UMTS) will provide a significant improvement to cellular networks with features such as lower cost, higher capacity (2Mbits/s), global roaming and other advance services. As part of the enabling technologies to bring Internet content to mobile users, the wireless application protocol (WAP) has been developed by the WAP forum [2]. The WAP specifications define services and functionalities of a common application protocol stack that is designed to handle mobile network characteristics. BlueTooth [3] technology aims to interoperate BlueTooth-enabled mobile devices. The objective of BlueTooth is the replacement of all proprietary cables. For example, a user may send messages directly through his/her mobile device without explicitly linking his/her mobile phone. Also we could imagine users forming an ad-hoc network at a conference to obtain slides and papers without the need to form a fixed infrastructure. An important issue of consideration is to make mobile devices vendor independent.

Ubiquitous connectivity as envisioned by cellular companies is considered an important component of ubiquitous computing. The vision of ubiquitous computing (Ubicomp) [4, 5] was first proposed by Mark Weiser in the early 90s, is that computer usage will one day become a necessity in our daily lifes. Ideally, these computers are integrated into the physical environment and invisible to users [5]. As mentioned above, the current trend is ubiquitous connectivity. However, as the use of mobile computers become common, they will not be limited to accessing the Internet but will perform a wide range of capabilities. Household electronic devices can report to mobile devices carried by users that are required to interact with the environment. Mobile users administer various devices locally or remotely through their mobile devices. For example, a mobile device saves useful information ranging from location of car keys to stock quotes.

The convergence of telecommunication networks and the Internet, entails increased support for packet switched networks (i.e., IP) to route packets generated by mobile users and demands smooth interoperation of Internet protocols [6]. This convergence is in accordance with the aim of integrated services where all traffic types are transmitted on the same network resulting in cost reduction and smooth implementation of new services. However, one of the fundamental problems of bringing Internet content to mobile users is that widely deployed Internet protocols assume fixed hosts, which means current protocols operating over the Internet have to be upgraded. Furthermore, these protocols must be scalable to millions of mobile subscribers and adaptive to mobile characteristics. The update and augmentation of existing protocols introduce many problems and of particular importance is routing in mobile environments, both unicast and multicast. Current protocols such as the IP and TCP were optimized for fixed networks, and do not directly support the mobility of hosts. As a result, various problems have emerged due to the incorporation of mobile hosts (MHs). It will be shown in this thesis that existing protocols require modifications before they can handle mobility.

Mobile devices must work seamlessly over current network infrastructure. The incorporation of mobile devices in current network infrastructures requires new hardware and new protocols. Common entities that are involved include: base station (BS), MH and mobile switching centre (MSC)[1]. BSs act as gateways or access points for MHs such as personal digital assistants (PDAs) or notebooks equipped with wireless access (e.g., infra-red). Each BS is equipped with a transceiver and the coverage area of a transceiver's signal is called a *cell*. When a MH migrates into a *cell*, the MH establishes a radio connection with the BS that is managing the cell. Once the connection has been successfully established, the MH is free to transmit/receive data. The MH may have access to resources such as a printer in the foreign network given the appropriate privileges. When a MH visits a particular network, the MH is allocated a care-of-address by the BS. The care-of-address is then used by the MH in its outgoing packets and to identify the MH uniquely in the current network.

The process by which a MH crosses the *cell* boundary and establishes a new connection with the BS is called a handoff. There are three kinds of handoff: hard, soft and seamless[7]. In hard handoff, only one active data path exists when the MH switches from one BS to another. In soft handoff, the MH is able to receive data from the new as well as previous BS. Here, the MH has the option to choose a BS based on some criterion like lower signal to noise ratio. Seamless handoff is quite similar to soft-handoff where two connections are set up between the MH and the BSs. However, only one connection is active. In other words, the MH can only receive data on only one of the connections.

Figure 1.1 depicts how mobile devices are 'linked' to the fixed network infrastructure. In Figure 1.1, two types of wireless accesses are depicted, wireless LAN and cellular.

[1]In cellular networks

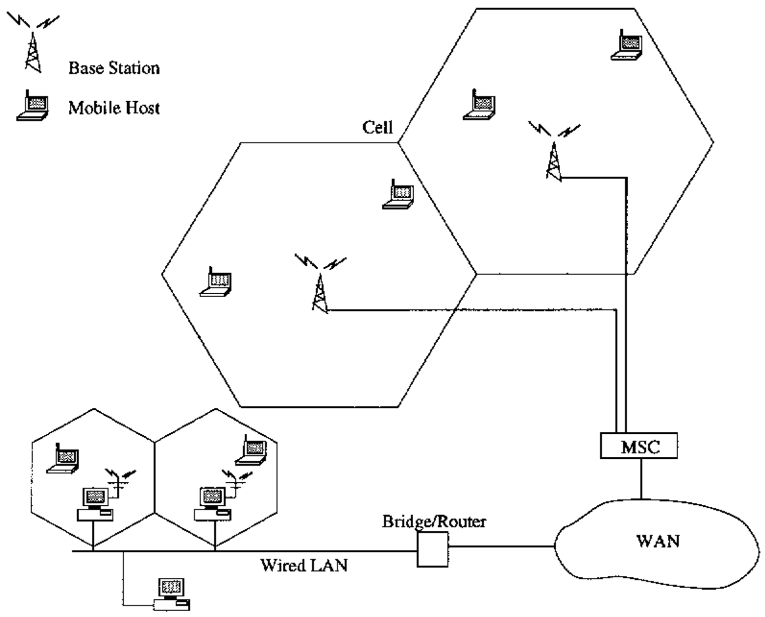


Figure 1.1: An example mobile network

Future wireless devices are likely to be equipped with multiple wireless interfaces. This allows for vertical handoffs [8] where the mobile user may use different wireless interfaces depending on his/her location. For example, infra-red maybe used when the user is at his/her desk but when he/she exits the office, radio is used.

Mobile users are likely to expect the same QoS that is available on their desktop computer for such services as electronic mail, file transfer, real-time gaming, video conferencing and video telephony. Such applications are dependent on the underlying mechanisms: unicast and multicast routing. In mobile networks the main problem in routing is the redirection of packets. The common scheme adopted for unicast communication is to have an entity called the home agent (HA) maintain the location of MHs under its care. The HA acts on behalf of the MH when it is away, whence packets destined for the MH are tunneled to the MH. In multicast communications, the multicast tree needs to be updated to reflect the mobility of group members. In connection-oriented networks such as ATM, connection(s) need to be rerouted and QoS renegotiated. In both multicast communications and protocols in connection-oriented environments, mobility poses a severe problem compared to unicast protocols in connection-less networks. This is due to the states maintained within the router/switches. These states need to be updated wherever a MH migrates. There may be a number of states involved, thus the update mechanism deployed may result in scalability problems. For example, as the number of multicast group members grow, the number of routers/switches maintaining states pertaining to the session increases as well[2]. Hence, the states maintained at the routers/switches need to be updated during each migration.

1.1 Active Networks

In ANs, routers/switches are programmable. The positive aspect of programmability is that the network can be viewed as a computational engine so that the network can be programmed to adapt to changes. Apart from the basic functionalities of routing packets, ANs also provide a computation space in which programs are executed. A new breed of protocols can be developed to take advantage of the computation space offered at routers/switches. The programs executing at the routers/switches are either permanently installed or they exist for the duration of the session. AN-based protocols aim to widen the solution space by allowing network-based control of protocol functions. It is envisaged that the above approach leads to simple and efficient solutions to network problems because the active routers/switches are better positioned to act dynamically. In other words, by viewing the nodes in the network as a collective computational engine, the benefits and problems that can be solved by deploying fully AN-based protocols over these active nodes can be investigated. This shift in paradigm cuts through the limitations of deploying protocols from the edge of the network. Now that the network is no longer viewed as a black box, a protocol designer need not probe the network to determine the current state it is in. The network adapts to the protocol when the state changes. Use of ANs results in significant reduction of signaling overheads. Moreover, AN-based solutions react faster to the changing dynamics of the network.

Once routers/switches are made programmable, three categories of protocols are envisaged to exist within these routers/switches: (i) default protocols, (ii) AN-based protocols, and (iii) enhanced protocols. An active node must at least provide default processing to allow backward compatibility with existing protocols. In (iii), the

[2]The growth rate is not one to one since a router/switch may handle N members

enhancements installed at routers/switches are transparent to the operation of conventional protocols. Conventional protocols do not specifically take advantage of the installed functionalities and operate as normal. However, active programs have the ability to process packets pertaining to a given protocol. Since AN-based protocols exist in the underlying computational environment(s), they may run for the duration of the communication session or be installed semi-permanently so that they can be reused in other sessions.

1.2 Thesis Objectives

In essence this thesis embraces two visions of future computing, namely UbiComp [4] and ANs [9]. ANs are central to the thesis objectives. This thesis points out fundamental problems in existing Internet protocols under conditions of mobility. Then the AN paradigm is used as a tool to overcome these problems. The objectives of this thesis are as follows:

1. To investigate the viability of the ANs paradigm in solving routing problems in connection-less (IP) as well as connection-oriented (ATM) networks.
2. To enhance/augment existing mobile aware protocols; the IETF mobile IP and connection rerouting in ATM networks.
3. To design and investigate AN-based multicast routing protocols to address the shortcomings of existing multicast protocols.
4. To investigate active program deployment to maximise the performance of protocols.

To the best of our knowledge, this thesis is the first to investigate the viability of ANs in enhancing routing and multicasting in mobile networks. As shown in later chapters, by installing simple programs to augment the operations of existing protocols, unnecessary mechanisms which result in increased complexity or overheads can be avoided. The main aim of this thesis is to show that ANs provide a new paradigm for solving problems in realising the UbiComp vision.

To carry out these investigations, problems pertaining to routing in mobile networks were identified. By determining how routers/switches can be programmed dynamically in accordance with the operation of the protocol, additional features can be added at the routers/switches to mitigate the effects of mobility. By designing new AN-based protocols which execute at the computation space for the duration of the communication session, protocols can be made to adapt to varying network conditions. For unicast protocol, this thesis shows how existing protocols can be enhanced by augmenting their operations transparently. For multicast, new AN-based protocols were developed and these protocols exist for the duration of the communication session and they adapt to subscriber migrations.

1.3 Contributions

These research reports on four new contributions pertaining to ANs and their application to mobile communications: Also, there are numerous contributions specific to the investigated protocols. The next subsection summarise the contributions of this thesis.

1.3.1 ANs and Mobile Communications

The main contributions resulting from the investigation on the application of ANs to the problem of routing in mobile communications are:

1. *Use of programmability in routing to MH.* To the best of our knowledge, little research has been done to investigate the application of ANs in mobile communications, specifically routing. The work presented here is novel in that it considers the use of active routers/switches to solve problems related to routing in mobile communications.

2. *AN-based solutions and also augmentation of existing protocols.* The programmability paradigm provided by ANs is shown to be effective when applied to routing in mobile communications. This has been validated with AN-based solutions for both unicast and multicast protocols. The main contribution here is that, the active solutions developed in this thesis take advantage of all nodes in the network rather than strategically placed programmable nodes in the network. Therefore, this work provides an insight into how this new-breed of solutions can be developed to tackle mobility.

3. *Reuse of states.* ANs are applied in problems that maintain a significant amount of states information (e.g., soft state) in the network, for example, connections in ATM networks and multicast routing. In the mobile environment, due to frequent host changes, an efficient update mechanism is required and ANs provide such a paradigm that allows for efficient solutions. This aspect is crucial in third generation networks where there may be millions of users. Therefore, ANs can aid in the scalability of Internet protocols.

4. *Integration of active programs and protocol operation.* A model to deploy active programs along side signaling messages inherent in a given protocol is outlined. The implication is that positioning of active programs either manually or through computation is unnecessary. The active programs are deployed only on paths in which the packets of interest flow. Most important, the deployed programs react in accordance to events. These programs are dynamic in that the role changes from one node to the next, adapting to mobility.

1.3.2 AN-Based Solutions

Due to the investigations carried out, various contributions to the problem areas have been made. These contributions (by chapter) are:

- *Chapter 4.* The IETF mobile IP [10, 11] protocol is further improved with ANs. These enhancements are: (i) efficient delivery of location update messages using a multicast like scheme, (ii) use of SmartBuffer to shield TCP from long handoff delay, (iii) use of BusStation framework which allows the transport of aggregated messages. The framework has been demonstrated to reduce the number of location update messages in the network.

- *Chapter 5.* In the AN-based solution proposed in this chapter, (i) there is no need to make a trade off between path reuse and path optimality, (ii) automatic removal of loops without requiring intervention from end-hosts, (iii) paths are optimized incrementally when needed to maximise path reuse and optimality.

  In mobile ATM networks, existing connection rerouting algorithms need to make a tradeoff between signaling overheads and path optimality. Here, no such trade-off is necessary. Moreover, no network probing is required, thus signaling overheads are reduced.

- *Chapter 6.* Current solutions concerning IP multicasting in mobile networks suffer from inefficient routing and tunnel convergence. This chapter presents an AN-based solution that (i) avoids the tunnel convergence problem, (ii) incurs minimal signaling and update overheads and, (iii) is scalable to increasing group members.

- *Chapter 7.* In this chapter a solution to the problem of rerouting multicast connections in ATM networks is presented. An AN-based solution presented in this chapter, called **M**ulticast **Co**nnection **Re**routing (MCoRe), contributes as a seminal work in the area of rerouting multicast connections in mobile networks. M-CoRe is based on an adaptive source-rooted-tree that minimizes network latency. MCoRe features include, (i) efficient rerouting of multicast connections, (ii) loop prevention, (iii) preservation of cell ordering, (iv) minimal cell loss and (v) low signaling overheads.

1.4 Thesis structure

Thus far in this chapter a brief introduction to mobile networks and the scope of this thesis were presented. The following provides a brief description of the chapters to follow:

- *Chapter 2.* This chapter reviews existing protocols covered in this thesis and highlights their limitations. Two categories of protocols are covered: connection-less and connection-oriented. This chapter gives an overview of routing and multicasting in mobile IP as well as ATM networks. Literature review on the performance of TCP over mobile networks is also presented.
- *Chapter 3.* This chapter presents the vision of ANs. AN based solutions and related architectures are reviewed. Finally, the rationale for using the ANs paradigm in this thesis is discussed and detail arguments on the application of ANs in mobile computing environment will be presented.
- *Chapter 4.* In this chapter, the enhancements to the IETF mobile IP models with active routers are presented. This chapter shows how the mobile IP models can be improved significantly by augmenting the operation of mobile IP models with active programs.
- *Chapter 5.* This chapter is concerned with the application of ANs to the problem of connection rerouting in mobile ATM networks.
- *Chapter 6.* The previous two chapters were concerned with unicast protocols. This chapter covers AMTree, an active multicast protocol for IP networks.
- *Chapter 7.* In this chapter, multicasting in ATM networks is considered. Also a set of requirements are presented which serve to provide design guidelines for subsequent work in the area of rerouting multicast connections in ATM networks.
- *Chapter 8.* The thesis is concluded with a summary of the outcomes of the research work. Open research issues that arose from the investigations carried out are also discussed.

Chapter 2

Protocols over Wireless Data Networks

The scope of this thesis was introduced in the previous chapter. Further, a brief introduction to the entities within a wireless data network were presented. In this chapter, routing protocols developed specifically for mobile hosts are reviewed. Two network paradigms will be considered, IP and ATM. In each of these paradigms, protocols related to unicast and multicast routing will be presented. Specifically, IETF Mobile IP [10, 11], connection rerouting in ATM networks, and multicasting in both ATM and IP networks.

This thesis encompasses connection-less and connection-oriented environments in the areas of rerouting unicast connections and multicasting. Two cover these two areas, this chapter is organised as follows. Firstly unicast routing in mobile networks will be presented. Section 2.1 presents unicast routing in IP networks followed by routing in mobile ATM networks in Section 2.1.2. After unicast protocols related to this thesis have been reviewed, later sections will present literature related to multicasting. The literature concerning IP multicasting in mobile networks are reviewed in Section 2.2.1 followed by multicast in mobile ATM networks in Section 2.2.2. Finally, a summary of the chapter is presented.

2.1 Unicast Routing in Mobile Networks

2.1.1 IP and Mobility

Mobile IP research is sparked by the need for a protocol which supports mobile devices whose point of attachment changes frequently. Current protocols such as IP assume that hosts are stationary and have fixed addresses. Therefore, they cannot provide transparent support to mobile devices such as mobile computers. Current protocols require changes to a MH's address, modifications to a number of configuration files and restart of all communication sessions when moving from point A to B. These processes are time consuming and error prone [12]. The above problems stem from the use of hierarchical addressing and routing schemes which were designed with stationary hosts in mind [12]. According to Bhagwat et al. [13], there are two possible solutions to mobility; (1) redesign the current internetworking protocols or (2) provide additional services on top of existing protocols. The first solution requires significant modifications which are infeasible in the current scenario. Issues such as transparency above the network layer and security need to be resolved with the second option. A MH must provide the impression of a stationary host and its point of attachment should be hidden from protocols and applications [13]. The main goal of mobile IP research is to introduce transport layer transparencies and to eliminate the need for reinitialisation when the correspondent host (CH) changes its point of attachment [14]. Another goal is route-optimisation. Route optimisation requires caching the MH's *care-of-address* and updating it when the MH changes its point of attachment. Work covering caching in intermediate nodes can be found in [15, 16]. Myles et al. [17] outline two main goals of mobile IP; operational and performance transparency. Operational transparency implies that the user is not required to perform any special actions during host migration. As for performance transparency, it is concerned with optimum routing, efficient and robust migration, and efficient use of network resources. Other issues include cost, host impact, infrastructure requirements and backward compatibility.

2.1.1.1 Solutions

Myles et al. [18] and Perkins et al. [19] describe the Internet Mobile Host Protocol (IMHP) which supports route optimisation and authentication. The working of this protocol is similar to the proposed IETF Mobile IP [10] standard. The only difference is that optimal routing is considered as an extension in the IETF's proposal. Myles et al. [18] propose a near-optimal routing scheme, the MH's *care-of-address* is cached at intermediate nodes and the previous foreign agent is notified of the MH's *care-of-address*. Work on mobility support in IPv6 is proposed by Perkins et al. [20] who incorporate cache update for optimal routing. The difference between IPv4 and IPv6 is that nodes learn and cache care-of-address to enable them to send packets directly to the MH using IPv6 Routing Header. Updates to nodes are done through IPv6's *IPv6 Destination options* namely Binding update and Binding Acknowledgement.

Johnson [15, 21] proposes the Mobile Host Routing Protocol (MHRP) which is scalable and uses MHRP source addresses for cache updates. The use of list-of-source-addresses field for solving route looping coupled with cache invalidation and updates make MHRP a robust scheme. Location updates are sent when the destination node (e.g., FA) detects that a CH does not have the MH's care-of-address. If a MH is at home, the MH needs to inform all CHs communicating with it to discard their cache bindings. The use of a mediator node for supporting MHRP for hosts which do not support MHRP leads to little modifications at hosts and MHs above the IP level. Ioannidis et al. [22] use a set of Mobile Support Stations (MSSs) to facilitate mobility. Each MSS is in charge of a cell. During handoff, the MH notifies its previous MSS of the new MSS's address. If a packet is tunneled to a MSS which does not have the intended MH, a broadcast is made to all MSSs and the MSS serving the MH will respond. Performance and implementation details of the protocol are outlined in [22] and [23]. The performance issues presented are: cell switch latency, MSS load, IP-in-IP overhead and space complexity, fault tolerance, load sharing, and scalability. Their approach is limited to a campus wide environment and is not scalable. Perkins et al. [14] use IP's Loose Source Route (LSR) option to solve the mobility problem. LSR enables each packet to have enough routing information for the remote host to send a reply back along an optimal path. Problems with the LSR approach are as follows, (1) not all implementations of protocol stacks support LSR (2) non optimal processing of packets at routers (3) no LSR support for UDP packets and (4) possible compromise of security [17] [23] [13]. Myles et al. [17] provide extensive comparison of mobile IP research from IBM, Sony and Columbia based on issues such as backward compatibility, optimum routing, Intra-cell communications, migration procedures, packet losses and processing bandwidth.

In [24], the notion of *friend* networks is used to cache *care-of-addresses*. *Friend* networks are basically trusted networks with a mobile support router. Several schemes for cache update such as selective update, complete update, lazy update and no update are investigated. Results show that in the long run, selective updates incur the least cost during host migration. Selective update is a scheme where only *friend* networks get location updates. The lazy scheme needs less location updates because queries are only made when no binding is found.

Bhagwat et al. [13] provide a generic view of mobile IP by identifying fundamental entities which must exist within any mobile IP protocol. Baker et al. [25] eliminate the use of FA for supporting mobile hosts. Instead, MHs are equipped with FA functionalities. This means that little modification is needed to the current infrastructure and MH is able to function in networks which do not have a FA installed. On the other hand, the complexity and functionalities of the MH increase where it has to support encapsulation and decapsulation of IP-in-IP packets and also obtain a temporary IP address at the remote site. Moreover, optimal routing issue is not addressed. A different approach to handling mobility is outlined in [26]. Badrinath et al. [26] approach requires the MSS to be mobility aware. MSS acts as proxy for MHs where it provides the main communication needs for MHs. CH sees the MSS as the MH because the MSS binds the MH's IP address during communication with the CH. The main drawbacks of Badrinath et al. [26]'s approach are: no end-to-end semantics, need for application relinking, software overhead when data copying between threads at the MSS and handoff latency [27]. In [28], Okanoue et al. aim to improve on the IP-in-IP encapsulation model. The main idea is to append an additional header containing the MH's care-of-address. Also intermediate nodes are able to process the encapsulated datagrams. Their scheme achieves efficient routing path and requires low mobility binding maintenance.

A novel approach (MSM-IP) was proposed by Mysore et al. [29] where IP multicasting is the sole mechanism for addressing and routing packets to MHs. MSM-IP differs from IETF Mobile IP in the following ways:

- The MHs in MSM-IP uses class D addresses.
- Since multicast is the delivery protocol, multicast router(s) must be present at every subnet to support multicast delivery.
- Unlike Mobile-IP where the HA stores the bindings of MHs, MSM-IP uses a distributed directory.
- Finally, MSM-IP is able to provide seamless handoff because multiple BSs in the area can subscribed to the multicast group.

Some remaining problems that have yet to be resolved are (i) interactions with the address resolution protocol (ARP), (ii) modifications to TCP to accept destination as multicast address, and (iii) inadequate handling of IP multicast resource discovery and location update with regard to location management. Moreover, MSM-IP may have scalability problems given that each MH is assigned a class D address. Then in [30], the performances of MSM-IP was presented. Mysore et al. [30] showed that MSM-IP is effective in supporting seamless handoff and showed improvements in performance compared to conventional mobile IP.

A scheme combining forward pointers and caching to achieve low cost binding and delivery operations was proposed by Bejarano et al. [31]. The scheme involves the maintenance of a chain of forwarding pointers (HA to MH). Each pointer covers a geographical area and these chain of pointers are recalculated after each migration. The CH is notified of the nodes in the forwarding pointer and when the CH wants to send a packet to the MH, the message is forwarded to a chosen node. The intercepting node forwards the message to the next downstream pointer. The downside of this approach is that both CH and MH need to be aware of the existence of pointers. Further, a fairly complicated maintenance protocol is used to maintain the validity of pointers. Moreover, an optimal path to the MH can not be ensured due to the placement of pointers. Besides that, each pointer needs to monitor its parent pointer node to ensure the upstream node is alive. If the parent node dies, the ancestor node is notified to mend the chain of pointers.

IETF Mobile IP

IETF Mobile IP [10] augments IP to support MHs by providing a transparent scheme to route datagrams to MHs. IP was not designed to handle mobility because it assumes that end-hosts are fixed and their addresses do not change frequently. The Mobile IP specification solves the IP routing problem by identifying a MH by its home address. When the MH migrates from its home cell, the HA is notified of the MH's current point of attachment. This is done by sending a mobility binding to its HA which associates the MH's address to a care-of-address. The care-of-address is usually the address of the foreign agent (FA)[1] or a co-located address. When the MH is in a foreign network the HA acts as a forwarding agent for the MH. The HA intercepts packets destined for the MH. These packets are then tunneled to the MH's care-of-address. Tunneling is achieved by encapsulating the IP packet destined for the MH within another IP packet, hence the term IP-in-IP [32] is also used. When the FA receives an IP-in-IP packet, the packet is decapsulated and the FA checks whether the MH is in its visitor list, if so, the packet is sent to the MH, otherwise the packet is discarded [10].

Johnson [11] defines enhancements to Mobile IP (referred to as optimised Mobile IP in this thesis) in the form of extra binding update messages generated for the CH and the MH's previous FA. In optimised Mobile IP, the handoff commences when the MH at the foreign agent sends a registration request in the same manner as defined in the Mobile IP specification. The HA, in addition to the registration reply, sends a binding update to the CH notifying it of the MH's new care-of-address. This enables the CH

[1]The FA is a base station acting on behalf of the MH.

to tunnel packets directly to the MH thus bypassing the HA. The MH also sends a binding update to its previous FA. The previous FA tunnels any packets for the MH to the MH's new location.

Improvements to IETF Mobile IP

As mentioned previously, in optimized Mobile IP [11], MH informs the previous FA of its care-of-address. However, sending binding updates to the previous FA is not beneficial if the data packets precede the update [33]. Consider the case where a MH migrates when a train of packets is just about to arrive at the FA. Assuming the packets are not buffered at the FA, these packets will be lost due to MH migration, making the update useless. Perkins et al. [33] have presented a solution in which buffering is performed at the previous FA to catch in-transit packets, and such packets are tunneled to the old FA prior to the arrival of the binding update at the CH. After the binding update arrives, the previous FA forwards the buffered packets to the MH's current location. A scheme to reduce Mobile IP's handoff latency with hierarchical foreign agents was proposed by Perkins [34]. Basically, the idea is to send registration requests to a local/regional FA and have the FA respond with a registration reply message. The hierarchy of FAs is maintained as follows. The MH is informed of the hierarchy through advertisement messages transmitted by the FAs. In each advertisement message a list of care-of-addresses are included which represent the hierarchy of FAs. Therefore, as the MH changes its point of attachment, it can compare care-of-addresses obtained from the previous BS and the current BS. Once the MH has determined the lowest level of the hierarchy, the MH notifies the FA at the next higher level of its new care-of-address. This scheme requires the maintainence of the FA hierarchy. A subsequent proposal that aims to reduce the number of registrations directed at the HA and also lower handoff latencies is presented in [35]. The main idea is to use a gateway foreign agent (GFA) where the HA is notified of the GFA's address instead of the MH's address. Subsequent registration requests are directed at the GFA as long migrations are within the domain managed by the GFA. Later in [36] El Malki et al. extended the hierarchical FAs to perform seamless handoff. In their scheme a mobility agent uses multiple bindings and the MH takes an active role in determining which FAs maintain multiple bindings. As a result, packets destined to a given MH are multicast to a set of BSs, during handoff.

A scheme called cellular IP [37] also aims to reduce the high handoff latency of Mobile IP. Cellular IP is targeted for LAN environments and defaults to Mobile IP in the wide-area case. In cellular IP, each node within a LAN maintains mappings of MHs' current location. The mappings stored consist of the mobile's IP address and the interface which leads to the MH. Therefore, any packets destined to the MH are routed hop-by-hop to the MH. These mappings are refreshed by route-update packets or packets transmitted from the MH itself. MHs that are not actively communicating create a page cache mapping which is maintained at specific nodes only, for example the gateway of the LAN. This enables MHs that do not have a route cache to receive packets. After handoff, new mappings are created at network elements by packets transmitted from the new BS. Since network elements that are on the path to the old BS contain mapping of the MH also, packets will be directed to both BSs until their timer expires. In the case where the MH does not have anything to send after handoff, the MH is required to send route-update packets to create or update the necessary mappings.

An idea that considers QoS is presented by Ramjee et al. [38] where during handoff, the path is extended from the previous BS to the current BS. A crossover router, is at the intersection between the path from the root router to the old BS, and the extended path. Therefore, after handoff, packets may be forwarded from the previous BS as well as from the crossover router. With the above set up, the path from the source to the crossover node remains unchanged, only the QoS from the crossover node to the MH's new location needs to be re-negotiated. The idea is similar to those of connection rerouting in ATM networks which is explained in Section 2.1.2. In [38], the MH sets up a path to a given domain's root router.

Tan et al. [39] proposed a fast handoff scheme based on the following ideas. Firstly, a resident domain FA hides mobility within the given domain. Secondly, multicast is used to ensure seamless handoff. The main limitations of their approach is the usage of multicast which incurs bandwidth and buffering at neighboring BSs. Furthermore, the cell layout must be known for each BS to inform adjacent BSs to subscribe to a given multicast group. Hence the scheme proposed by Tan et al. is not scalable since the number of multicast sessions within a LAN is proportional to the number of visiting MHs. The use of multicast mandates support for multicast at each foreign network.

2.1.2 ATM and Mobility

Unlike connection-less networks where each packet is self-contained, cells in ATM are switched along a dedicated path. Care must be taken to ensure that cell losses are minimal and most importantly cell ordering is preserved. Moreover, the QoS previously allocated must be maintained. Hence connection rerouting is crucial in ensuring seamless handoff.

Generally, the main operation of any handoff protocol in ATM networks is the connection rerouting process. Note that some schemes such as those in [40] avoid the rerouting process by assuming a static crossover switch, but may suffer from suboptimal path. As part of the rerouting algorithm, a crossover switch (CX) needs to be located. The algorithm to locate the CX contributes significantly to the observed handoff latency. Moreover the location of the CX determines the extent of original path reuse, buffer required and optimality of the resulting path. The next paragraph explains the problem of connection rerouting using an example. Then different categories of connection rerouting algorithms are presented in the next section.

To present the connection rerouting problem consider Figure 2.1. In this example, BS_1 denotes the previous BS or current BS and BS_2 denotes the BS the MH is handing off to. The geographic cell is represented in italics (*cell*) and the default font represents a message cell to avoid any confusion. In the network shown in Figure 2.1, the fixed host (FH) has an established connection to the MH. When the MH migrates across *cell* areas, the connection needs to be updated. Referring to Figure 2.1, the two possible switches as the MH migrates from *Cell 1* to *Cell 2* and again from *Cell 2* to *Cell 3* are CX_1 and CX_2 respectively. When the MH migrates from *Cell 1* to *Cell 2* a new connection is setup from CX_1 to *Cell 2*. After the connection has been updated at CX_1 any cells destined for the MH are forwarded onto the new path. Before the BS forwards

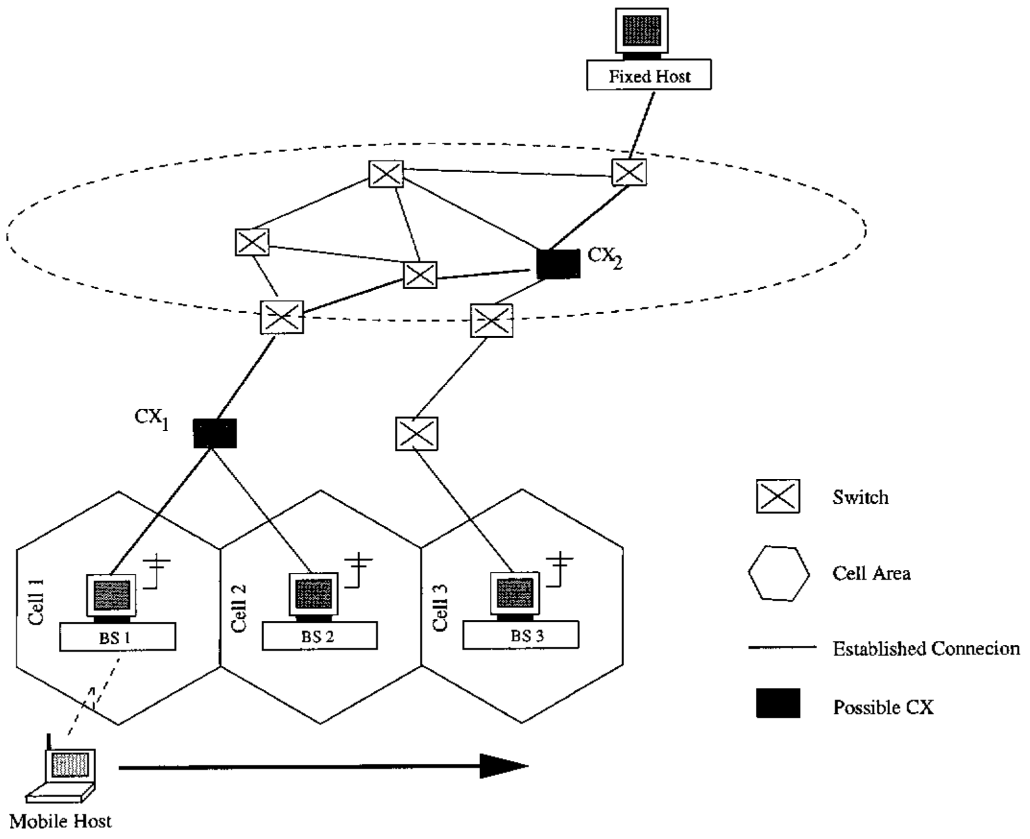


Figure 2.1: Example of mobile ATM

any cells from the new path, cells from the old path need to be drained. Once the last cell from the old path has been forwarded, cells from the new path are forwarded to the MH. Buffering is required at the CX or BS during the connection rerouting phase because preservation of cell ordering is crucial in ATM networks. The size of the buffer is dependent on the traffic rate and the length of the path to be drained. The issues of cell loss and QoS are dependent on the application. For example, time sensitive applications are tolerant to loss and are adaptive to varying QoS after each handoff. Once the cells are drained the connection from CX_1 to BS_1 is torn down. This process is repeated at CX_2 when the MH migrates from *Cell 2* to *Cell 3*. Usually during the connection rerouting process, a CX discovery algorithm is used to determine the CX which satisfies a given performance metric. A tradeoff to be consider in connection rerouting is between path reuse and optimal path. By maximising reuse states, that have already been allocated can be reused. However, maximising reuse may not result in an optimal path. Another important metric is QoS. QoS needs to be taken into consideration if the chosen CX is unable to support the requested QoS.

2.1.2.1 Solutions

Work on rerouting in wireless ATMs can be broadly categorised into the following:

- *Full Path Extension* [41]: In this approach, the original connection is extended from the old BS to the new BS resulting in low latency handoff and minimal cell loss. The main assumption is that BSs are connected by pre-established virtual circuits (VCs) which allow fast forwarding. The main limitations are increasing end-to-end delay of the route and looping. Therefore a path optimisation is required to address these limitations.

- *Incremental re-establishment* [42, 43, 44, 45, 46]: This class of methods involve invoking a CX discovery algorithm during handoff. Handoff latency is affected by the discovery algorithm, update time of translation table at the CX and connection time between the BS and MH. This class of algorithms require buffering at the CX or BSs to avoid cell loss and cell misordering. The algorithm used can be one of Prior Path CX Discovery, Distributed Hunt CX or Backward Tracking CX Discovery as discussed in [47].

  Akyol et al. [43][45] use a scheme called *Nearest Common Node Rerouting* (NCNR). In Akyol et al.'s scheme a switch common to both zones (old and new zones) is chosen as the CX. The CX then forwards cells to both zones until the handoff is finished. NCNR works well in a hierarchical topology and time-sensitive traffic (e.g., audio) is not buffered. Buffering is used to avoid cell loss and multipoint connections from the CX are used to preserve cell sequence for throughput dependent traffic (e.g. data transfer).

  Bejerano et al. [46] proposed two rerouting algorithms. Both of their algorithms are based on finding the minimum cost path. In the first algorithm, after handoff a comparison of setup cost is made between the extended path and the direct path from the CH to the MH's new location. If the extended path is α ($\alpha \geq 1$) times the cost of the direct path, then a new connection is created directly from the corresponding host to the MH. Otherwise the extended path is taken. In the second algorithm, after handoff the closest switch (on the original path) to BS_2 is chosen as the CX. Then the cost of alternative paths from the MH's current location to the CH is determined. Finally the path with least cost is chosen. Their analysis only included cost of setup and holding of resources. In their experiments they show that connection re-establishment has the lowest hold cost (time in which resource are held up), but high setup cost. The converse is true with the minimal path update algorithm [48]. As for the proposed algorithms, a balance is achieved between setup and hold costs. Issues that remain to be addressed are, prevention of cell loss, cell ordering, signaling cost, information required about the network topology and buffering requirements are not addressed in their work.

- *Path splicing* [49]: Path splicing is invoked by the new BS. A path is established from the new BS to the FH. If the path coincides with the original path, then the CX will splice the new path onto the existing path. However, path splicing does not guarantee that the resulting path is optimal.

  Archarya et al. [49] use path splicing. The CX discovery algorithm used involves sending messages back to the FH. Buffering at the BSs is used to prevent cell misordering. Undelivered cells at the previous BS are forwarded to the new MH via BS_2. Cells arriving from the CX are buffered until the previous BS finishes forwarding.

- *Tree Routing or Multicast Based* [50][51]: This class of algorithms aims to reduce the handoff rate by clustering BSs onto a root node. Each BS has a preestablished connection to the root node. During cell forwarding the root node forwards the cell onto the correct path. This scheme prevents cell misordering and cell loss through multicasting but suffers from cell duplications. The main limitations include: existence of cell duplicates, excessive buffering and pre-reservation of bandwidth [47]. Furthermore, this scheme may not be suitable for asymmetric traffic such as video delivery [52].

  Ghai et al. [51] use a supervisory host (SH) which groups clusters of cells. The SH acts as a gateway between the wired and wireless networks and is responsible for multicasting cells to a group of *cell* areas, tracking MHs and maintaining connections. Such a scheme suffers from poor scalability due to the location management algorithm used.

Acampora et al. [50] introduced the Virtual Connection Tree (VCT). VCT is similar to the SH approach, but the amount of responsibilities at the root node of VCT are greater. In the VCT scheme, VCs are prestablished to each BS. Upon entering a VCT, MHs are given virtual circuit numbers (VCNs) for each BS in the cluster. The MH then uses these VCNs to communicate with the root node and the root node translates these VCNs to the VCNs used in the fixed network. In such a scheme, preestablished connections are used and new connections need to be setup if the MH migrates across VCTs, requiring additional overhead.

- *Anchor Rerouting* [53, 40, 54]: In anchor rerouting a designated switch at the edge of the network acts as the CX. In other words, no CX discovery is required, and a new connection can be made to the new BS. This results in faster handoff but the resulting path is not optimal. Chen et al. [53] use an anchor based scheme. The location of the anchor node is predefined and its location is maintained by the mobile switching centre (MSC).

- *Full Connection Re-establishment* [50][42]: In full connection restablishment, the original connection is torn down and a new connection is made to the MH's current location. The main disadvantages are high handoff latency due to signalling overheads and poor performance at high handoff rates. On the other hand, an optimal path is guaranteed.

  In the VCT [50] approach a full re-establishment is needed once the MH moves out of its VCT. Keeton et al. [42] propose the use of full path re-establishment while cells are being forwarded from BS_1 to BS_2. Their approach is a combination of path extension and full path re-establishment, the use of predictive handoff was also discussed.

- *Hybrid* [55, 56]: Ramjee et al. [56] use a combination of path extension and distributed connection server scheme for CX discovery. They showed that a simple handoff is sufficient. Path extension is used during handoff. Later the connection server is queried and the server determines the CX along the path. In [55], a two state connection rerouting was used. To prevent looping, the MH is required to keep track of visited BSs.

A different approach was taken by Dommety et al. [57] for connection rerouting. They propose two optimization schemes based on PNNI [58] to optimize the route of a connection, namely *optimal* and *minimal* crossover node determination. In the *optimal* scheme, the hierarchical addressing structure of PNNI is used. For example, if both the old and new BSs are in the same peer group then the CX is the ingress border node of the peer group in which the BSs reside. On the other hand, if the CH and old BS are both located within a peer group but the new BS is located in another, then the CH is designated as the CX. Unlike the *optimal* scheme, the *minimal* crossover determination scheme requires modifications to PNNI. In the *minimal* scheme, the CX is determined as follows. Given, all concerned nodes are on the same level, the old BS uses the topological information from the underlying link-state routing protocol to determine the switches from the CH to both old and new BS. Once the routes are determined, they are compared and the intersecting switch between the two paths is designated as the CX. In the general case, where each party is at a different level, a crossover peer group is determined using the *minimal* scheme described above. At the egress border node of the crossover peer group, a *minimal* scheme is invoked again to locate the CX (switch or peer group) within the crossover peer group. The CX discovery process continues recursively until a CX is found. To date no performance studies have been presented to determine the impact of *minimal* and *optimal* algorithms on handoff latency, cell loss and cell ordering.

A comprehensive study on CX discovery algorithms is presented in [47]. Various CX discovery algorithms were investigated and each algorithm is evaluated based on its convergence hop count, circuit reuse, resulting new path length and efficiency. The *Prior Path* and *Distributed Hunt* CX discovery algorithms were found to perform the best. *Prior Path* discovers the CX by querying a connection server for the switches along the path. The new BS calculates the shortest path to each of these switches and selects the switch with the minimum number of hops to BS_2. *Distributed Hunt* is similar to *Prior Path*, instead of querying the connection server, each switch maintains a connection table. After handoff, a broadcast is made and switches along the route answer the broadcast. These algorithms are used in Toh's [48] hybrid handoff scheme. Performance studies of various connection rerouting described above can also be found in [59, 60, 61]. In [59], Banh et al. provide analytical comparisons of the rerouting schemes listed above. Performance parameters measured are handover delay, communication disruption delay, buffer requirements, complexity (number of signalling messages involved) and bandwidth requirements. Banh et al. showed that multicast rerouting techniques have lowest handoff latency but at the cost of high complexity and bandwidth. Then in Wong et al. [61]'s work, different schemes for performing path optimization were investigated. Their investigations where motivated by the following questions: (1) how to minimize disruption time during path optimization and (2) when and how often should optimization be performed. The probability distribution investigated are exponential, periodic, and Bernoulli. The above mentioned distributions determine the frequency in which optimization is performed. For example, connection rerouting is invoked periodically. It was found that invoking connection rerouting using the Bernoulli distribution yields the lowest cost per call.

2.2 Multicasting in Mobile Networks

The following subsections present work related to multicast in mobile networks. As in the unicast case, the literature review is divided into two subsections, namely IP and ATM networks.

2.2.1 Problems with IP Multicasting

Current multicast protocols such as distance vector multicast routing protocol (DVMRP) [62], multicast OSPF (MOSPF) [63], core based tree (CBT)[64] and protocol-independent multicast (PIM) [65] are designed with static host in mind and hence are prone to problems in mobile networks. Typical problems with multicasting in mobile networks are:

1. After migration, multicast protocols that are based on shortest path tree such as DVMRP and PIM-sparse mode may route packets incorrectly or drop packets due to reverse path forwarding.

   Consider Figure 2.2 where a source-rooted tree is constructed using reverse path forwarding. The source of the multicast session is mobile. The arrows in Figure 2.2 show the direction in which a source's packets are expected to arrive. When the source MH migrates, any packets sent from the new care-of-address will be discarded due to reverse path forwarding. This is because in reverse path forwarding the multicast router expects packets to come from an upstream interface. Since routers expect packets to arrive on the shortest path from the source, packets originating from the new location are dropped.

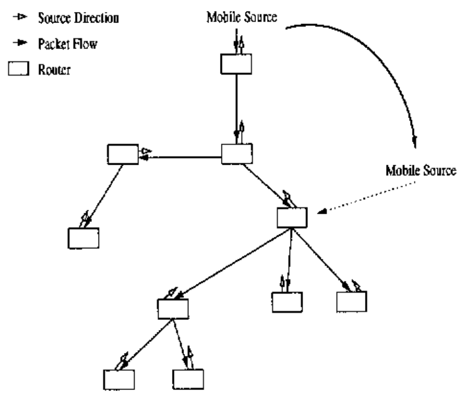


Figure 2.2: Problem with source migration in a shortest-path tree.

2. In shared-tree approach such as PIM [65] or CBT [64], an algorithm is needed to determine core(s) or rendezvous point (RP) strategic location(s) in the network and ensure minimal delay at the receivers. The core's location is usually selected at the start of the multicast session. In mobile networks if sources/receivers are mobile then a core's or RP's position will be sub-optimal after each receiver/source migration. What is needed is an algorithm to relocate the core (dynamic core), but such algorithm(s) may prove to be infeasible due to high signaling messages involved and multiple constraints that have to be satisfied due to host movement.

3. At the receiving end, when a MH migrates to a cell[2] with no other group members, it will experience delay. This is mainly caused by subscription delay, tree rebuild or non-existent multicast routers in the region.

   When a receiver migrates, three issues need to be considered. Firstly, the foreign network may not support multicast service. Therefore, the receiver is unable to rejoin the multicast session until it migrates to a network which supports multicast. Secondly, the foreign network may support multicast service but does not join the multicast group(s) in which the visiting MH is subscribed to. Hence the MH has to wait for Internet Group Management Protocol (IGMP)[66]'s next membership query cycle (at most one request per minute) and endure rejoining delay. Thirdly, in case where the foreign network has joined the multicast group, the receiver may receive duplicate packets or only subsequent packets.

4. The time to live (TTL) specified may be inappropriate. For example a TTL set for one region maybe inappropriate for another. Once the MH migrates out of a region, the specified TTL value earlier maybe too small. Further, since multicast groups are not unique, two MHs belonging to two different groups using the same address may receive unwanted packets [67]. Another problem outlined by Chikarmane et al. [67] is the use of a multicast address such as 224.0.0.1, which means "multicast to all hosts on this LAN". There are three interpretations for delivery: (1) all hosts on the LAN including visitors, (2) all hosts belonging to LAN only, or (3) all hosts on the LAN and those that are away. If choice (1) is chosen, then messages will be transmitted to a visitor leading to possible unwanted distribution of information. The opposite happens in choice (2), where all hosts that are away do not get transmitted packets. Finally for choice (3), when the packets are tunneled to MH, that are currently away, care has to be taken to ensure that the tunneled multicast packets are not multicast to all hosts in the foreign network, since the packets' address is 224.0.0.1.

[2]Cell here means radio coverage area

2.2.1.1 Solutions

Archarya et al. [68] extended Columbia's version of Mobile IP [22] where unicast tunnels are setup between mobile support routers. In other words a virtual link layer connectivity is established that enables DVMRP to work properly. This work was then extended in [69] where mechanisms for resolving duplicate packets and delivery of in-sequence packets were proposed.

The methods proposed in the IETF Mobile IP [10] specification are as follows:

- *Bi-directional HA tunneling.* The idea for a bidirectional tunnel between the MH and its HA was presented initially in [70]. The HA becomes the source/receiver of multicast traffic. Whenever the MH migrates to a new subnet a bidirectional tunnel is created from the MH's care-of-address to the HA. As a result, any traffic generated by the MH or directed towards the MH has to traverse through the HA. The tunnel is then used to send packets going to/from the MH. From the receivers point of view the source never left its subnet at all. Obviously this incurs a high handoff latency as the MH moves further away from the HA and introduces a problem called tunnel convergence. This is where multiple MHs are serviced by one foreign agent (FA) and some/all MHs have different HAs. As a result, each HA will have a tunnel to the FA. To solve the convergence problem, Wang et al. [71] and Harrison et al. [72] have looked at ways of reducing tunneling costs. Harrison et al. [72] investigated different policies for choosing the FA for multicast transmissions. The convergence problem is solved by having only one bidirectional tunnel to the designated HA. Wang et al. [71] then extended the work of [72] using an architecture consisting of multicast agents (MAs). Basically their idea is to have MAs serving a unique set of FAs, therefore the HA is required to tunnel only to these MAs in the network. In the former, a tunnel is setup directly to the MA and no algorithms are employed for choosing a designated HA.

- *Remote subscription.* A care-of-address is allocated when the MH is at a foreign network. The care-of-address is then used for multicast. Alternatively, the FA can subscribe to the multicast group on behalf of the MHs it is managing.

The disadvantage of subscribing at a remote side is that if a source-rooted tree is used, the tree needs to be rebuilt from the new location (source scenario). On the receiver side, the foreign network may not provide multicast service. Although end-to-end latency maybe lower, the handoff latency may be much higher because of the time taken to rebuild the tree. Due to mobility, packets directed towards the MH (such as negative acknowledgements (NACKS)) might not be forwarded correctly and reconstruction of the tree each time the MH migrates is inefficient.

An approach called RBMoM which encompasses both bi-directional HA and remote subscription was presented by Lin et al. [73]. Their idea is based on having a multicast home agent (MHA). Instead of tunneling through the HA, the MHA is responsible for delivery of multicast packets to the MH. Unlike bidirectional HA, where the HA's position is fixed, the MHA assigned to the MH changes as the MH moves out of the MHA's range. The MHA's then rejoin the multicast session if the MHA that the MH hands off to has not subscribed yet. By having MHAs and depending on the given range, the need to rejoin the multicast session after handoff is considerably reduced. Despite the advantages, RBMoM suffers from tunnel convergence since multiple MHAs may tunnel packets to a given FA. Further, considerable delay is incurred during MHA migration due to the query delay to the HA, computation of closest MHA and rejoining delay to the multicast session if the MHA is not subscribed. Another issue that remains to be addressed is the problem of updating the multicast tree in a source rooted approach.

A different approach was taken by Kim et al. [74]. Kim et al.'s scheme entails construction of precomputed base tree(s)[3] infrastructure that supports low latency joins, leaves and handoffs. The multicast tree is constructed as a subset of a chosen base tree. A link augmentation algorithm is used to optimise the sub-optimal tree in stable regions of the tree.

[3]A base tree is basically a spanning tree of a network with certain properties

2.2.2 Problems with ATM Multicasting

The problems of multicasting are more complicated in mobile ATM networks due to their connection-oriented service model. This means that there are multiple connections to be updated during migration. Furthermore, any tree update must ensure cell ordering and preservation of allocated QoS. Schemes involving tearing down of branches must ensure that cells are salvaged first to minimize loss.

One of the major problems with current multicast model is that no consideration is made for mobile group members. In user-network interface (UNI) 3.0 [75] the multicast model employed for ATMs is source controlled and unidirectional with no support for multicast group addressing. In source controlled models, the source has to be notified of join requests. The source explicitly informs the network to add/delete receivers from the multicast tree. When a receiver decides to leave, the source is notified. Unidirectional approach is employed due to the problem of cell interleaving [76]. Cell interleaving occurs when cells from multiple sources arrive at a merge point of the multicast tree. As a result, these cells are inter-mixed and receivers are unable to distinguish the sources of cells. Readers interested in solutions to the cell interleaving problem are referred to [77]. Clearly the model proposed in UNI 3.0 [75] does not consider the possibility of the source/receiver being mobile. When the source is mobile, a new multicast tree will have to be rebuilt from the source's new location. As a result, a high signaling overhead is incurred and worsens when handoff rate is high. Furthermore, subsequent join requests will be directed to the source's previous location.

Currently, native mode multicasting protocol in ATM networks assumes a point to multi-point model (PTM). VC mesh and VC tree are used to emulate multi-point to multi-point (MTM) connections (i.e., many-to-many connections). In VC mesh, a PTM tree is built from each member to every other member of the multicast session. As a result, the number of connections incident on a particular member increases with group size. Furthermore, all members need to be notified of any changes such as add/delete in group membership. Therefore, the VC mesh scheme becomes infeasible in mobile environments, especially when large number of signalling messages are transmitted in high mobility cases.

The second MTM model is the VC tree. This is realised with the use of multicast server (MCS). The MCS's job is to terminate VCs from sources, reassemble AAL_SDU and create another PTM to session members [78]. MCS suffers from single point of failure and congestion at the root. Moreover, quality of service (QoS) is determined by the MCS rather than by receivers/sources. Apart from that, end-to-end latency increases due to suboptimal path and AAL_SDU processing [79]. A performance study of MCS is presented by Talpade et al. [80]. Also, in [80] the proposed distributed MCS avoids the problems of single point of failure and traffic concentration. The advantages of the VC tree model are: reduced receiver resource consumption, host connectivity independent of group size and one point of control for bandwidth [79]. Further, unlike the VC mesh model, less signaling is incurred since joining/leaving operations do not affect sources/receivers due to the abstraction provided by the MCS. The MCS is dependent on the multicast address resolution server (MARS) [78] for multicast group membership management. MARS was introduced to provide multicast group address abstraction which enables IP multicast to work over ATM networks. The main objective of MARS is to maintain a mapping between IP/ATM addresses and multicast group address. Receivers inform the MARS server when they want to join a particular multicast group. The receivers' addresses are relayed to the sender (MCS address if VC tree model is used) and receivers added onto the multicast tree. To date, no support for mobility has been added to the VC tree model. One disadvantage is that the MCS must be present after migration since without a MCS in the current network the receiver maybe forced to connect to its previous MCS which maybe far away.

Another method for MTM communication currently under investigation is the use of a single shared VC for both receiving and sending [81]. This means that shared tree approach (e.g., CBT [64]) can be employed in ATM networks where multicast group members only need to setup a connection to the core for sending/receiving data. The main problem with this approach is cell interleaving from multiple sources at merge point. The core is to be chosen when the multicast session first starts up. After each migration by a source or receiver, the path to the core becomes suboptimal, hence increasing end-to-end latencies. Recomputing the core's position after handoff is computationally expensive.

In UNI 4.0 [82] the leave initiated join (LIJ) was introduced. LIJ is issued by receivers interested in joining a multicast session. The network then responses with a leaf setup request. The root/sender does not need to know of receivers' existence. If no node on the tree is encountered, the leaf setup request is processed at the source and a network is notified to add the corresponding receiver. In [83], Tedijanto et al. investigate suitability of PNNI [58] in supporting LIJ. Due to the inability of a node to determine which nodes downstream are already on the tree, a snapshot of the tree is required to determine whether a loop will form if a LIJ is initiated. To overcome the problem of looping, Tedijanto et al. [83] defined the notion of LIJ domain. A LIJ domain contains a proxy root that is responsible for setting up branches. Each proxy has a view of subtree (nodes subscribed to the multicast session) emanating from it. Hence it can safely determine whether any loops will form given a join request. Nodes that have more than one receiver inform upstream nodes about receivers that are downstream. The signaling required to obtain a snapshot of the multicast tree and relaying membership information to upstream nodes entails a high overhead in mobile environments. Due to the dynamic nature of members, signaling messages would need to be sent constantly in order to keep an updated snapshot of the multicast tree and also group memberships downstream.

2.2.2.1 Multicasting in Mobile ATM Networks

Toh [84] extended his unicast connection rerouting methods [47] to multicasting in ATM. In Toh's model, the MH signals to the previous FA prior to handoff and the connection rerouting process starts even as the MH establishes a new connection. The connection rerouting algorithm employed is the prior path knowledge (PPK) algorithm [48]. PPK works as follows, during connection rerouting a query is made to the connection server. In response to the query, the server returns the nodes that are on the multicast tree. The MH then uses the map of the tree to calculate a crossover switch. To locate the CX, firstly a distribution point is determined. The distribution point is found by traversing from the old BS up to a point where the degree of the node is more than two. The CX is any node between the old BS and the distribution point which satisfies a given performance metric. Toh's scheme [84] considers migration within LAN only and as a result suffers from packet loss in the case of hard handoff and suboptimal path. Furthermore, Toh's scheme requires a snapshot of the multicast tree which is infeasible due to frequent change of group membership.

2.3 TCP over Wireless Networks

So far we have looked at routing to MH. In this section, we move to the transport layer, specifically TCP. Here the problems of inter-operating TCP over wireless links are presented. TCP is covered in this chapter because TCP is used to show the benefits of the active delivery scheme in Chapter 4.

2.3.1 TCP over Wireless Link

In the wireless environment, packet loss is usually caused by high error rates at the wireless link or errors during handoff [85, 27, 86, 87]. TCP's response to packet loss is to decrease the congestion window and retransmit lost packets [88]. The congestion window is then increased using a congestion avoidance scheme such as slow-start [89]. The high error rates of a wireless link during handoff combined with the slow-start scheme significantly reduces the performance of TCP in the wireless environment. Caceres et al. [87] have commented that these effects will reduce interactive TCP based applications to an unacceptable performance level in the wireless environment. Caceres et al. also showed that if both acknowledgement and data packets are lost during handoff, a full transmission window maybe lost during each handoff. The problem is that TCP interprets packet loss as congestion in the network. The general assumption taken by TCP is that the network is reliable and any packet loss is a result of congestion. Unfortunately this is not the case for mobile networks where packet losses are more common particularly at wireless links and during handoff. The performance degradation to TCP due to wireless errors is well documented in [90, 87, 91]. A study concerning different flavors of TCP over wireless link was presented by Samaraweera et al. [92]. They found that the New-Reno and SACK modifications enhance TCP's performance significantly.

2.3.2 Solutions

To shield TCP from packet loss, Bakre et al. [93] proposed Indirect TCP (I-TCP). The main idea of I-TCP is to split the connection into two parts. The first part is from the CH to the BS, and the second over the wireless link. Therefore, in the second leg of the connection, a different protocol that is better suited to the characteristics of wireless link can be used. The limitations of I-TCP are, breakage of TCP's end-to-end semantic, relinkage of applications and considerable overheads due to the additional protocol stacks at the BS [94]. Further, I-TCP does not guarantee smooth handoff since packets maybe lost while states are being transferred from the previous BS. A reliable transfer of states from BSs that are robust to varying migration scenario is presented in [95]. The split connection approach was further improved by Wang et al. [96] and Brown et al. [97]. Both Wang et al. and Brown et al. terminates TCP connections at the BS or supervisor host (SH). Then, a protocol (e.g., M-TCP [97]) which is optimized for the wireless link is used on the last hop.

Caceres et al. [87] suggested fast retransmission after handoff. After a MH migrates it sends three duplicate acknowledgements to the CH. As a result, fast retransmission is invoked and TCP recovers faster. However, the authors only address handoff latency rather than wireless loss. Caceres et al.'s work was later extended by Manzoni et al. [85] where TCP is modified to perform slow-start only when communicating with a MH. A feedback mechanism is used to inform the CH that the communication involves a MH. A feedback approach was also proposed by Bakshi et al. [91]. In [91], the source is notified to re-adjust its timeout value whenever the BS detects that no packets are going through to the MH. The BS then buffers all subsequent packets from the source. An acknowledgement is sent when communication between BS and MH resumes. However, Bakshi et al. did not consider the case when the MH migrates from one BS to another. Maruthi et al. [98] also proposed a feedback like mechanism where the transport layer is notified of any interference. In [98], interference occurs when the receiver is unable to extract data from the radio waves. Maruthi et al. proposed three algorithms to increase TCP's performance. Firstly a pipe snapshot algorithm is used to record the state of TCP before interference since any measurements taken during interference are inaccurate. The states are restored once interference is over. Secondly, an aggressive retransmission which enables fast recovery after retransmission is proposed. Finally, an algorithm is used at the receiver to overcome the inadequate recovery rate at receiver's side due to interference.

The *snoop* protocol [94] involves the installation of an agent at the BS. This agent is transparent to both communicating ends. The main function of this agent is to keep track of TCP packets transmitted to the MH. The agent caches unacknowledged data thereby allowing local recovery if a given packet is lost at the wireless link. The main advantage of snoop is that it does not require modifications to current protocol stack and only BSs need to implement snoop. One shortcoming is that it uses multicast for handoff and is specifically developed for TCP. Moreover it is susceptible to high handoff latency. A comprehensive performance study of snoop, I-TCP, TCP with selective acknowledgements and link layer approaches such as SMART [99] is presented by Balakrishnan et al. [100].

Bakshi et al. [101] take a different approach to improving TCP. Their technique uses a multicast scheme where the source multicasts packets just before handoff to a neighboring BS, so when the MH arrives at the new BS, packets are already there and communication can continue directly without disruptions. The novelty of this approach is that multicast is only initiated when the MH is about to handoff. On the other hand, the use of multicast which maybe costly, requires cell topologies to be known. Some form of tracking and the existence of a group management agent to oversee the handoff process is required. In [102], Bakshi et al. point out the lack of work in the area of seamless handoff, and propose a selective multicasting scheme which is based on the user's profile. By taking the MH's profile into account, only likely *cells* in which the MH may migrate to receive the multicast during migration, thus conserving bandwidth.

Goff et al. [103] propose Freeze-TCP. In Freeze-TCP, when a handoff or disconnection is detected, the MH tells the receiver to reduce the window size to zero. The trick here is determining when an acknowledgement with window size of zero should be sent. It was found that the performance of Freeze-TCP is proportional to the bandwidth delay product. Furthermore, since window probes sent by a sender backoff exponentially, an idle period occurs after reconnection. To overcome this idle period, the MH in Goff et al.'s scheme [103] sends three duplicate acknowledgements back to the sender causing fast retransmission. Freeze-TCP is invariant to high handoff latency and encrypted traffic. Furthermore, buffering is not required at BS and does not require the transfer of states from a BS to another. The main problem with Freeze-TCP is the difficulty in determining the exact time at which the advertisement of zero window size is sent, given that the time of sending is dependent on the RTT from the CH to the MH. Moreover, in Freeze-TCP, the assumption is that handoff can be predicted, therefore allowing the sending of zero window size.

2.4 Summary

In this chapter protocols for both unicast and multicast have been reviewed. From the literature reviewed, we can see that there have been extensive studies on mobile IP, TCP over wireless link and connection rerouting protocols. However, little progress has been made in solving the problem of multicasting in both IP and ATM networks. Furthermore, existing multicasting models suffer from suboptimal path and are inefficient. In the area of TCP over wireless, most work has concentrated on shielding the sender from high handoff latency due to delay in locating the next new BS. As a result, delay occurs when sending acknowledgements back to the sender thus causing a time-out. In general, the natural solution to ensuring seamless handoff is to use multicast. On the other hand, multicasting requires tracking of MHs, buffering, and additional bandwidth.

The next chapter looks at the AN concept. AN provides the necessary platform in which solutions presented in this thesis are realised. Since the main objective of this thesis is to show the viability of ANs in mobile networks, an understanding of the AN paradigm and the state-of-the-art in AN infrastructures is crucial in determining what is feasible and what can and cannot be realised.

Chapter 3

Active Networks

In the previous chapter we reviewed unicast and multicasting protocols in mobile IP and ATM networks. Before we describe the problems to be addressed in this thesis and present solutions, an overview of ANs is necessary: this chapter serves that purpose. Also a review of services and capabilities available at active routers/switches in various AN prototypes is presented. This chapter shows why network elements (i.e., router-s/switches) are in a better position to adapt to changes than end-hosts. Hence a new breed of protocols can be developed to address limitations in current protocols.

This chapter is structured as follows. Firstly, the vision of active networks and its advantages are outlined. In Section 3.2, work that has utilized the active network paradigm to improve existing protocols is discussed. Then AN infrastructures and security protocols that provide the necessary foundation for this thesis are presented in Section 3.3. Finally, the motivation for employing ANs is presented.

3.1 Overview

In the current network environment, the intermediate nodes are transparent to any two end systems. In Figure 3.1, each host (A,B,C or D) assumes that packets injected into the network cloud at one end will appear at the other end with their content intact

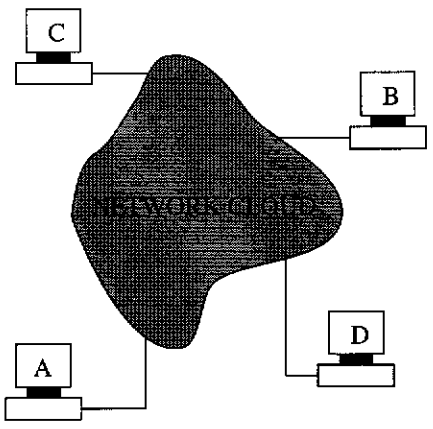


Figure 3.1: Current View of a Network Environment

and error free. The number of routers, bridges and the underlying network topology are transparent to both communicating hosts. Moreover, routing algorithms used at routers and bridges on each packet from one host to another are transparent to both hosts. In traditional routing schemes, hosts process only the header; they do not process the rest of the packet.

ANs were proposed by Tennenhouse et al.[9]. This new paradigm clears the network cloud so that end-nodes are aware of the programmability of network elements. ANs allow applications (such as video conferencing and network management) to inject customised programs into the network. This mechanism is realised by AN elements where each has a computational environment which can be injected with programs. The program can either extend the capabilities of the given network element(s) or provide customised computation on a set of packets (i.e., per flow computation) [9][104][105]. There are two proposed methods for computation at network elements [9]:

1. *Programmable switches.* This approach allows users to inject their programs into required nodes using a special tag located within the packet. When packets arrive at the node, the user specific program will be invoked to process the packet's content. Further, the program may perform computation on subsequent packets belonging to the current connection. The programmable switch approach is similar to the IP Router option specified in [106], where a new IP option type called *Router Alert* is defined. Routers supporting the *Router Alert* option can

then process the packet accordingly. Examples of this approach can be found in [107, 108].

2. *Capsules approach.* This approach is similar to how a Postscript printer evaluates its content. In ANs, each packet carries a set of instructions which is interpreted by network elements. During evaluation of the instructions which take place in a transient environment, the capsule has access to built-in routines for accessing resources. For example, the routing table. Apart from that, the capsule can leave a state (a static variable or a program) before it leaves the node.

   In the capsule approach, as capsules arrive at a given node, besides forwarding, the router is also required to execute the program embedded within the capsule. The program loaded may act on data streams for the connection or output new capsules. One of the main issues with this approach is that of security and a generic virtual environment must be in place at each router. Furthermore, the type of primitives to be made available are still under investigation.

ANs approach solves three main problems in the current network paradigm [9]:

- Integration of new standards and technologies into network infra-structure. The standardisation of protocols and communication technologies is a slow process. In the ANs paradigm, a standard computational model is provided and new protocol stacks can be installed at intermediate nodes whereas in current systems this is possible at end system only. Such a scheme is useful in applications such as video distribution because the technologies for distributing video applications changes rapidly.

- ANs provide flexibility. In their work on performance analysis of protocol layers [109], Clark et al. note that having a generic protocol is insufficient to cater for a wide variety of applications. In other words, a flexible protocol that caters to a wide range of applications is required.

- ANs allow existing services offered by network elements to be extended easily. Programs can be injected into the network to complement existing services.

In the above list, we see that the main benefits are the dynamic extension of current protocol stacks. This thesis exploits ANs to achieve benefits beyond those described above. By treating network elements as computational engines the network can be made to be adaptive. For example, if a router notices that there are huge number of update messages headed towards a given host then the router can automatically invoke an aggregation program. Another good example is in layered multicasting where receivers subscribe to different multicast addresses to improve their QoS. If a request to a higher layer causes congestion in the current domain then the request can be dropped. Furthermore, different scaling algorithms can be employed at different sub-tree levels (assuming source rooted tree) to cater for changing network conditions. The advantage of this is that the source can be transparent to such varying requirements of receiver. This thesis elaborates on such beneficial aspects of ANs and demonstrates how significant improvements can be made in mobile networks.

ANs can be considered as a subclass of the open signaling movement. The open signaling working community (OPENSIG)'s main aim is to rapidly create and deploy transport, control and management architecture [110]. A comprehensive review and comparison of programmable networks can be found in [111]. The main problem with today's network devices is the lack of separation between hardware and control software. As a result, the augmentation of these devices with more powerful and programmable services is impossible [112, 111]. The work in open signalling can be considered a subset of the ANs paradigm. The ANs paradigm extends the programmability model further by requiring dynamic deployment of programs. Since the codes used are from end-hosts, issues such as code mobility and representation, security and efficiency of the execution environment at each router/switch need to be addressed. Besides that, resource management and primitives to be provided at routers/switches require further investigation. The work on ANs is quite similar to that of mobile agents. The main difference is that work on ANs is concentrated at the network layer whereas mobile agents target the application layer. Although the targeted application maybe different, both ANs and mobile agents have similar traits. Both require safe transfer of code and robust execution environment. The main difference being, mobile agents reside at end-hosts whereas in ANs these agents exist at routers/switches. However, it is entirely possible for mobile agents to reside within routers/switches. A good overview of mobile agents can be found in [113].

3.2 Solutions Based on Active Networks

In this section, AN-based solutions are highlighted and the advantages obtained over existing solutions presented. This section, categorises active-based solutions into the following areas: mobile/wireless networks, network management, traffic management, filtering and information routing. Kulkarni et al. [114] also provided different categories of AN solutions, namely filtering, combining, transcoding, security, network management, routing control and supplementary services.

3.2.1 Mobile/Wireless Networks

Current work has mainly concentrated on injecting programs to address wireless link characteristics, for example high error rate. Apart from wireless link characteristics, services at the base station can be dynamically augmented. For example a video s-caling algorithm can be injected into the base station if the current base station does not support it. The use of active programs to address wireless link characteristics have been addressed by Kulkarni et al. [114] where adaptive forward error control, content base buffering and active caching schemes are presented. The ability to program BS was shown in the Mobiware [115] framework. Mobiware was developed to address the varying channel capacity of wireless links and varying QoS. Some of the services implemented in Mobiware include active media filters and adaptive forward error control filters. To address the variation of physical and data link services, Bose et al. [116] propose soft radios. Soft radios allow access points and mobile devices to implement physical layer functionality in software. In other words, services such as dynamic assignment of channel location and selection of modulation and coding techniques are implemented in software.

Kounavis et al. [117] outlined a programmable mobile architecture which allows the dynamic programming and composition of mobile services. They demonstrated their architecture by implementing two handoff services: (i) *multi-handoff access networks* and (ii) *reflective handoff services.* In (i), multiple styles of handoff are implemented, for example network controlled handoff and mobile controlled handoff. In (ii), mobile terminals are able to reprogram their protocol stacks as they roam across heterogeneous environments. Apart from the above, they presented a programmable MAC which is programmed by applications given a set of service requirements. Hence, different QoS can be tuned to a specific application.

3.2.2 Network Management

In the area of network management, traditionally systems are based on polling. To address the inefficiency of polling, Yemini et al. [118] deploy agents for network management. Yemini et al. [118] use the Netscript environment for implementing ANs. The work presented in [118] is similar to their earlier work [119] on agent deployment for network management. In [118], agents are deployed at intermediate nodes such as routers running the Netscript virtual environment whereas in [119], the agents are only executed from end-hosts. Once agents are installed, computation can be performed locally. Furthermore, agents can be programmed such that errors are handled locally rather than remotely, i.e., at the management station. An architecture which is based on ANs to localise the management of routers and switches was developed by Schwartz et al. [120]. Their system, Smart Packets, aims to improve on the existing network management paradigm by making network management decisions at switchs/routers, hence polling is not required.

Raz et al. [121] provided a system for network management. In their system, a Cisco 2514 router is connected to a personal computer (PC). The PC acts as the execution environment for the mobile code written in Java. Any packet encapsulated with the ANEP header is redirected to an adjacent PC for processing. Communication between the PC and router takes place using the simple network management protocol (SN-MP) [122]. The advantage of such a system over traditional systems is that current approaches incorporate multiple layers of abstractions which cause the cost of management to be obscured. Furthermore, the system reduces the number of redundant and unimportant information.

3.2.3 Congestion Control

Most congestion control algorithms detect congestion when packet loss occurs, and upon timeout or receipt of duplicated acknowledgements. A fast feedback mechanism is required to reduce the number of packets transmitted into the network in the event of congestion. Apart from that, during congestion, routers drop packets without consideration of traffic type. In Faber's work [123], active routers are used to resolve and prevent congestion. This is done by calculating the congestion window size at the routers and sending the new congestion window size to the sender. Thus, enabling the sender to react faster to congestion since notifications are sent from the routers. More importantly, the sender has a better idea of the congestion state. Bhattacharjee et al. [124] analyse the performance of an active router in performing congestion control by taking into consideration traffic type. The active router has a set of discarding functions specific for MPEG frames to achieve congestion control. These functions can be loaded or unloaded dynamically, in order to intelligently discard MPEG frames based on MPEG's file structure during congestion. Later in [125] they proposed an application specific buffering scheme. The advantages of having application specific buffering are that application's goodput is maximised and congestion reduced. For example, during congestion, video traffic is dropped based on frames information rather than the traditional packet discard policy such as "Early Packet Discard".

Williamson et al. [126] implemented an active explicit rate (ER) algorithm for controlling available bit rate (ABR) traffic. Unlike conventional methods, parameters of the ER algorithm can be set by a network management station located at the edge of the network. Therefore, customised ER congestion can be deployed by end-user instead of relying on vendor specific designs. Hence a suitable ER algorithm can be deployed based on network parameters. In [127], an AN-based solution is used to improve the performance of TCP over ATM's ABR service. The problem of concern is the increase in latency due to the increased buffer size at the source once ABR has reached a steady state. Furthermore, if buffering is finite, packets are discarded at the source. The solution entails generation of duplicate acknowledgements by switches when the source's buffer exceeds a threshold. The duplicate acknowledgements (after three have been sent) cause TCP to invoke TCP's congestion control. Thus, cell loss at the source is prevented and buffering requirement minimized.

3.2.4 Multicasting

In the area of multicasting, Wittmann et al. [128] employ ANs by injecting QoS filters at strategic positions in the multicast tree to cater for low end receivers. Therefore, multimedia traffic such as video is adapted based on receiver capabilities. The problem of multicasting in a heterogeneous receiver environment is also addressed in [129] and [130]. Bhattacharjee et al. [129] investigated the performance of various techniques for discarding MPEG traffic. Amir et al. [130] build a video gateway. The gateway manages incoming and outgoing video streams using transcoding and rate control. The improvement to current techniques for adaptation is that receivers are not required to subscribe to different multicast address. Furthermore, receivers do not have to perform network probing to determine whether they should degrade or upgrade their subscription.

Lehman et al. [131] used ANs to suppress duplicate NACKs and limit the delivery of repair packets to receivers experiencing loss. Furthermore, in their work multicast packets are cached on a best-effort basis for local retransmission. A similar approach was also taken in [132] where QoS filters and error control mechanisms are embedded within active nodes. The active nodes in the architecture of [132] comprise different service modules (MPEG-1 filters) that deliver varying QoS streams depending on available bandwidth. In other words, different branches of the multicast tree can have different qualities of service. Lau [133] worked on an active receiver driven layered multicast protocol. The protocol does network filtering and achieves bandwidth convergence via the use of active routers. In the area of video delivery, Baldi et al. [134] designed a video conference system based on ANs. Their systems allow the cloning of video servers and customisation of packet delivery.

An implementation of video distribution over active routers was presented by Keller et al. [135]. The active router architecture used is an extension of the router plugins architecture proposed by Decasper et al. [136]. Current method uses layered multicast [137], where the sender distributes varying video layered over different multicast groups. By utilizing active routers, the distribution of video can be adapted according to available bandwidth and allows fine-grained adaptation. For example, if a particular branch has limited bandwidth then it can be scaled accordingly instead of requiring receivers to subscribe to a lower layer. Moreover, scaling algorithms can react faster to changes in bandwidth [135]. Another advantage is due to the ever changing video encoding and decoding schemes. Hence newer and better schemes can be used once they become available.

3.2.5 Security

Work related to providing security in ANs is progressing rapidly though not much work has been done in the use of ANs to solve security problems. Therefore, more work is likely to appear in the future where security services offered by ANs are augmented with those of end-hosts. In [138], Van demonstrated the use of active networking for defending against address spoofing. Van solves the address spoofing problem by dynamically deploying a filter which inspects all packets destined for a given host. Another security related work is the development of an active firewall. This task is currently under development at University of Pennsylvania. A brief discussion can be found in [139].

3.2.6 Caching

In the area of caching, Legedza et al. [140] showed how an AN-based solution reduces the time needed to find unpopular cached and uncached documents. A cache server injects *redirectors* into the network, the *redirectors* redirect cache requests to other servers. The location of *redirectors* is based on a route trace of cache requests leading up to a given cache server. The *redirectors* are then injected into "hot" spots of the route. The advantages of their scheme are: the load of the cache server is reduced, caches do need to be configured with neighboring cache servers' addresses and requests for items are not redirected away from original shortest path to their destination.

Another application that can be improved with caching and filtering at intermediate nodes is online auction [141]. When an online auction server detects a heavy load, the auction server deploys a filtering program. The filtering program drops bids that are lower than the current price for a given item.

In [142], Bhattacharjee et al. presented self-organising wide-area caches. Traditionally, caches are located at strategic points in the network. In [142], a different approach was taken where small sized caches are kept at routers. Proper organization of router-based caches are critical in ensuring high hit ratio. Bhattacharjee et al.'s active scheme distributes caches using a radius parameter, where caches are distributed in a concentric circle with the server at the centre. Consequently, these small number of caches within routers form a large virtual cache.

3.2.7 Telephony

A proposal to apply the concept of ANs in telephony was presented by Maxemchuk [143]. In telephony, the application of ANs enables the implementation of adaptive coders. Furthermore, since lines are busy more often and longer due to users being online all the time, the use of ANs enable some lines to be reused during idle periods thereby utilising available lines and saving on the cost of additional lines. Further, new functionalities such as conferencing tools with restrictive access, intelligence to switch between circuit and packet switching during periods of congestion and leasing of multicast tree from the telephone network can be introduced.

3.2.8 Information Routing

In information routing the aim is to route based on the location of information rather than destination. Therefore, packets are routed based on user's interest. This is particularly useful for searching information on the WWW where the ability to find a given information efficiently is lacking. Furthermore, current search engines are based on intensive probing of databases which increases the network load. Zhang et al. [144] outline a model for information routing to overcome the above problems. ANs are used where filtering and binding operations can be injected dynamically to obtain information. The information provided by information providers are structured using an ontology based information hierarchy. Two types of routing are supported, query and distribution. In query routing, a packet searching for particular information is directed at information providers. In distribution routing, packets containing information are distributed to interested users.

3.3 ANs Enabling Technologies

As can be seen, there has been a proliferation of AN-based solutions in most areas of communications. For these solutions to be possible a platform is required which provides an environment in which AN-based solutions can be realised. A number of AN infrastructures are being prototyped and these architectures or infrastructures will be reviewed in the remainder of this chapter. This section, primarily reviews work in the context of ANs and other enabling technologies that can be extended to realise the ANs paradigm.

3.3.1 AN Architectures

The following subsections review architectures that are built specifically to realise ANs. As mentioned previously, ANs consist of two types of protocol deployment, capsule and programmable switch.

3.3.1.1 Capsule

One of the early works on prototyping ANs was done by Wetherall et al. [145], where a small TCL code is inserted into the IP option field. A stripped down version of the TCL interpreter is then incorporated into the Linux kernel. At each node the interpreter provides primitives for accessing packet header, creates new packets and obtain attributes pertaining to the current node. There are three limitations; the embedded Safe-TCL code is limited by the length of IP option field, no security issues are addressed (except for validation provided by Safe-TCL) and no resource allocation controls are provided. Wetherall et al. [146] have built a capsule based AN toolkit called ANTS using Java [147]. In ANTS, packets are replaced with capsules. In its simplest form a capsule only contains a reference to a routine to be executed at the active node. More complicated codes require code to be loaded by a code distribution mechanism. A novel code distribution scheme was proposed where the required code in a capsule is pulled along the traversed path. Basically when a capsule arrives, a check is made, and if the required code is not available a request is made to the previous node. The program from the capsule is then suspended until the arrival of code from the downstream node. Active nodes in ANTS provide the following categories of primitives: environment access, capsule manipulation, control operations, caching and rendezvous. Authentication is provided using MD5 [148] message digest and security of execution is as provided by the Java [147] virtual machine.

Fernando et al. [149] proposed PANTS, an extension to ANTS. The nodes in PANTS are more dynamic whereby nodes run-time execution environment can be changed dynamically and allow capsules to dynamically rewrite their code, which is difficult to achieve in statically typed language like Java. Further, capsules are able to form dynamic group of similar interest within active nodes without intervention from a central authority.

Instead of pulling the code along the route as in the ANTS architecture, Decasper et al. [150] used code caching in their architecture, where packets only have to carry pointers to digitally signed modules. During execution, when a program (from a capsule) refers to unknown functions, the processing is suspended, a query made to the code server, and the required modules are loaded on a demand basis.

A kernel based capsule approach called PAN was proposed in [151]. The performance overhead documented for the processing of a capsule (1500 bytes on a Pentium Pro 200Mhz running linux) was shown to be as little as 13%. Their results also showed that the use of Java bytecode increased the overheads by three fold. Apart from that, the PAN architecture enables the loading of different mobile code systems. Currently, two code systems are supported, namely Java bytecode and native Intel ix86 object code. The capsules in PAN contain information which specify the interfaces and code object to use during execution. If a code object is unavailable it is dynamically loaded as in ANTS [146]. PAN also provides soft-state and the ability to export interfaces related to the stored soft-state. This allows the soft-state to be maintained safely.

Another Java based architecture is presented by Kulkarni et al. [152]. The architecture presented in [152] is based on the capsule (referred to as SmartPacket in their work) and programmable approach. The capsule approach is used to inject new functionalities into the node where the programmable approach is used to invoke specific routines in the node. Each node has the following managers: *Port, Resource, Routing and Small State*. Each manager provides predefined interfaces which can be accessed by SmartPacket during execution. Also, the manager keeps track of resources allocated to a given program. For example, the resource manager keeps a limit on the amount of CPU time and also bandwidth allowed. Two applications were implemented to show the workings of their architecture; mSMTP and mHTTP. In mSMTP, there exist an agent based mechanism where the mail is carried by a SmartPacket to the remote system and interactions required in traditional SMTP are done locally. The functionalities in mHTTP include caching/accessing of customized data and allows users to specify a list of HTML pages and images to be cached.

An architecture implemented using the M0 (M-Zero) language was presented in [153] where the capsule is called a messenger. The M0 language is a high level language similar to the Postscript language. The system is similar to the ANTS model [146], however, in M0, resource management uses an economy-based model where resources are associated with cost and all codes required for execution are available within the messenger.

Smart Packets [120] address short comings of current network management models. A Smart Packet contains program, computed results or messages. The messages in Smart Packets are encapsulated in the active network encapsulation protocol (ANEP) [154]. For backward compatibility, Smart Packets use the Router Alert option specified in [106]. Consequently, routers that do not support active packets will ignore the Router Alert option. The language used, Sprockets, in [120] is derived from the C language. Sprockets inherit C's grammar and keywords but remove constructs such as enumerations, typedefs, structures and unions. After compilation, Sprockets are converted to a *spanner*. Spanner is the assembly language for Smart Packets. A virtual machine was developed for interpreting spanner code. Authentication and security features are also implemented.

So far, work on ANs has mainly extended existing architectures and utilised existing network services. A different approach was taken by Hicks et al. [155], called PLANet, where an inter-network of active nodes were constructed based on OCaml 4.0 [156] and the Packet Language for Active Networks (PLAN) [157]. PLANet implements network layer services and these services are extensible through programs written in OCaml. This means PLANet implements its own addressing scheme, packet formats, error handling, and so forth. Packets in PLANet carry programs and contain a tag which specifies the destination node at which the packet is to be evaluated. Hence, selective evaluation is possible. Apart from that, packets have the option to specify whether to use the default routing protocol or a customised routing protocol. Issues affecting the performance of their architecture include: kernel crossings, use of threads, copying and garbage collection cost.

The paradigm envisioned by Hartman et al. [158] is similar to ANs. In [158], they attempt to apply mobile code technologies to all layers of the communication stack. Instead of a specific code, executing on a specific machine, the code "flows" from one machine to another, thus the term liquid software. The aim here is to transport and transform data efficiently. One main result from the liquid software idea is Joust

[159]. Joust is basically re-implmentation of Java virtual machine running on the Scout operating system. The Java virtual machine is optimized to take advantage of Scout's features such as path abstraction. To benchmark the performance of Joust, Hartman et al. [159] used the ANTS [146] architecture and it was found that ANTS runs with much lower communication and layering overheads. This is mainly due to the tight coupling between the virtual machine and the operating system.

3.3.1.2 Programmable Switch

Active bridge was proposed by Scott-Alexander et al. [107]. The active bridge allows different spanning tree algorithms to be injected. The spanning tree algorithms are written using the Caml language and executed in a thinned Caml execution environment for safe operation. Caml comes with a security model such as strong typing and safe execution environment for injected programs. Primitives provided include time, thread, logging and network related functions. They successfully showed that a modular approach in which basic functions loaded initially on the bridge can be extended dynamically. Furthermore, they showed the transition between different protocols in a coordinated and automatic manner. Moreover, should the transition fail, it is able to fallback on the previous working module. The transition between different spanning tree algorithms is triggered when receiving different tree spanning packets, for example DEC and 802.1D.

Murphy [160] built an active node which is able to route ordinary IPv6 as well as active IPv6 packets. This demonstrates the co-existence of ANs and traditional network infrastructure. Kulkarni et al. [152] outline another AN architecture implemented in Java [147]. In their architecture, each node has a set of managers which have primitives for managing a node's resources such as CPU cycles, ports and routing tables.

A programmable approach was outlined by Bhattacharjee et al. [124] for congestion control in ATM networks, packets contain control information for identifying the predefined functions to be executed at each node. Further, a set of labels are included whose functions are to select the corresponding state information used at the switch. Due to the small packet size in ATMs, the header information is placed after the ATM adaptation layer (AAL)'s header. This means that switches must first reassemble AAL data units before they can examine the header. Since this process is costly, reassembling does not happen to every cell.

3.3.2 ANs Node Operating System

The Bowman operating system was specifically developed for active networks [161]. The performance measured on Bowman is encouraging where a throughput measure of 100Mbps was achieved while forwarding IP packets over fast Ethernet. The features of Bowman include, per-flow processing, efficient and flexible classification algorithm, fast path for non active packets, support for implementation of virtual networks, reasonable performance, support for multiple processors, and different execution environments. Examples of programming interface made available to programmers of these execution environments include libraries for accessing resources such as computing and storage. In another work, Meregu et al. [162] detailed the implementation of composable active network environments (CANEs) on top of the Bowman operating system. CANEs implement a slot processing model. The slot processing model consists of two steps, the first of which is a base functionality (i.e., default processing). Within this default functionality are slots where customisation programs can be injected. For example, after receipt or prior to forwarding a packet, a user can insert a program that takes the packet and performs operations on it before forwarding [162]. Two applications, namely congestion control and self organising caches have been implemented in CANEs and the details can be found in [142].

In [136], Decasper et al. proposed a router with plugin capabilities. Plugins are dynamically loaded and configured at run-time. The difference with Boulis et al.[163]'s work is that plugins are loaded from an end-host (e.g., MH) whereas in [136] plugins are present in the local machine. Moreover, no provisions for security and authentication are provided in [136]. In Decasper et al.'s architecture, they have shown the ability to bind different plugins to individual flows. Furthermore, the increase in performance overhead was shown to be 8%. Their router plugin architecture was then extended and used to realise an AN.

The secure active network environment operating system (SANE O/S) [164] is currently under development. SANE O/S is based on the FreeBSD operating system and uses Java as the execution environment. Moreover, the AEGIS boostrap integrity system proposed in [165] is incorporated. Apart from that, SANE O/S implements ANEP [154] and allow *N* active nodes to mutually authenticate each other.

3.3.3 Programmable Networks

The idea of programmable network precedes that of ANs but more restricted in nature. The work in programmable networks can be viewed as a subset of the ANs paradigm. The difference to ANs is that programmable network looks at opening up the access to switches, routers and base stations. Therefore in comparison, it does not cover distribution and execution of active programs. In the following paragraphs we will look at technologies in this area. The literature in this area can be extended to ANs.

In 1991, Hutchinson et al. [166] outline an architecture for composing and constructing network protocols called *x*-kernel. The *x*-kernel provides an independent address space, light-weight processes and an architecture for implementing and composing network protocols. Some of the objects that enable the creation of protocols include protocol, service and message. Protocol objects basically create session objects and demultiplex packets to the corresponding session. The session represents end-point network connection that are created during run-time. Finally, the message object provides a mechanism in which messages are passed between protocol objects.

Application-specific handlers (ASHs)[167] have similar objectives as the *x-kernel*. Here ASHs are dynamically injected into the kernel invoked after a message has been demultiplexed. All operations and run-time of ASHs are bounded thereby promoting safe code. Some of the abilities of ASHs include, message and control initiation and dynamic protocol composition.

Protocol booster [168] has the same goals as ANs, that is rapid protocol evolution and customisation of protocols in heterogenous environments. To achieve both of these goals, protocol booster can be incorporated transparently to "boost" existing protocols. For example, incorporation of a booster which adds forward error correction codes. The error correction codes are transparent to the operation of IP. In the protocol booster model, removing booster modules does not affect base functionalities of protocols where boosting takes place. With the protocol booster model, protocols are first designed for best-case scenarios (e.g., homogeneous networks) followed by subsequent tuning or tailoring performed using booster modules.

Boulis et al. [163] outlined an active base station. Their architecture is based on a packet classifier which multiplex packets onto a set of packet processing filters (PPF). PPFs are integrated in a pipelined fashion. These PPFs are loaded using the PPF loader program which contains the necessary code and instruction to instantiate and terminate a PPF path. To add functionalities, a MH injects the BS with a PPF. As part of the injection process, the MH specifies to the PPF loader how the injected PPF should be integrated.

Mobiware [115] is an adaptive architecture based on distributed object technology. Mobiware contains programmable network interfaces which enable the building of new services using CORBA, DCOM or Java. Mobiware was developed to tackle the varying QoS requirements in mobile networks. The programmable aspect of Mobiware is based on the xbind broadband kernel [169]. Three main components of Mobiware are, QoS controlled handoff (scaling of traffic during handoff), mobile soft-state (provide mobile devices the capability to respond to changes in wireless or mobile QoS) and flow bundling (exploits the common routing representation to speed up handoff). Furthermore, the link layer in Mobiware is programmable in that it can be programmed with different services to provide minimum wireless QoS assurances.

Amir et al. [170] argued that a large number of applications do not need full fledge implementation of ANs. They developed the active service framework as an intermediate goal towards ANs and showed that most applications can be enhanced without a full featured ANs architecture. In the framework, a pool of one or more clusters provide services that can be instantiated by clients. Once these services are instantiated (referred to as servents), clients are able to dynamically control and reconfigure their operations.

Sudame et al. [171] proposed transformer tunnels. Transformer tunnels are established like any other tunnels in a network. In the transformer tunnel model, services are attached at the beginning and end of the tunnel only. Example services are transcoding, reassembly and compression. This means traffic is "transformed" before entering the tunnel and reconstructed when it exits the tunnel. Transformer tunnels can be used to construct limited ANs, similar to MBONE. The viability of transformer tunnels has been shown in car pooling [172]. The car pooling idea uses transformer tunnels to aggregate packets such as TCP acknowledgements. As a result, load on the routers is reduced. Unlike ANs, where computation is performed at every router/switch, transformer tunnels only perform computation at the entry and exit point of the tunnel.

In [173], the inflexibility resulting from an integration of control architecture and switching plane in an ATM switch is addressed. Since there are no standards on the interaction between both planes, different vendors have implemented their own solutions. To address this issues, van der Merwe et al. [173] introduce a framework for programmability in ATM switches. Their framework partition switch resources into switchlet, and each switchlet is controlled independently by different virtual network managers. The direct implication of this is that multiple instances of the same or different control architectures can be applied to the same switch [173].

Campbell et al. [112] detailed a spawning network. The architecture is built over the Genesis kernel. In their architecture, virtual networks are created on the fly. For example, cellular IP [37] can be spawned to overcome high handoff latency in mobile IP [10]. Spawned networks inherit architectural components from their parent networks. These components include, transport, programming, life-cycle and management. The proposed architecture is generic in that different architectures can be built, for example spawning a virtual ANs.

3.3.4 Security

The main focus of security in ANs is providing safe execution environment and authentication of distributed code. This translates to the following requirements an ANs architecture must have, (1) correct evaluation of the program (2) methods to check validity of programs (3) authentication of user (4) enforces limits on execution [120]. Moreover the execution of programs must be based on the authorizations and security policy. In addition, the ANs infrastructure must be secure and provide assurance that all modules are checked prior to loading [165].

The architecture used for Active Bridging [107] uses a restricted environment feature called module thinning where a set of functions and their data structures can be grouped into a module. Each of these modules have a signature that describes the content of the module [107].

SANE[165] currently offers the following security services: cryptographic primitives (e.g, symmetric key encryption), packet authentication, packet confidentiality and key establishment protocol. SANE uses Caml [156]. Other security measures implemented include: access control, authentication and authorization of a principal (network node or user) requesting action and dynamic resource naming. Besides the above, a mechanism in which active programs are distributed and security between routers pertaining to a session needs to be addressed. The performance of SANE has been documented in [174].

Shapiro et al. [175] propose the use of the Extremely Reliable Operating System (EROS) for providing a safe execution environment. Resource allocation control is done using capabilities. Certificate mechanism is used to authenticate injected programs and a dedicated and controlled area for each application or process is provided.

In the ANTS [146] architecture, MD5 [148] is used to name on-demand loaded programs. However, this only provides unique naming rather than security. ANTS execution environment depends on the Java [147] virtual machine which provides static checking and run-time restrictions.

PLAN [139] is a functionally restricted language for use in ANs. The security properties in PLAN include, resource and expression limited to prevent resource denial-of-attack, termination guarantee (i.e no ability to define unbounded iteration), strong typing and garbage collection mechanism. Remote evaluation primitives and service functions to be invoked at active nodes are the other functionalities included in PLAN.

Thibault et al. [176] extended PLAN [139] to a generalised ANs programming language. The resulting system called PLAN-P provides programmability as well as meeting security and safety demands. The main advantage being, it is based on IP and allows PLAN-P programs to use UDP and TCP. The performance measured showed that PLAN-P is two times faster than compiled Java and 2.5 times faster than Caml. The performance was achieved using the idea of program specialisation. The program specialisation process transforms a program based on some known inputs. Program specialisation was used on the PLAN-P interpreter where the interpreter is specialised to the input program during run-time. Some of the properties enforced by PLAN-P include guaranteed termination and delivery, and ensuring that packets are not duplicated exponentially. This system has been demonstrated to improve distributed protocols in [177].

Menage [178] developed an AN environment with the design goal of restricting resource usage. Network resources such as CPU, memory and bandwidth have to be restricted because unchecked programs may cause denial of service attacks. The run-time environment used is the OCaml system and the PLAN interpreter is used to provide a restricted execution environment. Security issues covered by Menage [178] include sessions (i.e., authenticating the owner of a session), resource accounting, CPU scheduling, Network input/output, memory and service functions (creation and manipulation of services).

Yeh et al. [179] describe how a dynamic path going through different domains with different security policies is set up. Yeh et al. describe signaling and path repair mechanism. A path repair mechanism is needed when link failure occurs, either due to congestion or change in security conditions when a path failure is detected by the nearest router, an alternative path can be set up quickly.

3.3.5 Code Representation

The encoding of active programs has not been standardized as yet. According to Maxemchuk [143], programs should be represented with the flexibility of the hypertext markup language (HTML). This because HTML is generic in that different browsers can evaluate it differently and yet still produce almost similar results.

The ANEP [154] is developed to provide a generic header in which the payload's execution environment and security associations can be specified. The goal of ANEP is to enable active routers to quickly identify the execution environment for a given packet. Besides that, if no matching evaluation is found then default and minimal processing can be performed. Moreover, information that does not fit in the program could be specified in the header. However, the downside of ANEP is that it only provides specification for packet header. Therefore, the encoding of packet payload is left to the developer. To address encoding of active programs, Williamson et al. [180] employed abstract syntax notation one (ASN.1)[181]. They have successfully demonstrated the encoding of network management tools such as ping and traceroute. The advantages of using ASN.1 are:

- ASN.1 is already standardized. Thus it allows the integration of ANs with existing protocols that utilised ASN.1.
- Debugging is made easy since the program is treated as a packet specification and any packet analysers can decode the program.
- States can be specified formally and in the event of router shutdown, states can be moved to another router.
- Support for different encoding schemes and provision of security definitions.

3.4 Why Use Active Networking in Mobile Networks?

ANs are being successfully employed to address several network related issues by using installed programs at strategic points. As can be seen from the literature no work has investigated the applicability of ANs in routing packets to MH. Thus, the aim is to investigate and quantify the viability of ANs in alleviating problems due to mobility, specifically routing. This thesis shows how ANs can be applied to the protocols reviewed in Chapter 2.

The term ANs takes on different meanings and is viewed differently by different researchers. Most prominent is its association with fast protocol deployment and extensible routers/switches. The view of "active" here is the ability to perform computation at routers without requiring the probing and computing associated with the collection of data at end-hosts. Current work has mainly concentrated on proving the viability ANs. This thesis takes a different approach where the goal here is to show the performance and problem solving abilities of ANs. By viewing the network as a large collection of computational engines, new breed of protocols may emerge that overcome problems encountered in today's protocols. The idea here is to obtain better solutions to network problems such as location management of MHs compared to solutions which are deployed from the edge of the network. In the area of mobile networks, intra network processing is advantageous due to the dynamic location of MHs and QoS requirements. To ensure seamless execution of applications running at MHs a fast response time is required to mask the disruption caused by handoff. The view taken in this thesis is that individual network elements are in a better position to adapt to changing situations rather than receive instructions from end hosts. With this view, any change in the network can be adapted in accordance with current information. For example when a MH migrates into a high congestion network and advertises a window size (assuming TCP), and if this window size is high it can aggravate the situation. With ANs, the window size can be changed according to the network situation to avoid degradation of performance.

The main benefits of applying ANs in mobile networks are:

1. Management of location. The network manages the MH's location instead of a HA. The CHs can establish connections with the MH faster (i.e., bypassing the HA), resulting in efficient routing of packets. Multiple points in the network are aware of MH's location unlike existing schemes, thus avoiding *single point failure* problem.

2. Traffic customisation. Customisation is invoked depending on the current state of network. For example, if a particular host (e.g., web server) is receiving a significant number of binding updates then the network can load an aggregation program which decreases the number of update messages, thus conserving bandwidth and reducing the number of packets to be processed at routers.

3. Customized architecture. To reduce the number of signaling messages a common practice is to utilize a hierarchical network topology. Hierarchical foreign agents [34] do not necessarily ensure low handoff latency. AN-based solutions provide a generic solution that is independent of any topology and also reduce bandwidth requirements of signaling messages. As we shall see in later chapters, programmability offers better adaptation to different network topologies.

4. QoS adaptation. Variation in QoS is handled internally rather than waiting for end-hosts to adapt. Having the network handle QoS has the effect of distributing the work load associated with adaptation among the network elements.

5. Security. Security features in ANs are still evolving and security may be available as an added feature in all AN-based services. This is because ANs enforce and address a wider range of security issues [165], therefore when ANs are realised, network elements would provide security as a service to end-hosts.

6. Reuse of states. ANs are particularly useful in connection oriented networks and multicast protocols where a lot of information is maintained within the network. The advantage over end-hosts based solutions is that these states can be reused. If an end-host solution is used, a lot of probing is required to figure out the location of information within the network before any update is performed. This point will be shown when rerouting both unicast and multicast connections to ATM and IP networks are presented.

7. No strategic positioning. A significant benefit of dynamic program deployment is that no pre-calculation is required to determine the strategic point in the network in which programs should be injected. This is crucial in mobile networks due to host movement, since strategic points maybe invalid after host movement. One example is the shared-tree where the core of the tree is fixed *a priori*. Due to host migration, the core maybe invalid, thus becomes sub-optimal. In ANs, programs become strategically positioned given the correct event. For example, when a given router/switch discover that all its downstream subscriber are mobile, it can invoke traffic customisation to cater for the low bandwidth. Once these subscribers migrate, the customisation program can be unloaded. A given node only remains active (or strategically positioned) while it is needed. The time in which a node becomes active is not determined *a priori*, and its invocation depends on the state of the communication, for example after a migration.

The performance of ANs is being investigated and increasingly better performance architectures have emerged. For example, in PAN [151] the computational overhead is only 13% on a Pentium Pro 200 MHz running Linux. The Bowman [161] node operating system for ANs is able to sustain 100Mbps throughput while forwarding IP packets over fast Ethernets. No work has yet implemented and investigated the performance of ANs on a dedicated router. The computational tasks to be performed by a router in a typical AN-based environment would involve managing multiple execution environments, classification and forwarding of packets. Such tasks can be handled by an embedded processor.

The schemes proposed in this thesis could also exploit mobile aware networking infrastructures. The downside of having custom "mobile-aware" network elements is the restricted functionalities provided to the MH. An ANs approach is capable of providing enhanced and programmable network elements. Most important is that these "mobile-aware" network elements need to be placed strategically within the network. In later chapters, we will see active programs distribution model used in each of the presented solutions. The distribution of active programs do require precalculation of strategic position(s).

The work presented herein does not concern with issues such as architecture and management of active programs. Given a programmable network, the main concern is to determine the improvements that can be made to existing protocols, such as mobile IP. Due to software extensibility provided by ANs, significant amount of improvements can be made if computations are performed at the routers/switches. In general, improvements can be broadly categorized into: traffic customisation and local processing. Traffic customisation deals with rerouting of connections/packets and aggregation of packets. In local processing, computation is positioned closer to information. There are two main implications to local processing. First, end-hosts do need to probe the network. This improves the scalability of any scheme since the number of signaling messages is reduced. Second, processing is only done on the latest information. Due to the dynamic changes of the network, computation can be performed when events happen and not after it has happened. These improvements, in particular to routing in mobile networks, have not been explored and contribute greatly to ANs research and provides a new perspective on possible solutions that maybe incorporated into existing protocols.

3.5 Summary

In this chapter we introduced ANs. Many infrastructures have been prototyped and a significant amount of work has been done since their inception in 1996. At the beginning, as a proof of concept, the IP option field was used and embedded with TCL. After that, Java was the natural choice due to its static checking and safe execution environment. In the latest developments we have seen the development of operating systems tailored especially for ANs. On the security front, AN communities have made progress with the development of execution environments containing active programs.

The distribution of active programs have not received much attention in active networking communities. The majority of the work has concentrated on router architectures, security and applications. Little work have specifically address how and where active programs are positioned within the network or how instantiated services cooperate.

As shown in later chapters, strategically positioning active programs would defeat the purpose of having ANs, due to active solutions being topology independent. Also, s-trategically positioning services in mobile networks is infeasible due to frequent host migrations.

To sum up, this thesis takes on the observation that routers/switches are in a better position to adapt to network changes rather than end-hosts. More importantly, little progress has been made in applying ANs in the area of routing in mobile networks. This thesis is not concerned with the architectural aspects of ANs. The solutions presented in Chapters 4 to 7 only requires an execution environment that allows transient programs to be executed.

Some areas that require further investigation include, efficient security mechanisms, program encoding, and standardization of services available at active nodes. An efficient security mechanism is vital to the fast deployment of active programs especially in mobile networks where different parts of the network maybe loaded or unloaded to keep up with mobility. A set of standard services must be present at each active node. Therefore, the design of an AN-based solution can be simplified if certain services already exist.

In the following chapters, the proposed AN-based solutions will be investigated and evaluated. These solutions take the view that the entire network is programmable and constitute a virtual computational network. As indicated in Chapter 2, this thesis covers both unicast and multicast protocols.

Chapter 4

Enhancements to Mobile IP

In the previous chapter the ANs paradigm and the functionalities that it allows were introduced. Besides that, the rationale for applying ANs to routing in mobile networks have been discussed. In this chapter the first of the AN-based solutions investigated in this thesis will be presented.

In this chapter, a scheme called active delivery scheme (ADS) is proposed. ADS comprises a set of enhancements to IETF mobile IP [10]. ADS consists of three novel functionalities: smooth handoff through the use of SmartBuffer, a multicast like delivery of binding updates and the BusStation framework. Firstly, SmartBuffer ensures smooth handoff without the need for multicasting to FA and is resilient to high handoff latency or delay in the delivery of binding updates. Secondly, in ADS the HA is not required to send binding updates to CH. The ADS program keeps track of CHs associated with the MH and determines whether it should send a binding update during handoff. As a result, binding updates are sent from the active routers (ARs) closest to CH. Hence CHs experience lesser binding update latency. Furthermore, the HA's load is reduced and bandwidth conserved. Finally, the third contribution concerns with the use of the BusStation framework for aggregating location update messages. Through aggregation, a number of messages heading towards a given server are reduced considerably, and the load on routers reduced since there are less packets to process. Apart from the above, ADS provides for efficient cache binding maintenance at the routers and CHs.

The above mentioned novel features of ADS were developed based on the following observations and short comings of the IETF mobile IP models:

- *Registration latency*: During handoff, the MH and CH depend on the HA to respond with a registration reply and binding update respectively. This update time is dependent on the RTT between the MH and the HA or CH. Therefore, the reliance on the HA increases, the handoff latency and the number of packets tunneled to the MH's old care-of-address.
- *Binding update latency*: It is crucial that handoff time is minimized and the MH's care-of-address be made known to the CH quickly. This is required since packets dropped during handoff [85, 27, 86, 87] and handoff delay will cause disruptions to transport protocols, such as TCP. In TCP, packet loss results in the invocation of slow-start congestion avoidance [89] based upon the assumption that any packet loss is a result of congestion. The binding update latency is crucial in scenarios where the CH is mobile. When the MH and CH migrate, binding update must be sent as soon as possible because the acknowledgement packets as well as the data packets are being tunneled to the old care-of-addresses. This is detrimental to the performance of TCP as will be shown later.
- *In-Transit Packets*: Network propagation delay may cause some in-transit packets to be forwarded to the MH's previous location. The amount of traffic directed towards the old address depends on the time taken (a function of RTT) to update the CH with the MH's new care-of-address and also the time it takes for the MH to establish a link with the new BS.
- *Smooth Handoff*: Buffering at the previous FA as proposed by Perkins et al. [33] to solve the problem of packet loss at the previous FA due to delay in binding update. In scenarios where the handoff latency plus the time it takes for the binding update to get to the old FA is large, the TCP sender may experience a retransmission timeout. Thus, causing unnecessary retransmission of packets and reduction of congestion window.

- *Inefficient Binding Update Delivery*: There may be N CHs associated with a given MH. For each CH the HA needs to send an associated binding update. To help reduce the load of the HA and conserve bandwidth a multicast like mechanism can be used to deliver the binding updates.
- *Aggregation of Update Messages*: From a router's perspective there maybe k MHs associated with the same HA. In addition, there maybe a number of binding updates directed towards a given server. For example a busy server might be serving a large number of MHs and during high handoff rate a high number of binding updates will be directed towards the server.

This chapter is organised as follows. In Section 4.1 the router architecture and the ADS program structure are presented. The architecture presented is an abstract representation of an active router and it also represent the program layers in the simulation. After that, AN-based enhancements which improve the short comings of IETF mobile IP are presented in Section 4.2. Then experiments carried out to verify and quantify the performance of ADS are outlined in Section 4.3. Following from the results, a discussion of the related work and how ADS addresses problems and its differences to those in the literature are presented in Section 4.4. Finally conclusion is presented in Section 4.5.

4.1 Architecture

The router architecture used in the simulator is based on the generic router architecture illustrated in Figure 4.1. The implemented features in the simulation include, a classifier and an active program manager which provides packet/capsule manipulation services. Other modules such as TCP/IP and scheduler are provided by the simulator. All active packets with an ANEP [182] header are passed to the Active Program manager (APM). The main job of the APM is to figure out the execution environment in which the active program is to be loaded and also interact with the corresponding environment to load any necessary routines needed by the program.

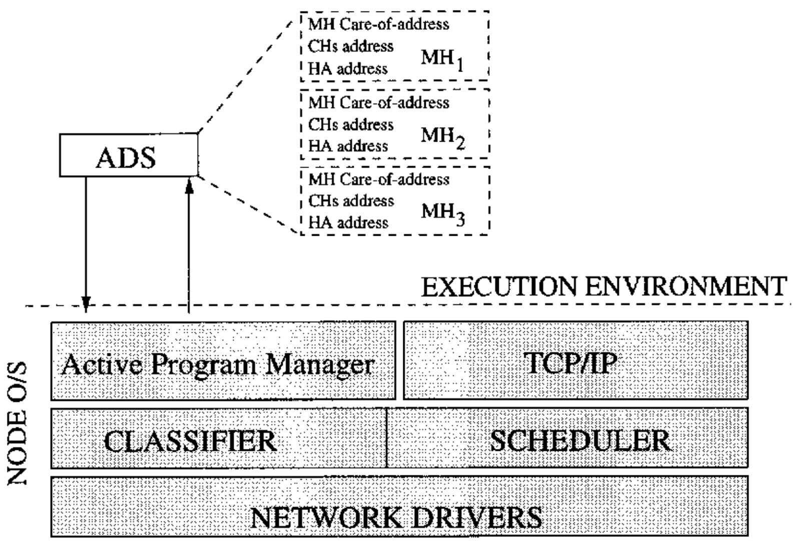


Figure 4.1: Active router's architecture.

In the simulation, a programmable switch approach is used where the program has already been loaded onto each router. Therefore, we only need to specify an active tag which corresponds to the ADS program to be loaded. When a packet arrives with an active tag, the APM looks up a table that maps the tag to the corresponding ADS program. The APM also handles requests by active programs. Generally, these requests are directed at the classifier. We envisaged the main services provided in an active node operating system include, packet/capsule manipulation (e.g., encapsulation, decapsulation, creation, etc), security services and soft-state creation. Security services are omitted in the simulation since this work is not concerned with security issues.

In ADS, packets going to the MH are of main interest and hence the classifier is notified of such packets. The ADS expects packets headed towards the MH as input, hence providing per-flow customisation. The output from an ADS program comprises data packets (possibly encapsulated), hop discovery capsule, location update messages and security related control messages.

The classifier is notified of the ADS's interest by specifying a tuple such as {dest=MH address, src=*, encapsulated=yes}. The classifier basically parses an incoming packet, looks up its list of registered tuples and any packets matching the set of registered tuples are forwarded to the corresponding active program. Otherwise the packet is passed to the TCP/IP module for normal processing. Note that as the MH changes its care-of-address the ADS would need to notify the classifier. If more than one active program is interested in a given packet, then a copy is made. More complicated approaches may involve pipeline processing where the packet is processed by one active program after another. In other words, the operation performed by one active program maybe negated by another, therefore, some form of cooperation or set of rules are required (perhaps by the node operating system). In the design of ADS, the above approach was not chosen because this is specific to the node operating system and at any one time no more than one process is interested in packets heading to the MH. Therefore, another copy of the packet is made and passed to the corresponding active programs.

Upon setup, each ADS creates a data structure which stores the mapping between the MH's home address and its care-of-address. Also stored is a list of CHs communicating with the MH and the MH's previous FA if any. This information allows the AR to tunnel packets to the MH and send binding updates. The data maintained by each ADS for a given MH is referred to as a *state* and is shown in Table 4.1. The corresponding *states* maintained for each CH are shown in Table 4.2. The ADS program is maintained using soft-state and a timer is used to determine its time of service. The timer is refreshed each time the ADS processes a packet.

Data	Size (Bytes)
AR Identification	8
MH's IP Address	4
MH's Care of Address	4
MH's HA address	4
CH addresses	N
Expiration Time	4
Security Association	8

Table 4.1: State: Data maintained by an AR for a given MH.

Only one ADS needs to be loaded at each router. A router with multiple instances is illustrated in Figure 4.1, where multiple instances of Table 4.1 are maintained by the ADS in order to conserve resources at ARs where only one ADS is loaded to handle all connections to/from the MH. As shown from Tables 4.1 and 4.2 the size of each *state* maintained by each ADS program per MH is $32 + (N \times 15)$ bytes where N is the

Data	Description	Size (Bytes)
Address	IP address of CH.	4
Update Status	This flag indicates whether a binding update has been sent.	1
Hops	The number of hops to the CH.	2
Latency	Observed packet latency.	4
Responsible AR	IP address of the AR responsible for sending binding request.	4

Table 4.2: CH data maintained by AR.

number of CHs. For example, an AR located at a baseball stadium with 10,000 fans, would be required to handle about 18.2 Mbytes of data if all fans each with a MH were were communicating with 10 CHs.

4.1.1 Implementation of ADS

The ADS program is injected into ARs using the programmable switch approach [107]. In ADS, ANEP header [154] is used to represent an active tag which determines whether to inject an ADS program or not. In the simulation when a packet with an ANEP header is received, it is passed to the APM. Currently, the header only contains a reference to the ADS program to be loaded. The ADS program is loaded into the AR during the registration reply phase of the handoff procedure. The handoff procedure will be elaborated further in Section 4.1.2. Once loaded, the ADS states its interest and then waits for the classifier to pass its packets. Packets processed by ADS are passed to the scheduler for transmission.

Each ADS program serving a given MH, tracks packets going to/from the MH and refreshes its soft-state timer each time data for the MH are processed. When a MH migrates, the loaded ADS might not intercept any packets directed towards the MH after time t (no larger than the registration lifetime) where the soft-timer expires and the ADS program for the given MH is removed. This enables ARs to deallocate resources and prevents misdirection of packets due to out-dated data. An alternative method is to unload ADS explicitly. After migration when a binding update is sent to the previous FA, each ADS program that processes the binding update checks whether it has processed a registration request recently. If not the ADS for the given MH is unloaded.

4.1.2 Distribution of the ADS Program

The ADS programs are distributed among ARs using one of the following methods. Firstly, ADS can be loaded through an active version of the registration request or through an active registration reply sent by the HA. The difference between these two methods is the location/point of distribution of the ADS program. Embedding ADS into the registration request message allows for a faster adaptation to mobility. This saves valuable time because while the MH is waiting for a registration reply, ARs are able to forward any in-transit packets destined to the MH and also inform CHs sooner. Secondly, the program could be embedded in the registration reply message where the HA is responsible for distributing the program itself. The second approach is taken because having a centralised point of delivery in this case allows ease of code management. The adaptation to mobility is dependent on the time it takes for the registration request to reach the HA and the time for the HA to program ARs. To enable a faster adaptation, the MH stores the program it receives from the HA during registration reply. Since, the programmable switch approach is taken, an active tag is used, thereby allowing the corresponding ADS program to be loaded as discussed in the previous section. As a result, when a MH initially migrates out from its domain, ARs are not programmed, thus there is no performance enhancement. During subsequent registration requests, possibly when migrating to another domain the MH embeds the program in its registration request. At this point, since some routers have already been programmed, the ADS takes action to improve the performance of Mobile IP.

The above ADS distribution process can be improved as follows. Instead of requiring the MH to store the program, an AR (with ADS loaded) that intercepts a registration request from the MH is responsible for programming ARs leading to the MH's new location. The HA has to program the network once only, that is after the MH migrates out of its domain. Subsequent programming operations are handled by ARs. After processing the registration request message, the AR sends a control packet which loads the ADS program in downstream routers. If a capsule approach is taken, the ADS program is sent in a capsule which gets loaded in downstream ARs. This has the benefit of transparency since only the HA needs to be aware of the programmability of the network.

The deployment of ADS is from a single source, namely the HA. The ADS users for a given domain can deploy their own version of active programs. Hence all MHs originating from different domains may use different AN-based programs without interfering with other MHs from different domains. Of importance in this issue is the backward compatibility of AN solutions and the Mobile IP models. Another advantage is that the HA may request for ADS from a known active programs repository. The benefit here is that third party active program developers do not need to scatter active services strategically around the network. As a result, no mechanism is required by the HA or any other deploying agent to locate these services. This is the main motivation for ADS.

4.2 Active Delivery Scheme

4.2.1 Overview

The approach proposed here assumes a network interconnected by ARs. These ARs are loaded with the ADS program as described in Section 4.1.2. The main functions of the ADS are:

- Reroute in-transit packets during handoff
- Handoff quickly and smoothly
- Send binding updates to CH
- Aggregate location update messages.

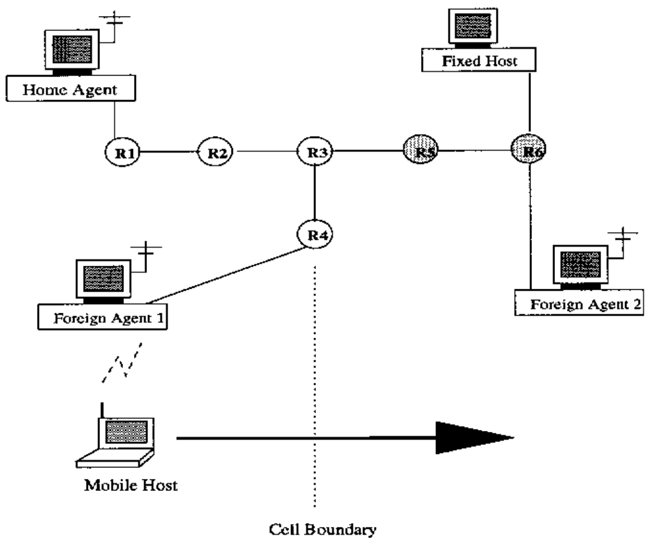


Figure 4.2: The setup of ARs before and after handoff.

Figure 4.2 gives an overview of how ADS enhances the implementation of Mobile IP. In Figure 4.2, R1 to R4 are programmed as described in Section 4.1. The MH is moving from FA_1 to FA_2. When the MH has migrated to FA_2 it sends an active registration request to inform its HA. As the active registration request traverses through ARs, the ADS program is loaded onto such routers as R5 and R6 which are not programmed *a priori*. At R3, no reprogramming is performed but the ADS program within R3 notes the MH's new care-of-address and forwards the active registration request. At this point in time any packets destined for the MH will be retunneled to the MH's new location by R3. In addition, a binding update and registration reply message is sent to the CH and MH respectively. Here, security associations between ARs, CHs, HA and the MH are assumed to exist. Thus, enabling the CH to tunnel packets directly to the MH's new location and reducing handoff latency. Besides that new CHs wishing to communicate with the MH have a faster connection setup time since ARs can redirect packets to the MH's current care-of-address without going through the HA. Security issues will be discussed further in Section 4.4.3. The ADS program at each AR keeps track of messages going to the HA and the number of hop counts to each CH. This enables efficient delivery of binding updates.

The main assumption taken here is that after migration, ARs along the route are able to intercept any packets destined to the MH. If for example the CH has a direct link to the HA that bypasses ARs then ADS poses no advantage. In such cases, the CH will receive its corresponding binding update when the registration request arrives at the AR local to the HA and therefore the binding update latency is reduced, assuming that the paths are symmetric: the paths taken by registration request and reply are the same. This assumption simplifies the update of ADSs when the registration reply passes through. Although in asymmetric network ADS could impose a hop-by-hop forwarding mechanism in which each AR knows its upstream and downstream ARs. This complicates the operation of ADS because an additional protocol is required to keep track of the programmed ARs between the HA and MH on a hop-by-hop basis.

4.2.2 Hierarchical Handoff

Previous work [183] has shown that MH migration is usually local (within a subnet, or a given domain). In the ADS handoff process, a registration reply is sent to the MH or receipt of a registration request is processed. Hence handoff latency is reduced since the MH does not rely on its HA. The ADS works both locally and globally; the ARs are dynamically set up to promote scalability. This is because as the MH migrates to a new domain, the local router gets programmed and handles all registration requests within the given domain. The central idea in ADS, as in cellular IP [37] is to shield mobility from CHs and HAs. In cellular IP [37], as long as MHs migrate within a domain maintained by the cellular IP protocol, the HAs/CHs do need to be updated. ADS aims to provide a generic mechanism in which the same functionalities can be achieved regardless of whether the migration is local or global.

When an ADS receives an active registration request, it sends a registration reply back to the MH confirming the handoff process. As a result, the handoff latency is not dependent on the delay between the MH's new location and its HA but is dependent on the delay between itself and the closest programmed AR (the first AR encountered by the active registration request which contains an ADS program for the given MH). Refering to Figure 4.2, R3 generates a registration reply to the MH once it has processed the active registration request.

In ADS, all active registration requests are relayed to the MH's HA. Alternatively, the ADS program could relay the request only if the MH has moved across subnets. As a result, the number of active registration requests directed towards the HA is reduced. The benefit of this feature is crucial in a pico cell environment where handoff rate is high. Due to the high mobility rate, the HA experiences a higher number of registration requests from the MH, hence increasing bandwidth used and HA load [1].

4.2.3 Packet Redirection

When a MH migrates, packets are redirected at each ADS during the migration itself. Simultaneously, the CHs are updated with the MH's new care-of-address. During this interval, ADS prevents packets from being forwarded to the wrong address. When the ADS receives an ordinary packet it checks whether the packet is going towards the correct destination. If the packet is headed to a wrong address then the packet is redirected. The check also determines whether the packet is encapsulated. If so, the packet is decapsulated and re-encapsulated with the MH's care-of-address and sent to the MH.

The forwarding process can be augmented with customised functionalities. For example when the MH has a high handoff rate or the MH is unreachable (due to non-overlapping cell areas), the ADS can be programmed to delay the delivery of messages until the MH becomes available. In this paper, no special policies are implemented.

4.2.4 Smooth Handoff

As mentioned, buffering at the previous FA might not be useful if the handoff latency plus the time for the binding update to get to the FA is large. This scenario happens when the MH in a WAN or the MH is not able to locate the next BS. For example, Ludwig et al. [184] reported an average latency of 592ms over global system for mobile communications (GSM) network. Suppose that a user was connected to an infrared link

[1]Here, assuming that there are no mobile switching centres and whenever the MH crosses a cell boundary an active registration request is generated.

previously, when the user exits his/her office and migrates to a GSM based network, the handoff latency will increase significantly. Furthermore, the window size used previously maybe inappropriate due to the reduction in bandwidth. If the original window size is used, the sender may overflow buffers at the BS causing packet loss.

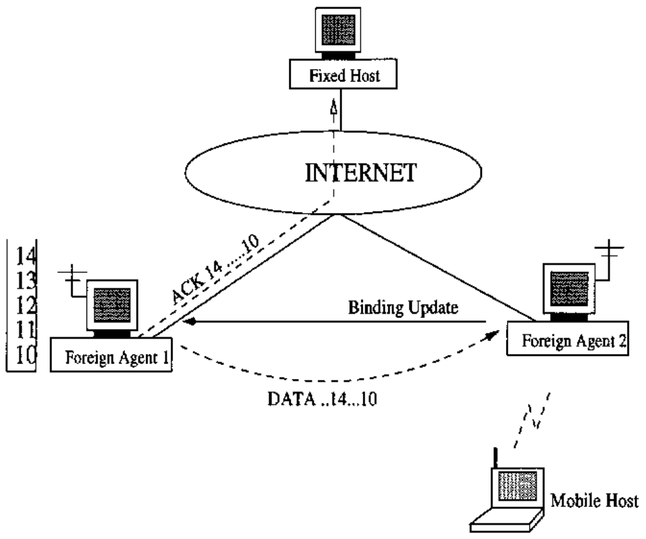


Figure 4.3: The SmartBuffer protocol.

To overcome the problem described above, the previous FA is programmed with SmartBuffer. SmartBuffer sends acknowledgements for packets buffered at the FA while waiting for binding updates from the MH. SmartBuffer monitors all data segments and acknowledgement packets going to and from the MH. SmartBuffer only sends acknowledgement for packets that have not been acknowledged due to migration and new packets that arrived after migration. In other words, when SmartBuffer detects a MH migration (explicitly notified by the MH as in soft handoff or when acknowledgements are overdue), it proceeds to act on behalf of the MH by sending acknowledgement packets. Note that here this study is not concerned with the characteristics of wireless links, and the problem is dealt from a network layer's point of view.

In Figure 4.3, it is shown that FA_1 has buffered packets numbered 10 to 14. SmartBuffer detects that the MH (when acknowledgment is overdue or notification from the MH in a soft-handoff scenario) has migrated and proceeds to send acknowledgements for the buffered packets. Once the binding update from FA_2 arrives, the buffered packets are forwarded to the MH. As a result, the sender is shielded from any delays in the handoff latency. Note that if the distance between FA_1 and FA_2 is large then the CH may send further packets to FA_1. Therefore, for each of the newly arrived packets an acknowledgement packet is sent to the CH. The main assumption here is that all link-layer packet losses at the wireless link are handled by an existing link-layer mechanism such as *snoop* [185].

SmartBuffer measures the RTT of acknowledgement packets across the wireless link as in [94]. The measured RTT is used to clock the rate at which acknowledgements are sent to the sender. This is needed to avoid the ACK compression problem [185], since TCP uses acknowledgement packets to determine the rate of outgoing packets.

The transfer process between FA_1 and FA_2 needs to be reliable because the data received at FA_1 would have been acknowledged. In the simulation model a simple go-back-N protocol between the FAs is used. TCP is not used because TCP's connection setup process would delay the forwarding of data. After migration, a new SmartBuffer program is installed at the current FA by the MH and the MH informs the SmartBuffer program of the next data segment it is expecting. Each newly arrived segment is checked and if the data segment is the next segment, the MH is expecting then the segment is forwarded to the MH. Otherwise, the data segment is buffered until packets from the previous FA have been forwarded onto the MH. Note that acknowledgement packets are generated by the MH as a result of receiving data forwarded from the previous FA. These acknowledgement packets are forwarded to the sender by the SmartBuffer program because acknowledgement packets from the previous FA to the sender might be lost. This does not cause the sender to reduce its congestion window since the sender only invokes fast retransmit and recovery after receiving three duplicate acknowledgements.

To avoid over-buffering at the new BS, an acknowledgement specifying the available receiver buffer size (i.e., the advertised window) is sent to the sender. By sending the allowed window size to the sender, excessive buffering at the BS can be avoided due to inappropriate window size at the sender. There are two ways in which the appropriate buffer size is determined. First, if a SmartBuffer program is running at the new BS, the average advertised window size used by other connections is used. The SmartBuffer program only needs to record the allowed window size from the acknowledgement packets. Therefore, the average window size can be easily calculated from connections managed by SmartBuffer. Secondly, if the SmartBuffer program is not installed, then the buffer size allocated by the active environment is used given that the window size advertised by the MH is larger. In other words, if the MH is requesting more than it should, then the advertised window size is replaced with the buffer size allocated by the active environment.

There are two ways in which SmartBuffer can be unloaded. In the first method, SmartBuffer is unloaded once it has forwarded its buffered data to the new FA. Secondly, when the MH's registration lifetime has expired, with SmartBuffer, the Mobile IP's registration lifetime is used to determine the time for unloading. The lifetime is refreshed each time the MH sends a registration request to renew its registration. At the new FA, the MH can choose to deactivate the SmartBuffer program after forwarding of data from the previous FA. Therefore, during normal operation no buffering is required. Before migration the SmartBuffer program is reactivated and the operations described above are carried out.

Another issue that needs to be addressed is when the previous FA goes down or the communication link between FAs is severed. This means that the new FA must determine a waiting time before concluding the previous BS has died. In this scenario, further investigations are required to determine the performance degradation due to loss of packets at the previous FA. Furthermore, we need to determine whether these data are recoverable from the sender since these data have already been acknowledged.

Depending on the mechanism used to detect handoff, the time in which SmartBuffer is injected or activated affects the RTT measured. To obtain an accurate measure, several samples are needed. Installing or activating SmartBuffer too early may disrupt the sending of acknowledgement packets. Therefore, during simulation SmartBuffer is partially deactivated after data from the previous FA have been forwarded. This means that SmartBuffer only measures the RTT of acknowledgement packets and does not buffer any data packets until the MH is ready to migrate.

4.2.5 Efficient Delivery of Binding Updates

In this section, an efficient binding update delivery scheme is presented. The idea here is similar to having a multicast like approach where the CHs are the receivers and the HA is the sender. Unlike multicast, the CHs are neither required to join a multicast group nor a multicast tree built. In ADS, each ADS program keeps a hop count to a given CH. During processing of the active registration request message, the ADS that is closest to a given CH generates a binding update message. As a result, the CH is notified earlier rather than waiting for the HA to notify the CH of the new care-of-address. Moreover, the ADS reduces load on the HA since the HA is not required to generate binding updates to the CHs and also the bandwidth consumed is reduced.

4.2.5.1 Hop count

Firstly, an explanation of how hops to each CH are measured by each AR. The number of hops observed is used by ADSs to determine whether they should send binding updates to a CH. Each router records the number of hops required to reach a given CH that is communicating with a MH. To calculate the number of hops, an Active Discovery Capsule (ADC) is used. ADC contains three pieces of information, hop count and security key[2] and *Resp_AR*. The security key is basically a security association between CH and routers, and serves to authenticate all binding update messages and most important of all, to re-register after migration. This will be explained in Section 4.2.5.3. The *Resp_AR* field contains the address of the AR which is responsible for sending a binding update during handoff. The details of which ARs take responsibility will be elaborated further in the next section. The capsule is sent by a CH wishing to receive binding updates sooner. During connection setup, the ADC can be piggybacked with setup messages, for example the synchronization message during the setup phase in TCP. The destination address of the packet containing the ADC capsule is the MH's current location. At each AR downstream from the CH, the code within the ADC increments the hop count value. Once an AR that contains a state for the given MH,

[2]e.g., public/private key pair.

the AR records the hop count value from the ADC, increments the hop count value before forwarding it towards the MH. Therefore, each router has a fair idea of the distance to the CH.

4.2.5.2 Which AR sends the binding updates?

In order to determine which ARs along the path from the HA to the MH are responsible for sending binding updates during handoff, the following algorithm is used:

Upon receipt of an ADC capsule, the AR checks whether it is the first to intercept the capsule by accessing the *Res_AR* field within the ADC. Note that the AR processing the ADC capsule must also be serving the corresponding MH.

```
  /* Get the CH's address from the capsule */
  CH_addr = Get CH address from the capsule
  /* If we have record, then hop count > 0.  Otherwise return */
  /* 9999 to denote no route */
  CH_hop       = Get_hop_count_to_CH(CH_addr)
  /* Get the AR address which is responsible for the given CH */
  R_AR_addr    = Get Resp_AR field from capsule
  /* Current hop count recorded */
  Recorded_Hop = ADC.hop_count
  IF R_AR_ADDR is empty THEN {
    /* This means we take the responsibility */
    Create a record for the CH (See Table 4.2)
    Set Responsible AR variable to local IP address
    Set I_SENT flag to local IP address
   }
  ELSE IF R_AR_ADDR == local IP address THEN {
    IF CH_hop == Recorded_Hop THEN
       ADC_Discard = TRUE
  }
  ELSE {
   /* Check whether we have a better route to CH_addr */
   IF Recorded_Hop < CH_hop THEN {
    Send an update to the address stored in the Responsible AR variable.
    Set Responsible AR variable to our IP address.
   }
 }
 IF NOT ADC_Discard THEN {
   Increase_hop_count(ADC.hop_count);
   Forward_Packet(ADC)
 } ELSE
     Remove_packet(ADC)
```

Algorithm 4.1: Determining hop count to each CH

Note that in Algorithm 4.1 we can simply assume that the responsible AR is the router that first intercepted the ADC and the ADC can be discarded after processing. Although the bandwidth and processing consumed can be conserved since the ADC does not need to travel all the way to the MH's care-of-address, but the hop count information will be outdated once the MH or CHs migrate. Apart from that Algorithm 4.1 can be augmented to include other parameters such as load/traffic. For example if the AR that intercepts the ADC has a high load, then it can choose to be not responsible for the CH. Hence the downstream AR will handle the capsule. To determine whether a shorter route exists, the AR searches its list of CHs (maintain for each MH) for the CH's address. If a record exists and contains a smaller hop count than the advertised hop count, the current AR takes responsibility. Once a new AR takes responsibility, the previously responsible AR is notified by the message CHANGE_AR. The message CHANGE_AR contains the address of the AR that is taking over the responsibility for a given CH. When an AR receives a CHANGE_AR, the AR resets the variable *Responsible AR.* This also includes intermediate routers that intercepted the CHANGE_AR message.

Example

To illustrate binding update delivery scheme further, consider Figure 4.4. For this example, assume that the MH and CH_1 have not migrated to the BSs shown in Figure 4.4. Also, except for CH_1, all CHs reside at the corresponding FAs shown in Figure 4.4.

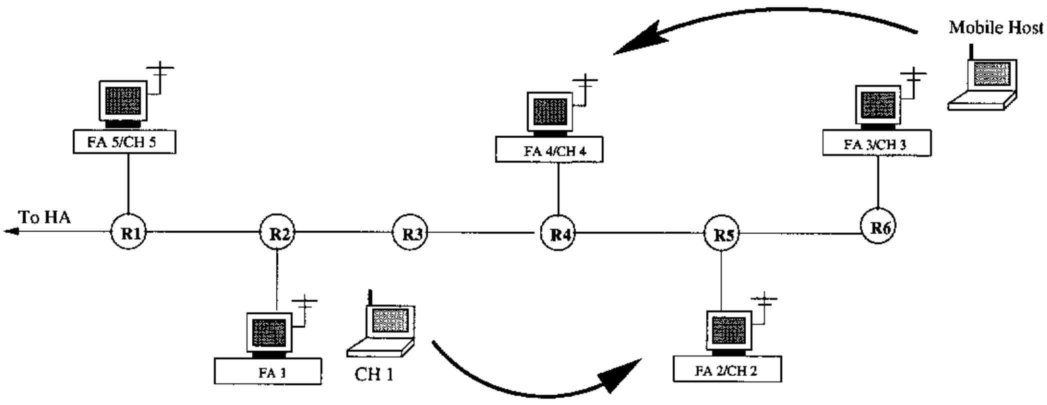


Figure 4.4: Example topology, a MH communicating with four fixed hosts.

Router	CH_1	CH_2	CH_3	CH_4	CH_5
1	None	None	None	None	1
2	1	None	None	None	2
3	2	None	None	None	3
4	3	None	None	1	4
5	3	1	None	2	5
6	4	2	1	2	6

Table 4.3: Hops information for CHs maintained by each AR.

In Table 4.3 we see that upstream ARs have shorter hop(s) to a given CH (given the MH's current location). For example the number of hop counts from AR_3 to CH_1 is two whereas that from AR_2 to CH_1 is one. When a registration request is received, an AR checks its list of CHs and generates binding updates only to those CHs it is responsible for. If an AR observed an alternative route to a given CH a control capsule is then sent to the responsible AR (possibly an upstream AR). This scheme is passive in that ARs do not actively exchange control messages and do not use an elaborate discovery mechanism. Hence low overheads are observed. In the event of ARs observing the same number of hops, the upstream AR is used.

4.2.5.3 MH Migration

When the MH migrates, the router previously responsible for the CH may no longer be valid. Therefore, an update mechanism is required to determine whether the responsibility roles have changed. In ADS, the update process is initiated during the handoff process. In other words, when registration request is received or upon receipt of a binding update sent to the previous BS. When any of these messages are received, a binding update is sent towards the CH in which it has responsibility. In addition, each CH is informed to generate an ADC towards the MH's new location. Also, in the ADC, the local IP address is included, which is echoed in the ADC generated by the CH. When a router serving the MH intercepts the new ADC, the ADC is authenticated. Algorithm 4.1 is then executed to determine whether the roles have changed. If the *Resp AR* variable in the ADC is similar to that of the local IP address, then no roles have changed. Hence the ADC is discarded. On the other hand, if the *Resp AR* variable does not match the local IP address, and the recorded hop count is smaller, then an invalidate message is sent to the router recorded in the *Responsible AR*. Here the *Responsible AR* could be empty when the MH migrates upstream, for example in Table 4.3, AR_1 does not have an entry for CH_1 to CH_4.

In Figure 4.4 the MH migrates to FA_4. As a result, registration request will be sent to the HA and also a binding update generated to FA_3. Referring to Table 4.3, before handoff AR_5 has a one hop count to CH_2. After migration the responsible ARs for CH_2 and CH_3 have to change. Note that upon handoff, the corresponding states for the given MH at R_5 and R_6 are unloaded. Therefore, the responsible AR after handoff is R_4. After receiving new ADCs, the hops information shown previously in Table 4.3 is updated to that of Table 4.4.

Router	CH_1	CH_2	CH_3	CH_4	CH_5
1	None	None	None	None	1
2	1	None	None	None	2
3	2	None	None	None	3
4	3	2	3	1	4
5	None	None	None	None	None
6	None	None	None	None	None

Table 4.4: Hops Information for CHs maintained by each AR after MH migration.

4.2.5.4 CH Migration

In the case of CH migration, the responsible AR is likely to change. In Figure 4.4, given that the MH has migrated to FA_4, CH_1 migrates from FA_1 to FA_2. As a result, the responsible AR has to change from R_2 to R_4. After migration the CH sends a new ADC to the MH's current location. Since the hop count recorded in the ADC will be lower than that of the recorded by R_4, the responsible router for CH_1 is R_4. Upon receiving the ADC, R_4 accesses the *Responsible AR* variable to obtain the address of the previous responsible AR. A CHANGE_AR is then transmitted, resulting in the reseting of *Responsible AR* variables in routers R_3 and R_2. The hop count information after both MH and CH_1 migrate is shown in Table 4.5.

Router	CH_1	CH_2	CH_3	CH_4	CH_5
1	None	None	None	None	1
2	None	None	None	None	2
3	None	None	None	None	3
4	2	2	3	1	4
5	None	None	None	None	None
6	None	None	None	None	None

Table 4.5: Hops Information for CHs after MH and CH 1 migrated.

4.2.6 The BusStation Framework

A novel feature here is the aggregation of binding updates and registration requests through the use an adaptive BusStation framework. The idea of the BusStation framework is similar to a bus in real life, thus the name BusStation. Note that the term 'bus' is due to the similarity of the operation of buses in real life and it does not refer to the definition of bus in the computer communications context. The bus in ADS is represented as a capsule containing multiple location update messages. Each AR acts as a BusStation where buses wait for a particular time. While the bus is waiting at an AR, it takes on passengers (i.e, location update messages) and it moves on to the next station downstream when the waiting time expires. The details of each aspect of the BusStation framework will be presented in the next section.

To demonstrate the BusStation framework, its applications to the aggregation of location update messages will be demonstrated. The rationale for aggregating location update messages is based on the likely possibility that in the near future as MHs become ubiquitous, Internet's traffic would constitute a sizeable number of update messages. The objective here is to reduce the number of packets forwarded to HA or CHs. The direct consequence of this is the reduction of load at routers and conservation of bandwidth. Badrinath et al. [172] have shown the benefits of aggregating small packets. This observation can be applied to registration requests as well since each registration request size is only 28 bytes. As registration requests are handled by ADS, MHs do not depend on their HAs for fast handoff and CHs need not wait for HAs to send them binding updates. In other words, a delay in location update messages does not reduce the performance of Mobile IP given that ADS is deployed. The aggregation of binding updates is more suitable for CHs with a high number of MHs, for example a web server. A high mobility rate results in the flooding of binding updates towards the CH which severely increases its load. However, this scheme might not be useful when the number of location update messages are low. In the simulation, a scheme in which each AR tracks the rate of incoming location update messages and decides whether to invoke the BusStation framework described later was implemented. Hence only heavily loaded ARs are required to activate this option. Besides that users are able to specify in the message itself whether they want their messages aggregated. For example, a user might not have ADS enabled.

For brevity, this section will only describe the procedure for aggregating registration requests. Note that the same procedure applies to binding updates. When the BusStation is enabled, the ADS invokes a program called ABus. The ABus program described in following the paragraph basically tracks all location update messages going to the same HA. Upon arrival, a location update message is processed by the ADS and passed to the ABus. The ABus then determines whether there is a bus waiting to take packets to the given HA. If a bus exist then the packet is 'loaded' onto the bus. Otherwise a new bus is created and the packet is 'loaded'. A bus is associated with a waiting timer. This determines how long a given bus waits for location update messages. Once the waiting timer expires, the bus takes off and it moves to the next hop to pick up packets. When the bus reaches the AR local to the HA, the bus unloads its 'passengers'. Alternatively if the HA supports ABus then the bus is unloaded at the HA. As a result, the number of packets traversing towards the HA is reduced.

The use of the BusStation framework results in location update messages not being directed to their appropriate ADS program. Since the bus may contain multiple location update messages with associated ADS programs the ABus program needs to update the corresponding ADS programs on behalf of each location update message. This basically involves enquiring the APL for the associated ADSs given the MH's address. In the simulation, the APL returns the object instance of the matching ADS. The ABus then calls the update state procedure and passes it the information within the location update message. This is repeated for each packet in the bus. Therefore, the bus has a minimum waiting time due the update operations. While the operation is in progress other processed location update messages can board the bus.

4.2.6.1 The Bus

The bus created by the ABus program is shown in Table 4.6. Basically the bus encapsulates a sequence of packets. The header information is used by the ABus program to manage buses. Once the bus is created ABus allocates a slot for it. Assuming a bus size of 576 bytes, a maximum of 19 registration requests (passengers) with 28 bytes per registration request is possible. If this limit is exceeded a new bus is created and the full bus's status is updated to express, which means it does not stop at any node. The express bus is then forwarded to its destination. Note that each bus has a maximum waiting time and age parameters. These parameters define the thresholds in which the bus waits at a particular AR and the upper limit on the packet latency.

The algorithm executed at the ABus when an location update message or a Bus arrives is shown below:

```
wt = Calculate_Waiting_Time_For(HA address)
IF LOCATION_UPDATE_MESSAGE THEN {
  Dest = Get_Destination_From_Packet(HA address)
  IF Exist_Bus_Going_To(Dest) THEN {
    Load_Packet_Into_Bus(Bus_No, LOCATION_UPDATE_MESSAGE)
  } ELSE
     {
        Create_New_Bus_To(Dest, wt)
        Load_Packet_onto_Bus(Bus_No, LOCATION_UPDATE_MESSAGE)
     }
}
IF BUS and NOT express mode THEN {
   Dest = Get_Destination_From_Bus_Header(Bus)
   MaxWait = Get_Max_Wait_Time(Bus)
   IF NOT Exist_Bus_Going_To(Dest) AND wt < MaxWait THEN {
     Install_Bus_At_Slot_No(Dest, wt)
   } ELSE forward bus to next hop
}
```

Algorithm 4.2: Pseudocode for ABus Program

Field	Size (Bytes)	Description
Destination Address	4	Uses by ABus program to determine where bus is headed.
Age	2	How long should the bus stays in the network.
Size	1	The bus's size.
Status	1	1 means express 0 otherwise.
Max Wait Time	2	The maximum time it waits at a given node.
Seat Numbers	4	The number of packets in the bus.
Passengers	$N \times 28$	The packets loaded onto the bus. 28 here means the size of of registration request

Table 4.6: ABus packet information.

4.2.6.2 Bus Management

As mentioned each bus is allocated a slot by the ABus program. This is implemented as a hash table where the destination address is the key. When a bus arrives ABus checks to see whether a bus exists, if not then a bus is inserted in its slot. A waiting timer is assigned to the bus. When the timer expires the bus is forwarded to the next hop.

4.2.6.3 Waiting Time

The waiting time affects registration request's latency and the number of pickups by a given bus. If the waiting time is too long then the bus might overflow and a new bus is created. Thus, increasing the number of packets heading towards the HA. Furthermore, the registration request's lifetime might expire before the bus reaches its destination. The objective here is to find a tradeoff between the reduction of packets directed towards the HA and the waiting time.

The waiting time is calculated based on the following policy. The waiting time for a given bus is dependent on the rate of arrival of registration requests. In other words each AR has to monitor the frequency of registration requests heading toward a particular destination. It then calculates the rate of registration requests for each HA. This rate is then used to calculate the waiting time for a given bus. If the rate is high the bus's waiting time is longer and vice-versa. A different view can also be undertaken whereby given a high rate the waiting time is reduced given that the rate in which the bus is filled is high, therefore less time is needed. In ABus, the first approach is taken because it was found that the first approach results in higher number of aggregated packets.

The calculation of the waiting time is similar to that of TCP's calculation of RTT [89]. The difference here is that a waiting time that maximizes packet reductions given different arrival rates is considered. The formula for waiting time and inter-arrival time are as follows:

$$\text{Update Inter-arrival Rate (IntA)} = \alpha IntA + (1 - \alpha) IntA_{old} \tag{4.1}$$

$$\text{Waiting Time} = \alpha \left(\frac{Capacity}{IntA_{new}} \right) + (1 - \alpha) IntA_{old} \tag{4.2}$$

The variable $IntA_{new}$ and $IntA_{old}$ in equation 4.2 refers to the new and old inter arrival rate ((from equation 4.1) of registration requests respectively. α is a smoothing factor which determines how much weight is given to the old inter-arrival time. Capacity refers to the bus's seating capacity. The seating capacity is used because a bus's capacity determines how long it waits. If it has more available space then it is likely to wait longer.

During simulation, *IntA* is calculated at each AR for each destination (i.e HA). Each HA's *IntA* and its current waiting time is kept in a hash table with the destination as an index. Each AR will have a different set of *IntAs* and waiting time. Therefore, as the bus traverses through the network its waiting time varies as it passes through ARs with different location update message arrival rates. When a bus arrives no calculation of $IntA$ is required since it does not reflect the rate of registration requests and packets are not transferred from one bus to another. The associated waiting time for the bus is then accessed and equation 4.2 is used to calculate the waiting time.

To reduce the states maintained at each AR, an expiration timer is associated with the *IntA* and waiting time of each destination. Due to MH migrations the rate of registration requests at the AR vary over time. As a result, registration requests from a given MH might not pass through an AR again, therefore its state needs to be removed. This improves on the scalability since there is no need to keep track of all MHs currently in foreign networks. The timer is set to one second in the simulation. When this timer expires the state for the destination is removed. When there are new registration requests then the rate and corresponding waiting time are recalculated for the given HA.

4.3 Evaluation

To investigate the performance of Mobile IP coupled with ADS the *ns-2* simulation package developed at Lawrence Berkeley National Laboratory [186] is used. In the simulator the Mobile IP code[3] (version 1.4) from the *ns-2* distribution was used and augmented the code to include optimized Mobile IP. The full functionalities as presented in [11] were not implemented. The augmentation only enables CHs to receive binding update, which result in the loading of the tunneling code that encapsulates all outgoing packets from the CH. The topology used in the simulation is shown in Figure 4.5. All links interconnecting ARs (R1 to R10) and end-hosts are set to 10Mbps and the wireless link has a data rate of 2Mbps. Depending on the experiment involved, the MHs are set to migrate at a predetermined time to either FA_1 or FA_2. In all experiments, MH_1 is designated as the CH. Except for Experiment 3, MH_1 remains stationary for the duration of the simulation. Configuration parameters and topology variation that are specific to a given experiment are explained later in this section.

In the simulation studies, a comparison between the performance of TCP (NewReno) over Original Mobile IP, Optimized Mobile IP and ADS was carried out. One objective of the simulations is to determine whether ADS provides any benefits to TCP's throughput due to handoff. Interested readers not familiar with the behavior of TCP in a heterogeneous network are referred to [86, 100]. Further details concerning different flavors of TCP and performance studies on multiple packet loss can be found in [187].

[3]Included in *ns-2* distribution

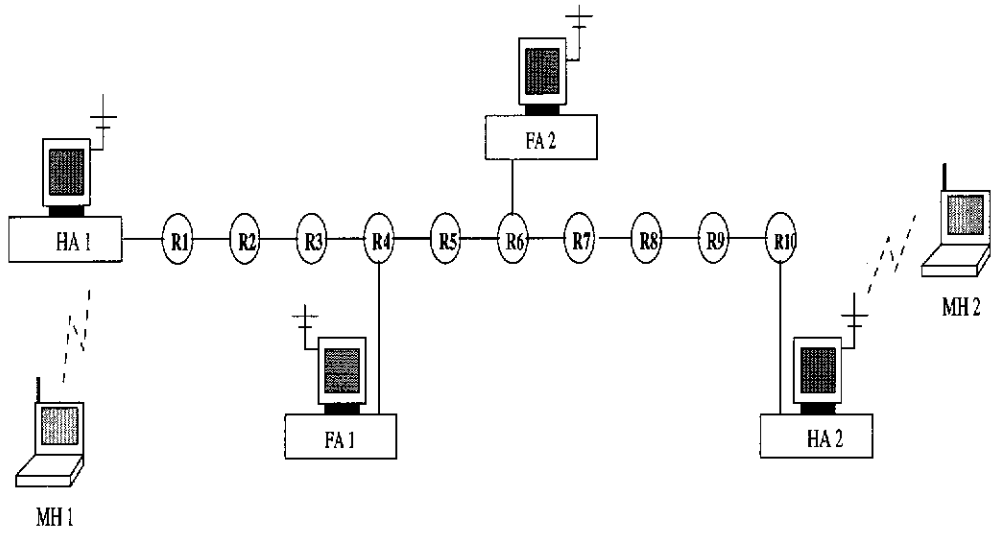


Figure 4.5: Network topology used for simulation.

The active delivery process was tested on on random mesh and hierarchical topologies. In the mesh topology there are 25 nodes connected randomly with a degree of two. In the hierarchical topology a binary tree with four levels is considered. At each simulation run, CHs, FAs and the HA are connected randomly to the topologies considered. Therefore, the MH experiences a combination of inter and intra domain handoff. Only one MH migrates at any one time and migration time is set randomly for each run. At each handoff the number of hops required to send the binding updates from an AR and the update latency for each CH are recorded. The update latency is the time taken for the binding update message to reach a given CH from the time of handoff.

Data	Description
Average Hops to CH	This is the average number of hops observed by an AR to a given CH.
Binding Update Latency	The time interval between handoff and the receipt of binding update at the CH.
Binding Update Savings	Percentage of the number of binding updates sent over the number of CHs in the list.

Table 4.7: Simulation data collected for active delivery.

The topology shown in Figure 4.5 was used to simulate the BusStation framework. Instead of HAs, MHs and FAs the network is connected by traffic generators. These traffic generators are responsible for generating different registration request rates (30 distinct HA identifications). All the generators generate packets of size 28 bytes with an exponential distribution. The burst and idle time is set to 30ms and 1ms respectively. A *sink* is connected to **R10** which counts the number of buses and registration requests and also other statistics such as packet latencies. Apart from that each generator node has a TCP session. This creates the cross-traffic on the network and we also investigated the impact of aggregating ARRs on end-to-end latency of TCP's traffic. To study the benefits of the BusStation option we performed two experiments. Firstly we investigate the impact of constant waiting time given varying traffic rates. In this experiment all generators are set to transmit at rates ranging from 20 to 200 kilobytes. The second experiment investigates the use of the adaptive waiting time with the rates shown in Table 4.8.

Connection	Rate (Kilobytes)
R1	40
R3	100
R5	60
R7	200
R9	10

Table 4.8: *Configuration of traffic generators for BusStation simulation.*

The following sections present the results obtained from the exhaustive simulation studies. These results show that TCP recovers faster with the use of ADS. Moreover the performance of ADS's binding update delivery mechanism is analysed. Note that in each experiment, a specific functionality of ADS is investigated. For example in Experiments 1 and 2, the SmartBuffer scheme was not incorporated. This enables independent evaluation of each ADS feature.

4.3.1 Experiment 1: LAN Migration

In this simulation the MH is set to migrate to a new cell at $t = 50.0$. The persistent source is set to transfer 80 packets each of size 536 bytes from the CH to the MH [4].

[4]In the simulations no variables are dynamic or random, hence these simulations runs are repeatable. Furthermore, similar behaviors are observed given different handoff time.

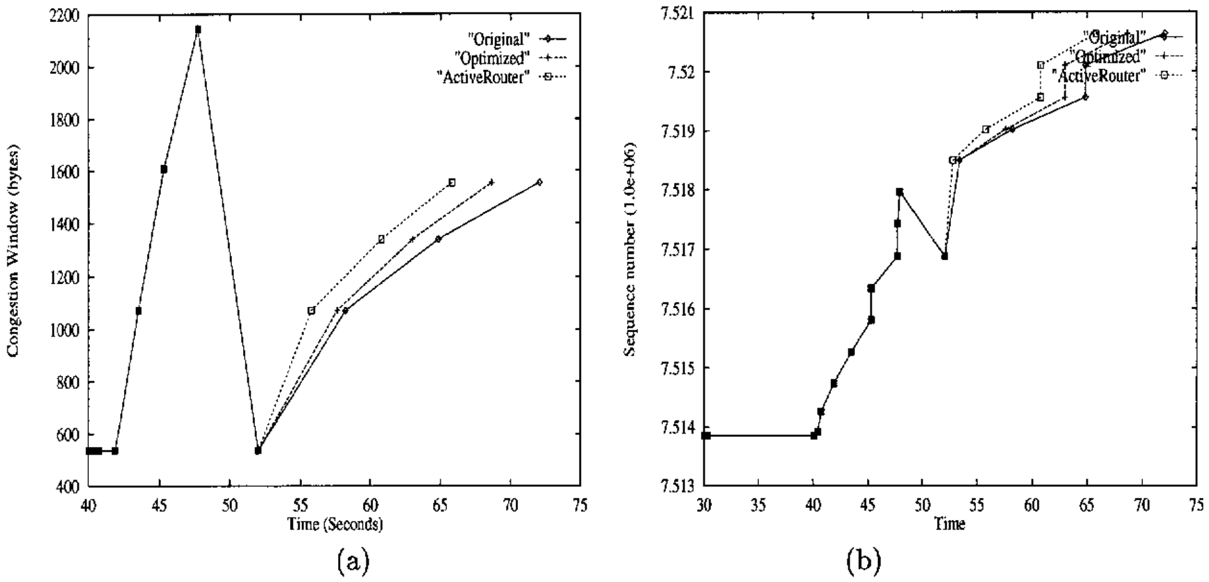


Figure 4.6: Congestion window size and sent sequence number for Experiment 1.

From Figures 4.6(a) and (b), it can be seen that ARs perform well above the mobile IP models since the CH and the MH no longer rely on the HA for the care-of-address binding update. As can be seen from the figures, at time $t = 45$ TCP timeouts which causes the reduction in congestion window in response to congestion in the network [89]. However in our experiment, the packets were dropped due to handoff instead of congestion. In slow start mode, the rate at which TCP recovers depend on the rate in which acknowledgement packets are returned and the number of packets lost. Figure 4.6 shows that TCP slow-start recovers fastest when the ARs are programmed with ADS. A measure of the improvement is shown by the sequence numbers increasing faster in the ARs curve than it does with either the optimised or original Mobile IP curves. This is mainly due to the ADS redirecting packets sooner, thus reducing the number of packets lost at the previous BS. As we shall later, the redirection of acknowledgement packets is crucial when both communicating hosts are mobile. This is because the sender is notified earlier, thus indicating to the sender that packets are getting through. This means that congestion has subside and the sender can start transmitting new packets onto the network.

Figure 4.6(a) also shows that Mobile IP (labelled Original) does not perform well since the CH needs to send each packet to the HA where it is tunneled to the MH. This causes unnecessary traffic which is inefficient particularly when the MH is located near the CH, meaning that longer RTT induced delays may occur. Moreover congested link to the HA can be avoided. The congestion window for Mobile IP increases at $t = 58.2$, a delay caused by TCP's exponential backoff. In the optimised Mobile IP, the congestion window starts increasing after $t = 57.6$ and for the ARs approach the congestion window advances at $t = 55.8$. This faster recovery enables the sender to increase its sequence number sooner which results in an overall improvement over the other Mobile IP models. The time taken to transmit 80 packets each of size 536 bytes for the three models is different. The ARs scheme has the highest throughput due to faster recovery after handoff. In this simulation the CH gets updated with the new

	Mean	Standard Deviation
Original	1.43	1.0
Optimised	1.35	0.87
Active Router	1.29	0.77

Table 4.9: RTT mean and standard deviation.

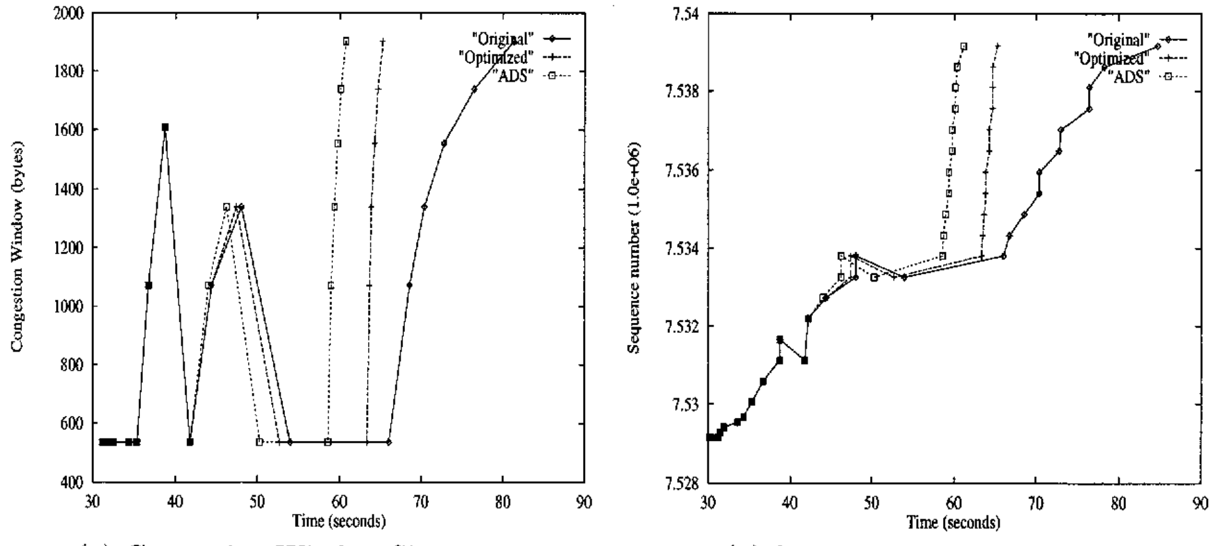
care-of-address, 2.9ms after handoff for the ARs approach, and 9.1ms for the optimised Mobile IP. No binding update is sent to the CH in the original Mobile IP specification. Handoff takes 2.8ms using in ADS and 7.4ms for the Mobile IP models.

4.3.2 Experiment 2: WAN Migration

This experiment considers the migration of a MH in a WAN. The network topology is modified as follows. The links between R10 and HA_1, and R4 and R5 are modified to a 2Mbps link.

MH_2 is set to migrate from its home network to FA_2 at $t = 40$ (where t correspond to simulation time unit). MH_2 stays at FA_2 for 10 seconds before it migrates to FA_1. When MH_2 has migrated to FA_1, its registration request is intercepted by R6 which in turn sends a registration reply. In the other models for Mobile IP the HA handles the MH's requests which is clearly inefficient in this topology since there is a large distance between the MH and the CH. Furthermore, since the HA sends a binding update to MH_1 after it has migrated, the binding update is delayed by the time it takes for the registration request to reach the HA plus the time for the binding update to reach MH_1. In ADS the binding updates (for FA_2 and MH_1) are generated once R6 receives a registration request. Hence the updates are generated at ARs that are closest to the CHs and old FA. Note that after migration to FA_1 any further migration within the FA_1 subnet will be handled by R4. The AR model achieves a significant improvement in performance by generating a binding update to the CH. A measure of the improvement can be seen in Table 4.10, with R6 sending the binding update, the CH gets updated 4ms sooner.

Table 4.10 shows the update time and registration time compared to the ARs approach. The registration and update time for both the original and optimised Mobile IP are the same. A better solution is to have the R4 handle the handoff and proceed to send an update to CH and registration reply to MH. The original registration request is forwarded onto the HA. Therefore, the CH is updated sooner and handoff time is greatly reduced. In experiment 2, the handoff times are chosen arbitrarily. In the simulation at $t = 40$ the MH moves to a predetermined location. This does not mean that the MH has to be at the final location before a radio connection is established. The handoff time in Table 4.10 does not include the latency of radio establishment. It is latency from the time the MH sends out a registration request to the receipt of registration reply. Figure 4.7(a) shows two drops in the congestion window, at $t = 40$



(a) Congestion Window Size (b) Sequence Number for WAN Migration

Figure 4.7: The MH is set to handoff at $t = 40$ and $t = 50$.

Time	Original	Optimised	Active Routers
Handoff 1	6.4ms	6.94ms	2.30ms
Handoff 2	39.22ms	11.30ms	2.78ms
CH Update 1	2009ms	2009ms	2005ms
CH Update 2	12ms	10ms	0ms
FA_1 Update	10ms	10ms	20ms

Table 4.10: Handoff and binding update latencies for WAN migration.

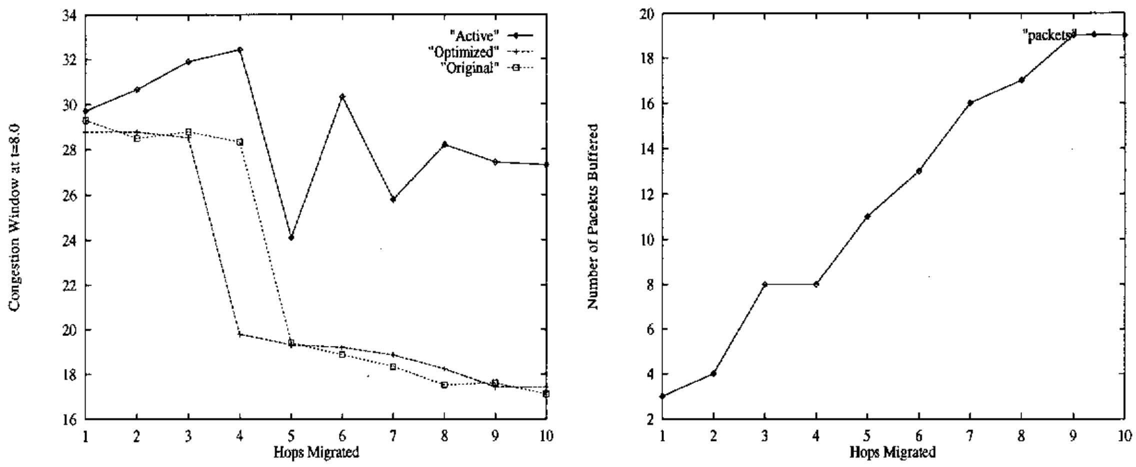
and $t = 50$. In both cases the AR approach recovers faster compared to the other two approaches. It is interesting to note that in the first handoff, the congestion window in the ARs approach recovers faster. This is because R6 sends a binding update and registration reply sooner than the Mobile IP protocols.

4.3.3 Experiment 3: Migration Factor

In this experiment the resulting performance when the MH migrates further away from its HA is presented. For each simulation run the FA's distance (hops) is increased while the CH is fixed. Although the MH is migrating further away from its HA, it does not necessarily mean that it is migrating closer to the CH. Figure 4.8 shows a snapshot of the congestion window size at $t = 8.0$. The time of handoff is $t = 5.0$. As the MH migrates further away from its HA, TCP's congestion window for both the Mobile IP models takes longer to recover. Due to rerouting of in-transit packets and earlier notification of the CH TCP recovers faster with ADS. Figures 4.8(a) and (b) show the congestion window size and packets buffered at the previous FA when the MH increasingly migrates further away from its HA. The details of buffering are explained in section 4.3.5.

4.3.4 Experiment 4: Impact of CH's Migration

In this experiment the detrimental effects of CH migration are shown. This is where the MH is communicating with another MH across the fixed network. In general, the distance between the MH and CH (mobile) either decreases or increases. In either case the sender's retransmission timeout value becomes invalid. This is because the data



(a) Congestion Window at t=8.0 (b) Number of Packets Buffered

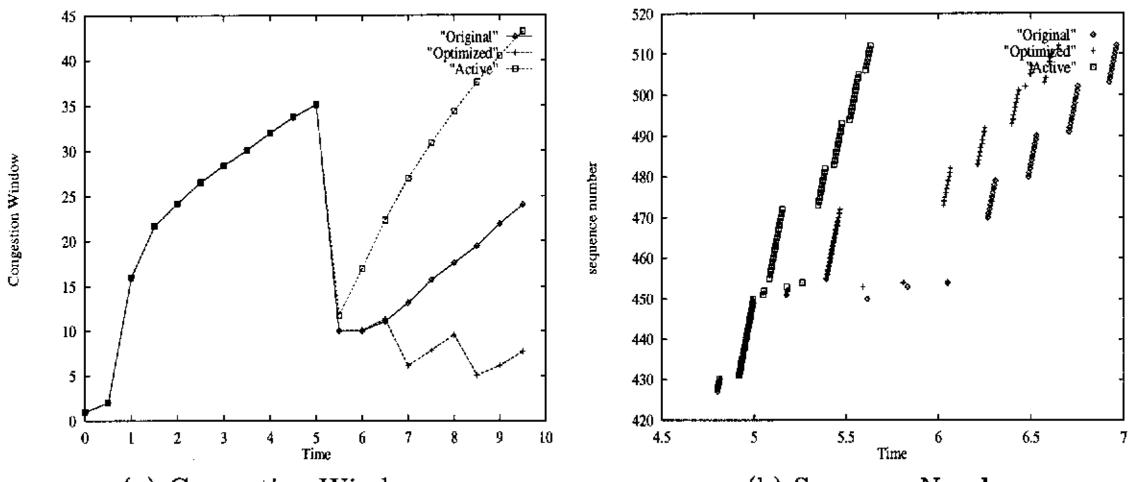
Figure 4.8: Scenario at previous FA when the MH migrates further away from its HA.

segments and acknowledgement packets that have to be rerouted through the HAs of each MH may cause multiple retransmission timeouts. In the optimized Mobile IP case, delay in binding updates may cause the loss of acknowledgements which results in waiting for a retransmission timer expiration before re-initiation of data flow.

The migration time for the CH and MH is set at t=5.0. In this scenario both data segments and acknowledgement packets may experience losses. The performance of TCP in this scenario for both Mobile IP models and ADS is shown in Figure 4.9. As can be seen both Mobile IP models recover poorly. This is due to the reasons discussed above. In Figure 4.9(b) it can be seen that only two packets need to be retransmitted with ADS. For the original Mobile IP it only manages to recover after t = 5.5. That is after both HAs have received registration requests from their respective MHs. For the optimized Mobile IP, due to delays in binding updates some packets are lost at the MH's previous location (packets 452 and 453), thus it takes longer for TCP to exit from its fast recovery phase.

4.3.5 Experiment 5: SmartBuffer

In this experiment, the SmartBuffer scheme was implemented and tested with varying handoff latency. The optimized Mobile IP is augmented with buffering as proposed in [33]. Figure 4.10(a) shows the congestion window of the sender when the beacon period

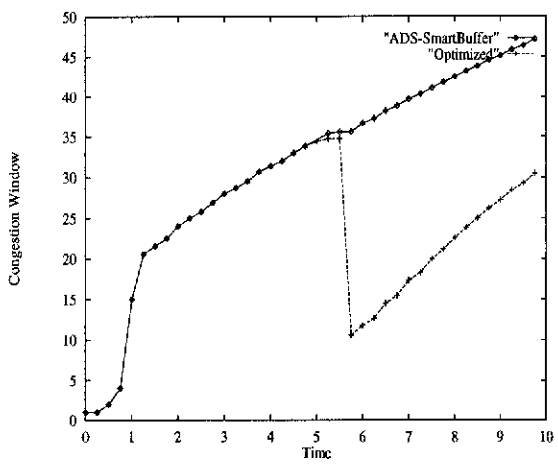


(a) Congestion Window (b) Sequence Numbers

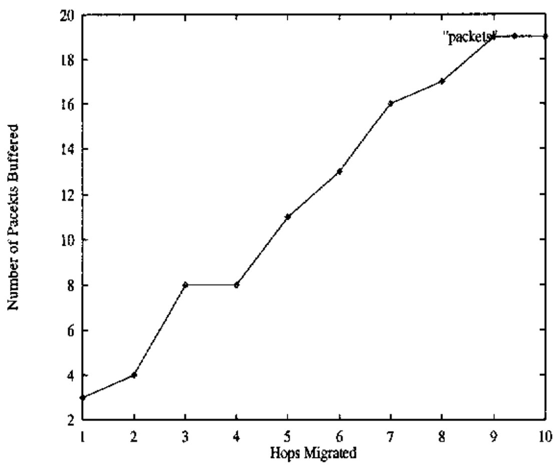
Figure 4.9: Mobile CH

at the BS is set to one second (as recommended in the Mobile IP specifications [10]). As can be seen, ADS is able to avoid reduction in congestion window due to handoff. Hence ADS using SmartBuffer is more resilient to the effects of migration such as high handoff latency. Here the network stores and forwards packets to MH only after determining its address. From an end-host point of view, the handoff latency is transparent. In the optimized Mobile IP, a retransmission timeout occurs due to the high handoff latency. Use of SmartBuffer results in the following features, (i) CHs are not susceptible to high handoff latency and (ii) no feedback messages to sender are required to adjust transmission. Since the handoff rate is transient in nature, sending feedback to the source may prove premature. It can be seen that programmability can help the performance of applications during this temporary disruption in communication.

Here, another investigation concerning buffer requirements in relation to the distance (hops) between the previous FA and the current FA. In the experiments, the average packets buffered are between four and five. However, it is interesting to quantify the buffer requirements when the MH migrates increasingly away. This is shown in Figure 4.10(b). At ten hops the number of packets buffered is 19. The number of packets buffered is determined by the time it takes for the binding update message to reach the previous FA. Therefore, there are two aspects that affect the source's transmission: (i) time it takes for the MH to re-establish radio communications with the new FA and (ii) the delay between the current and previous FA. In both cases SmartBuffer is able



(a) Congestion Window



(b) Number of Packets Buffered

Figure 4.10: ADS with SmartBuffer. MH migrates at t=5.0 and packet size is 512 bytes

to prevent any disruptions to the flow of packets.

4.3.6 Experiment 5: Delivery of Binding Updates

In this experiment the efficiency achieved by having ARs monitor and send binding updates appropriately is quantified. The average hop count recorded by ARs and the average latencies observed by CHs are shown in Tables 4.11 and 4.12. From the tables it can be seen that for both topologies ARs perform better than optimized Mobile IP.

Schemes	Average Hops	Average Latencies (seconds)
AR	2.9	0.22
Optimized	5.9	0.38

Table 4.11: Randomized mesh topology.

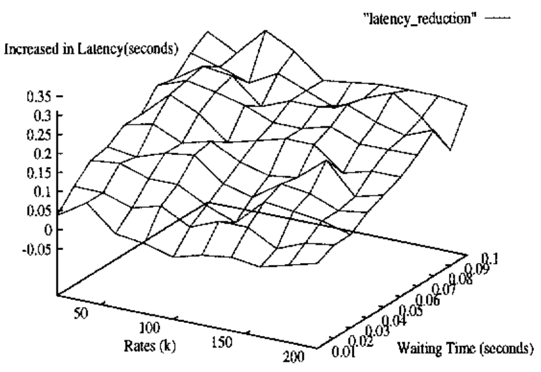
Schemes	Average Hops	Average Latencies (seconds)
AR	1.43	0.18
Optimized	4.9	0.81

Table 4.12: Hierarchical topology.

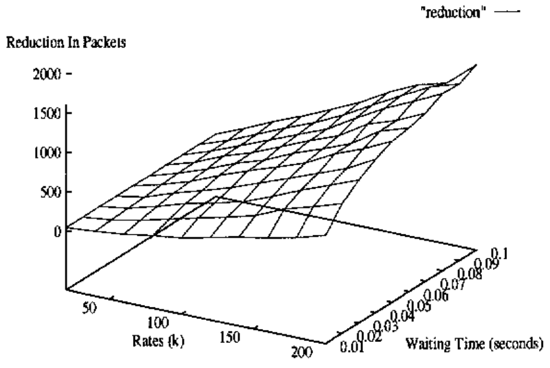
4.3.7 Experiment 6: The Benefits of Aggregating Registration Requests

This experiment investigates the benefits of enabling the BusStation option. Firstly, the effects of varying waiting time is studied. Note that the waiting time here is not calculated based on equation 4.2. The reason being, the amount of reduction in packets that can be achieved and its impact on end-to-end latencies need to be quantified. The results of this experiment are shown in Figures 4.11(a) and (b). Secondly, an adaptive waiting time using equation 4.1 is incorporated to determine the rate of registration requests which in turn dictates the waiting time required using equation 4.2. The corresponding results when the adaptive waiting time is incorporated are shown in Figures 4.12(a) and (b).

In Figure 4.11(a), it can be seen that with increasing rates the number of packets aggregated is high. At lower rates the bus needs to wait longer to maximize the number of aggregated packets at the cost of increased end-to-end latency. It is evident in Figure 4.11(a) that at higher rates the end-to-end latency of packets is reduced due to reduction in packet processing at each router. A fixed waiting time does not adapt well since different routers would experience different rates. For example, ARs close to a busy web server are more likely to experience high rates therefore a higher waiting time is required. In Figures 4.12(a) and (b), the results shows the performance of ARs with an adaptive timer. As can be seen from Figure 4.12(a) the lowest latency is when $\alpha = 0.1$. The end-to-end latency remains fairly constant except when $\alpha = 1.0$. This means that the bus stops at ARs that have a high pickup probability. A comparison of TCP's round-trip time (RTT) with and without the BusStation option is also investigated. The average RTT values with and without BusStation option are 2.33 and 2.50 seconds respectively. Hence other traffic such as TCP benefit when the BusStation option is enabled.

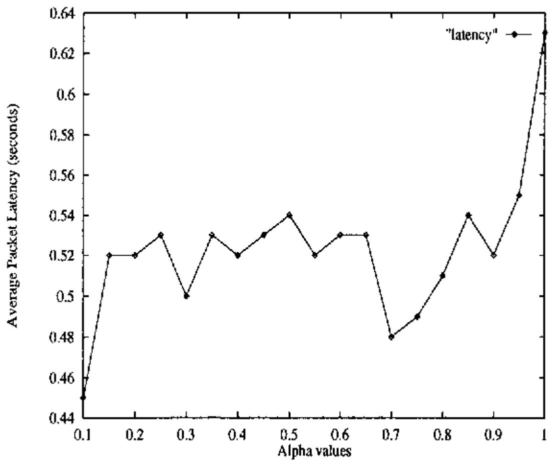


(a)

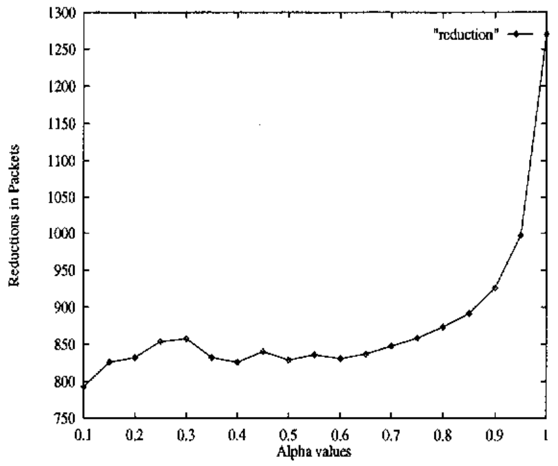


(b)

Figure 4.11: (a) Average reduction in registration request latency. (b) Number of packets reduced with ABus.



(a)



(b)

Figure 4.12: (a) Average latency with varying α values. (b) Average packet reduction with varying α values.

4.4 Discussion

4.4.1 Redirection of Packets

The redirection of packets in ADS is similar in nature to those in Cellular IP [37] and Sony's VIP scheme [16]. The Cellular IP proposal [37] is targeted at local mobility and uses Mobile IP for wide-area mobility. Although there are major differences (for example ADS is targeted at wide-area mobility) between ADS and Cellular IP, here only similarities will be considered. The main similarity is that the nodes in Cellular

IP maintain mappings of MHs. These mappings are used to route packets hop-by-hop to the MH's current location [37]. The mappings in Cellular IP are refreshed by route-update packets whereas in ADS we rely on registration requests. Moreover, Cellular IP uses a time-out mechanism to remove mappings information from nodes leading to the previous FA. ADS also uses time-out mechanism, but the timer can be explicitly removed by binding update sent by the MH to the previous FA. In Cellular IP [37] no update message is sent to the previous FA which may result in loss of packets.

Routing of in-transit packets is also performed by the Sony VIP scheme [16]. Although the main concept of Sony VIP redirection of packets is the similar to that of ADS, there are two main differences. Firstly, the state within each ADS is not explicitly updated by the HA. Secondly the ADS program for a given MH is only active for ARs between the HA and MH's current location. This is a better approach because maintaining mappings in ARs leading up to each CH results in difficulty in maintaining mappings if the MH is communicating with multiple CHs and the problem becomes aggravated if the CHs are mobile. Furthermore, the MH might be connected to a given CH for a short period of time thus maintaining a mapping only wastes resources.

A similar approach using chain of pointers (Anchor-Chain) was proposed in [31]. The differences between ADS and Anchor-Chain are, (1) optimal path is guaranteed in ADS (2) no external mechanisms are required to maintain states at routers since in ADS the refreshed mechanism is based on Mobile IP's default registration request (3) the operations of ADS are transparent to end-host and (4) each router maintains the MH's current location whereas in Anchor-Chain the next downstream pointer is maintained. As a result, Anchor-Chain is susceptible to node failure.

4.4.2 Regional Registration

The local registration performed by ADS programs is similar to the proposed regional registration request/reply [34]. Perkins's [34] proposal is based on a hierarchy of FAs. During handoff, the MH only has to register with a FA that is the parent of the current and previous cell areas. This hierarchy of FAs reduces the number of registration requests/replies to the HA and provides for a fast handoff.

The main objectives of the hierarchical FAs [34] (HFA), Caceres et al.'s [183] hierarchical handoff scheme and the hierarchical handoff in ADS are to reduce handoff latency and fault tolerance (MH is able to make a registration request during network partition). The following paragraph outline the differences between ADS's hierarchical handoff scheme and HFA[34].

The main differences between ADS and HFA [34] are as follows:

- The only performance enhancement provided by HFAs over the Mobile IP scheme [10] is faster handoff. As shown in the results the main concern is the redirection of packets and the binding update time of the CH, hence no significant performance is observed by application(s) at the MH (or CH).
- ADS programs do no need to cooperate to form a hierarchical topology. The HFA requires the maintenance of FAs which use control messages to determine the hierarchy a FA belongs to. ADS is generic in that it does not assume any network topology. If the network topology is tree based then ADs along the HA and MH will naturally form a hierarchy.
- The MH in ADS does not assume the existence of ADS programs. The MH function as specified in the Mobile IP specification by sending a registration request back to the HA. In the HFAs specification the MH is required to keep track of the hierarchy of FAs serving its new point of attachment [34].

The differences between ADS and Caceres et al.'s scheme [183] are as follows:

- ADS does not need any modifications to the Mobile IP specifications, that is it only enhances Mobile IP. Caceres et. al. [183]'s scheme needed modifications to the Mobile IP specifications to support mobility by providing a different mobility infrastructure which involved a hierarchy of FAs.
- In [183], Caceres et al. exploit locality of MH movements, Mobile IP is only used if the MH migrates globally. This suffers from the limitations addressed in

Section 1. ARs alleviate these problems and provide an efficient handoff scheme. The ARs along the route to the MH's current location are programmed with the ADS program. In effect, this further reduces the RTT from the MH's to an AR and also increases the probability of packet redirection.

4.4.3 Security

The importance of security in any mobility related protocols is of importance. Currently security issues in ANs are still being investigated and remain an open issue. Readers who are interested in how active programs are loaded and distributed securely are referred to [165, 188]. In this chapter, detailed discussions on security have been excluded intentionally because the incorporation of security into ADS requires further investigation and justification which are beyond the scope this thesis. Also this is also partly dependent on the maturity of security in ANs. Some of the issues that are being investigated are, identifying sender of a program (both end-host and network elements) and allocating and controlling the use of resources.

There are many possibilities of incorporating AN securities to Mobile IP. Let us assume that secure active network environment (SANE)[165] is being used. Current security services offered by SANE include cryptographic primitives, packet authentication, packet confidentiality and key establishment protocol. Given these services, the HA and MH may request the network to provide a secure channel without worrying how the security measures are taken. Alternatively, HA and MH can use own mechanisms. Here the assumption is that the MH and HA share a security association or allocated by the network (e.g, using session-key establishment [165]). Besides the above, the key establishment protocol can be used between each AR and CHs where secret keys and certificates are established. Since each AR serving a given MH maintains security association, when the CH migrates and re-registers with any of the ARs serving the MH, an authentication can be performed safely. Therefore, the intercepting AR can verify and identify the CH. Apart from that, location update messages can be signed and checked at each AR to ensure the integrity of the message before being passed to end-host. Here, since ARs have already established security associations with MHs, CH can safely determine the source of update messages. As for code authentication during distribution of the ADS program, the naming services provided by SANE can be used. The naming services allow identification of programs for different names, semantics and trust dependencies to be encoded [165]. Hence different active programs deployed from different HAs can be loaded without colliding with each other.

4.4.4 SmartBuffer

In the *snoop* protocol, the idea is to shield the sender from any loss at the wireless link and use multicasting during handoff [189]. On the other hand, the idea of SmartBuffer is to shield the sender in the event of high handoff latency and delay in binding update delivery. Furthermore, with SmartBuffer there is no need to multicast packets to surrounding cell areas or track the MH's movement. This is the main novelty of SmartBuffer because *snoop* does not address high handoff latency, especially during vertical handoffs. When handoff latency is high multicasting aggravates the problem of retransmission timeout at the sender. Furthermore, it may cause high number of packets to be buffered in the surrounding BSs. Therefore, it is beneficial if SmartBuffer is incorporated into the *snoop* protocol to overcome errors at the wireless link as well as retransmission timeout due to high handoff latency.

The work presented by Wang et al. [96] is similar to SmartBuffer. The following highlight the main differences:

- SmartBuffer does not require the splitting of TCP connections. This results in simpler solution because states (e.g., states allocated due to socket calls) pertaining to the connections do need to be transfered to the new BS during handoff.
- In SmartBuffer, TCP is not used to transfer states from the previous BS. SmartBuffer avoids the connection setup time of TCP, resulting in a faster forwarding time.
- In Wang et al. [96]'s scheme, the sender views the BS as the communication endpoint where the BS returns acknowledgements on the MH's behalf, thus breaking

the semantic of TCP. However, in SmartBuffer, the end-to-end semantic is only violated when the handoff latency exceeds a given threshold.

4.4.5 The BusStation Framework

The bus waiting time is calculated based on a simple exponential averaging. Due to lack of traffic data on registration requests or binding updates from a real mobile network it is difficult to determine the appropriate methods to use and the distribution update messages exhibit. In this chapter it was shown that given varying arrival rates, at each node the waiting time computed adapts to the varying rate of registration request arrivals at each node and is sufficient in reducing the number of packets headed towards a given host.

A concern with deploying the BusStation scheme is that the amount of benefits achieved is dependent on the correlation between geographic location and position of multiple MHs with the same HA or CH. The number of signaling messages headed towards a given HA maybe small, thus does not warrant the use of BusStation. On the other hand, it is quite likely that in the near future as the number of mobile users increase, popular servers will likely be swamped with location update messages (given that optimized mobile IP is used). Note that the BusStation is not positioned strategically (possibly hot spots in the network) within the network. The benefit here is that aggregation is only performed on demand.

BusStation provides a framework for future aggregation services. A direct application of BusStation framework is in overcoming ack implosion or local error recovery in multicast. The BusStation provides a framework in which any AN-based solutions to the aforementioned problems can deploy the BusStation service to reduce acknowledgements headed to the source of the multicast or transporting recovery packets to group members.

4.5 Conclusion

This chapter has investigated the application of intra-network processing in the context of mobile communications, specifically improving the operation of mobile IP. As shown, ANs play an important role in realising these solutions where the network elements are programmed to work in concert with end-hosts. The implications and potential performance improvements of ARs programmed with ADS have been analysed. Main limitations of the Mobile IP models have been identified and this chapter has shown that ADS can overcome these limitations. ADS was shown to be most beneficial in fast rerouting and notifications of CHs especially in a scenario where both communicating parties (MH and CH) are mobile. Apart from that, the benefits of three novel schemes: smooth handoff through the use of SmartBuffer, efficient delivery of binding updates and aggregation of location update messages have been shown to be viable approaches to increasing the performance of the mobile IP models. Utilization of these four schemes improve the operation of the Mobile IP models by reducing the bandwidth and processing cost of location update messages. Furthermore, Mobile IP is resilient to high handoff latency due to the use of SmartBuffer. Another important aspect to point out with the application of ANs is that, hierarchical FAs or special servers to reduce handle handoff within a domain are not required if ANs are used. As can be seen in the hierarchical handoff described in this chapter, the local router handles all requests from the MH. The main implications of this are that AN-based solutions are generic and do need to make assumptions about the underlying network topology.

In this chapter, the viability of ANs in enhancing the routing operation in packet switching networks has been shown. The benefits of ANs addressed above also applies to other solutions presented in this thesis. To show these benefits further, the next chapter employs the programmability of ANs to rerouting unicast connections in ATM networks, i.e., to show the advantages of ANs in connection-oriented service model.

Chapter 5

ACR: Active Connection Rerouting

In the previous chapter the benefits of ANs in enhancing mobile IP were presented. It was shown that maintaining the location of the MH and the CHs communicating with the MH enhances the performance of TCP over IETF mobile IP. In this chapter the problem of rerouting unicast connection in ATM networks will be addressed. The problem presented in this chapter is quite different to those presented in the previous chapter. In a connection-oriented environment, the problem is concerned with rerouting of connections. With connection rerouting, the problem of minimising cell loss and preserving cell ordering is critical. This chapter investigates how programmability can be used to provide a better solution to the connection rerouting problem. The resulting AN-based solution called ACR is efficient in terms of signaling overheads, low latency handoff, minimal cell loss and is transparent to end-hosts.

The ACR scheme employs a two part optimization scheme. The first part involves extending the path to the MH's current location during handoff. The main advantages of using path extension are low latency handoff, preservation of cell ordering and little or no cell loss. Another reason for using path extension is based on the following observation. After the MH migrates, cells continue to be forwarded to the previous BS and these cells are discarded if no buffering is performed at the previous BS. To salvage these cells some form of forwarding scheme needs to be in place. In [49], after path splicing, an external mechanism is used to forward cells over. Hence some form of forwarding is required. Therefore, in ACR, the forwarding of cells is made an integral part of the handoff process. Path extension, may lead to an increase in end-to-end delay and loops. To overcome these two problems, a partial optimization scheme is used to remove loops automatically after handoff. The invocation of partial optimization is transparent to end-hosts. From the end-hosts' point of view, the network is able to resolve any loops. The second part involves an incremental optimization scheme that optimises the resulting extended path. Although, the idea of a two part connection rerouting process is not new [55], ACR has the following functionalities which further improve current methods:

- Loop elimination at the switches. Current methods detect loops when the MH revisits a given BS. A MH keeps track of the BSs it visits and detects a loop when a BS is revisited. However, this method only detects loops at BSs but there maybe other multiple loops formed within the network thereby consuming unnecessary resources. With ACR, the use of active switches prevents loop formation.

- Incremental optimization. In order to optimize the resulting extended path current methods employ CX discovery algorithms or re-establish a new connection to the CH. In CX discovery, a tradeoff must be made between path reuse and optimization. That is to say the algorithm has to determine whether it needs to reuse as much of the original path as possible which in effect may mean that the resulting path is sub-optimal. Re-establishing a new connection to the FH ensures optimal path but not path reuse. This chapter demonstrates how ACR optimizes the connection incrementally while a significant part of the original path is reused. Thus, no CX discovery algorithms need to be invoked from end-hosts and no tradeoff policy needs to be in place. As a result, ACR reduces the signaling messages and buffering required.

This chapter is organised as follows. The active switch architecture used in this thesis will be presented in the next section. Then the ACR scheme is presented in Section 5.2. This is then followed by a description of the analytical model used to evaluate ACR. The analytical models derived are also used to compare ACR with other rerouting algorithms reviewed in Section 2.1.2. A simulation model was also used to determine the viability of ACR. The simulation methodology and the results obtained are presented in Section 5.4. Then a brief discussion is presented in Section 5.5 followed by conclusions in Section 5.6. For clarity, a table of signaling messages used in ACR is presented in Appendix B.1.

5.1 Architecture

In ATM, active switches perform per virtual connection (VC) or virtual path (VP) customisation. In our approach, ASs detect and reroute VC(s) or VP(s) pertaining to CHs and MHs. The roles played by these ASs are connection rerouting and location management. ASs keep track of MH location enabling ASs to inform FH(s) of a MH's current location during connection setup. The main advantages of ACR over other schemes are loop elimination at switches and incremental optimization that allows the reuse of connection before handoff. A general overview of AS used in this thesis is shown

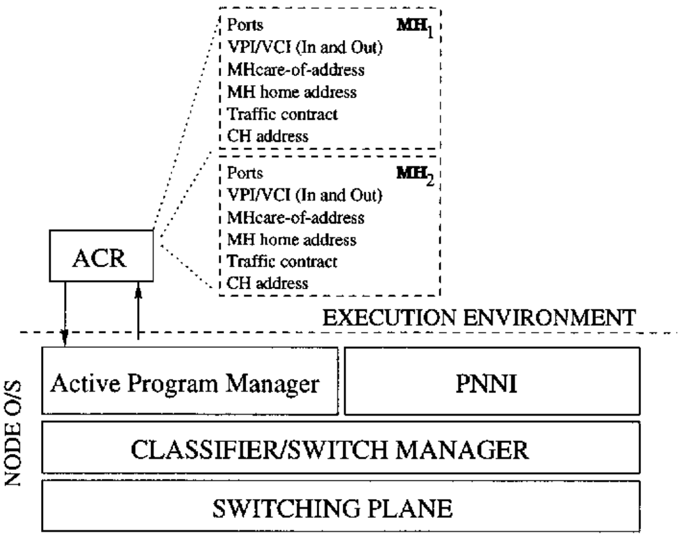


Figure 5.1: Active switch's architecture.

in Figure 5.1. The classifier or switch manager layer basically provides an abstraction to the underlying switch hardware. It exports a set of interfaces which allow the control of the underlying hardware. For example the creation/deletion of VPI/VCI, query on the number of ports and so forth. The role of the classifier is to parse any cells that contain an active tag. Due to the size of each cell, the capsule approach was not taken.

Note that the dynamic loading of program from an external source is possible where the program is fragmented into multiple cells. Once a cell with an active tag is parsed, the cell is passed to the active program manager. The program manager then loads the corresponding program. The ACR program then takes control of the connections pertaining to MH it is managing. The corresponding data allocated for each MH is shown in Figure 5.1. The program manager provides an interface to active programs where it exports a set of interfaces for managing connections, forwarding of cells and route queries. Furthermore, the program manager also informs the switch manager that any cells that arrive on a given VPI/VCI are to be passed for active processing. Otherwise these cells are passed to the PNNI layer.

The ASs only need to be programmed once. If there are multiple connections, the ASs only need to record the connection information. This improves scalability since only one program is loaded for all connections going to the MH. Furthermore, the data maintained at each AS requires minimal storage. Table 5.1 shows the data maintained by the AN-handoff program at each AS.

Data	Details
Pointer to the Translation Table	This is needed for path rerouting during optimization.
VPI/VCI	The connection identifiers
Ports	The outgoing and incoming ports
MH's care-of-address	The MH's current location.
MH's HA address	The MH's home address.
CH address	The corresponding host (CH) (i.e., calling party).
Traffic contract	Traffic contract specified (e.g., QoS).
Identification	This is used to create a security association between the FH and AS.
Optimization Status	Optimization in progress flag.
Original or Extended Flag	This flag marks whether the AS is on the original or extended path.

Table 5.1: Data structure maintained by AS

5.2 The ACR Model

5.2.1 Overview

ACR reroutes connection using a simple path extension scheme after handoff followed by a lazy optimization scheme. Although the idea of performing a two phase rerouting is not new, ANs enable this process to be improved significantly. To give an overview

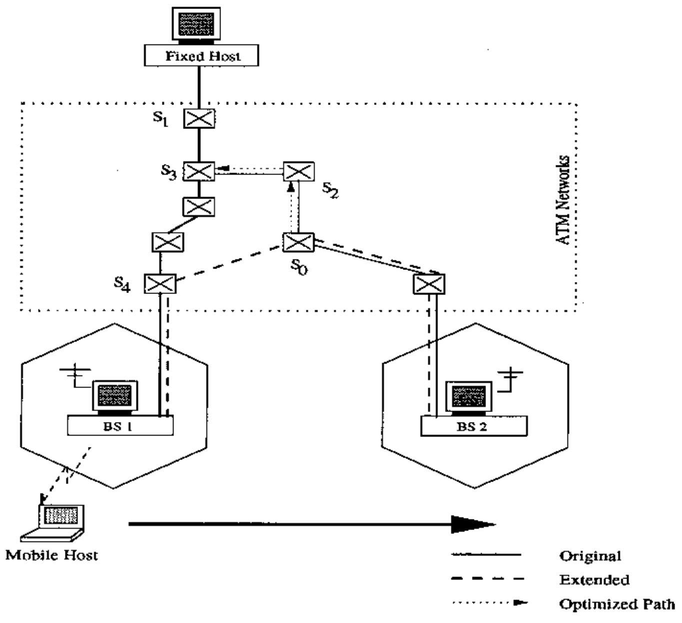


Figure 5.2: Overview of the ACR model.

of the ACR process consider the network shown in Figure 5.2. When the MH migrates to BS_2 the path is simply extended from BS_1 to BS_2. What follows is the route optimization phase. This phase is separated into two parts. During the path extension phase from BS_2 to BS_1 an active setup message is used which determines that the extended path has intersected at S_4. After the path extension process, S_4 short-circuits the extended path (referred to as partial path optimization). Once the ACR program within S_4 has updated the translation table at S_4, the path extends from S_4 to BS_2 instead of going through switches between S_4 and BS_1. Notice that this is a form of loop removal. In traditional path extension method the only way a loop can be detected is when the MH realizes that it has revisited the same BS. In such networks, a loop removal algorithm can be invoked, but the delay of loop detection may cause multiple loops to exist within the network itself. The next stage of the optimization process is called full path optimization. The time to invoke a full optimization is based on a given policy determined by the MH. For example, when the MH has migrated from the current domain or the QoS on the extended path has to be renegotiated. Full optimization is initiated by sending an active setup message back to the FH. As the setup message traverses the network, each switch checks whether the setup message is currently being forwarded on the old path or diverted to a new path. If the setup message is forwarded to a new path (probably a shorter path leading to the FH) then it notes the AS at which the path diverges. In Figure 5.2 the path diverges at S_0. No operation is performed at this stage until the setup message encounters another AS that is on the original path (S_3). This AS maybe the FH's local switch. Then an optimization process is performed in which a new path is created from S_3 to S_0. The implication of this is that path can be reused from the MH up to S_3 and FH to S_2. In conventional methods, a lazy optimization using a full connection establishment method would not be able to reuse the original path. Other methods which involve CX discovery would require a probing of the network and finding the switches S_2 and S_3 which involves additional overheads.

5.2.2 Programming ASs

In ACR, the integrated approach for location management proposed by Acharya et al. [49] was adopted. The scheme proposed by Acharya et al. [49]'s is similar to location management in Mobile IP [10]. The FH issues a SETUP[1] message to a host without knowing whether the end-host is a MH. When the home agent (HA) of the MH intercepts this message, either a CONNECT or RELEASE is returned. A CONNECT response is generated when the host is static or when the MH is in its home network. A RELEASE is generated if the MH is in a foreign network. The RELEASE contains the MH's current location allowing the FH to setup an optimal path to the MH. The CONNECT and RELEASE messages contain an element which identifies the connection to be a mobile connection. This allows the FH to determine if it is communicating with a MH in order to determine whether switches need to be programmed. Having the HA

[1]Additional Information Elements are added in signalling message. This include SETUP, CONNECT and RELEASE

deploy the ACR program allows for a single point of distribution. This means that all MHs originating from the HA's domain will be using the same ACR algorithm.

ACR returns an active CONNECT or RELEASE message to the HA instead of the ordinary CONNECT and RELEASE messages. Note that if a switch containing the ACR program is encountered, an active CONNECT or RELEASE message is returned by the switch instead of the HA. Thus, connection setup time is reduced. These *active* messages contain the required security association and an embedded program to be used for loading the ACR scheme into ASs. The advantages of having the HA deploy handoff programs are that only the predefined copy of the active program is loaded and the security association is distributed from the HA only. Upon receipt of a CONNECTION message, a FH proceeds to establish a connection to the MH's current location. This sequence of events are described in Figure 5.3. Referring to Figure 5.3 an active SETUP message is initiated by the FH and sent to the MH via the MH's home network. The HA intercepts this message and replies with a RELEASE message, assuming that the MH has migrated to a foreign network. Embedded within this RELEASE message are two elements, the first is used to load the ACR scheme at ASs and the second is used to indicate to the FH that the corresponding host is a MH. Using the MH's current address embedded within the RELEASE message, the FH then makes a direct connection to the MH. ASs along the path that intercept the setup signal sent by the FH establish the required security association and load the ACR scheme. Once the program is loaded it has access to information pertaining to the call. Other accesses such as translation table or new call creation are through predefined interfaces.

5.2.3 Handoff

In the ACR scheme handoff is achieved firstly by extending the current path to the new BS. After that the path is lazily optimized using the proposed incremental optimization scheme. The follow paragraphs present the handoff algorithm and path optimization in detail.

In Figure 5.4 when the MH detects that it is in the vicinity of Cell 2, it sends a

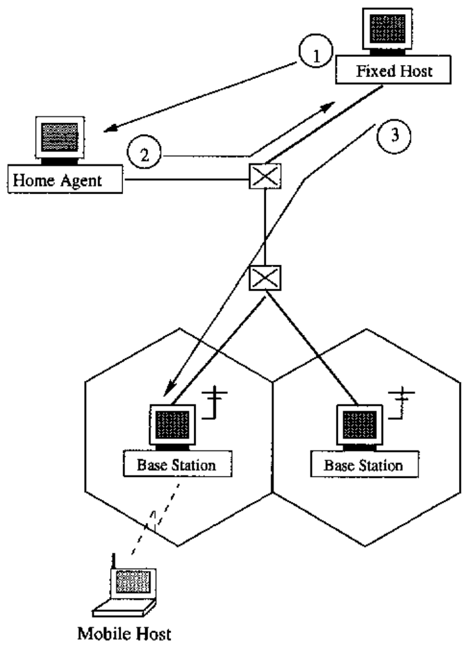


Figure 5.3: Location management in mobile ATM

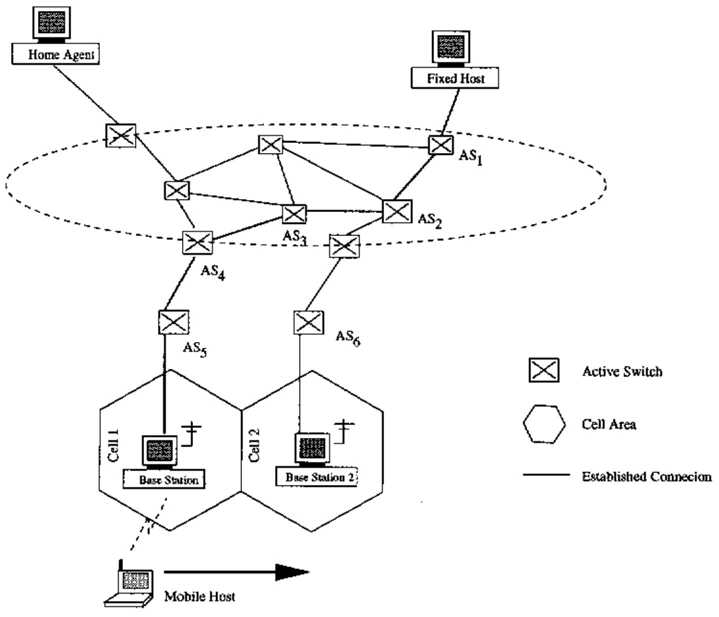


Figure 5.4: Handoff in wireless ATM

HO_REQUEST to BS_1 and this message is forwarded to BS_2. If BS_2 is able to support the required QoS, BS_2 then sends an active SETUP message to BS_1. In addition to a new path being formed, all ASs between BS_1 and BS_2 are programmed. This completes the path extension process [2]. Once the MH has migrated, BS_2 sends a HO_DATA_FORWARD message to BS_1. BS_1 then updates its translation table to merge the two paths. Once this is done, cell will be forwarded onto the new path. Note that using preestablished connection between BSs will further improve handoff latency. Finally a HO_REPLY is sent from the BS_2 to the MH through BS_1. If the path extension fails a HO_REJECT is returned.

In the above, the assumption is that the MH is able to detect signal strength from neighboring *cell* areas and determine which BS it is approaching next. This enables the MH to preallocate the extended connection before handoff occurs. In the case of *hard handoff* (no prediction of handoff), after the MH establishes a radio link with the new BS (BS_2), the active setup message is sent from BS_2 to BS_1. BS_1 automatically updates its translation table without waiting for the HO_DATA_FORWARD message. Once the translation table is updated at BS_1 the handoff process is completed. Note that all uplink connections from the MH are routed through BS_1 after handoff. These links are then optimized in a similar manner to the downlink connections.

The ACR scheme maintains cell ordering and prevents cell loss, thus no disruptions to traffic flow are encountered and results in low handoff latency. Note that handoff latency is a factor of the signaling latency of the path extension and it is unaffected by the path optimization performed by the AS.

5.2.4 Partial Route Optimization

Once a path has been extended and all the ASs along the path have been programmed, ASs determine whether optimizations are needed. The AS in which the extended and original paths intersect is designated as the CX. In other words, the AS that has reachability to both BS_1 and BS_2 is designated as the CX. The CX then updates its translation table to splice the extended path onto the original path. This process is explained in Section 5.2.6.

[2]We assumed that BS_2 has sufficient bandwidth for maintaining the MH's QoS. Work on satisfying QoS is not within the scope of this paper. Interested readers are referred to [190]

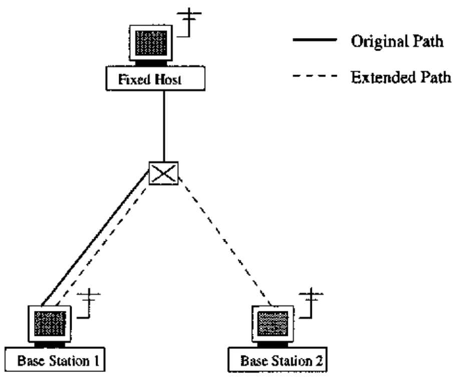


Figure 5.5: Example topology for path extension.

Referring to Figure 5.5, BS_1 and BS_2 share a common AS. After path extension, cells have to traverse through BS_1. This can be avoided by updating the translation table at the AS. Moreover, this scheme does not require any further signaling. During the path extension process when the setup message from BS_2 passes a CX, it is noted by the ACR program within the CX. After the connection has been extended and cells start flowing on the new path, the program at the CX proceeds to reroute the connection by updating its translation table. Hence the connection is partially updated. A question that arises is whether this step can be performed right after handoff. Partial optimization right after handoff may result in the loss of cells from the CX to the previous BS unless a special buffering mechanism between BSs is used to prevent loss.

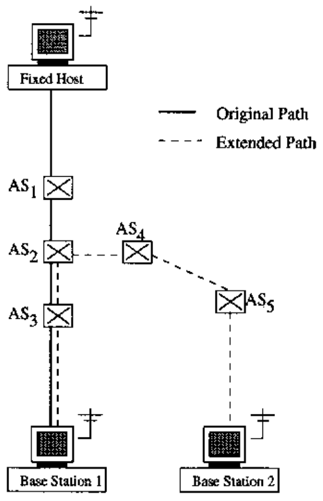


Figure 5.6: Example topology for path extension with multiple switches.

Figure 5.6 shows another example but with multiple ASs involved. This figure shows why a connection setup needs to come from BS_2. In Figure 5.6, AS_1 to AS_3 are ASs on the original path. When a connection is made from BS_2, the extended path intersects the old path at AS_2. In Figure 5.6 a loop has formed between AS_2 and BS_1. To resolve this loop AS_2 is designated as the CX. Once AS_2 updates its translation table and cells forwarded onto BS_1 are drained, the loop AS_2-AS_3-BS_1-AS_3-AS_2 is torn down. If BS_1 was to extend the path, then AS_3 will be chosen as the CX resulting in a loop between AS_2 and AS_3.

During cases where the MH moves backward and forward (or zig-zag) between BS_1 and BS_2 the partial optimization process will result in BS_1 being chosen as the CX. Hence the extended path from BS_1 to BS_2 is torn down. Due to the inherent partial optimization in ACR, loops are automatically removed and thus there is no need for expensive sophisticated loop removal algorithms.

5.2.5 Full Route Optimization

This step is performed after partial optimization. It serves to check existence of alternative paths to the FH. If an alternative path is found, optimization is performed. To optimize the extended path, BS_2 makes an *active* connection setup to the FH. The setup message contains a short program which checks whether a state exists at each AS as it traverses to the FH. As the setup messages traverse through each switch, the program within the SETUP message determines whether the AS holds state information for the connection between the FH and MH. If a state exists, the program records the AS's address because the SETUP message was forwarded on the original path. If the SETUP message has diverted onto a new path it will encounter an AS that doesn't have the required state. At this point, the OPTIMIZE state within the program is set to true. When the SETUP message encounters the next AS which holds the state between FH and MH, the optimization process is invoked.

As shown in Figure 5.7, there is an alternative path leading to the FH. As the setup message traverses through AS_5 and AS_4 it detects whether these are programmed during path extension. Upon arrival at AS_6 the program detects that AS_6 has not

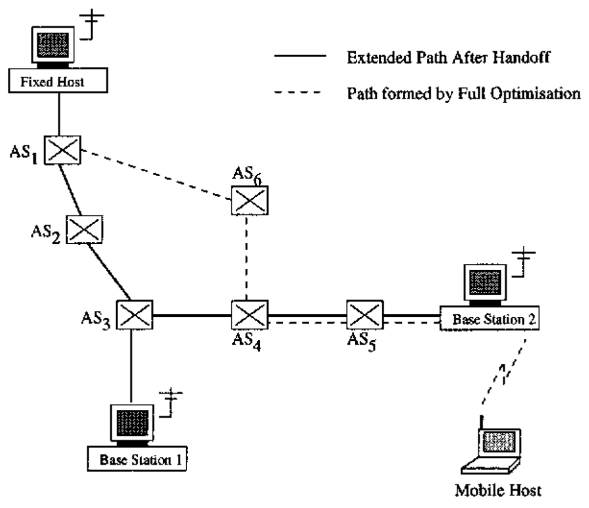


Figure 5.7: Example of full optimization.

been programmed and hence the setup message has been routed through an alternative path. Therefore, full path optimization is needed. When the setup program arrives at AS_1 the program realizes that AS_1 is on the original path. At this point the program stops traversing and updates the translation table to splice the new path onto the old path. The setup message stops at AS_1. If AS_6 is on the original path and if no alternative route has been taken then the program is terminated and any allocated resources are freed by returning an active RELEASE message. If the CX chosen does not have the appropriate resources to maintain the required QoS, the setup message is forwarded along the route towards the source. The message continues to be forwarded until a suitable CX is found.

The above two stage approach takes advantage of the locality of MH's movements and fast handoff provided by path extension. More important is the fact that an incremental optimization process is used where the original path is reused as much as possible and only optimize portions of the original path. Although the example shown in Figure 5.7 only has one optimization process, there can be multiple optimization processes which work on different portions of the original path. Besides that, having an incremental optimization reduces buffering required during connection rerouting. In conventional schemes a new connection is setup to the FH, all cells on the old path have to be drained before cells from the new connection can be accepted. Comparing the buffering requirements to ACR it can be seen that in ACR, cells are drained from a portion of the path whereas in conventional method old connection from the FH to MH needs to be cleared. Also, incremental optimization conserves resources. As can be seen from the optimization process only new states are created when the path needs to be optimized, therefore any states pertaining to the original path are reused. Furthermore, ACR provides a generic solution which provides an optimal path regardless of the underlying topology. This will be shown in Section 5.4.2.

5.2.5.1 Mobility at the Source

Up until now the source is considered to be stationary. It is likely that the FH maybe mobile. In the rest of this section, a FH that is mobile will be referred to as CH. Therefore, if the CH moves, then during the full optimization phase, the destination address used might be the CH's previous care-of-address. As a result, the resulting route is sub-optimal and may fail since the CH is no longer at the recorded care-of-address. One solution to this is to query the HA for the MH's new care-of-address resulting in an additional overhead. Another problem is when the CH and MH migrate simultaneously. As a result, two optimization processes are invoked from both ends of the connection leading to ambiguity in the optimization process.

To overcome the above problems, after BS_2 has sent the HO_DATA_FORWARD message, the MH sends a HO_UPDATE message containing its current care-of-address along the connection to the CH or vice-versa. As the HO_UPDATE message traverses the network, each AS on the path updates the MH's care-of-address. If the HO_UPDATE is sent by the CH then the *CH* field in Table 5.1 is updated instead.

The full path optimization process uses the *CH* or MH care-of-address during the optimization process. The CH or MH address in the setup message maybe invalid due to recent migration. Therefore, when an updated care-of-address is encountered, the current optimization process is canceled and a release message containing the updated address is returned. The release message also contains an error code specifying why the optimization process was aborted. When a local AS (to the MH) receives the release message it restarts the optimization process with the MH's new care-of-address.

To avoid any ambiguities in the optimization process due to MH and CH optimizing the connection simultaneously, the optimization process from the MH (or the callee of the connection) is given precedence. During the optimization process a check (*Optimization Status* in Table 5.1) is performed at each AS to determine whether an optimization is in progress. If the MH is currently performing an optimization then the optimization process from the CH is canceled, otherwise, the optimization from the CH continues. In the case where the optimization process from the MH encounters an optimization process from the CH no operation is performed.

5.2.6 Connection Rerouting

This process is needed to splice the new path onto the old path. This involves updating the translation table at the CX and draining cells from the old path. When the CX detects a setup request, it forwards the request to BS_1. After the connection has been setup cells that are in the original path need to be drained. Firstly, the CX sets up a connection to BS_2 given the traffic contract of the original path. When a path has been successfully setup the translation table is updated. A capsule is sent along the new path to inform BS_2 to buffer all incoming cells from the new path until a marker cell from the old path has been received. The CX then sends a marker cell along the old path to mark the last of the cells forwarded. Upon receipt of a marker cell, BS_2 forwards buffered cells. The paths from BS_2 to BS_1 and from the CX to BS_1 are torn down.

The following points outline the sequence of events during connection rerouting at the CX:

1. Inform BS_2 to buffer incoming cells from the new path.
2. Update translation table. New cells from the FH will be forwarded onto the new path here after.
3. Inject a marker cell onto the old path. When this marker cell appears at BS_2 the buffered cells can be forwarded to the MH.
4. BS_2 then informs the CX to tear down the original path.

5. Finally BS_2 tears down the extended path to BS_1.

The above sequence of events are similar to rerouting in the full path optimization stage. The only difference is that instead of BSs, the rerouting involves switches. For example in Figure 5.7, the path to drain is the old path between AS_1 and AS_4.

5.3 Analytical Model for ACR

This section provides an analytical study of the ACR algorithm and compares its performance to other connection rerouting schemes in the literature [42, 43, 48, 191, 50]. The analytical derivations outlined are based on the work by [59]. A brief summary of Bui et al.'s work will be presented in the following section [59].

5.3.1 Basis of study

Bui et al.'s work compares the performance of various rerouting schemes [59]. These include *Connection Re-establishment* [42], *Path Extension*[43], *Anchor Rerouting*[53], *Dynamic Re-routing* [48], *Multicast Based Rerouting* [191] and *Static Virtual Connection Tree based Re-routing* [50]. These rerouting schemes are compared based on their complexity (number of signaling messages required), handoff latency, communication disruption, bandwidth requirements and buffer requirements. Algebraic equations are derived for each of the above performance parameters. These equations are derived based on the processing and traversing times of signaling messages generated during handoff. Tables in Appendix A list the input parameters which are used to generate expressions to analyse a given scheme. The various parameters listed in the table shown apply only to the ACR scheme. Other parameters can be found in [59]. With regard to the parameters the following assumptions were made. Firstly, the propagation delay of the wireless link is $3\mu s$ which corresponds to 1 kilometer cell diameter. Signalling message processing time in the Mobile Switching Centres (MSCs) have been derived based on the Forerunner ASX1000 switch [59]. Due to Base Station Controllers (BSCs) having low processing power, their processing time is twice that of a MSC whereas the processing time in base transceiver stations (BTSs) is four times that in a MSC [59].

5.3.2 Handoff Delay

In the ACR scheme, handoff latency is dependent on the time it takes to establish a radio connection with BS_2 and the time to setup (including programming ASs) path(s) to BS_1. In schemes such as dynamic rerouting, handoff latency depends on fast CX discovery. Handoff delay needs to be at a minimum to ensure seamless operation of real-time applications during handoff. In this section, a comparison of the handoff delay of ACR to other schemes [59] is presented. Note that the notations used in the derivation of the expressions to follow are defined in Appendix A.

The expression for handoff delay for the ACR scheme can be seen in Table 5.2. Also shown is the expression for handoff latency if BSs have direct connection to each other and soft handoff scenario. The procedures for deriving these expressions can be found in Appendix A.2. The expressions for other rerouting schemes mentioned above can be found in [59]. In ACR, the main delays are due to programming/setup of ASs between BS_2 and BS_1 and transmission of messages over the radio link. In connection re-establishment the latency is dependent on the number of switches between the FH and BS_2 (denoted by $N_{old-org}$).

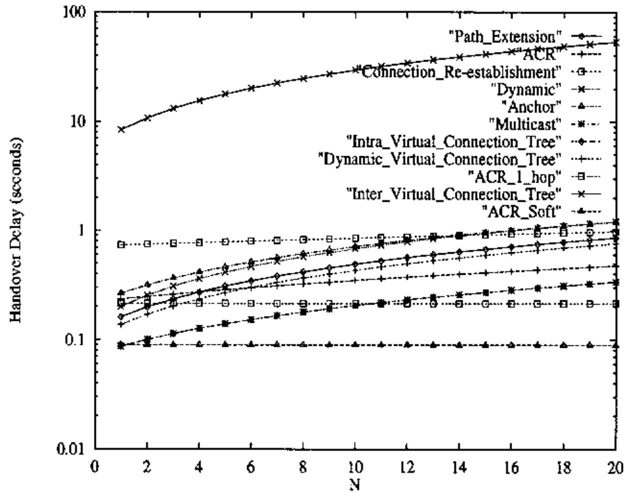


Figure 5.8: Handoff delay.

Figure 5.8 shows a plot of handoff delay against N (the number of switches between

Rerouting Scheme	Handoff Latency ($T_{handover}$)
Connection Re-establishment	$N_{old-org}(2T_{sw} + 2S_{stp}) + N_{new-org}(4T_{sw} + 2S_{stp} + S_{sw(setup)} + S_{sw(other)} - 6T_{sw} + 6T_{bsc-sw} + 4T_{bsc-bts} + 2T_w - 4S_{stp} + S_{sw(setup)} + S_{bsc(setup)} + 4S_{bsc(other)} S_{bts(setup)} + 2S_{bts(other)} + S_m$
Path Extension	$N_{old-new}(4T_{sw} + 2S_{stp} + S_{sw(setup)} + S_{sw(other)} - 4T_{sw} + 8T_{bsc-sw} + 4T_{bsc-bts} + 2T_w + 2S_{bsc(setup)} + 4S_{bsc(other)} + S_{bts(setup)} + 2S_{bts(other)} + S_m$
Dynamic	$N_{old-cos}(2T_{sw} + S_{sw(COS)} + S_{stp}) + N_{new-cos}(4T_{sw} + 2S_{stp} + S_{sw(setup)} + S_{sw(other)} - 6T_{sw} + 6T_{bsc-sw} + 4T_{bsc-bts} + 2T_w - 3S_{stp} + S_{sw(setup)} + S_{bsc(setup)} + 4_{bsc(other)} +$
	$S_{bts(setup)} + 2S_{bts(other)} + S_m$
ACR	$3T_w + 3S_{bts(other)} + 2S_{bts(setup-active)} + S_{bsc(setup-active)} + S_{sw(setup-active)} N_{old-new} + 4T_{bsc(sw)} + S_{sw(other)} N_{old-new} + 2T_{sw} N_{old-new} + S_{bsc(other)} + 2S_{stp} + S_{sw(upd-tbl)}$
ACR-1-hop	$4T_w + 3S_{bts(other)} + 2S_m + 2T_{bsc-bts} + 2S_{bts(setup-active)} + T_{sw} + 4T_{bsc-sw} + 3S_{bsc(other)} + S_{sw(upd-tbl)}$
ACR-Soft	$2(S_m + T_{bsc-bts} + 4T_w + 3S_{bts(other)} + S_{bsc(other)}$

Table 5.2: Expressions for handoff latency.

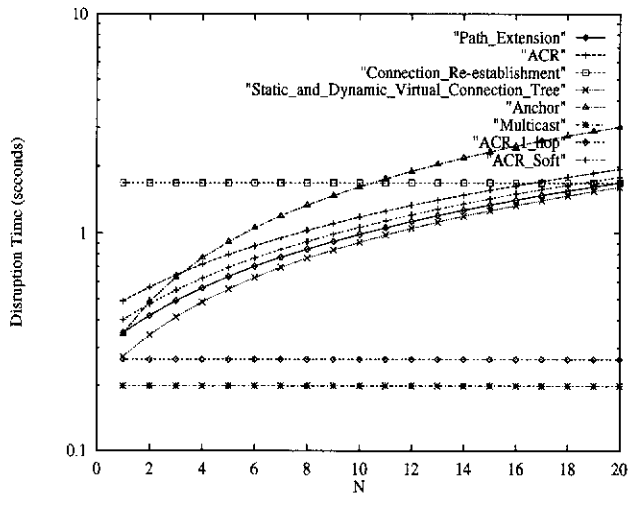
BSs or the number of switches above the BSC where the CX resides). We can see from Figure 5.8, connection re-establishment has a constant handoff latency and has the second highest handoff latency. The handoff latency recorded for the ACR scheme is almost the same as the path extension scheme. Although there is a 20% overhead with the ACR scheme it still outperforms the non-active path extension scheme. This is due to the reduction in signalling messages required compared to the path extension scheme outlined in [59]. Soft handoff has the fastest handoff latency followed by the ACR scheme with 1-hop configuration. Multicast has the lowest handoff latency due to its higher bandwidth cost.

5.3.3 Disruption Time

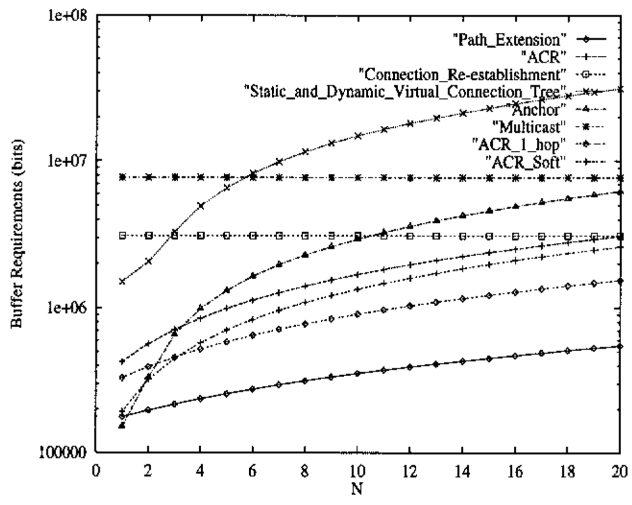
The disruption time determines the interval in which no cells are forwarded to the MH. Figure 5.9(a) shows a comparison of disruption times for various schemes. The behavior of this plot is quite similar to the plot showing handoff latency (Figure 5.8). This is mainly due to the direct relation between handoff latency and disruption time. The ACR has the benefit of minimising disruption time since it is only dependent on the time required to setup the extended path and the time taken for the first cell to be forwarded from BS_1. In the dynamic scheme, delay is incurred mainly in the CX discovery phase.

Rerouting Scheme	Disruption Delay ($T_disrupt$)
Connection re-establishment	$N_{new-org}(T_{sw} + S_{stp} + T_{d(sw)} + S_{sw(sw)}) + T_{bsc-sw} + T_{bsc-bts} + 4T_w + T_{d(bsc-sw)}T_{d(bsc-bts)} + T_{d(w)} + S_{sw(other)} + S_{bsc(other)} + S_{bsc(sw)} + 2S_{bts(other)} + S_{bts(sw)} + 2S_m$
Path Extension	$N_{old-new}(T_{sw} + S_{stp} + T_{d(sw)} + S_{sw(sw)}) - T_{sw} - T_{d(sw)} + 2T_{bsc-sw} + T_{bsc-bts} + 4T_w + 2T_{d(bsc-sw)} + T_{d(bsc-bts)} + T_{d(w)} + 2S_{bsc(other)} + S_{bsc(other)} + 2S_{bts(other)} + S_{bts(sw)} + 2S_m$
Dynamic	$(N_{new-cos} - 1)(T_{sw} + S_{stp} + T_{d(sw)} + S_{sw(sw)}) + T_{bsc-sw} + T_{bsc-bts} + 4T_w + T_{d(bsc-sw)} + T_{d(bsc-bts)} + T_{d(w)} + S_{sw(other)} + S_{bsc(other)} + S_{bsc(sw)} + 2S_{bts(other)} + S_{bts(sw)} + 2S_m$
ACR	$T_{HO-latency} + N_{old-new}T_{d(sw)} + T_{d(w)} + T_{d(bsc-sw)} + T_{d(bsc-bts)} + S_m$
ACR-1-hop	$T_{HO-latency-1-hop} + T_{d(bsc-bts)} + T_{d(w)} + S_m$
ACR-Soft	$3T_w + S_m + 2_{bts(other)} + N_{old-new}(T_{sw} + S_{sw(other)} + T_{d(sw)} + T_{d(w)} + T_{d(bsc-bts)} + T_{bsc-bts} + S_{bsc(other)} + 2(T_{bsc-sw} + T_{d(bsc-sw)}$

Table 5.3: Expressions for disruption delay.



(a)



(b)

Figure 5.9: (a) Disruption time (b) Overall buffer requirements

5.3.4 Overall buffer requirements.

As mentioned, the ACR schemes require buffering at the MH and BS to prevent cell loss and misordering. It can be seen from Table 5.4 that for ACR the buffer requirements at the MH (upstream data) are dependent on disruption time. On the other hand the buffer requirements at BS_2 are dependent on the propagation delay of the marker cell

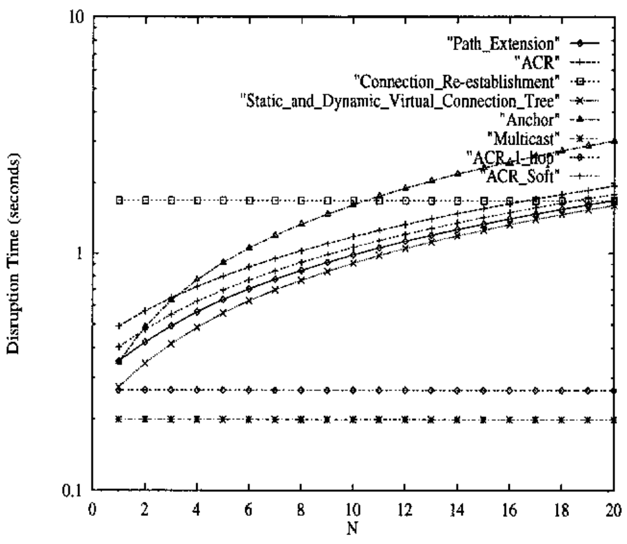


Figure 5.10: Disruption time.

from the CX, that is for the path $CX - BS_1 - BS_2$.

Rerouting Scheme	Buffering Nodes	Buffer Requirements
Connection re-establishment	SW_{org}	$((N_{new-org})(T_{sw} + S_{stp}) + T_{bsc-sw} + 2T_{bsc-bts} + 4T_w + S_{sw(other)} + S_{bsc(other)} + 3S_{bts(other)} + 2S_m)BW_d + \lceil((N_{old-org} - 1)(T_{d(sw)} + S_{sw(sw)} + T_{d(bsc-sw)} + T_{d(bsc-bts)} + T_{d(w)} + S_{bsc(sw)})BW_d/M_d\rceil M_d$
	BSC_{new}	$(2(N_{new-org} - 1)(T_{sw} + S_{stp}) + 2T_{bsc-sw} + T_{bsc-bts} + T_w + S_{sw(other)} + 2S_{bsc(other)} + S_{bts(other)})BW_d$
Path Extension	BSC_{old}	$(N_{old-new}(T_{sw} + S_{stp}) - T_{sw} + 2T_{bsc-sw} + 2T_{bsc-bts} + 4T_w + 2S_{bsc(other)} + 3S_{bts(other)})BW_d$
	BSC_{new}	$(2N_{old-new}(T_{sw} + S_{stp}) - 2T_{sw} + 4T_{bsc-sw} + T_{bsc-bts} + T_w + 3S_{bsc(other)} + S_{bts(other)})BW_d$
Dynamic	SW_{cos}	$((N_{new-cos} - 1)(T_{sw} + S_{stp}) + T_{bsc-sw} + 2T_{bsc-bts} + 4T_w + S_{sw(other)} + S_{bsc(other)} + 3S_{bts(other)} + 2S_m)BW_d + \lceil((N_{old-cos} - 1)(T_{d(sw)} + S_{sw(sw)} + T_{d(bsc-sw)} + T_{d(bsc-bts)} + T_{d(w)} + S_{bsc(sw)})BW_d)/M_d\rceil M_d$
	BSC_{new}	$(2(N_{new-cos} - 1)(T_{sw} + S_{stp}) + 2T_{bsc-sw} + T_{bsc-bts} + T_w + S_{sw(other)} + 2S_{bsc(other)} + S_{bts(other)})BW_d$
ACR	MH	$T_{HO-latency} \times BW_d$
	BSC_{new}	$((N_{cx-old} + N_{old-new})(T_{d(sw)} + S_{sw(sw)}) + T_{d(sw)}) \times BW_d$
ACR-1-hop	MH	$T_{HO-latency-1-hop} \times BW_d$
	BSC_{new}	$((N_{cx-old} + N_{old-new})(T_{d(sw)} + S_{sw(sw)}) + T_{d(sw)}) \times BW_d$
ACR-Soft	MH	$T_{HO-latency-soft} \times BW_d$
	BSC_{new}	$((N_{cx-old} + N_{old-new})(T_{d(sw)} + S_{sw(sw)}) + T_{d(sw)}) \times BW_d$

Table 5.4: Expressions for buffer requirements.

The total buffer requirements for each scheme are shown in Figure 5.9(b). From Fig-

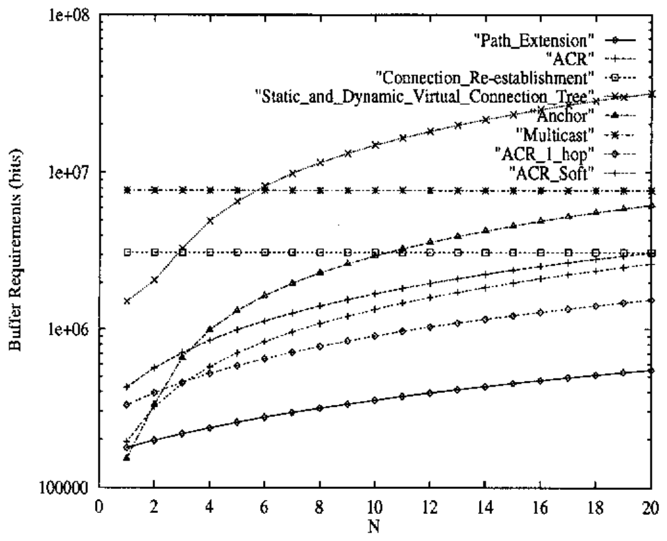


Figure 5.11: Buffer requirements.

ure 5.9(b) the overall buffer requirements for ACR are in the range 12.5 - 800 Kb. Comparing between ACR-soft and ACR scheme, there is only a slight reduction in buffer requirements. If a topology with BSs connected to each other is used, buffer requirements are significantly reduced.

5.4 Simulation Methodology

The simulation studies were done using Opnet [192]. In these simulations the buffer requirements during path optimisation and performance of ACR in different network topologies are studied. Three network topologies were experimented upon; mesh, tree and redundant tree. These topologies can be found in Appendix A.1.

A video conferencing application initiated by the FH was used. The application is set to generate frames at 30 frames/sec with frame size of 65K. The application uses CBR service over links of 155Mb/s. At the start of the simulation each switch has been loaded with the ACR algorithm. During connection setup the program is enabled when a setup signal passes through the given AS. The program is loaded by the AS management module and is only allowed to access call information pertaining to the call. The call information contains a traffic contract and interfaces to the translation table and management module. During connection rerouting, the translation table is updated to reflect the new route changes. In addition, the program has access to the ATM management interface which allows it to setup or tear down connections.

In the simulation a distributed, asynchronous adaptation of the Bellman-Ford shortest path algorithm [193] is used. The cost of each link is a factor of the available bandwidth. In other words, lower bandwidth entails higher cost. If no bandwidth is available the cost is infinite. Each node holds a routing table containing the routes to any given node in the network with the best route at the top of the routing table. During routing, the first route on the table is chosen. If the route satisfies all requirements of the call then bandwidth is allocated for the given route. Otherwise the next best route is chosen.

In the simulator, ten BSs are randomly connected to the ASs. Each BS has an ATM interface which allows it to create connections to another BS or CX. Furthermore, ACR is loaded during connection setup enabling rerouting and path extension during handoff. The BSs broadcast advertisement messages every n seconds. The advertisements are needed to enable MHs to detect that they are in the cell area of the given BS. Three values of n have been tested and their affect on handoff latency analysed. In each simulation, following data are recorded:

1. **Original Path Length** (OrigPL). This is the number of hops (ASs only) from the FH to the MH (when MH is at BS_1).
2. **Extended Path Length** (EPL). The number of hops from BS_1 to BS_2.
3. **Optimised Path Length** (OpPL). The optimised path length after full optimisation.
4. **Partial Optimised Path length** (POPL). The path length from the FH to BS_2 after partial optimisation.
5. **Convergence Count**. The number of hops needed before a CX is found.
6. **Partial Optimisation Flag**. Recorded whether partial path optimisation was performed.

This section outlines results from the simulation studies. Firstly, we highlight the buffer requirements during route optimization given varying delays on the extended path. The significance of these results is the quantification of the number of cells being buffered given varying drainage times. Finally, the performance of ACR is evaluated for the three network topologies discussed in Section 5.1. The simulation studies demonstrate that ACR provides a reasonable tradeoff between optimal path and path reuse for the mesh and tree topologies. Low path reuse or high optimization gain implies higher latency in the extended path, hence higher drainage time. It is crucial to provide a tradeoff between these two variables to minimise buffer requirements.

5.4.1 Buffer Requirements During Route Optimization

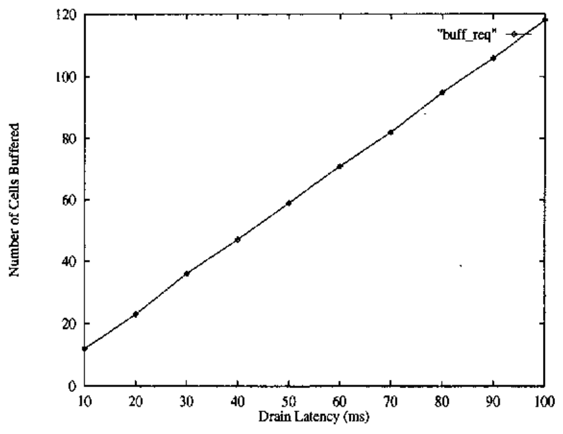


Figure 5.12: Buffer requirements vs. delay in the original path. (CBR Traffic)

Figure 5.12 shows that in the event of large end-to-end delay in the extended path, more cells are buffered. In LANs with delays less than 10ms the buffer requirements are less than 10 cells. The buffer requirements can be reduced further by buffering only throughput dependent traffic such as data transfer which are susceptible to loss. The overall buffer requirements can be further improved if custom architectures such as [51] are used to reduce handoff rate. An observation we made with CBR traffic is that the size of the buffers remains stable after handoff due to similar incoming and outgoing rates at the buffer.

5.4.2 Performance of Varying Network Topologies

The results of the experiments on mesh, tree, and redundant tree topologies are shown in Table 5.5. The results shown in Table 5.5 are average values of performance parameters from ten simulation runs. The results show path reuse and optimization gain from each optimization stage. The objective of these experiments is to investigate the effectiveness of the proposed scheme in achieving optimal path and maximal path reuse. The following items explain each performance parameter in Table 5.5:

- *Path Reuse (Partial Optimization)* is the percentage ratio of the number switches common to original path and partially optimized path, and the number of switches in the original path. If the value is high then full optimization is likely to yield significant optimisation gain.

- *Path Reuse (Full Optimization)* is the percentage ratio of the number of switches common to original path and fully optimized path and the number of switches in the original path. The path includes new ASs (if any) from BS_1 to BS_2. A high value implies a low optimization gain.

- *Optimized Gain 1* is the gain when full optimization is performed on the extended path before partial optimisation is performed. In other words, this gain gives an indication of how much shorter is the optimized path compared to the extended path. This value is calculated as:

$$\frac{(OrigPL + EPL) - OpPL}{(OrigPL + EPL)} \times 100 \tag{5.1}$$

- *Optimized Gain 2* is the gain due to the full optimization stage after partial optimisation has been performed on the extended path. This value is calculated as:

$$\frac{OpPL}{POPL} \times 100 \tag{5.2}$$

- *Optimized Gain 3* is a measure of the gain achieved after partial optimization is performed on the extended path. A high value means partial optimization is sufficient in the given topology. The expression for this value is:

$$\frac{(OrigPL + EPL) - POPL}{(OrigPL + EPL)} \times 100 \tag{5.3}$$

- *Partial Optimization Runs* is the ratio between the number of instances when the partial optimization is executed and the total number of simulations run.

	Mesh	Tree	Redundant Tree
Path Reuse (Partial Optimization)	77.11%	41.4%	58.3%
Path Reuse (Full Optimization)	39.3%	41.4%	40.73%
Optimized Gain 1	42.0%	47.6%	43.0%
Optimized Gain 2	19.6%	0.0%	17.4%
Optimized Gain 3	18.3%	47.6%	26.5%
Partial Optimization Runs	50%	100%	20%

Table 5.5: Performance parameters for network topologies.

	Mesh	Tree	Redundant Tree
Maximum number of hops	7	4	7
Minimum number of hops	1	1	2
Average number of hops	4.4	3.4	4.6

Table 5.6: Number of hops for convergence.

The mesh topology has the highest path reuse under partial optimization. This is because in most cases during handoff the path is extended from switches local to BS_1 and BS_2. Therefore, we see all the ASs on the original path being reused in the extended path. Since the CX is the ASs local to BS_1, partial optimization does not yield any optimization gain as shown by Optimized Gain 3. In the tree topology, the extended path traverses up the hierarchy to reach BS_2. After partial optimization, ASs from the CX to BS_1 are omitted, hence we observe fewer ASs from the original path being reused.

When full optimization is run we see that the mesh topology has the lowest reuse value. The values for path reuse for the tree topology are similar because no full optimization was needed. It is observed that partial optimization is sufficient, and full optimisation did not result in a shorter path.

In Optimized Gain 1 we see that full optimization results in significant reductions in path length. Furthermore, we see that a reasonable tradeoff is observed between path reuse and optimized path. This is important because the main problem with any rerouting algorithm is the drainage latency on the old path. As we have seen in Figure 5.12 high drainage latency would lead to high buffer requirements. To keep buffer requirements at a minimum we need to make a tradeoff between path reuse and optimal path. Maximisation of path reuse would entail more cells to drain. The buffer requirement at the BS is proportional to the propagation delay of the extended path. A high optimization gain means the extended path length is significantly longer than the optimized path. Therefore, an increase in drainage time will be observed if the path is significantly optimized[3]. From the results it is clear that the AN-handoff protocol provides a reasonable tradeoff between optimization gain and path reuse.

It can be observed that partial optimization is sufficient for the tree topology. This is shown by the 0.0% gain recorded for Optimized Gain 2. Both the mesh and redundant tree require full path optimization to be performed to yield a further 19.6% and 17.4% gain respectively.

The low Optimized Gain 3 value for the mesh topology indicates that partial optimization is not sufficient. A slightly higher value is observed for redundant tree because in most cases during handoff, the extended path needs to traverse up the hierarchy involving two or three ASs on the original path. As we can see partial optimization for both mesh and redundant tree is insufficient to yield total gains of 42.0% and 43.0% respectively.

In retrospect, we observe that in the partial optimization stage approximately 50% of the optimization gain is achieved. The proposed two-stage scheme has the advantage of limiting the end-to-end of the extended path. If full path optimization alone is performed, the extended path is significantly longer compared to the extended path after partial optimization. For example in the mesh topology, the extended path is 58% longer than the optimized path. Given that partial optimization does not require

[3]Only the number of hops are considered. A shorter path might have longer delays given factors such as bandwidth and traffic on the given path.

any additional signaling overheads the cost incurred is minimal. Apart from that we can program the ASs to invoke full path optimization under given constraints. For example, full path optimization is omitted in tree networks or full path optimization is invoked after the MH has migrated across multiple domains.

Table 5.6 shows the rate at which the CX was found. At worst ACR needed 7 hops before the CX is found. On average we needed 4.4, 3.4 and 4.6 hops for the mesh, tree and redundant tree topologies respectively.

5.5 Discussion

Our results show a number of benefits in using the ANs approach to MH rerouting. As can be seen from the results, the optimization scheme is not biased towards optimal path or path reuse. This is mainly due to the incremental optimization process where only portion of the path that is sub-optimal is optimized. Therefore, the drainage delay of cells from the old path is kept at a minimum. This is important since we have shown in Section 5.4.1 that buffer requirements are a function of the propagation delay on the extended path. The use of ANs technology allows processing at each AS to determine whether optimization(s) are needed. This leads to an efficient rerouting scheme. For example, in partial optimization no additional signaling is required to optimize the path. Most importantly processing is done at the switches instead of probing the network for information and processed at end-host(s).

Traditionally, the use of path extension has been limited due to its increased end-to-end delay [42, 56, 47] and chances of introducing routing loops. The advantages of using path extension are preservation of cell sequence, minimal cell loss and fast handoff [56]. The incorporation of partial path optimization removes any looping which might occur during path extension. Path extension fits into the nature of handoff in wireless ATMs. Since cells need to be forwarded from BS_1, a connection has to be made to BS_2. By making path extension as the initial stage of the handoff protocol we were able to overcome problems encountered in other schemes. This takes the assumption that the application is intolerable to loss and all untransmitted cells at the previous BS need to be salvaged.

The application of ANs enables the design of a generic protocol that works seamlessly with any network infrastructure. Other works [50, 194, 51] used a predefined topology to reduce the rate of handoff thereby reducing the number of path optimizations needed. The use of custom architectures does not provide a generic solution to all networks and the main limitation of these architectures lies in their CX discovery algorithm. In this study the main concern is with inter-domain migration. Therefore, the use of topologies such as that in [51] will maximise the performance of the handoff algorithm.

ASs along the path are able to resolve the MH's current location when a SETUP message is encountered. Consequently the FH experiences faster connection setup time. ASs can intercept any connection setup to the MH if required, hence reducing traffic directed towards the HA.

5.6 Conclusion

In this chapter, the performance of ACR has been extensively studied through simulation and analytical studies. ACR provides the benefits of path extension without the side effects of increased end-to-end delay and looping. From the analytical study, the performance of ACR was found to be comparable to that of traditional path extension. Moreover ACR provides a reasonable tradeoff between optimal path and path reuse. Simulation results have shown that ACR works well in the mesh, tree and redundant tree topologies. Hence ACR is generic, scalable and efficient. By taking advantage of the intra-network, computation loops can be removed at the switches and the reuse of the connection can be maximised. Apart from that, ACR does not need to deploy CX discovery algorithms from end-host(s), hence reducing overall signaling messages overheads. Moreover, no additional computation are performed by end-hosts and also end-host(s) are not required to probe the network for information.

In this chapter, the effectiveness of ANs have been evaluated in enhancing traditional methods for connection rerouting in mobile ATM. Furthermore, the advantages of performing connection rerouting within the network are shown and limitations of traditional methods are addressed. To emphasize further, ANs enable the following functionalities to be implemented: (1) the ability to efficiently determine which parts of the connection needs to be optimized and (2) automatic loop removal. The main advantage over an end-hosts based scheme is that no intensive probing is required and information is processed locally. These two steps reduce the signaling overheads considerably and more importantly results in fast adaptation to mobility.

Chapter 6

AMTree: Multicasting in Mobile IP Networks

In the past two chapters the problems of unicast connection have been addressed using ANs. One of the most significant contributions was shown in the previous chapter where it is possible to perform incremental optimization, resulting in high reuse. Hence, states allocated within the network are reused. The advantage of reuse is even more critical in multicast routing. Due to the large number of states required in multicast routing it is crucial that most of these states are retained during handoff. AN-based solutions preserve or update part of the multicast tree efficiently. In the past, work on applying ANs to multicasting has mainly concentrated on catering for heterogeneous receivers. In this chapter (and the next), the routing of multicast connections in the event of mobility is presented.

As mentioned in Section 2.2.1 there are four main problems concerning multicasting in mobile IP networks. In this chapter the main objective is to solve problems one and three. Problem one deals with source migration in a shortest path tree and problem three is concerned with receiver migration. The AN-based solution, called AMTree, addresses both of these problems. AMTree takes advantage of the processing capabilities at routers which enable MHs to continue sending packets to receivers after migration. Hence the multicast tree can be maintained with minimal modifications and minimal packet latency resulting in low handoff latencies and minimal disruptions to packet flow. Furthermore, due to intra-network processing, signaling overheads are low and tree is updated dynamically in an efficient way. AMTree uses the source-rooted tree (SRT) approach. The multicast tree incorporates the bidirectional state of core-based tree (CBT) [64] and is not dependent on the state maintained by the underlying routing protocol. Although AMTree incorporates some features of CBT [64], no selection and management of cores are required. In AMTree, the term core refers to an active router (AR) that has a degree greater than one and its main functionality is to provide abstraction during handoff. In AMTree, the designated cores to send packets to are dynamic and distributed. Hence no traffic concentration is experienced. A core can be easily programmed with monitoring and traffic management protocols applied only to parts of the tree. Unlike DVMRP [62] where one separate tree is constructed for each source, in AMTree only one multicast tree is required for a given session. Furthermore, periodic messages are not required to acquire topology changes and routers do need to maintain prune information. Senders that are not members of the multicast session can send packets to the tree although in this work only one primary source is assumed. AMTree encompasses the advantages of both source-rooted and shared tree with adaptation to host migration. The work herein also can be compared to other work that uses the shortest-path tree[1] such as protocol independent multicast (PIM) (sparse-mode)[2]. In PIM (sparse-mode) [65], receivers have the option to switch from an existing shared-tree to the shortest-path tree [65]. Therefore, if the source is mobile the problems discussed in the next section will need to be addressed. As mentioned, this chapter also considers receiver migration. The migration scheme proposed works in foreign networks where no multicast service is present and does not suffer due to rejoining delay. Furthermore, no packet loss and/or packet duplication are incurred.

A brief comparison of AMTree and existing multicast protocols is shown in Table 6.1, the term NC is short for New Connection and DNA means does not apply. In Table 6.1, **S** and **R** refer to the number of sources and receivers respectively. In Table 6.1, under the category *mobility support*, shared tree approaches support group members' mobility indirectly. When a source is mobile, the source only has to form a new connection to the core. As far as the core is concerned the new connection formed is from a new source

[1]Shortest-path tree and source rooted tree are used interchangeably
[2]PIM is separated into dense and sparse mode

given that the mobile source has obtained a new care-of-address. At the receiving end, receivers may experience delays due to the reconnection but the multicast tree formed is still valid. Hence traffic can still flow to receivers whereas in approaches based on SRT, traffic flow ceases.

Category	AMTree	DVMRP	MOSPF	CBT	PIM-Dense	PIM-Sparse
Tree Type	SRT	SRT	SRT	Shared	Shared	SRT
Multiple Sources	Yes	No	No	Yes	Yes	No
Number of trees in Multiple Sources	1	S	S	1	1	S
Router's State	$O(R)$	$O(S \times R)$	$O(S \times R)$	$O(R)$	$O(R)$	$O(S \times R)$
Use of Core	Yes	No	No	Yes	Yes	No
Mobility Support	Yes	No	No	Indirectly	Indirectly	No
Mobility Effect to Tree	Adapts	Invalid	Invalid	NC	NC	Invalid
Core's Adaptability To Mobility	Yes	DNA	DNA	No	No	DNA
Cores' Position	Dynamic	DNA	DNA	Static	Static	DNA

Table 6.1: A brief comparison of AMTree with other multicast protocols.

The rest of this chapter is structured as follows. The next section introduces the AMTree scheme. This is then followed by the simulation methodology used to gauge the performance of AMTree. The results obtained from the simulations are presented in Section 6.3. Finally, the conclusion is presented in Section 6.4. For clarity, a table of signaling messages used in MCoRe is presented in Appendix B.3.

6.1 The AMTree Protocol

The rationale of AMTree is to enable the adaptation of multicast tree during handoff. Current methods [68, 69, 71, 72, 70] are inefficient, require significant computational overheads, or do not consider the source to be mobile. Updating the tree from the edge of the network involves signaling overheads and large number of updates, proving it to be inefficient. AMTree is an efficient solution that requires no modifications to the distribution tree after handoff. This enables a multicast source to be mobile while still being able to send data down the tree. Receivers have the option of optimising the tree wherever an alternative path to an optimal portion of the tree is found. Furthermore, no periodic control messages are used to catch topology changes. This is because receivers are required to join the tree explicitly, thus data only flows over links that lead from the source to receivers. In the case where the link to the parent AR is down, an optimization process is executed to graft to another portion of the tree. AMTree does not suffer from traffic bottlenecks encountered in PIM [65] and CBT [64] since the tree is basically a source rooted tree. ARs dynamically filter out unnecessary control messages. For example join/optimization/NACK/ACK messages are filtered out by ARs nearest to the subscribing receiver.

The AMTree protocol is separated into three phases: (i) construction of active multicast tree (ii) update process during migration and (iii) tree optimization by active nodes.

6.1.1 Building the Multicast Tree

In the implementation of AMTree, a distributed location directory (LD) service is assumed to exist in the AN which maintains the contact point of a given group, in this case the source. The distribution and access to the LD can be implemented similar to the domain name service (DNS). The active router architecture used is similar to the router architecture presented in Chapter 4(Section 4.1). The construction of the multicast tree comprises three processes: join, leave and send.

6.1.1.1 Join Process

As in the traditional IP multicast protocol a receiver indicates its interest in a multicast session to its local router[3]. If an active session already exists then no request is made to the LD. Otherwise the local router queries the LD for the contact point. The contact point in this case is the source address or the router local to the MH and it is updated whenever the source migrates. A JOIN_REQUEST message is then sent hop-by-hop towards the contact point. Once the JOIN_REQUEST message is intercepted by an AR that is subscribed to the session, a JOIN_ACK is sent back to the initiating router. This completes the join process. If a receiver is the first to be subscribed then the processing AR loads the AMTree program and creates a state pertaining to

[3]Assuming that this router supports the multicast protocol

the session. The router then is subscribed to the session. Each router does not know how many receivers are subscribed to its child nodes. An AR only knows how many subscribers (end-hosts or ARs) are directly subscribed to it. Each router maintains a link to an upstream AR and a list of subscribers (which can be a neighboring AR or end-hosts). The states created at each AR for each session are shown in Table 6.2. The corresponding number of bytes for each data stored are shown in Table 6.2. As can be seen from Table 6.2, properties can be easily associated to each subscriber which enable employment of service differentiation such as filtering and scoping to parts of the tree. Besides that each state (parent and subscribers) is allocated a TTL value. The TTL value is refreshed after data are forwarded or received. When the TTL expires, for example when the link leading to a downstream AR is severed, the corresponding states are removed. Interested readers are referred to [195] for an analysis and design of active multicast services. Of importance are the states required to store receivers. For a given multicast session, the number of receiver states maintained at each AR is less than or equal to its degree (i.e the number of neighbors an AR has).

Data	Details
Upstream AR's IP	A link to the upstream AR is required to return any responses from receivers
Subscribers' IP	The downstream ARs/end-hosts to multicast packets to
Multicast Address	The multicast address associated with the current session
Time to live	The maximum duration in which the state is maintained
Contact point	The root node of the multicast tree and is obtained from the LD or HOP_DISCOVERY message
Forwarding IP	This is used to tunnel packets to a recently migrated receiver.

Table 6.2: Data structure maintained by each AR.

Data	Bytes
Upstream AR's IP	4
Subscribers' IP	$Degree \times 4$
Multicast Address	4
Time to live	$(1 \times NumberofSubscribers) + 1$
Contact point	4
Forwarding IP	4

Table 6.3: Number of bytes maintained at each AR.

In the multicast tree, ARs with at least one receiver are termed *core* routers. An example of a multicast tree with cores is shown in Figure 6.1. The main function of these routers is to provide an abstraction during handoff. In other words, no modifications are required to routers that are downstream from the core router during handoff. Furthermore, the shortest path from each connected receiver to a core is maintained given that migration happens higher up in the tree. In addition, control programs to monitor and customise traffic at a given subtree can be loaded at the core.

Although the cores in AMTree are formed dynamically they can be used to interconnect different networks using different multicast protocols. For example, if a network uses CBT then the $core_{CBT}$ (that belongs to CBT) becomes one of the cores in AMTree. An illustration of AMTree working with different multicast routing protocols is shown in Figure 6.1. Note that for a network using SRT, the source in the network is a pseudo root. As far as the receivers are concerned the source of the multicast is rooted at the pseudo-root. The only requirement is that $core_{CBT}$ understands the control messages sent by the root. Also $core_{CBT}$ is fixed unlike cores formed naturally in AMTree. Since in this chapter the main concern is with AMTree, the interoperation of AMTree with other multicast protocols requires further investigation.

MH migrations are generally local [183]. This means MHs more likely to stay within a domain. which translates to movement at the upper part of the multicast tree. Given this observation, modifications to the multicast tree happen more often at the upper part (i.e near the source's subnetwork) than at the leaf nodes. For example when a multicast tree spans several clusters of networks. Modifications to the tree are confined to the cluster in which the MH is located. Furthermore, these core routers are responsible for filtering out optimization messages if an optimization has been completed or the core router is currently undergoing optimization. Since the tree is a source-rooted tree, no traffic concentration occurs and cores in AMTree are distributed. This improves the scalability of AMTree.

As can be seen from Figure 6.1, the cores within the multicast tree form a structure that is quite similar to a CBT. In contrast the *cores* in AMTree are dynamically assigned and receivers do not join using cores as rendezvous point. The tree is built with each

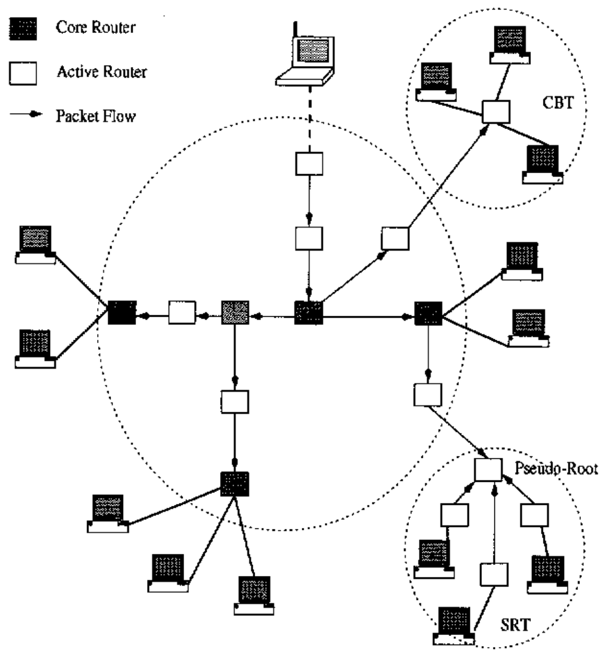


Figure 6.1: Example of a multicast tree building using the AMTree protocol.

receiver having a shortest path to the source and subsequent optimizations are updated on a demand basis. After handoff, some receivers might experience lower latency while others might experience increased latency. For example, when a MH migrates to a domain containing leaf nodes, receivers in that cluster will experience lower delay while others may observe higher delays. Depending on the observed latency, a receiver can request for an optimization if the latency exceeds a threshold.

6.1.1.2 Leave Process

Leave operation in AMTree is explicit. Each subscriber wishing to stop receiving packets[4] sends a LEAVE_MESG to its corresponding AR. As a result, the subscriber will be pruned from that AR. Apart from that each subscriber maintained at each AR then checks whether it has other subscribers. If no subscribers are found, then a prune message is sent upstream. The upstream AR then removes the downstream AR from its list of subscribers. A check is then performed to determine whether there are any subscriber(s) left, if none then a prune message is generated. AR has an associated

[4]Applies to receiver migration as well.

TTL value. The TTL value is refreshed after packets pertaining to the session are processed. When a TTL value for a given subscriber has expired, its entry is removed.

6.1.1.3 Send Process

A source interested in sending to the multicast tree indicates its interest in creating/transmitting to a multicast tree through its local router. The local router then registers with the LD stating the current contact point for the session (i.e specifying the root of the tree). The AR does not send any packets unless there is at least one receiver subscribed to the session. Once the AR has a subscribed member (either AR(s) or receiver(s)), it proceeds to multicast any packets sent by the source. When a packet arrives at an AR, the session details are accessed and a list of subscribers to the router is returned. The packet is then duplicated and forwarded to each subscriber.

In this discussion, only one source per multicast tree is considered although multiple sources are possible. Unlike DVMRP, AMTree is bidirectional and considers only one primary source. This simplifies the optimization algorithm which will be presented in Section 6.1.3. Due to the bidirectional nature of the state maintained at each AR, any non-member source that wishes to send packets to the session can do so. The storage space required at each AR is O(N), where N is the number of receivers subscribed to the session. This is similar to CBT [64] and PIM [65].

6.1.2 Handoff

The migration of the source entails two main processes: registration of the source's care-of-address and connection to the nearest core. Here the assumption is that the radio establishment between the base station (BS) and the MH is handled by the link-layer at the BS and MH. Also a mechanism such as dynamic host configuration protocol (DHCP)[196] is used to allocate a new care-of-address to the MH. Once the connection to the BS is established, the handoff protocol executes the following:

1. A REG_COF is sent to the LD. This updates the source entry at the LD with

the MH's current care-of-address. If an AR that has subscribed to the session is encountered by the REG_COF message the intercepting AR sends a CORE_CONNECT to the MH's current location. The CORE_CONNECT message is used by the AMTree core discovery protocol to determine the nearest node/core to connect to after handoff. The details of the protocol will be explained in the next section.

2. A HO_UPDATE message containing the MH's care-of-address is then sent to the previous contact point. A HOP_DISCOVERY with the MH's current care-of-address is then generated and multicast along the tree (rooted at the old source). Cores on the tree receiving this message will then generate a CORE_CONNECT to the address specified in the HOP_DISCOVERY message. The HOP_DISCOVERY message continues to be forwarded along the tree until it reaches a leaf node where it is discarded.

3. A HO_COMPLETE message is returned to the MH once the local AR has processed the CORE_CONNECT message.

6.1.2.1 Core Discovery

Core discovery is performed after handoff and is initiated by the core routers within the multicast tree. The discovery protocol uses the computation at each AR to determine which core/router is nearest to the MH. The designated core is then used by the MH to unicast packets, the core in turn will multicast packets along the tree.

The CORE_CONNECT message contains four pieces of information: hop count, designated core, multicast address and MH's care-of-address. The message (destined for the MH) is then processed, by each AR between the core and the MH. At each AR the hop count is compared with the observed hop-count (given that it has processed another CORE_CONNECT message) and incremented if required before being forwarded. If the message is the first to be processed, a new state is created and information in the HOP_DISCOVERY message is copied. Furthermore, if the AR is not part of the multicast tree, data shown in Table 6.2 are created. Basically, the AR is now subscribed to the session. If an AR that has already subscribed to the tree is encountered and it is not a core router then the hop count is set to zero and the designated core field is set to the current AR's address. A release message is then sent back to the old designated core. Unlike shared-tree methods such as CBT [64] the MH does not need to connect to any of the predefined cores.

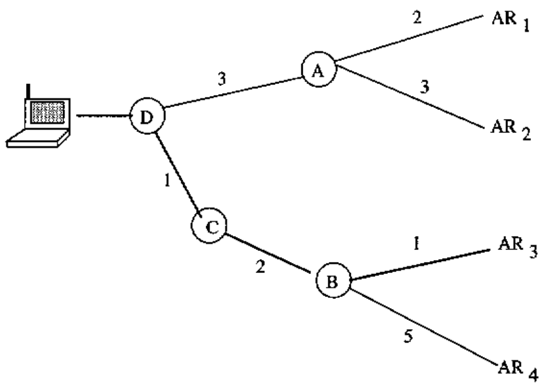


Figure 6.2: A network example showing hop count.

In Figure 6.2, AR_1 to AR_4 are core routers and A, B, C and D are routers. As can be seen, for core routers AR_1 and AR_2 hop counts to router A are different. Consider A and that a soft-state for AR_1 has already been created with a hop count of two. When the CORE_CONNECT message from AR_2 arrives at A, router A compares the hop count within the message with the hop count stored in its soft-state (i.e the hop count for AR_1). Since the stored hop count for AR_1 is smaller, the CORE_CONNECT message from AR_2 is discarded. Then a RELEASE_MESG is sent to AR_2 and all soft-states allocated between routers A and AR_2 are freed. Note that the release message only affects states created by the CORE_CONNECT message. At D the CORE_CONNECT message from AR_1 has incurred a hop count of five. On the other hand, the message from AR_3 has only incurred a hop count of four. Therefore, D updates its state with AR_3 and sends a RELEASE_MESG to AR_1 which unsubscribes all routers along the path to AR_1. As a result, multicast packets are then sent down the path D-C-B to AR_3. Note that the multicast tree is augmented with routers D, C and B. Apart from that if a CORE_CONNECT message passes through a core router, the message is dropped. Thus, unnecessary signaling messages are prevented from flooding the network during migration. For example if C is a core router, clearly the shortest core to connect to is C, therefore any subsequent CORE_CONNECT messages received by C are discarded and release messages sent. The pseudo-code in Algorithm 6.1 shows the core discovery process at the AR.

```
  IF we are subscribed but not a core THEN {
    Generate a new CORE\_CONNECT message.
    Set hop count to zero
    Set designated core to our address
    Discard old CORE\_CONNECT message.
 } ELSE
     {
       IF no soft-state exists THEN  {
       Create soft-state(hop count, multicast address, core address)
       Increment hop count value
       Forward CORE_CONNECT message onwards to MH's care-of-address
     }
    ELSE
     {
        Get hop count from CORE_CONNECT message
        IF hop count < hop count in soft-state THEN {
          Replace soft-state data with data from current message
          Increment hop count value
          Forward CORE_CONNECT message onwards to MH's care-of-address
        } ELSE {
           Discard message
           Send a RELEASE_MESG back to originating core router
          }
      }
     }
```

Algorithm 6.1: Discovery Process at AR.

6.1.2.2 Packet Delivery After Handoff

Once the tree is augmented with the MH's new location, the designated core router has to multicast packets up the tree as well as down. If a non-member source sends packets to the session the algorithm presented in this section applies. The change in direction of packets after handoff is shown in Figure 6.3. In Figure 6.3, the routers leading up to the old contact point do not have any receivers subscribed therefore they are pruned from the tree. The algorithm to forward packets is as follows:

- Make a duplicate of the message and forward each message to downstream subscriber(s).
- If the source message did not come from an upstream AR then forward the message upstream given that the upstream AR did not prune itself from the session.

- The message is terminated once it reaches the old contact point.

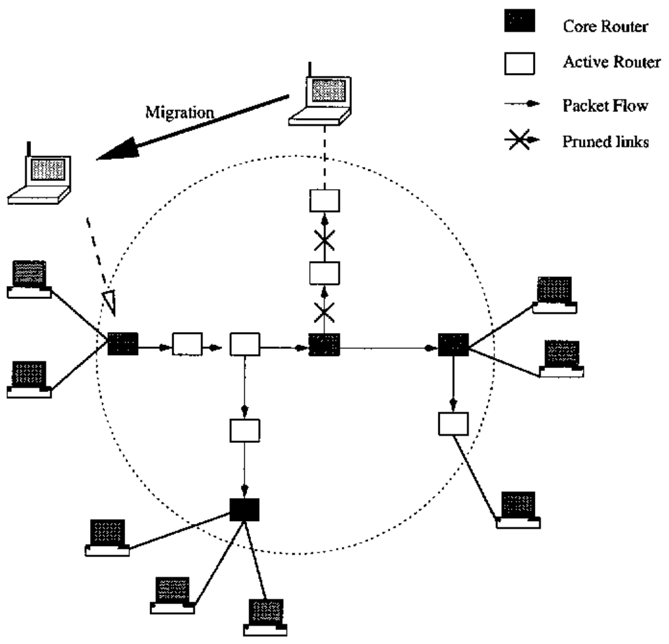


Figure 6.3: Example of packet flow after migration.

6.1.3 Optimization

Two questions arise with respect to optimisation: *when should optimization be invoked?* and *how is optimization performed?*. In the first question, optimization can be invoked when the receiver notices a significant increase in end-to-end latency.

In the receiver's case, the optimization process is performed on a demand basis. This is due to some receivers having lower latencies after handoff or vice-versa. Therefore, based on a given receiver's tolerance to delay, optimization might only happen periodically or randomly. Hence the optimization process is performed when QoS constraints are violated. Note that receivers are neither aware of source migration nor shorter paths. Without any probing of the network, the only indication (albeit not reliable) that a shorter path exists is when a significant increase in end-to-end delay is observed. In the periodic invocation case, core ARs invoke the optimization process periodically to determine whether an alternative path exists. The downside is that optimization may be performed when no source migration has taken place, thus causing unnecessary signaling messages. Finally, the optimization process can be invoked explicitly. After source migration, a message is multicast to all cores to invoke optimization. Each core determines whether the parent node is valid, given the source's new location. If not then a new parent is found. The approach taken by us consists of on demand routing and explicit notification. This means that receivers invoke the optimization when quality of service constraints are violated. For example when RSVP [197] is used, migration of source may cause the agreed upon contract to be violated. As mentioned, AMTree also adopted the use of explicit notification. For simplicity, the source invokes the optimization process explicitly when it migrates across LANs. This is because migration within a LAN will not cause significant increase in delay. Moreover updating is only required at the local core, thus subscribers downstream are not affected. Besides the above, the optimization message is also used when topology changes and the link to the parent AR is severed. Note that ARs themselves can choose to invoke the optimization when necessary. At the AR, when the TTL of the parent node expires and the multicast session has not ended yet[5], the core AR then invokes the optimization process. As a result, given an alternative path, a new parent link is formed which links the core AR to a different part of the multicast tree. In the following paragraphs the optimization process is described from a receiver's point of view. Note that similar processes can be invoked from a core AR as well.

The next question that needs to be addressed is: *how is optimization performed?* The optimization process entails sending an OP_DISCOVER message to the MH's care-of-address. If a shorter path exists then optimization is possible. The multicast tree is then augmented and the original link pruned. In the optimization algorithm, only one primary data source is considered. Since the multicast tree is bidirectional (i.e., the receivers are able to send feedbacks to the source) there is no need to build a separate tree for each source. Hence the multicast tree has the benefits of CBT but no discovery is required to find the optimal core(s). Note that the OP_DISCOVER message is routed based on the recorded MH's care-of-address at any given time. Therefore, if the source migrates during the optimization process, and the OP_DISCOVER message and ARs along the path do not have the updated MH's care-of-address then the message will be routed to the recorded care-of-address. Therefore, the optimization process needs

[5]Given that the session does not end when link failure occurs

to be repeated for the given receiver(s). To prevent the network from being flooded with OP_DISCOVER messages, only one OP_DISCOVER message is sent for a given core. This is sufficient to ensure an optimal path for other receivers that directly subscribe to the core. For example, if an AR receives an optimization request from its downstream subscriber and an optimization has already been performed, the request is discarded. The pseudo-code for the optimization process at a given router is shown in the pseudo-code of Algorithm 6.2.

```
Get AR_ADDRESS (neighboring AR's address) from OP_DISCOVER
Check if we are subscribed to multicast session

 IF not THEN {
   create new multicast session state and add AR_ADDRESS to subscriber list
   Forward OP_DISCOVERY message with AR_ADDRESS set to our IP
 }
 ELSE {
  Check if AR_ADDRESS is one of our subscriber
  IF not THEN {
    Add AR_ADDRESS to subscriber list
    Send GRAFT_MESG to AR_ADDRESS
  }

  IF we have not optimized before THEN
   Forward OP_DISCOVER message with AR_ADDRESS set to our IP
  ELSE discard message
 }
```

Algorithm 6.2: Optimization Process at AR.

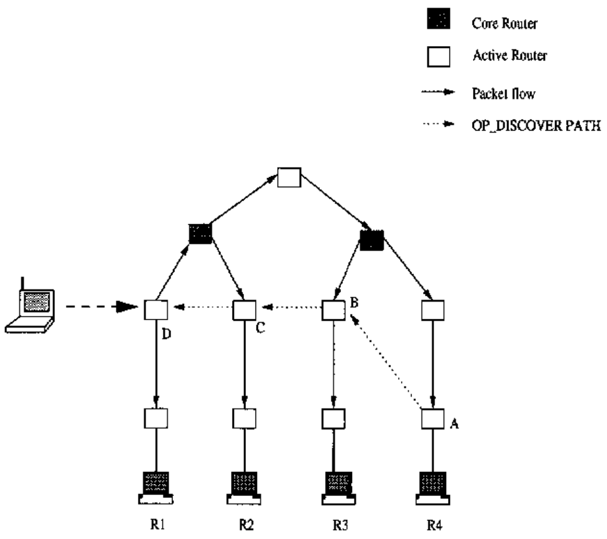


Figure 6.4: Example of an optimization being performed.

When a receiver decides to perform an optimization, a request is made to the local AR.

The local AR then transmits an OP_DISCOVER message with the MH's care-of-address as its destination. As can be seen in Figure 6.4, **R4** has initiated an optimization. The OP_DISCOVER message is relayed through other parts of the tree[6]. At each AR, a check is performed to determine whether it is subscribed to the session and if so whether its downstream link is where the OP_DISCOVER message originates from. If the AR is subscribed to the session (e.g the parent of **R4** in Figure 6.4) no computation is performed. Otherwise the session state in Table 6.2 is created. The AR now is subscribed to the session. In Figure 6.4, router *B* is subscribed to the session but it does not have router *A* as one of its downstream links. Clearly an alternative path exists. As a result, router *B* creates a new state for router *A*. A GRAFT_ACK is then sent back to *A* indicating a successful graft and the old link can be pruned. When the OP_DISCOVER message reaches *C* where no entry exists for router *B* as before a new state is created and a GRAFT_MESG sent. In this case, the GRAFT_MESG message is sent to router *B*. This process is also repeated at router *D* and resulting tree obtained is shown in Figure 6.5.

If a given router has already processed the OP_DISCOVER message, it assumes the route to the MH to be optimal. Therefore, any subsequent optimization requests by subscribers are dropped. If the subscriber requesting the optimization does not have an entry then an entry is created for the subscriber and GRAFT_ACK is sent. As shown in Figure 6.4, receivers **R1** to **R3** do not require any optimization once the optimization from **R4** has been performed. If an alternative path exists for **R3** then the optimization process is carried out.

6.1.3.1 Pruning

Upon receipt of a GRAFT_ACK message, an AR transmits a PRUNE_MESG on the up link to its parent AR. At the parent AR the corresponding subscriber specified in the PRUNE_MESG is unsubscribe. If no subscriber(s) is left then the AR prunes itself from its parent AR. In the case where the prune message reaches an AR that has

[6]Here the assumption is that the underlying routing protocol will route the message through the shortest path. If an alternative path exists then the path is assumed to be optimal.

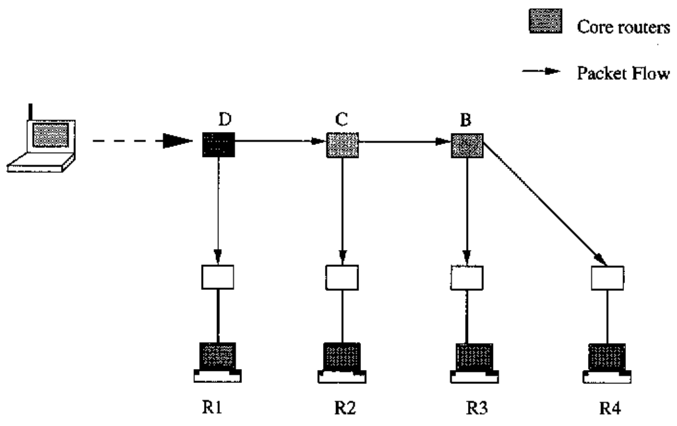


Figure 6.5: Resulting tree after optimization from R4

already pruned itself the message is discarded.

The PRUNE_MESG maybe lost because the services provided by IP are unreliable. Therefore, upstream ARs continue to multicast packets even though a prune message has been sent. As a result downstream ARs receive duplicate packets and loops may form. To overcome this problem, when a packet arrives from the parent node in which a PRUNE_MESG has been sent, the packet is discarded and a PRUNE_MESG retransmitted. If the packet contains sequence number, then it is also used. For example when an AR receives packets with similar sequence number from two or more neighbors. A copy of the packet is then multicast to all subscribers and a PRUNE_MESG sent out to all neighbors (except the parent AR) responsible for the duplicate packet.

6.1.4 Receiver Migration

So far only source migration process has been described. In this section, a simple scheme for receiver migration that does not assume the existence of multicast service at the foreign network is outlined. The AMTree scheme minimizes rejoining delay, packet duplication and packet loss for receiver migration as well. Note that the scheme proposed here is different to those of reliable multicasts where the aim is to ensure a given packet is received by all group members. Here the aim is to enable seamless handoff. From the receiver's view, during handoff it remains subscribed and causes little disruptions to traffic flow.

To ensure that the receiver continuously receives multicast data, a forwarding pointer is used. This means that after handoff, a receiver sends a notification to the previous BS while waiting for a rejoined notification at the current network. Once the receiver rejoins the multicast session successfully, the forwarding pointer is terminated. The advantages include:

- Receivers are able to receive multicast traffic in networks that do not support multicast service.
- A fast rejoin is possible since there is no need to wait for IGMP query request.
- Receivers are not susceptible to lengthy rejoin process in networks where no group membership exists.

6.1.4.1 Handoff

Figure 6.6 gives an illustration of the processes involved during receiver migration. When MH_1 migrates to FA_2 a check is performed to determine whether the current network has any members subscribed to the multicast session. If so, the receiver notifies the local multicast agent (FA or a multicast router) to update its multicast table. Otherwise, a binding update is sent to FA_1. The receiver also makes a request to rejoin the multicast session. Once FA_2 has intercepted the binding update message, it is considered subscribed to the session, therefore any subsequent receivers of the same group migrating into the current network are not required to send a binding update. As a result at any given time only one forwarding path exists, thereby avoiding the tunnel convergence problem. After processing the binding update message FA_2 forwards the message to FA_1.

The binding update contains the receiver's new care-of-address. If an AR that is subscribed to the multicast session intercepts the binding update message (as the case in Figure 6.6), the AR (C_1) updates its *Forwarding IP* state to the care-of-address stated in the binding update message. In this case, the binding update is not forwarded to the FA_1 but any multicast packets received by the AR will be tunneled to the MH until a STOP_FORWARD message is received. The STOP_FORWARD message is sent by FA_2 once the rejoin process is completed.

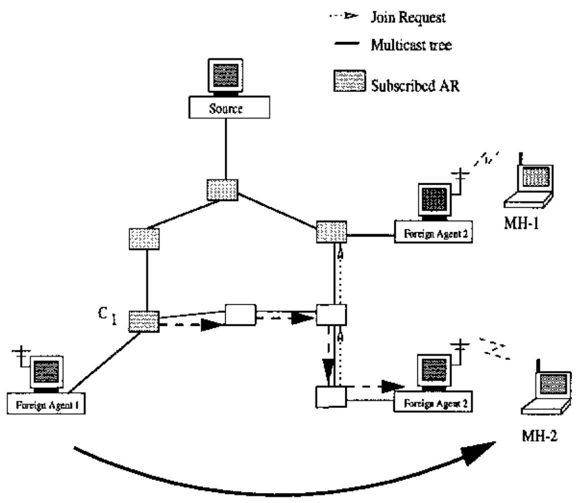


Figure 6.6: Illustration of receiver migration and rejoining multicast session

The receiver is not considered to be part of the multicast session until its rejoin request has been acknowledged. Since the destination address of multicast packets contain class D addresses, the packet has to be tunneled to the receiver. Therefore, the receiver needs to perform a check (*Forwarding IP* field) to determine whether forwarding is required, if so a copy of the multicast packet is encapsulated and forwarded to the care-of-address stored. This means if the *Forwarding IP* field is set, then forwarding is required.

Consider Figure 6.6 where a join message is heading up to the source. The join process is similar to that described in Section 6.1.1.1. Once the receiver has rejoined the multicast session, a STOP_FORWARD message is sent to FA_1. Upon receiving the STOP_FORWARD message, FA_1 or an AR removes the receiver from its *Forwarding IP* list.

Since the receiver does not need to wait for IGMP query or delay incurred by the rejoining process, a fast handoff is achieved. In networks, where no multicast service is available, the proposed handoff method is not affected since it is not a requirement that multicast service exists. The main advantage offered by AMTree is that while receiving packets from the previous BS, multicast service can be installed. Recall that in AMTree, services provided by ARs can be augmented. Due to the join request a foreign network that does not have multicast service will be able to support multicast service. As a result subsequent receivers migrating into the network can utilise multicast service. Unlike approaches such as [72] where a HA handoff is required, in AMTree the branch created by the first receiver persists until all group members have left the network. Since at any given time only one branch leads to a given network and the branch persist until all receivers leave, the problems associated with tunnel convergence are avoided.

6.2 Simulation Methodology

The simulation studies were done using Opnet [192] and the following investigations are carried out:

- A comparison between AMTRee and bidirectional HA method. Performance metrics measured were, the resulting end-to-end latency incurred before/after handoff and after optimisation. Furthermore, the handoff latency incurred for both schemes are investigated. In AMTree the handoff latency is calculated from the time the MH establishes a link with the BS to receiving a HO_COMPLETE message from the local AR. These experiments looked at how the latency varies with different receiver sizes.
- The resulting end-to-end latency after handoff due to the resulting non-optimal tree. This is to show that while no immediate modifications are done to the tree the end-to-end latency is tolerable.
- The effects of receiver sizes on the number of cores formed within a multicast tree. This is significant in that the number of cores represent the varying degree of abstraction achieved.

The network used is modeled after traditional networks with several enhancements. These enhancements allow customised programs to be injected into routers. Here the programmable approach is taken where a multicast join message loads the AMTree algorithm into routers. The join message issued by receivers are active packets in that they hold a flag which loads the appropriate active program in each router it traverses through. Once a program is loaded the router passes all packets related to the multicast session to the AMTree program accordingly. Furthermore, the router provides predefined interfaces for accessing routing information, creating soft-state and forwarding of capsule/packets. As in traditional network models the network modeled in this study provides "best-effort" packet delivery in that the underlying network is unreliable where packets can be lost, duplicated or delayed. Hence multicast applications are needed to provide reliable transport of data packets. The assumption is that for each subnet connected to the network there exists a base station (BS) which enables MHs to connect to the network. MH migrations in the simulation model are random in that they can migrate locally or globally.

The receivers are considered static in the simulation and each receiver that is interested in joining a multicast session contacts its local router. IGMP[66] is used to relay membership information. RIP [198] is used to route control messages such as join/leave and assume that given a destination address will route packets on the shortest path. The multicast tree built is symmetric in that each router only knows who its upstream and downstream routers are. Therefore, upstream and downstream packets will follow the same route.

The simulation was done on a 50 node mesh topology with degree ranging from three to six. The data rate of interconnecting links between ARs is set to 10Mb/s. The wireless links have a data rate of 2Mb/s. Bit errors at the wireless link are assumed to be handled by a data link layer protocol. Each FA (or BS) is assumed to manage a *cell* and overlapping of *cells* means there are no *silent* areas. The MH controls the time of handoff, meaning the time of handoff and the duration of the handoff procedure can be measured. The MH migrates randomly to any of the receiver's subnet. Hence the migrations are not necessarily local. The BSs broadcast advertisement messages at intervals of one second. The advertisements are needed to enable MHs to detect that they are in the *cell* area of a given BS.

The only traffic in the network are those solely of the multicast session and the AMTree protocol. At any moment in time there is only one ongoing session. This is in accordance with the design specification where a specific tree is built for a unique session. The packets generated by the source are of size 1024 bytes at a rate of one packet per second.

At the start of a simulation each receiver is randomly connected to the topology. The performance of AMTree on varying number of receivers (ranging from 10 to 50) was then determined. This serves to simulate low to high density multicast tree. The AMTree algorithm is loaded into ARs at the start of simulation. Ten simulation runs were performed for each case and the results averaged.

6.3 Results

6.3.1 Handoff Latency

The handoff latencies for different number of receivers are shown in Figure 6.7. The latencies range from 4.4ms to 4.8ms. As can be seen the handoff latency achieved by AMTree is much lower than that of bidirectional HA method [10]. In the bidirectional HA method the increase in latency when the number of receivers reaches 30 is mainly due to the increase in traffic. The latencies measured in AMTree are for cases where the full core discovery protocol is employed rather than of connection to a core encountered during registration of care-of-address (i.e core encountered from REG_COF message). A significantly lower handoff latency will be observed in highly dense groups due to the higher probability of encountering a core during registration. A significant feature of AMTree is that the discovery protocol does not stop in finding a core. The discovery protocol is executed further to determine the shortest path possible to a core.

6.3.2 Number of Cores

The number of cores discovered for different number of receiver sizes are shown in Figure 6.8. The higher the number of cores the higher the degree of abstraction. This means that less modifications are required, which leaves a large proportion of the tree unaffected after handoff. Although according to results in Figure 6.8(a) the number of cores is proportional to the number of receivers, the increase in cores does not translate

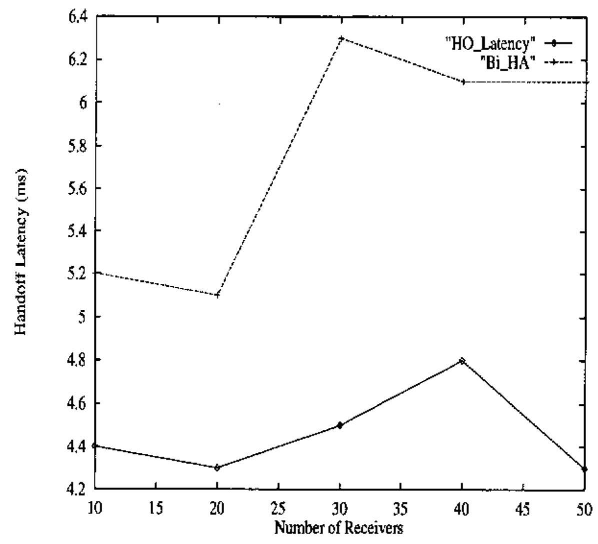


Figure 6.7: Handoff latency for varying receiver size

to a higher consumption of bandwidth due to core discovery and/or optimization. On the contrary, it increases the number of messages filtered out thereby conserving the overall bandwidth. As can be seen from Figure 6.8(b), as the group size increases with group size, the number of messages filtered increases. By observing Figures 6.8(a) and (b), when the multicast group size is at 45, there are 22 cores and on average two messages are filtered. That translates to 44 messages filtered in the entire network.

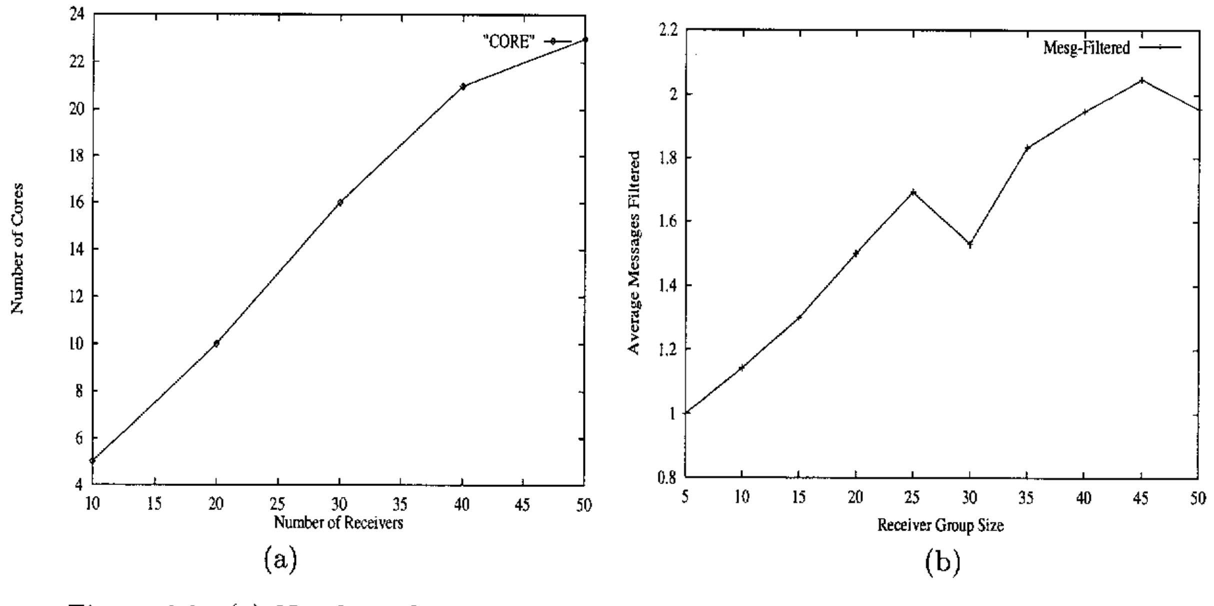


Figure 6.8: (a) Number of cores versus number of receivers. (b) The corresponding number of messages filtered.

Table 6.9 show the number of subscribers (which include routers and end-hosts) main-

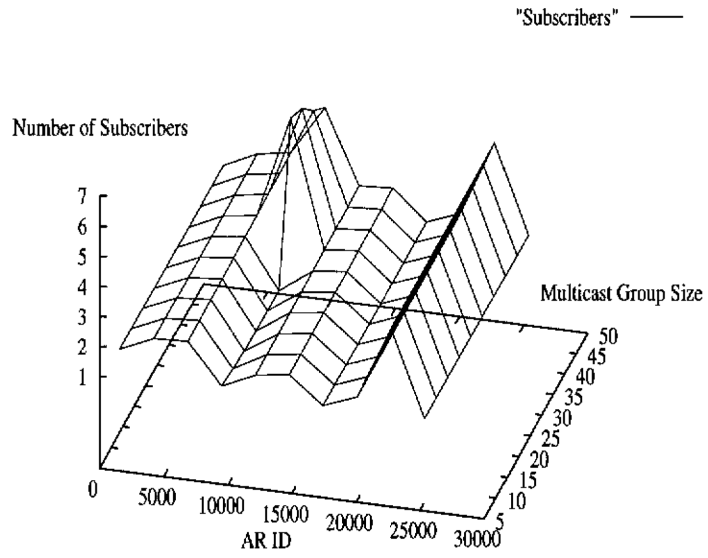


Figure 6.9: Number of subscribers maintained by each core with respect to multicast group size.

tained by each core router. The storage requirements for a given session are distributed across 20 cores. Note that ARs are also classified as subscribers. From observation, as the multicast group size increases, ARs at the edge of the network maintain a higher number of subscribers. This fact is only true in dense mode situation where most ARs will have at least one subscriber downstream and subscribers will join the multicast session at the AR that the *join message* first encountered.

6.3.3 End-to-End Latency

In this subsection a comparison of AMTree's end-to-end latency to the bi-directional HA method specified in the Mobile IP's specification [10] is described. Figure 6.10 shows the latency incurred by each receiver (40 receivers scenario) in the simulation.

It is shown in Figure 6.10, the bi-directional method incurs a high latency due to triangle routing. On the other hand, with AMTree the latency after handoff registered a slight increase. This increase is partly due to an increase in hop count to the migrated subnet. For example when the MH is at subnet *1*, receiver *i* might have a latency of

4ms. After the MH migrates to subnet *2*, the best possible latency obtainable is 4.5ms. Therefore, a slight increase in end-to-end latency is observed. This slight increase is unavoidable and video applications need to adapt to this slight increase in delay.

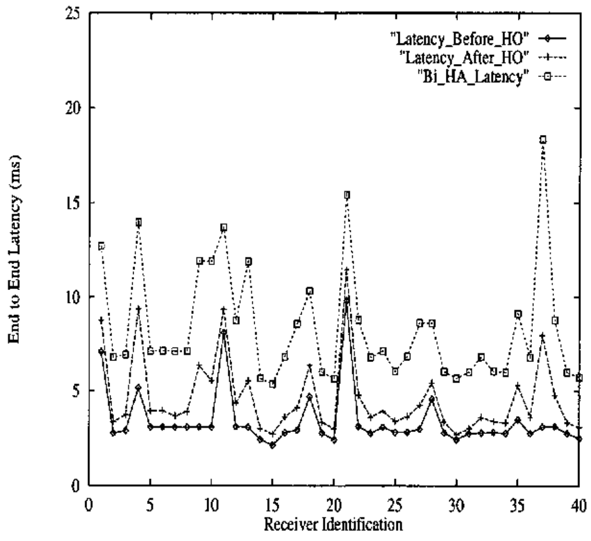


Figure 6.10: Comparison between AMTree and bidirectional HA method. The number of receivers in this 50 node topology is 40.

Apart from that a comparison between AMTree to shared-tree approach was carried out. Figure 6.11 shows a comparison between AMTree, Bi-HA and shared-tree with regard to end-to-end latency. The number of receivers is 40 in a network topology with 50 nodes. As can be seen from Figure 6.11, shared-tree generally has a stable end-to-end latency given that the RP is positioned in a strategic location. AMTree outperforms shared-tree on most occasions in terms of end-to-end latency. Also for shared-tree the results showed that the end-to-end latency observed does not vary as much as Bi-HA or AMTree. This is due to the position of the core/RP. Since the core/RP is manually positioned in the centre of the topology, all BSs have approximately equal distance to the core. Hence as the source migrates to each BS, the distance to the core will be approximately equal to that of its previous BS. Therefore, the delay experienced by the shared-tree approach in the experiment represents a "perfect" situation where end-to-end delay is not affected considerably by source migration. When receivers/sources migrate further away from the core, the performance of the shared-tree is similar to that of the Bi-HA method.

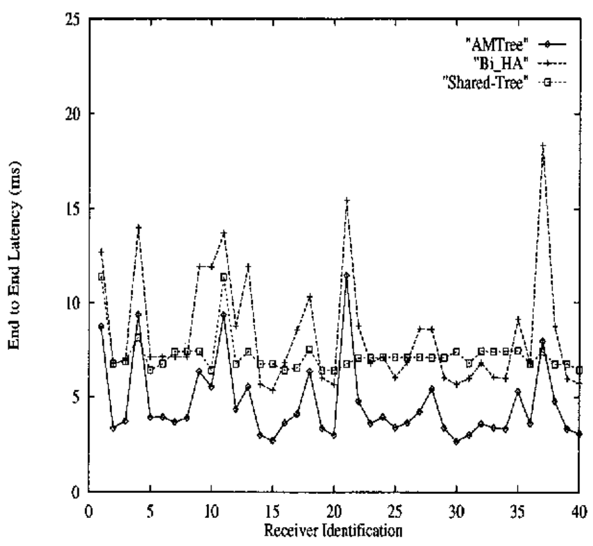


Figure 6.11: End-to-end latency of AMTree (no optimization) to shared tree approach.

6.3.3.1 Hop Count

The performance of AMTree in cases where the MH migrates with an increasing number of hops was also analysed. From Figure 6.12 the delay after optimization remains fairly constant and is significantly lower than the bidirectional HA method and also AMTree with no optimization. As the MH moves further away from its HA, the Bidirectional HA scheme's end-to-end latency increases rapidly due to triangle routing. On the other hand, in AMTree protocol, the increase in latency is restricted because the probability of encountering ARs is high with increase in the number of receivers. This is because AMTree takes advantage of the geographical placement of ARs in the tree and connect to the closest one. Therefore, as the number of receivers increases the probability of encountering an AR is high.

6.3.3.2 Optimisation

The optimization metrics measured are the end-to-end latency and hop count. Here results for a network with 40 receivers is [7] shown in Figure 6.13. As can be see from Figures 6.13(a) and (b), a significant reduction in number of hops is observed. Hence a lower end-to-end latency is observed. Note that in Figure 6.13 some receivers do

[7]Tests for receivers ranging from 10 to 50 was also carried out.

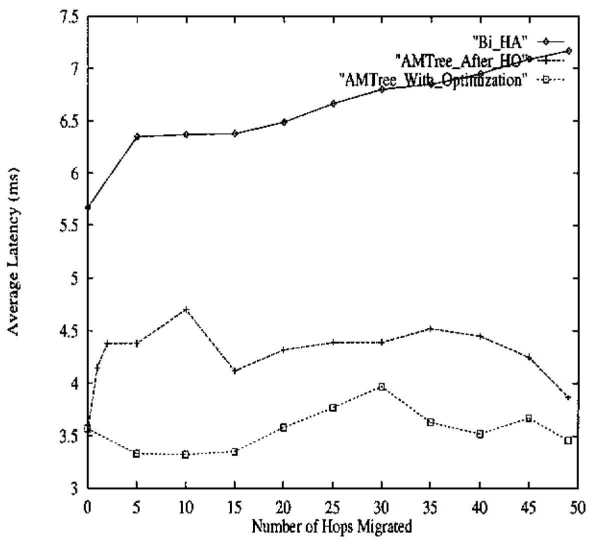


Figure 6.12: Latency vs. Migration Distance.

not gain from the optimization process and may suffer from a slight increase in latency. Certain cores sustain an increased number of receivers thereby increasing the processing time after optimization. Note that in AMTree the receivers join the multicast tree at the start and they are static throughout the simulations. If receivers are dynamic and some receivers join after source migration the path from those receivers to the source would be optimal.

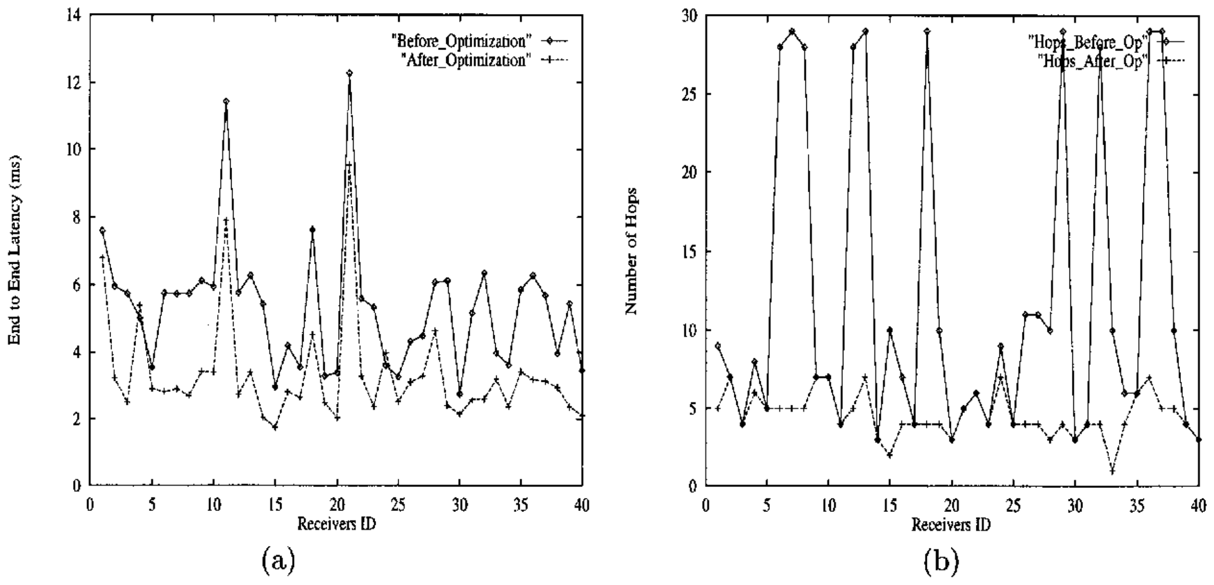


Figure 6.13: Hops and end-to-end latency after optimization on a 50 node mesh topology with 40 receivers.

Besides that an investigation was carried out to compare the AMTree's optimization algorithm to that of remote subscription where the entire multicast tree is rebuilt to accommodate the MH's new location. Hence the latencies incurred should be the best achievable for the given MH's location. The latency incurred for both algorithms are shown in Figure 6.14. From Figure 6.14, some receivers achieve better latency compared to remote subscription. Therefore, at best the AMTree's end-to-end latency is lower than remote subscription.

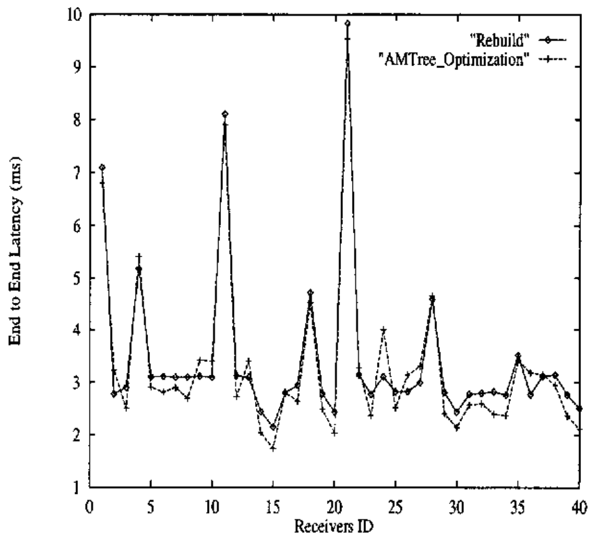


Figure 6.14: Compares the latency of AMTree's optimization to rebuilding a new multicast tree

6.3.4 Tree Efficiency

Table 6.4 shows the tree efficiency achieved by AMTree. Essentially the results in Table 6.4 justify the arguments that most of the tree can remain unmodified. Due to the fact that ARs only maintain the parent node, any changes due to source migration will only update the given state leading to the parent node. Downstream receivers are unmodified. As can be seen in Table 6.4, the group density increases the efficiency. This means that for a receiver size of 40 there is only 4.4% difference between our multicast tree and Total-Rebuild. Note that this high efficiency is due to increasing number of cores which modify the tree structure without requiring changes to downstream subscribers. Also note that due to the increased number of receivers, most subscribers will be concentrated at edge routers, therefore for most cases these receivers are unaffected.

Number of Receivers	Efficiency
10	73.5
20	68.75
30	76.4
40	94.6
50	97.3

Table 6.4: Efficiency of the AMTree.

6.3.5 Qualitative Comparison With Other Approaches

In Table 6.5, a comparison between the features of AMTree with those of other mobile multicast approaches is presented. The category *Join and graft delays* refers to the time it takes for receivers to rejoin the multicast session. For Bi-HA and MoM the join delay is dependent on the receiver's distance from its HA. As for RBMoM the join delay is range dependent. As mentioned in [73], the worst case and ideal performance of their scheme is that of Bi-HA and Rebuild respectively. In AMTree, the join latency is dependent on the distance to the previous BS. Since migration is generally local, the rejoin process incurs minimal delay. Note that in AMTree the second rejoin process was not taken into account because it does not affect the disruptions observed by the receiver. Table 6.5 also shows the transparency of each scheme. Transparency

Category	Rebuild	Bi-HA	MoM	RBMoM	Shared-Tree	AMTree
Optimal routing	Yes	No	No	Range dependent	No	Yes
Tunnel Convergence	No	Yes	Yes	Yes	Yes	No
Transparency	No	Yes	Yes	Yes	No	Yes
Tuneling	No	Yes	Yes	Yes	No	Minimal
Join & graft delays	Minimal	High	High	Range dependent	Moderate	Minimal
Seamless Receiver Migration	No	No	No	Range dependent	No	Yes
Agents Involved in Routing	FA	HA,FA	HA,FA	HA,FA,MHA	FA	FA

Table 6.5: This shows a comparison between AMTree and other mobile multicast approaches.

here means whether subscribers experience any disruptions due to mobility (source or receiver). Only Rebuild and Shared-Tree do not provide transparency. Shared-tree is included in Table 6.5 because it supports mobility indirectly and no modifications to the current protocols are required. The problems that arise with CBT and PIM are, (i) receivers suffer from high disruptions delay (ii) high packet loss during handoff (iii) possibly increased end-to-end delay. Further, packet duplication may occur when receivers rejoin the multicast session after migration. Despite these problems, packets do reach multicast group members after handoff unlike SRT approaches.

With regard to tunneling, note that unlike Bi-HA, AMTree requires minimal tunneling because tunneling is only used during receiver migration. Since tunneling is only used when there are no members at a foreign network and cease to exist after the receiver has rejoined the multicast session, the tunnel convergence problem is avoided.

As for seamless receiver migration, current multicast protocols do not ensure smooth handoff and rejoin. Bi-HA and MoM suffer from rejoin latency which is dependent on the HA's distance from the receiver's location. In Rebuild and Shared-Tree, the existence of multicast services is assumed and considerable delay is incurred if the foreign network does not have group members. RBMoM's rejoin latency is range dependent. Depending on the range, at worst, its performance may be similar to Bi-HA and at best similar to Rebuild.

6.4 Conclusion

In this chapter, an active multicasting protocol that enables the adaptation of the multicast distribution tree during handoff has been presented. AMTree enables the multicast tree to remain intact after handoff. The end-to-end latency incurred after migration is shown to be minimal and significantly lower than that of bidirectional HA and comparable to that of remote subscription after the AMTree optimization. The handoff latency is shown to scale well as the number of receivers increase. The handoff latency will remain fairly constant as the number of receivers increase due to higher probability of finding an AR that is subscribed to the session. In the bidirectional HA method the handoff latency is proportional to the round trip time from the MH's current location to its HA. AMTree can be easily optimised and end-to-end latency achieved is comparable to rebuilding the tree from the new location. Unlike remote subscription no new tree is constructed at the MH's new location. Apart from that AMTree's storage requirement is kept at a minimum and ARs take an active role in reducing un-necessary control packets. Furthermore, AMTree has demonstrated how the advantages of source-rooted tree and shared-tree can be incorporated to handle MHs. The functionalities embedded in ARs can be extended easily to include bandwidth adaptation programs if any of its subscribers are mobile. Besides that the problems associated with receiver migration have been addressed. The algorithm is simple, fast and does not assume the existence of multicast service at foreign networks.

An intermediate follow up of AMTree is to investigate the incorporation of multiple sources. The multicast aspects of having multiple sources are trivial but the challenge is in the optimization process. Another avenue in which AMTree can be extended is to overcome the problem of scoping, QoS and security.

In this chapter, the main objective of ANs was to determine a node (subscribed to the multicast session) that is nearest to the MH after migration. As can be seen the determination of this node is straight forward where routers cooperatively work out which node is closest. Another important aspect of this chapter is the filtering of messages so as to eliminate unnecessary optimization requests and hence improve the scalability of AMTree. Also, multicast states within the network are updated on-demand. This means that most of the multicast tree remains unchanged.

Chapter 7

MCoRe: Rerouting Multicast Connections in Mobile ATM Network

In the previous chapter AMTree was shown to be an efficient solution and adaptive to mobility. In this chapter rerouting of multicast connections in connection-oriented networks will be considered. Unlike the rerouting of unicast connection, in a connection-oriented multicast environment more issues need to be considered. These issues will be addressed in Section 7.1.

In ATM networks, the problem of unicast connection rerouting in the event of host migration is well studied. This can be seen in the numerous rerouting categories cited in Section 2.1.2. In contrast, rerouting multicast connections in the presence of mobile sources and receivers is an area that has not received adequate attention by researchers in the past. As can be seen in Section 2.2.2, existing multicast protocols such as UNI 3.0/4.0 [75][82] assume fixed end hosts. An existing work proposed by Toh [84] which considers MHs, suffers from sub-optimal path and assumes a LAN.

To overcome the aforementioned problems, this chapter outlines a protocol, called multicast connection rerouting (MCoRe), which provides flexibility to a source rooted tree (SRT) and is designed for rerouting multicast connections in WANs. SRT is ideal for delay sensitive applications, therefore it is crucial that the problem of mobility is addressed. MCoRe uses information within the switches for rerouting multicast connections. This chapter shows how MCoRe utilises intra-network processing to reroute multicast connections dynamically after source/receiver(s) migration. The MCoRe protocol has low signaling overheads, maintains cell ordering, incurs minimal cell loss, low handoff latency, low buffering requirements and requires minimal state (amount of information recorded) at switches. MCoRe is efficient and simple and does not employ complex algorithms in order to update the multicast tree after each migration. The most important aspect of MCoRe is that the existing multicast tree does not have to be reconstructed after source migration. Moreover, a majority of the states allocated before migration are reused.

The SRT referred to in this chapter is independent of the underlying unicast routing algorithm and is receiver initiated. In SRT approaches, traffic concentration at the core or RP and suboptimal path are avoided. Traditionally, the main drawback of SRT is scalability [199]. This is due to the fact that a separate tree has to be constructed for each host interested in transmitting to the multicast session. Thus, the amount of state at routers/switches is $O(S \times R)$, where S and R are the number of sources and receivers respectively. Later in this chapter, a discussion on making SRT scalable and adaptive to multiple sources will be presented. Unlike the unidirectional tree of UNI 3.0 [75], the connections in the MCoRe approach are bidirectional. Hence receivers are able to send feedback messages back to the source.

The problems addressed in this chapter are illustrated in Figure 7.1, where a MH, the source of the multicast, migrates from BS_1 to BS_2. Each receiver interested in joining the multicast session sends a request to the session source. The multicast tree shown in Figure 7.1 is receiver initiated and classified as a SRT. When the MH migrates to BS_2, some connections of the multicast tree are no longer valid and some multicast cells from BS_1 maybe lost. As we can see, rebuilding the tree would prove to be highly disruptive due to the time taken to setup up all the necessary connections. Furthermore, the rebuilding and tearing of connections would be costly in terms of signaling overheads. In receiver handoff, when receiver migrates to a different tree branch, no cell duplications may occur. At the new location, the receiver may receive cells already received before migrating. The problem is aggravated because ATMs

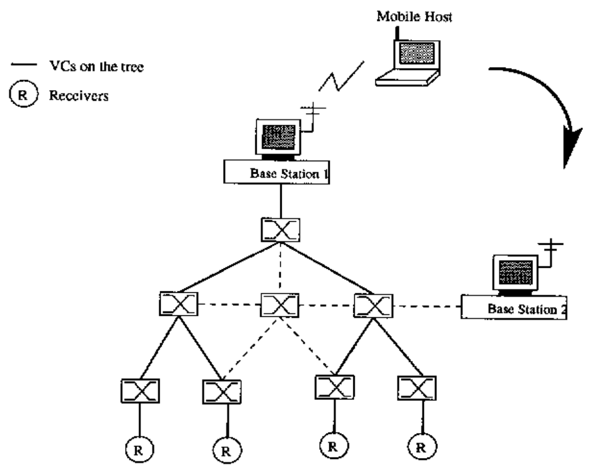


Figure 7.1: An example showing the effects of handoff in connection oriented networks.

do not incorporate the notion of sequence number in the cell structure. Therefore, a synchronization process is required to determine whether a given receiver has received a particular cell. Moreover, the synchronization process must be generic in that it must be independent of higher layer protocols. Another problem with receiver migration is that the network in which the receiver migrates to may not have multicast capabilities or does not have group member(s). If no multicast capabilities exist then the receiver's session will be interrupted until it moves out of the current network. In networks which support multicast but currently have no group members, the receiver experiences a high disruption time due to the rejoining process. On the other hand, if predictive handoff is used, a rejoin can be initiated in advance thereby minimising disruption time.

This chapter is structured as follows. In the next section a generic design specification is presented. The specifications provide requirements for future work in this area. Then the MCoRe scheme is presented in Section 7.2. This is then followed by a simulation methodology and the corresponding results. These are presented in Section 7.3 and 7.4 respectively. A discussion on future work and open research issues remain to be investigated are presented in Section 7.5. Finally, the conclusion is presented in Section 7.6. For clarity, a table of signaling messages used in MCoRe is presented in Appendix B.3.

7.1 Design and Assumptions

In the design phase of the MCoRe protocol a set of requirements were adhered to. These design requirements serve to provide a generic design specification for future work in this area. The requirements identified in this study are:

- Low handoff latency
- High reuse
- Low signaling overheads
- QoS
- Low cell loss and ordering
- Minimal state at switches
- Transparency
- Loop avoidance

Handoff latency should be minimal to ensure smooth handoff. High reuse of existing connections conserves resources because new connections entail overheads in terms of additional states and signalling. With regard to signalling, a scalable solution should not require the probing of switches for information, for instance to obtain a map of the multicast tree. Probing the network incurs high signaling overheads. By reducing signaling overheads, handoff latency due to delay in probing and computation of data collected can be minimised. Besides that, the QoS established in the original tree must be maintained across handoff. Therefore, during connection rerouting or when the multicast tree is rebuilt, the QoS agreed upon must be satisfied.

After handoff, cells that are in-transit must be salvaged. This depends on whether the application can tolerate losses. But depending on the speed of transmission and size of the multicast tree there maybe a significant number of cells in transit. Tearing down connections without salvaging cells in-transit may significantly affect QoS observed at receivers, hence resulting in high disruption time. The salvaging operation must preserve cell ordering. After handoff, once a new tree or connection(s) has been established, the solution must ensure that salvaged cells are forwarded first before new cells are forwarded to the receivers. In the case of receiver migration, when a receiver rejoins the multicast session it may receive duplicate cells or the next expected cell may have already passed. Therefore, a multicast protocol should have a mechanism to overcome the aforementioned problems when receivers rejoin the multicast session after handoff.

Switches that are part of a multicast session should maintain minimal state information regarding the session. Moreover, the state does not scale as the multicast group size increases. Furthermore, rerouting algorithm should make use of available information within the switches, for example information acquired from the underlying routing protocol. Moreover, the rerouting process must be transparent to the source and receivers of the multicast group. Hence receivers and sources are not required to rejoin or rebuild the multicast tree after migration. Transparency is attained by having the network adapt to host movement without requiring the intervention of end-hosts.

Loops cause cell misordering and cell duplication. It is imperative to eliminate loops in rerouted connections. Loops arise because a given switch does not know which downstream/upstream switches are already subscribed to the multicast session. Therefore, merging of connections without prior knowledge of switch subscriptions results in the occurance of loops.

In MCoRe, switches within the network are programmable and there exist standard sets of interfaces for manipulating VCs related to the current multicast session. In the active switch architecture used, the programs injected into ASs, reroute VC(s) or VP(s) pertaining to a given multicast session as mentioned in Chapter 5. The interfaces at ASs include functions for tearing and setting up connections, updating of translation table and control over which VCs are multicast. Note that in this paper issues such as security (in ANs) and QoS adaptation will not be addressed. These issues require further investigation and are beyond the scope of this paper. Besides the above, the work presented here does not attempt to map IP multicast models over ATM networks.

7.2 The MCoRe Protocol

In this section the MCoRe protocol is presented. The main objectives of MCoRe are to satisfy the design requirements introduced in Section 7.1 and show how programmability within switches can be used to reduce signaling overheads involved during rerouting. In this section, the operations that allow the multicast tree to be updated after source migration are presented. The update process causes no disruption to traffic flow, thus allowing for a seamless multicast session.

An overview of the MCoRe protocol, is illustrated with the help of Figure 7.2. As can be seen from Figure 7.2, a multicast session is rooted at BS_1. The receivers are represented as logical domains (LDs). Basically a LD represents a subnet and it contains at least one receiver. The role of a LD will be elaborated further in the next section. After handoff, the MH sets up a connection back to BS_1. In other words, a path extension process is performed from BS_1 to BS_2. Therefore, after handoff the MH is able to multicast and receive packets through the extended path. After handoff MCoRe invokes an optimization process that updates the multicast tree. The MH notifies all LDs of its new address. In response, the proxy root (PR) of each LD sends out a tree update message back to the mobile source. As the tree update message traverses, the network to the mobile source's current location, each AS on the path checks whether the next hop being forwarded is the parent node. In Figure 7.2, AS_x determines that the update message will not be forwarded to its parent node. Hence a new connection is setup as shown in Figure 7.2. Once the update message arrives at the source, the extended connection and connections that are invalid are torn down. In Figure 7.2, these connections are labelled VC_a to VC_c. Before tearing down any connections, cells that are on the old path must be drained properly to preserve cell ordering. This is done by injecting a marker cell along the old tree which signals core ASs to update their multicast state and start forwarding cells from newly created connections. Core ASs have more than one receiver subscribed to them.

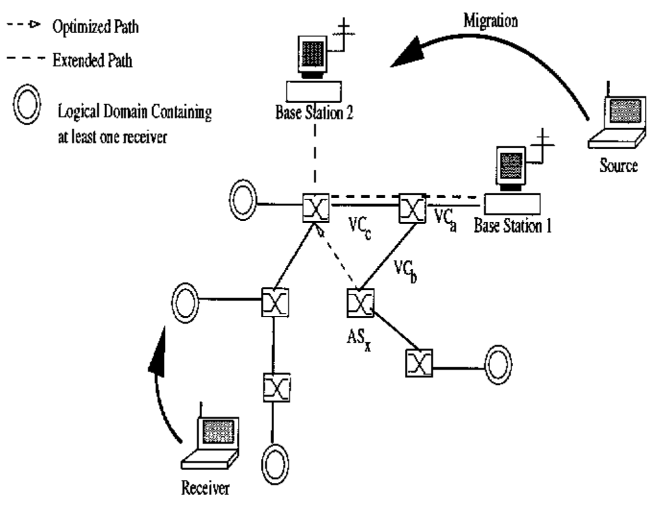
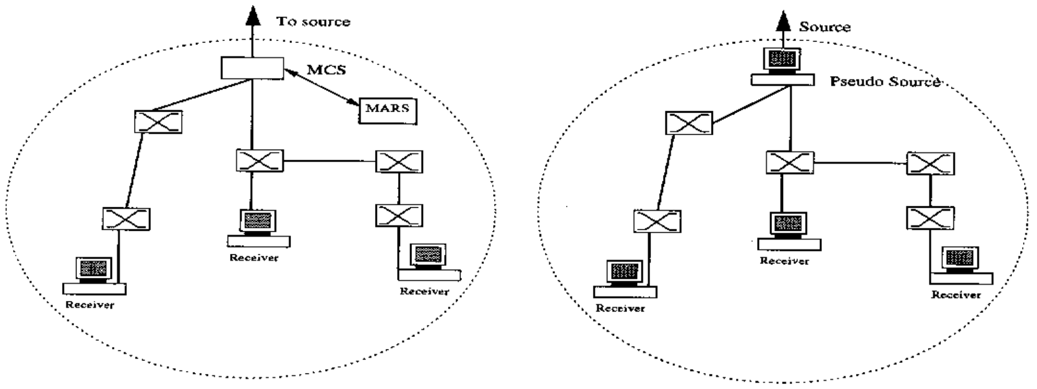


Figure 7.2: An Overview of the MCoRe Protocol.

Receiver migration also involves path extension. Other than the advantages such as fast handoff, path extension enables a receiver to continuously receive cells in networks that do not have multicast service. The next stage of receiver migration involves a rejoin of the multicast tree at the current network. Therefore, the extended path is maintained for a short period of time and is torn down after the MH has rejoined the multicast tree at the foreign network. The main problem that needs to be addressed is cell synchronisation to ensure that the receiver obtains the next expected cell after it has rejoined the multicast session.

7.2.1 Logical Domain

As mentioned in the previous section MCoRe assumes the notion of LD. LD is a logical grouping of receivers and possibly with its own multicast protocol. For example, all receivers belonging to a peer group can be classified as a LD. By introducing LD, migration within a LD can be handled using algorithms such as those in [84] and reduce handoff rate considerably. Another advantage of having LD is that connection rerouting process is transparent to those receivers within the LD since they are not required to participate in the rerouting process. In Figures 7.3 (a) and (b), two LDs are shown running two different multicast protocols. MCoRe assumes the existence of a PR within each LD. The PR of a LD can be the MCS or core (or rendezvous point) if shared tree is used. The role of the PR will be explained in the next section.



(a) MCS approach. (b) Source-rooted

Figure 7.3: Logical Domain with (a) MCS approach and (b) Source-rooted

7.2.1.1 Proxy Root

The PR handles all join requests on behalf of receivers within the domain. Further, the PR keeps track of receivers within its domain and administers the setting up and tearing down of connections. PR is programmable and responds to signaling messages sent as part of the MCoRe protocol. Depending on the multicast used, the PR may be setup dynamically or administratively. For example a PR might be set to the MCS which is handling the given multicast session. The process by which a PR is chosen within a LD is beyond the scope of this paper.

The PR must have the following functionalities in order to support the MCoRe protocol:

- Send ADD_PARTY message: to initiate a join to a multicast session. Source address of the multicast session is obtained from a session directory.
- Process join requests from receivers: new join requests are not forwarded beyond the LD if the PR is already subscribed.
- Tear down connections: if no receivers are subscribed within the domain, a prune message is sent to its parent node. This means the PR has to keep track of receiver(s) within its domain since a MH may migrate without explicitly unsubscribing from the session.

- Respond to HO_UPDATE message: upon receipt of this message the PR updates its mobility binding by recording the source's current care-of-address. Once the mobility binding is updated, the PR then sends a TREE_UPDATE message back to the source. If the PR is an AS, then the operations outlined in Section 7.2.4 are performed. Otherwise the TREE_UPDATE message is simply forwarded on the VC allocated (leading to the parent node) for the multicast session.

7.2.2 Tree Creation and Maintenance

In this section the joining process initiated by PRs is outlined. In MCoRe, the SRT is bidirectional and a single VC is used for receiving and sending. Hence receivers are not required to setup connections back to the source in order to send acknowledgements for received data. Each connection on the multicast tree is constructed using a single VC. This conserves resources and reduces the number of connections involved during connection rerouting. One of the main problems with the use of a single VC is cell interleaving, which occurs when cells from receivers are interleaved with cells from other receivers/source(s) at merge points (i.e., node degree greater than one). The problem of cell interleaving has been addressed in [200] and [77].

Data	Description
ParentVC	The data structure containing information about parent node. Information stored are: VCI/VPI, address, port and QoS allocated.
RD_i	This list contains information for each subscriber downstream. VCI/VPI, QoS allocated and prune flag.
GrpID	The multicast group identifier.
SAddr	The source's current care-of-address.
NConn	The VCs corresponding to new connections created or received during the optimization process.
ExConn	The VC information of the extended path during handoff.
LastRM	The sequence number from the last resource management (RM) cell seen.
CellCount	The kth cell after the RM cell has passed. The counter is reinitialized to zero each time a RM cell is processed.

Table 7.1: Data Maintained by AS for a given multicast session.

Each switch subscribed to the multicast session maintains the information shown in

Table 7.1. Basically each switch knows which of its immediate downstream ASs are subscribed to the multicast session. Note that the number of subscribers maintained for a given AS is minimal. This is because states for each receiver belonging to a multicast group are distributed among ASs in the network. Each AS is aware of subscriptions at downstream ASs and it does not need to store information of all subscribers of a downstream AS. During optimization, ASs distinguish between newly created VC and old VC on the old tree to prevent cells from being forwarded on the optimized VC. VCs which belong to new and old trees must be distinguishable in order to prevent loop(s) from occurring. This will be shown later in Section 7.2.4 how the optimization process takes advantage of the *NConn* variable and how different VCs are differentiated.

7.2.2.1 Initiating a Multicast Session

To start a multicast session, the source registers its intention with the PR of its LD by sending a packet with the following tuple {*source permanent address, multicast address, services*}. Note that the source's permanent address (i.e., home address) is used. When receivers interested in joining the multicast session transmit join requests, the source's permanent address is returned during multicast address resolution. Therefore, receiver requests will be directed to the source's home network. If the source has left its LD then a care-of-address is returned in the release message. The corresponding receiver then uses the care-of-address in a subsequent join request. The problem of location management has been addressed by Acharya et al. [49]. The approach taken by Acharya et al. [49] is similar to that in Mobile IP [10] where a home agent manages all MHs that have migrated from its domain. However, in [49] a MARS service is required to provide a mapping between multicast address and source's home address.

Upon receipt, the PR relays the registration to a multicast directory server (MDS). Here MDS refers to server that maintains a mapping between multicast address and source's ATM address, for example MARS. Here, the assumption is that multiple MDSs exist which provide distributed mappings of multicast addresses. When MDS confirms registration, an acknowledgement is sent to the initiating PR, which is relayed to the source. Unlike MCS [78], the PRs in MCoRe are not required to maintain a connection with the MDS for any information regarding group membership. Therefore, in the MCoRe model, the source is neither aware of the multicast size nor is it required to add receivers one at a time. This is because the management of multicast group members has been implemented in ASs and PRs are notified of the source's care-of-address.

After registration, the source's PR waits for join request(s) from receiver(s). When a join request arrives, the PR sets up a new connection to the source if one does not exist. Then an active setup message augmented with an active tag is sent to the receiver. The active tag is used by each AS to load the MCoRe program. The MCoRe program then creates state information as shown in Table 7.1. The MCoRe program controls VCs, signals and maintains states pertaining to the given multicast session. After the connection is setup the source starts transmitting. Readers interested in AS architectures are referred to [201][169].

7.2.2.2 Join Process

The signaling process in which receivers join the multicast session is adapted from ATM's native multicast mechanism, specified in UNI 3.0 [75] and UNI 4.0 [82]. Note that due to the limitations of the native ATM multicast mechanism, modifications to the signaling messages are required. These modifications allow the programming of ASs which enable bidirectional connections and installation of necessary states for connection rerouting.

A PR interested in joining a multicast session first makes a query to the MDS. In response, the MDS returns the multicast session source's ATM address. A LIJ message is then sent to the source containing the multicast group identification. Each AS checks whether it is subscribed to the multicast group specified in the LIJ message. If so a setup message is returned, which results in a new connection (branch) to the LD initiating the join request.

Consider Figure 7.4, here LD_1 has joined the multicast session. When the join request from LD_2 is intercepted by AS_x, a setup message is sent to the PR of LD_2. At AS_x, a new entry is then added to the multicast table (see Table 7.1) reflecting the new

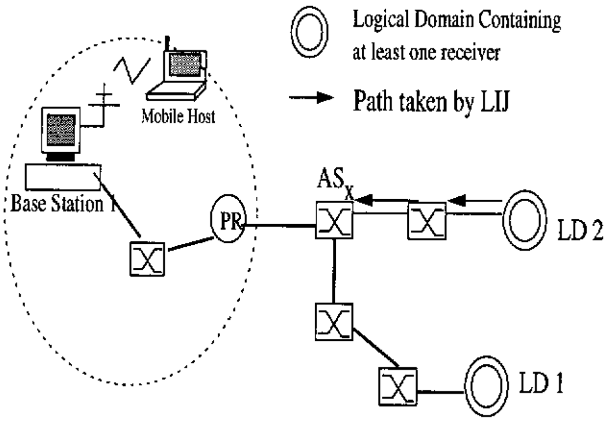


Figure 7.4: Illustration of Join Process.

connection to LD_2. Once added, incoming cells from the ParentVC are multicast accordingly. Thus, the latency of join process is the time for the join request to encounter an AS subscribed to the multicast session plus the connection setup time.

In a LD where no proxy root is assigned, the adjacent AS (programmed with MCoRe) takes the role of the PR. The receiver does not need to be equipped with the M-CoRe protocol. This is because the intermediate switch from the receiver is already programmed with the MCoRe program. One of the main advantages of ANs is dynamic augmentation of functionalities. When a receiver migrates to a LD that does not support multicast, ASs within the LD can be programmed with the MCoRe protocol. Subsequent receivers migrating into the LD will be able to utilize the programmed ASs. In this scenario, one of the ASs programmed within the LD will have to be selected as a PR. For simplicity, the designated PR is the AS located at the boundary of the peer group.

An exception in the joining process is that join messages originating from receivers in the same LD as the source must not graft a new branch on the connection that connects the source to the PR. When the source migrates, the connection from the source's previous location to the PR will be torn down. As a result, all receivers grafted on the branch will be dropped from the session. Alternatively the PR could perform an update on the source's old connection after migration. But since the assumption is that the rerouting process within LD is independent of MCoRe, the update of connections is left to the rerouting protocol employed in the domain.

7.2.2.3 Leave Process

The leave operation is performed by sending a prune message. When the PR detects that it has no receivers subscribed to a given multicast session it sends a release message on the outgoing VC for the multicast session (i.e the connection leading to the parent node). At the merge point, upon receipt of the release message, the AS removes states allocated for the given subscriber.

In the leave process it is assumed that the PR has a mechanism for detecting whether it has any receivers subscribed within its LD. This is crucial when receivers are mobile. A MH may migrate out of the current LD without explicitly tearing down its connection to the PR. A solution to this may involve a soft-state VC approach. Each VC is associated with a timer which is refreshed each time cells are received. Once the timer expires, the corresponding VC is torn down. The advantage of using soft-state is that no additional signaling needs to be invoked to tear down connections.

When tearing down connections, care must be taken because unlike connection rerouting in unicast communications, an old connection cannot be torn down without first checking whether there are receivers relying on the connection for data.

7.2.3 Handoff

The handoff process in MCoRe is a simple path extension process. This means that a new connection is setup from the source's new location to the source's old location. It has been shown in [56] that path extension has a low handoff latency, maintains cell ordering and has minimum cell loss. There are two main disadvantages with path extension: increased end-to-end delay and likely formation of loops. The problem of end-to-end delay will be addressed in the next section. As for preventing the formation of loops the partial optimization process proposed in Chapter 5 is used. MCoRe adapts partial optimization scheme to suit multicast connections.

To illustrate the partial path optimization process consider Figure 7.5. When the source migrates to LD_2 it sets up a path to BS_1. As can be seen in Figure 7.5, the

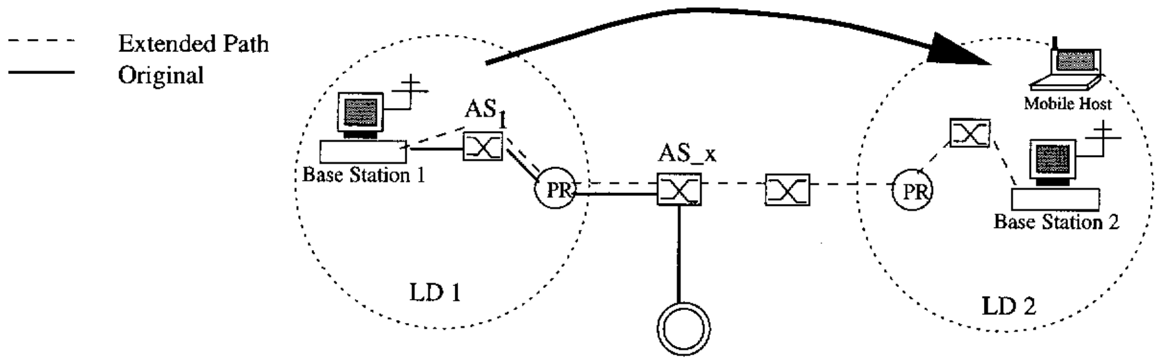


Figure 7.5: Handoff with partial optimization.

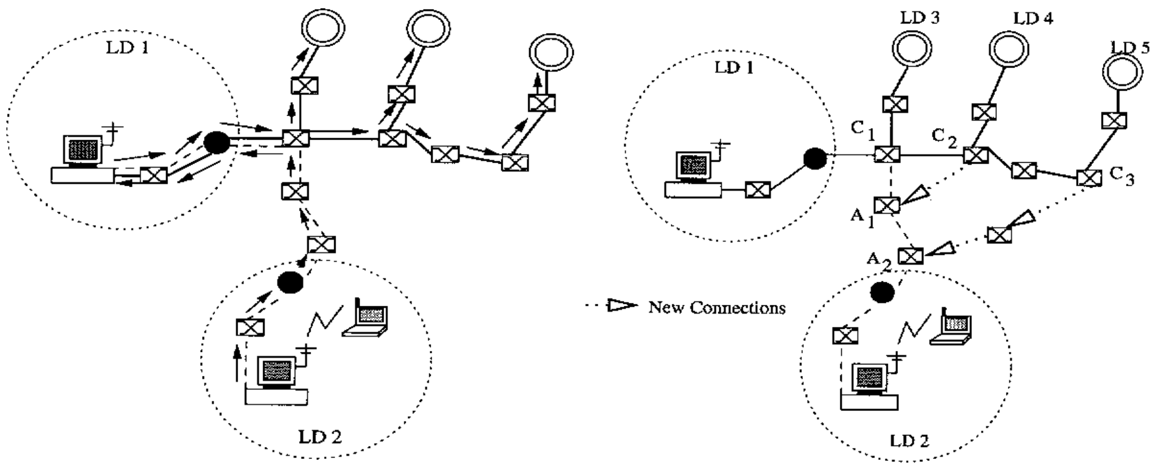
extended path can be short-circuited at AS_x, given that there are no receivers in LD_1 and on the branch leading to LD_1. If there are receivers in LD_1 then the path is short-circuited at the PR for LD_1. This is the reason why receivers in the same LD as the source are not allowed to create a branch on the connection between the source and PR. The partial optimization process is performed after the extended path has been established. Note that the partial optimization process is performed automatically by ASs without the knowledge of PRs. Moreover, no additional signaling are generated to invoke the optimization process. The only signaling involved are those used for tearing down connections.

The process in which the path is partially optimized is as follows. Firstly a check is performed by PR of LD_1 to determine whether there are any receivers. If there are receivers subscribed then the connection PR-BS_1-AS_1-PR is torn down after cells from the path are drained. Thus, the resulting path ranges from BS_2 to PR of LD_1. The tearing down process is similar to those in [202][203]. If there are no receivers subscribed, a PARTIAL_OP is sent by the PR on the extended path. At each AS, a check is made to determine whether the extended path has diverged from the current AS. From Figure 7.5, it can be seen that the extended path has diverged at AS_x, hence AS_x is chosen. In general, the AS with a degree more than one is chosen. This differs from Toh [84]'s scheme where a snapshot of the multicast tree is required.

The computation for determining whether the path has diverged is trivial. Since downstream AS(s) are subscribed to the multicast session, a path to the source's current location can be computed. This is similar to PNNI [58]'s computation of designated transit list (DTL). Here the DTL is computed for the source's current location. A check is then made to determine whether the next hop matches one of the entries in RD_i. If the next hop is not the parent node, the AS performs connection rerouting, hence draining and tearing down the original path from the AS to BS_1 and the extended path. In case the AS encountered is a branch point such as an AS with degree greater than one, referred to as AS_{br}, then the partial optimization process is performed by AS_{br}.

7.2.4 Optimization

The optimization process is initiated by PR of the LD in which the source is in. To optimize the multicast tree a HO_UPDATE message is multicast by the source (see Figure 7.6(a)). The HO_UPDATE message contains the source's current care-of-address. The main purpose of the HO_UPDATE message is to notify PRs of the source's care-of-address and begin the optimization process. When PRs receive the HO_UPDATE message, they note the source's care-of-address and transmit a TREE_UPDATE message. The TREE_UPDATE contains the source's address and is used by ASs to compute the corresponding DTL. Upon receipt of the TREE_UPDATE, the AS determines whether there is an optimal route to the source's current location.



(a) HO_UPDATE (b) TREE_UPDATE

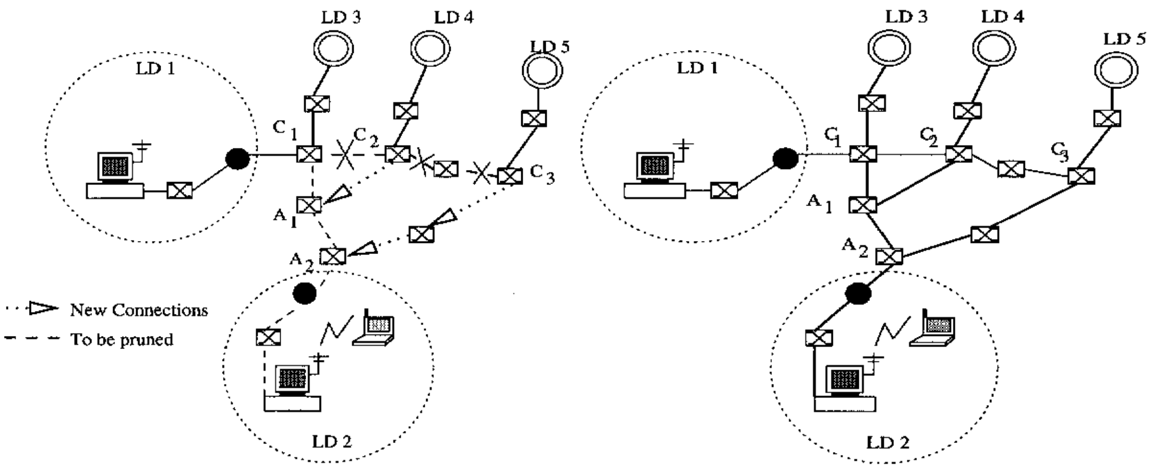
Figure 7.6: (a) The flow of HO_UPDATE message (b) New connections created after transmitting TREE_UPDATE

Figure 7.6(b) shows new connections created by the TREE_UPDATE message. After a PR has sent the TREE_UPDATE message, each AS computes a DTL using the source's address within the TREE_UPDATE message. The route obtained is then used to determine whether the next hop is the parent AS. If so, then no new connection setup is required. For example, ASs after C_3 to LD_4 do not have to create a new connection because the TREE_UPDATE message is being forwarded to the parent AS. If the next hop is not the parent AS, then a new connection is setup, for example C_3 in Figure 7.6(b). At C_3 the AS determines from the computed DTL that the next hop is not the parent AS. Therefore, it sends a setup message to the source with the MCAST_TYPE field set to NEW_TREE. As a result, a new connection is setup[1] towards the source. ASs that create new connections or have new connections incident on them are referred to as core ASs. In Figure 7.6(b), both C_2 and C_3 have created new connections to the source's current address. At C_2 and C_3 these new connections are marked as NEW. This is to prevent cells from being forwarded to any VCs marked as NEW and the occurance of loops. Note that C_1 needs to update its multicast table during optimization because currently its parent node variable is referred to LD_1. Here a new connection is not required because the extended path is reused.

When the setup message reaches an AS (e.g., AS_1 and AS_2 in Figure 7.6(b)) subscribed to the multicast session, the AS proceeds to merge the connection. The connections are merged using the following policy. If the AS contains a state for the extended path, in other words the extended path passes through it, then the connection is merged with the extended path. The newly formed branch is then marked as NEW. Once the connections are merged, the setup process for the new connection is terminated. A TREE_OP_ACK is then generated and transmitted to the source and the initiating core AS is informed of the successful creation of an optimized path. This confirms the optimization process from the corresponding LD. Note that the TREE_OP_ACK messages are dropped by an AS if the AS has previously processed the TREE_OP_ACK message. This reduces the amount of signaling involved, thus resulting in a scalable solution. As shown in Figure 7.6(b), AS_1 and AS_2 have merged connections from C_2 and C_3 onto the extended path. At this point, cells are still being forwarded on the old multicast tree. Note that these cells are not routed onto the new connections at AS_1,

[1]Given the required QoS requirements are met.

AS_2, C_2 and C_3.



(a) Connections pruning (b) Tree after optimization

Figure 7.7: (a) The connections to be pruned after receiving marker cell. (b) Resulting tree after optimization.

Once a core AS has setup a new connection, it sends a PRUNE_RDY message upstream. This message tells the upstream ASs to mark the VC on which the PRUNE_RDY message arrived for tearing down after a MIG_CELL is received and forwarded. In Figure 7.7(a), C_3 sends a PRUNE_RDY message upstream. Note that this message is only sent on the link leading to the parent node. All ASs such as C_3 and C_2 that are not designated as core mark the incoming VC as PRUNE. The PRUNE_RDY message from C_3 is terminated at C_2 and that from C_1 is terminated at AS_1.

7.2.4.1 Tree Transition

Once a new connection has been setup, the source can proceed to update the multicast tree. To preserve cell ordering all cells from the old tree must be drained first before new cells arriving on the new path can be forwarded. In order for a smooth update where cell ordering is preserved, MIG_CELL is multicast on the old tree. The MIG_CELL marks the end of cells flowing on the old tree and signals core ASs to update their multicast state. Also, core ASs tear down connections marked PRUNE. Furthermore, MIG_CELL is used to determine whether buffering is required at the core ASs.

As the MIG_CELL traverses the old tree, each core AS receiving the marker cell unmarks VCs that have been previously marked NEW. As a result, cells will be forwarded on the unmarked VC(s). Cells arriving on 'new' connections before MIG_CELL are buffered. For example in Figure 7.7(a), cells may arrive at C_3 before the marker cell is processed by C_3 since the MIG_CELL has to traverse through the sub-optimal path. Once the marker cell is processed, buffered cells are forwarded ensuring preservation of cell order during the transition process.

Upon receipt of MIG_CELL, connections previously marked as PRUNE are torn down. Note that the connections are torn down after the MIG_CELL has been multicast. This enables downstream ASs to tear down any marked connections. The pruning process avoids the problem of introducing duplicate cells during the transition process. After forwarding the MIG_CELL to downstream subscribers, the core checks whether the incoming VC in which the the MIG_CELL was received is marked PRUNE. If so then a release message is sent and the VC is torn down. Referring to Figure 7.7(a), connections that are marked PRUNE and torn down by the corresponding core after processing of MIG_CELL. The resulting tree after pruning down of connections is shown in Figure 7.7(b).

In the unlikely case where the MIG_CELL arrives before the "new" connection has been setup, a prune message is sent to the parent node once the "new " connection has been setup and synchronisation of cells have been performed. Once the the parent node receives the prune request, it removes the subscriber from its multicast subscribers list. Before the old parent connection is removed, synchronisation of cells arriving from the "new" and parent connection has to be performed. This is to ensure that receivers downstream do not get duplicate cells. The synchronisation process will be explained in Section 7.2.5.2.

7.2.4.2 Updating Multicast State

Apart from tearing down of connections, the states pertaining to the multicast session are updated. At each AS, new connections marked as "new" are unmarked. Referring to Figure 7.7(a), AS_1 and AS_2 have connections marked as "new" incident on them.

Upon receiving MIG_CELL, AS_1 and AS_2 update their multicast state and clear any connection marked NEW. As a result, subsequent cells will be multicast along the new link. Consequently C_2 and C_3 may receive cells from the optimized connection before receiving MIG_CELL. In this case the cells from the optimized connection are buffered until the arrival of MIG_CELL. Note that the extended connection is reused.

During the processing of MIG_CELL, a check is necessary to determine whether the MIG_CELL arrived from the extended path or from the parent node. If the MIG_CELL is at an AS with only extended path information then a new multicast state is created (see Table 7.1). States from the extended connection are then incorporated into the newly created multicast state. Any "new" connections are also incorporated.

At an AS where multicast table exists and the MIG_CELL does not arrive from the parent nodes no update is performed. This is because updating multicast table would cause cell loss and infinite buffering. Furthermore, if the ASs have connections marked for pruning, these connection will be prematurely pruned. As a result, the MIG_CELL does not get forwarded when it arrives from the parent node since the connections leading to downstream receivers have been pruned. This results in infinite buffering at ASs downstream.

A buffer limit is set at each AS, while waiting for the arrival of MIG_CELL. This is to ensure that infinite buffering does not occur when MIG_CELL is lost. Setting a buffer limit provides for a simple solution since it can be determined based on the available resources at a given AS.

7.2.4.3 When does the MIG_CELL gets sent?

The roles played by the MIG_CELL have been mentioned but one question remains, when should the tree transition take place. The transition process can be invoked when the PR of LD_2 receives a TREE_OP_ACK message. The side effect of this is that some connections may not be setup thus causing cell loss. For example, a neighboring TREE_OP_ACK message is bound to arrive sooner than those LDs which are a few hops away. Alternatively, MCoRe could wait t amount of time. Note that the waiting time does not result in cell loss because cells are still being multicast on the old tree. The only side effect is that additional resources (connections) are held up in the network. A more complicated approach involves estimating the number of LDs. The transition takes place once the TREE_OP_ACK of all LDs have been received, incurring additional signaling overheads.

The approach taken in MCoRe is to send a MIG_CELL once the PRUNE_RDY and TREE_OP_ACK messages are received. Since a prune message is only generated upon creation of a new connection and the message has to traverse through the elongated path (up to old source's point of contact and extended path), the time taken by the prune message provides a simple way for all core ASs to complete any optimization required prior to update. While waiting for the PRUNE_RDY and TREE_OP_ACK messages, there is no need to keep track of multicast group members. For example, informing upstream parent node about the number of subscribers (or core ASs) downstream. This means that periodically signaling messages flood the source of the multicast tree. In order to make MCoRe scalable, MIG_CELL is sent when the PRUNE_RDY and TREE_OP_ACK messages are received. This method is unreliable in that for some cases both messages may arrive before "new" connections are setup properly. Note that this is not in any way detrimental to the transition process because synchronisation is done at the core AS if a MIG_CELL is received before a "new" connection is setup. Further, since a prune is only sent after synchronisation, no cell loss occurs. Also buffering is required to ensure cell ordering. It will be shown later in the results section that buffering required is minimal, therefore buffering does not compromise the scalability of MCoRe.

7.2.5 Receiver Migration

So far only source migration was considered. In this section, the problems and solutions to receiver migration are presented. The main problem of receiver migration is that of cell ordering. Assume that cells have sequence numbers incorporated, at the previous LD the MH has received cells with sequence numbers up to i. When the MH migrates to a new LD and rejoins the multicast session, in the absence of a scheme such as

MCoRe, the MH may receive cells with sequence numbers less or more than i. The problem is aggravated because ATM cells do not have sequence numbers. Therefore, ASs are unable to determine whether a receiver has received a particular cell by looking at the header unless there is a global mechanism for distinguishing cells. To overcome the aforementioned problems the source (or the PR of the source's LD) sends resource management (RM) cells every k cells. Included in each RM cell are: sequence number and k (the number cells preceding the RM cell).

7.2.5.1 Handoff

As in source migration, to avoid cell loss and preserve cell ordering during handoff, a simple path extension is made. This is illustrated in Figure 7.8. After migration, the MH sends a rejoin request to the local PR. Once receiving the rejoin request from the receiver, the new PR sets up a connection to the receiver's previous location. Once the connection has been setup, the handoff process is finished. At this point in time, the receiver is receiving cells from the extended path, therefore from the receiver's point of view it is still in multicast session. The next phase is for the PR to send a rejoin message to the multicast session specified in the rejoin request. If the PR has already joined the multicast session then no rejoin is required. In this case the synchronisation process outlined in the next section is performed.

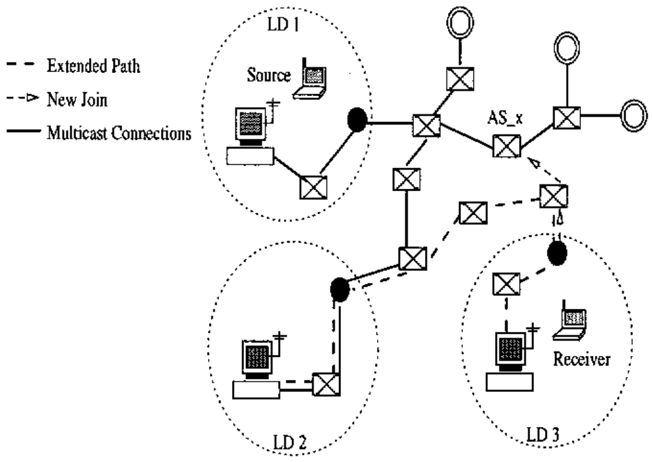


Figure 7.8: The handoff process of a mobile receiver.

The information available in the rejoin message sent by receivers are shown in Table

Data	Description
McastGrp	The multicast group to rejoin.
RM_seq	The last received RM sequence number at the previous LD.
Old_COF	The previous care-of-address.

Table 7.2: Information within the rejoin request.

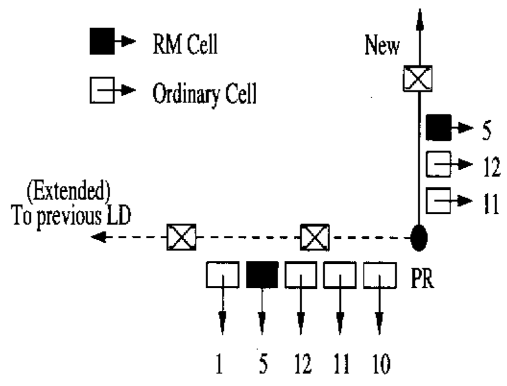


Figure 7.9: Illustration of Cell Synchronisation at PR.

7.2. Once the AS receives the rejoin message and the QoS conditions are agreed upon, a new branch is setup to the initiating PR.

7.2.5.2 Synchronisation

As a result, of rejoining the multicast session, the PR starts receiving cells from both the newly created branch and the extended connection. Upon receiving cells from the new branch, the PR prepares to tear down the extended connection. The synchronisation process must ensure that there is a seamless transition of cells from the extended and new connection. Figure 7.9 illustrates the synchronisation process performs by a PR. The PR has two connections incident on it and is receiving cells from both connections, new and extended. The RM cells from both connections are used to determine when the PR should switch connections. When a RM cell is received from a new connection all subsequent cells are buffered. Once the a matching RM cell from an extended connection is received, the PR switches connection and starts forwarding buffered packets. All subsequent cells from the extended connection are discarded and the extended connection is torn down.

In the case where the next RM cell number received from the new connection is less than that of the extended connection, cells from the extended path are forwarded until matching RM cell from the extended path is received. After that, the extended connection is torn down. Cells from the new connection are forwarded only when the RM sequence number is equal to cells from the extended path. An alternative solution may involve maintaining the extended path but this introduces the possibility of loops. The maintenance of extended path is useful in LDs with no multicast capabilities. If the receiver discovers that the LD does not support multicast then the extended path is maintained until it migrates to another LD which does. The algorithm for the synchronisation process is shown in Algorithm 7.1.

```
IF RM cell THEN {
  IF RM cell from new link THEN {
   opRM_seq       = Get_RM_sequence_number
   opCellCounter = Get_RM_cell_counter
  }
  END

  IF RM cell from Extended link THEN
   extRM_seq = Get_RM_sequence_number
  END

  IF opRM_seq != 0 AND extRM_seq != 0 THEN
   {
    IF opRM_seq > extRM_seq THEN
     set BUFFER_REQUIRED true
    ELSE IF opRM_seq < extRM_seq THEN {
     Send STOP_SENDING signal to previous BS
     set Discard_From_Ext true
    }
    ELSE {
     set RMs_EQUAL true
    }
   }
} ELSE {
  IF cell from extended path AND NOT Discard_From_Ext THEN
    extCellCounter += extCellCounter

  IF RMs_EQUAL THEN {
    IF opCellCounter > extCellCounter THEN
      set BUFFER_REQUIRED true
    ELSE IF opCellCounter < extCellCounter THEN
      Remove cell if from extended path
    ELSE
      Optimization process done
  }

  IF BUFFER_REQUIRED &&
     (opRM_seq != extRM_seq || opCellCounter != extCellCounter) THEN {
   IF cell from optimized path THEN
    buffer_cell(cell)
   ELSE
    send_to_receiver
   }
  ELSE
   Forward buffer to receiver
}
```

Algorithm 7.1: Cell Synchronisation.

Apart from the handoff procedure described above, a fast handoff is achieved by including the last RM sequence received as part of the path extension process. When a path extension is initiated, a join message containing the next cell sequence, the receiver is expecting is sent to the local PR. If the PR has the next expected cell (assuming the PR is already subscribed) then the PR forwards the cell and the path extension process is terminated. This results in conservation of resources.

As mentioned above, RM cell is used for synchronisation. The downside is that if the application decides to define k to be a large value then the synchronisation process may take considerably longer before another RM is received. To enable a faster synchronisation, AS and PR keep track of a counter call *CellCount.* This counter is incremented each time a normal cell is received and reinitialized to zero when a RM cell arrives. During the synchronisation process if both the RM cells received from the newly joined connection and extended path have the same sequence number then the *CellCount* is used. As part of the rejoin acknowledgement, the acknowledgement message contains the *CellCount* value of the AS that intercepted the rejoin request. Once the PR receives the *CellCount* value it compares with the *CellCount* from the extended path (given the RM sequence number is equal). Then the PR proceeds to cell synchronisation as in the RM case.

7.3 Simulation Methodology

To investigate the performance of MCoRe, the NIST ATM/HFC network simulator [204] was used. With the NIST ATM simulator, routes between two end-hosts have to be hard-wired, no call setup mechanisms and no mobility support are available. So the simulator was augmented with the link-state routing protocol and a simple call setup protocol was implemented. The call setup protocol does not involve any call admission algorithms. During a call setup, a VPI is allocated for the connection followed by the setup of translation table. If the translation table has already been setup, then it is updated with the connection's information shown in Table 7.3. As for mobility support, since this study is not concerned with the wireless link characteristics, handoffs are modeled by severing a particular link and restoring another. Once a particular link is up, the MH hears a beacon broadcast from the BS and assumes occurance of a handoff. Here the BS is set to transmit on the link (maybe severed initially) leading to the MH once every second.

A mesh topology (degree of four) consisting of 36 ASs was used. The links interconnecting ASs are set to 155Mbits/s with a distance of 0.5 kilometers. The link connecting the MH is set to 2Mbits/s. The delay to process a cell at switches was set to one millisecond. The source of the multicast session uses the constant bit rate (CBR) service and transmits at m Mbits/s. The value m ranges from 5 to 50 depending on the experiment involved.

Data	Description
VPI	The virtual path identifier
InPort	Incoming port
OutPort	Outgoing port
Conn-Type	Connection type (e.g., CBR)

Table 7.3: Input parameters.

Each switch was injected (pre-loaded) with the MCoRe program to facilitate recognition of signaling messages pertaining to the MCoRe protocol and also to maintain necessary states. During the join process, the injected program is activated and after processing each join message the AS is considered subscribed to the multicast session. In the simulations only one multicast session is running at any given time. Furthermore, all receivers join the multicast tree at the start of the simulations. In the simulations a total of 10 receivers were used corresponding to 10 LDs. A total of 10 simulations were run for each experiment with the MH migrating randomly. In the experiments, the data recorded were end-to-end latencies of receivers, signaling messages filtered and buffer requirements at the PR and ASs.

The performance of MCoRe is compared with that of CBT [64]. The core is manually positioned at the center of the topology with all receivers having an approximately equal distance to the core. The core remains fixed for the duration of the simulation. When the source migrates, a new connection is setup to the core and core proceeds to multicast any cells received from the new link. The old link is then torn down.

In the implementation of CBT no connection rerouting algorithm such as [60] was implemented. Since the connection is reestablished after handoff, the path from the MH to the core is thus optimal although a high disruption time to receivers is observed. A possible solution is to employ connection rerouting methods such as those investigated by Bui et al. [205]. The use of connection rerouting and the adaptation of shared tree approach to handle mobile group members is beyond the scope of this paper and will not be discussed further.

7.4 Results

7.4.1 End-to-End Latency

Figure 7.10 shows the end-to-end latencies observed by each receiver before and after handoff. A significant increase in end-to-end latencies is observed after handoff due to the path extension process. Once the path has been extended, optimization is performed and the resulting end-to-end latencies dropped significantly. For some receivers the observed end-to-end latencies increase because the MH's new location maybe further away from where the MH was previously. Therefore, as can be seen in Figure 7.10, receivers one, six and ten observed a higher end-to-end latency after optimization.

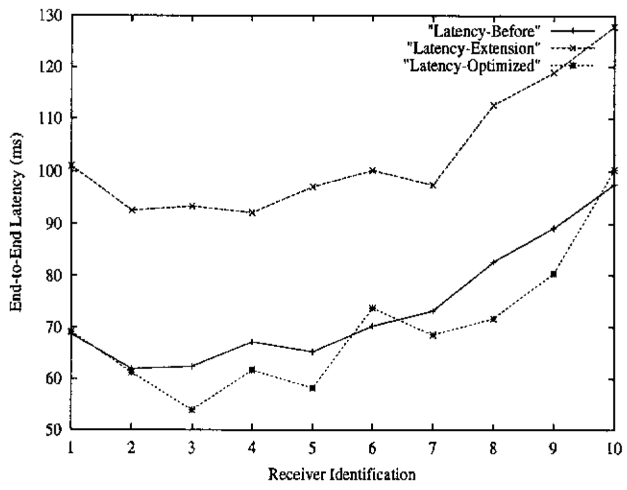


Figure 7.10: Average end-to-end latencies

7.4.2 Buffer Requirements

In this experiment buffer requirements at ASs during the optimization process are measured. Here the size of the buffer required is plotted against different transmission rates at the source.

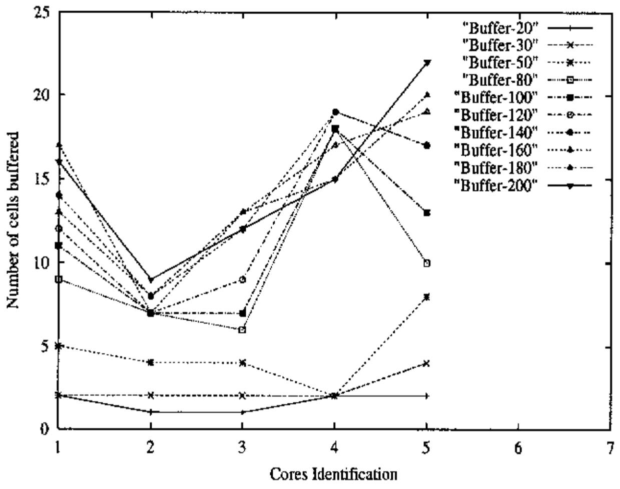
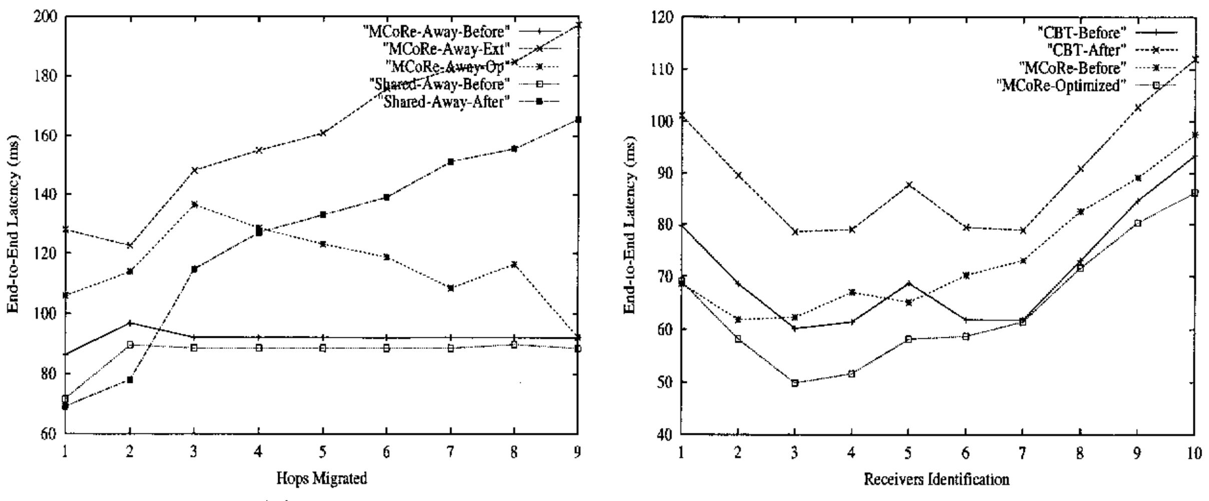


Figure 7.11: Average buffering requirements at core ASs, with source transmitting at 10Mbits/s

Figure 7.11 shows the number of cells buffered at each core given varying transmission rate. The buffering required is dependent on two factors: rate and the delay from a given core to the MH's current location. The delay from the core, the MH's current location determines the delay of the MIG_CELL. As a result, this delay determines the number of cells buffered. If the source transmits at a higher rate, the number of cells buffered will be higher since the arrival rate of cells from optimal link will be higher. In the simulation environment, the average number of cells buffered at core ASs range from 2 to 23 when the source transmits at 10Mbits/s.

7.4.3 MCoRe vs. Shared-Tree

Here a comparison between CBT and MCoRe is presented. Two experiments were performed. In the first, the MH is set to migrate increasingly away from its previous LD. The second experiment looks at the end-to-end latency in general for both MCoRe and CBT on varying topologies with receivers attached randomly.



(a) (b)

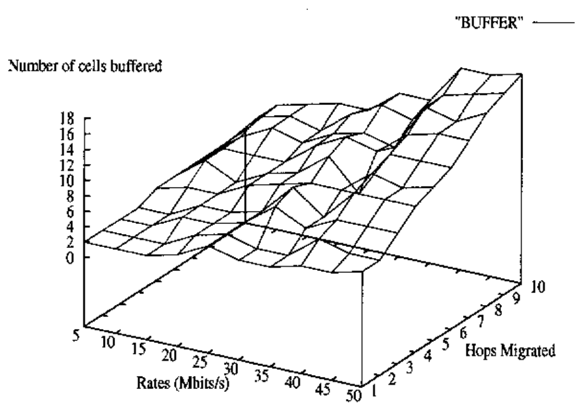
Figure 7.12: (a) End-to-end latency vs. Migration Distance. (b) Average end-to-end latency comparison between shared-tree and MCoRe.

As shown in Figure 7.12(a) as the MH migrates further away, the end-to-end latency increases with each migration. After migration it can be seen that the end-to-end latency dropped significantly. For CBT the end-to-end latency increases in a similar manner to that of MCoRe. Given that the core in CBT does not change after each migration, receivers are unable to improve on the observed end-to-end latency. This means that if CBT is used, the source has to stay within a certain range for the delay to be acceptable. In MCoRe the end-to-end latencies only persist, while the MH is receiving cells from the extended path. After optimization, the end-to-end latencies dropped dramatically as shown in Figure 7.12(a).

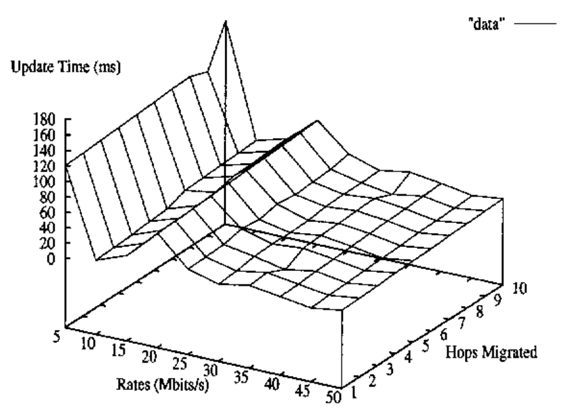
The comparison of the end-to-end latencies of CBT and MCoRe is shown in Figure 7.12(b). On average the end-to-end latencies observed by MCoRe are lower than those of CBT due to the sub-optimal path between the source and receivers in CBT. Note that the use of connection rerouting algorithms only reduces the disruption time due to handoff and does not improve end-to-end latencies. This is because the position of the core is the determining factor in the perceived end-to-end latencies by receivers.

7.4.4 Receiver's Migration

The performance under receiver's migration is dictated by the rate at which the multicast source transmits data and also how far the MH has migrated from its previous position. Other than that, the bandwidth available at the LD in which the MH migrated to affects the disruption time.



(a)



(b)

Figure 7.13: (a) Buffer requirements given different rates (b) Tree update time given different rates

As shown in Figure 7.13(a), with increasing rate the buffering requirement when the MH is 10 hops away is at a maximum (14 cells). This is due to the delay of the extended path where the PR has to buffer all incoming cells from the newly joined connections. The fact that MH migration is usually local only holds true when the migration is within a LD. During inter-LD migration, the extended path maybe traversed through a number of ASs. On the other hand, for local migration, it is likely that the extended path is relatively short. Therefore, at hops ranging from one to four the buffering requirements range from zero to six.

Apart from that, an investigation concerning latency of the synchronization process given increasing transmission rates. From Figure 7.13(b) it is shown that the update time is at its maximum at 5 Mbits/s. The update time increases when the MH migrates away from its previous LD. From 10Mbits/s onwards the synchronization time is relatively stable and is only affected by the length of the extended path.

7.4.5 Scalability

In this section the role played by ASs in reducing the number of signaling messages involved is investigated. Various experiments were run with increasing multicast group size (5-50) and recorded the number of TREE_UPDATE messages filtered during the tree optimization process. Further, the number of hops traversed by each message is recorded to determine how far a message has to traverse before being filtered. This means that if a message can be filtered as soon as possible then the traffic induced by signaling is reduced. As can be seen from Figure 7.14(b), when the group size is at 45 or 50, the signaling messages are removed once they arrive at the first switch.

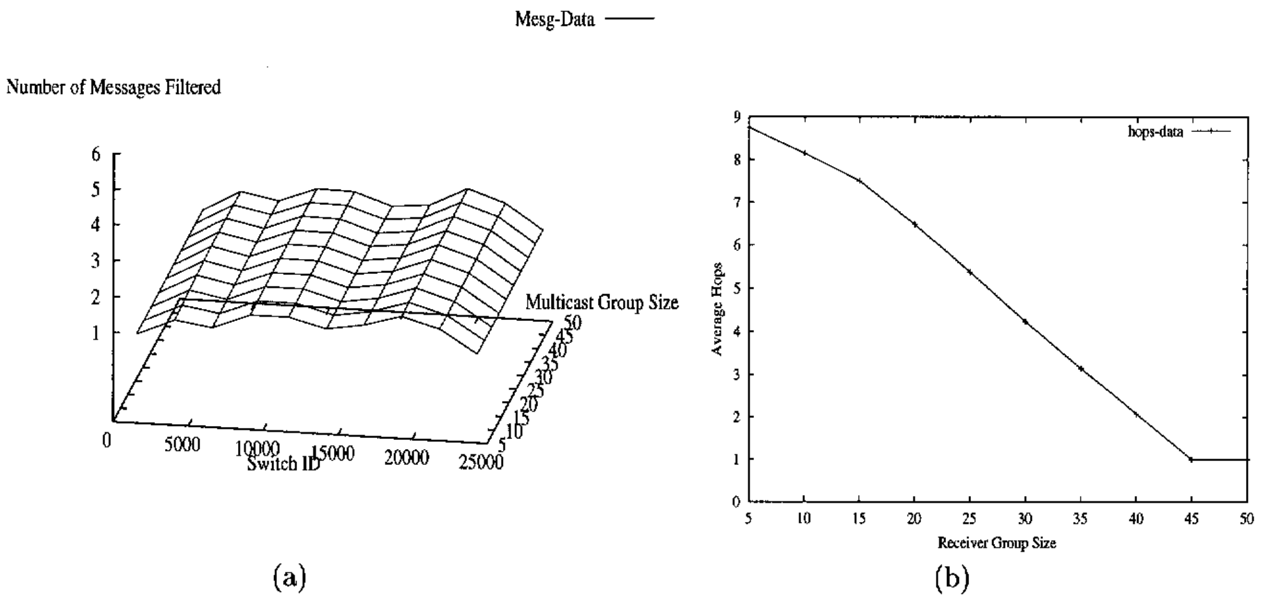


Figure 7.14: (a) Number of messages filtered. (b) Average Hops traversed by signaling messages.

Figure 7.14(a) shows the number of messages filtered as the multicast group size increases. Here, the number of messages filtered by core ASs is considered. For example the AS with an identification of 5000 has filtered five messages when multicast group is of size 50. Combining with other ASs (e.g., 5 ASs) assuming all of them have filtered out five messages, the total number of messages filtered is 25. Note that as the multicast group size increases the number of core ASs increases as well. Therefore, more signaling messages are likely to be filtered. This is shown in both Figures 7.14 (a) and

(b) where, as the multicast group size increases, the number of messages filtered increases. In Figure 7.14(b) the average number of hops traversed by signaling messages is inversely proportional to multicast group size. As a result, the signaling messages generated during handoff or optimization can be reduced dramatically.

7.5 Discussion

In the following paragraphs outline a list of areas that require further investigation.

- Multiple sources: As mentioned previously, currently MCoRe only supports one source. Given the data structure shown in Table 7.1 it is possible to have more than one source. The MCoRe program within ASs can be modified to multicast cells received from a given connection to all outgoing connections.
- Security: Currently MCoRe does not have security considerations incorporated and it remains to be investigated further. One of the main reasons for not dwelling on security at this stage is partly due to the immature nature of AN security technologies. Interested readers in AN securities are referred to [165]. The main concern here is on rerouting multicast connections, the issues concerning securities are beyond the scope of the work presented here.
- Adaptive Shared-Tree: Shared tree approach provides a simple solution to handling mobility but modifications are required to handle mobility. For example the core needs to be mobility aware. This means the core must contain a protocol in which it is able to perform handoff from an old connection leading to the mobile source's previous location onto the new connection to the mobile source's current location and vice-versa for receiver handoff. Furthermore, the core has to distinguish whether a new connection belongs to a source that just migrated or a new source. The handoff process can be improved if connection rerouting algorithms are incorporated. The incorporation of connection rerouting becomes more difficult when rerouting connections leading to receiver. Due to the dynamic nature of multicast group members, after each source/receiver migration the core's position may cause some receiver's QoS to be violated. To overcome this

problem we plan to look at the possibilities of applying active based solution in which a new core is computed given predetermined constraints.

- QoS: The issue of maintaining QoS after each MH migration requires further investigation. Some of the issues involve incorporating MH's QoS into the rerouting process and QoS adaptation. Current work in connection rerouting, both unicast and multicast, has not incorporated QoS policies.

7.6 Conclusion

In this chapter, the problems of rerouting multicast connections using an AN-based solution have been addressed. MCoRe utilises the computation at switches, resulting in a simple, efficient and scalable solution. Moreover MCoRe enables connections in a SRT to be adapted efficiently due to source/receiver mobility. MCoRe preserves cell ordering, prevents cell duplication and has minimal cell loss. Results have shown that MCoRe requires minimal buffering. Also the number of signaling messages is kept low with increasing multicast group size with the help of ASs. Moreover, compared to shared-tree approach MCoRe has lower end-to-end latencies and adapts to source migration. This means the multicast tree is always rooted at the source's current location whereas in shared tree the core is chosen during the initial setup of the multicast session and choosing another core after each host migration would prove to be an expensive process.

As can be seen programmability plays an important role in this chapter. These benefits outlined in this chapter were mainly realised using the programmability of switches. The most important factor that allows the rerouting of connections efficiently is the ability to distinguish between old and new connections. By differentiating between these two types connections, cells can be prevented from being forwarded onto new connections during optimization process and also allow tearing down of old connections. Another advantage of MCoRe is that all the operations performed do not require a snapshot of the multicast tree. This has added advantage since less signaling is required. Another benefit is the computation performed at each switch, which determines which connections remain intact after handoff.

Chapter 8

Conclusions

This thesis investigated the viability of applying active networking technologies to routing in mobile communications. Contributions to the mobile communication and ANs research areas have been outlined in detail. Various benefits of ANs in unicast and multicast routing for IP and ATM networks were demonstrated through extensive simulation studies. AN-based protocols presented in this thesis manifest in the following benefits: (i) efficient adaptation to mobility, (ii) reduced signaling overheads, (iii) high reuse of allocated network states, (iv) extensibility, (v) topology independence, and (vi) scalability.

As can be seen from the AN-based protocols presented in Chapters 4 to 7, updates are only performed on parts of the connection or multicast tree. Hence, ANs promote iterative update and reuse of allocated states. This was clearly demonstrated in both connection-oriented and multicast communications. In Chapter 5, it was shown that a connection in mobile ATM networks can be iteratively updated to maximise reuse. Similarly in Chapters 6 and 7, multicast states at routers (switches) were preserved and only parts of the multicast tree were affected. The ability to update portion of tree or connection is enabled by the computation at each router (switch) since routers (switches) can make their decisions based on the available information residing at their site and from neighboring routers (switches), they can determine locally whether updates are necessary under current situations. Therefore, routers (switches), in other words the intermediate network elements make the decisions rather than the end-host(s). The benefits of such intra-network computing capabilities in mobile environments have been demonstrated throughout this thesis.

It was shown that signaling message overheads are minimal compared to solutions deployed from end-hosts. This is due to the fact that computation is distributed among routers (switches). The main source of signaling overheads, that is, probing of routers (switches) for data to be processed at end-hosts is not required. For example, there is no need to obtain a snapshot of the multicast tree before any adaptation can be performed. Local computation can be performed *when events happen* and not *after*. This has obvious advantages in the area of congestion control where it is important to isolate the congestion and also notify end-hosts of the congestion state. In mobile networks, especially routing, routes need to be updated frequently, even more so in multicast communications, particularly when the host migrates. Therefore, it is crucial these updates are performed as soon as possible. The fast update process has been demonstrated in Chapter 4 where TCP's performance was shown to improve significantly with the faster arrival of binding updates. In the near future, the case for local computation is even more prominent where QoS issues are to be addressed. This is because we may want to preserve the allocated QoS during connection start-up, since re-determining QoS may not be feasible; for example solutions that reallocate QoS after each migration are not useful when there are a large number of subscribers of a multicast session.

In both AMTree and MCoRe we see that active routers (switches) take an active role in filtering out redundant signaling messages. As can be seen, unnecessary optimisation and join messages are removed by core nodes, thus conserving bandwidth. Note that core nodes are not limited to filtering service but can be extended to include other services such as providing different QoS to different parts of the multicast tree. The ability to filter out redundant messages is crucial in ensuring the scalability of the protocol as the number of subscribers increases.

Another advantage in using ANs can be seen in unicast and multicast connection rerouting of connection-oriented networks. The main reason is that programs can distinguish between 'old' and 'new' connections. In Chapter 7, switches distinguish between the new and old tree, thus preventing forwarding of cells on new connections which may cause looping or duplicate cells. The process of transition from the old to new was made efficiently without incurring high overheads due to the use of ANs. Note that the main signaling involved was due to connection setup and no other signaling is used to maintain the active programs instantiated. In Chapter 6, active routers help to determine the closest core. The routers subscribed are removed or added dynamically when they receive hop updates. This is an improvement over schemes which use a *ping* like mechanism to determine which subscribed router(s) has the minimum round-trip time.

The way in which active programs are positioned within the network has not received much attention in active networking communities. The deployment model presented in this thesis only programs routers (switches) that are on the communicating path. As a result, strategic positioning of programs is not required. Moreover, these programs can be unloaded easily after each migration. Since one of the many benefits of active solutions is topology independence, strategically positioning active programs would defeat the purpose of having ANs. As can be seen from the solutions presented in this research, frequent host migrations render strategic positioning services infeasible. Deployment of active programs alongside signaling messages, on the communication path facilitates identification of *strategic points*. For instance, in AMTree, the cores in the multicast tree are dynamic and in essence become *strategic points* based on group membership. Moreover, adaptive programs can be installed at these cores to cater to varying requirements of each downstream subscriber. Dynamic *strategic points* are crucial in mobile networks due to host movement. As shown in Chapters 6 and 7, the cores in the multicast tree remain dynamic. They can be unloaded or loaded based on group membership changes. As the cores in the AN-based solutions are dynamic, customisation programs are invoked when and where they are needed. In this thesis, the programs were deployed alongside the operation of the protocol, for example Mobile IP [10]. The goal of deploying programs alongside signaling messages used by a given protocol is that only routers (switches) that are involved with the session are programmed.

In order to have centralised source control, the active programs in this thesis are deployed from a single source which allows different domains to deploy their own unique protocols. The implications of the deployment model used in this thesis are that services can be downloaded from a known third party server, followed by any customisation by the deploying host (e.g., HA). This means that third party active program developers do not need to scatter their programs strategically within the network. As can be seen throughout the thesis, no external mechanisms are used to discover other active services corresponding to the communication session. This is in contrast to the proposal by Sivakumar et al. [206] where a specific infrastructure was introduced to track active services so that they may cooperate with each other.

Topology independence is another important feature of using ANs. Some schemes employed tree topology or dedicated LANs such as cellular IP [37] to reduce signaling messages and shield the correspondent host from the wireless/mobile characteristics. It was shown that AN-based solutions do not rely on network topology and can be dynamically customised to form a specific topology. This is evident from the solutions presented in Chapters 4 to 7. No assumptions were made regarding network topology. Hence the proposed AN solutions are generic in that they work across different topologies.

It can be argued that the solutions presented in previous chapters can be hard-coded into current routers thus facilitating immediate deployment of the proposed protocols. This certainly provides an immediate solution to the problems faced by protocols today. However, in the long run, ANs based solutions can be easily extended to address other issues such as QoS, security, network management, and others. We have also demonstrated how existing protocols can be upgraded or new protocols can be built to handle mobility.

To summarise, this thesis has investigated a *niche* area in mobile computing, that of employing AN technology in routing and multicasting in connection-less as well as connection-oriented networks. The exhaustive studies have demonstrated that ANs can be successfully employed in mobile computing environments.

8.1 Future Work

8.1.1 Large Scale High Mobility Multicast

An important area of future research work is to show the benefits of AMTree and MCoRe in large scale multicast groups with varying mobility of group members. In the near future it is likely that more users will be connected to the Internet through their hand phones. These users may be subscribed to multiple multicast groups, for example receiving video broadcast from multiple news stations. Therefore, it is interesting to compare AMTree and MCoRe with the performance of existing multicast protocols in such environments.

8.1.2 Security and Mobile Protocols

Security features in ANs are maturing and it would be interesting to incorporate AN security features with general security ones into mobile protocols; for example, the incorporation of SANE [165] into the operation of Mobile IP [10]. Current work that employs ANs to address security issues includes IP spoofing [138] and active firewalls [139]. Further work is required to investigate the benefits that can be obtained if the security features inherent in ANs are used to complement those of end-hosts. Since the security requirements of ANs are much more demanding, it is likely that security provided by ANs would prove to be beneficial.

8.1.3 QoS

QoS is another issue that remains to be investigated. As mentioned above, ANs provide an ideal solution for protocols that maintain states within the network. Therefore, ANs provide an ideal solution with regard to QoS because routers/switches can negotiate among themselves to maintain the allocated QoS or invoke an adaptive scheme during host migration. Furthermore, little work has investigated the application of ANs to QoS problems. In the area of QoS in connection-oriented networks, ANs can be employed to provide services.

8.1.4 Mobile Ad-Hoc Network (MANET)

AMTree and MCoRe in their current form cannot be applied directly in MANET. It will be interesting to see how AMTRee and MCoRe can be adapted into MANET. The problem itself is not straight forward given that in MANET, the cores are dynamic. In other words, adaptation within the network as well as that due to subscriber migration is required. This means that in-transit network elements (i.e., mobile devices) need to take an active role in maintaining the multicast session, which is similar to active routers/switches in the fixed network. However, due to limited resources available at these mobile devices, computation will be limited. This correlation between the roles of intermediate nodes in both ANs and MANET means that there is a high possibility that the protocols developed for the fixed network may apply directly to MANET with little modification. The reverse is also true for protocols developed for MANET.

Bibliography

[1] N. R. Sollenberger, N. Seshadri, and R. Cox, "The evolution of IS-136 TDMA for third-generation wireless services," *IEEE Personal Communications*, pp. 8–17, June 1999.

[2] WAP Forum, "Wap archictecture." http://www.wapforum.com, 2000.

[3] BlueTooth, "Bluetooth protocol architecture." http://www.bluetooth.com, 2000.

[4] M. Weiser, "The computer of the 21st century," *Scientific American*, pp. 94–100, Sept. 1991.

[5] M. Weiser, "Some computer science issues in ubiquitous computing," *Communications of the ACM*, July 1993.

[6] M. J. Riezenman, "Communications," *IEEE Spectrum*, vol. 37, pp. 33–44, Jan. 2000.

[7] R. Steele, J. Whitehead, and W. C. Wong, "System aspects of cellular radio," *IEEE Communication Magazine*, vol. 33, pp. 80–86, Jan. 1995.

[8] M. Stemm and R. H. Katz, "Vertical handoff in wireless overlay networks," in *ACM Mobile Networking (MONET), Special Issue on Mobile Networking in the Internet*, 1997.

[9] D. L. Tennenhouse and D. J. Wetherall, "Towards an active network architecture," *Computer Communication Review*, vol. 26, pp. 5–18, Apr. 1996.

[10] C. Perkins, "RFC2002: IP mobility support," Oct. 1996.

[11] C. Perkins, "IETF draft: Route optimization in mobile IP," Dec. 1997.

[12] D. B. Johnson, "Scalable support for transparent mobile host internetworking," in *Mobile Computing*, ch. 3, pp. 103–128, Kluwer Academic Publishers, 1996.

[13] P. Bhagwat, C. Perkins, and S. Tripathi, "Network layer mobility: An architecture and survey," *IEEE Personal Communications*, vol. 3, pp. 54–64, June 1996.

[14] C. E. Perkins and P. Bhagwat, "A mobile networking system based on internet protocol," *IEEE Personal Communications*, vol. 1, no. 1, pp. 32–41, 1994.

[15] D. B. Johnson, "Scalable and robust internetwork routing for mobile hosts," in *Proceedings of the 14th International Conference on Distributed Computing Systems*, IEEE Computer Society, June 1994.

[16] F. Teraoka and M. Tokoro, "Host migration transparency in IP networks: The VIP approach," *Computer Communications Review*, vol. 23, Jan. 1993.

[17] A. Myles and D. Skellern, "Comparison of mobile host protocols for IP," *Journal of Internetworking Research and Experience*, vol. 4, pp. 175–194, Dec. 1993.

[18] A. Myles, D. B. Johnson, and C. Perkins, "A mobile host protocol supporting route optimization and authentication," *IEEE Journal on Selected Areas in Communications, special issue on Mobile and Wireless Computing Networks*, vol. 13, pp. 839–849, June 1995.

[19] C. Perkins, A. Myles, and D. B. Johnson, "IMHP:a mobile host protocol for the internet," *Computer Networks and ISDN Systems: Special Issue on Selected Papers of the Annual Conference of the Internet Society/5th Joint European Networking Conference*, vol. 27, pp. 479–491, Dec. 1994.

[20] C. E. Perkins and D. B. Johnson, "Mobility support in IPv6," in *Proceedings of the Second Annual International Conference on Mobile Computing and Networking (MobiCom'96)*, (New York, USA), ACM, Nov. 1996.

[21] D. B. Johnson, "Ubiquitous mobile host internetworking," in *Proceedings of the Fourth Workshop on Workstation Operating Systems*, IEEE Computer Society, Oct. 1993.

[22] J. Ioannidis, D. Duchamp, and G. Q. M. Jr, "IP-Based protocols for mobile internetworking," in *ACM SIGCOMM '91*, pp. 235–245, Sept. 1991.

[23] J. Ioannidis and G. Q. M. Jr., "The design and implementation of a mobile internetworking architecture," in *Proceedings 1993 Winter USENIX*, (San Diego), pp. 491–502, Jan. 1993.

[24] R. Yuan, "An adaptive routing schemes for wireless mobile computing," in *Wireless and Mobile Communications*, pp. 39–50, Kluwer Academic Publishers, 1994.

[25] M. G. Baker, X. Zhao, S. Cheshire, and J. Stone, "Supporting mobility in MosquitoNet," in *Proceedings of the 1996 USENIX Technical Conference*, (San Diego, CA), Jan. 1996. http://mosquitonet.Stanford.EDU/mosquitonet.html.

[26] B. R. Badrinath, A. Bakre, T. Imielinski, and R. Marantz, "Handling mobile clients: A case for indirect interaction," in *Fourth Workshop on Workstation Operating Systems*, Oct. 1993.

[27] H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *ACM Wireless Networks*, vol. 1, pp. 469–482, Dec. 1995.

[28] K. Okanoue and T. Lhsawa, "IP mobility support with ip-squared encapsulation," *IEICE Transactions on Communications*, vol. e80-b, p. 11981207, Aug. 1997.

[29] J. Mysore and V. Bharghavan, "A new multicasting-based architecture for internet host mobility," in *Proceedings of ACM Mobicom*, (Budapest, Hungary), 1997.

[30] J. P. Mysore and V. Bharghavan, "Performance of transport protocols over a multicasting-based architecture for internet host mobility," in *ICC'98*, 1998.

[31] Y. Bejerano and I. Cidon, "An anchor chain scheme for IP mobility management," in *INFOCOM'2000*, (Israel), 2000.

[32] C. Perkins, "RFC 2003: IP encapsulation within IP," Oct. 1996.

[33] C. E. Perkins and K.-Y. Wang, "Optimized smooth handoffs in mobile IP," in *4th IEEE Symposium on Computers and Communications*, (Egypt), 1999.

[34] C. Perkins, "Mobile-IP local registration with hierarchical foreign agents." IETF-Draft, Feb. 1996. draft-perkins-mobileip-hierfa-00.txt.

[35] E. Gustafsson, A. Jonsson, and C. E. Perkins, "Mobile IP regional tunnel management." INTERNET DRAFT draft-ietf-mobileip-reg-tunnel-01.txt, Aug. 1999.

[36] K. E. Malki, N. A. Fikouras, and S. R. Cvetkovic, "Fast handoff method for real-time traffic over scalable mobile IP networks." INTERNET DRAFT, draft-elmalki-mobileip-fast-handoffs, June 1999.

[37] A. Valko, A. Campbell, and J. Gomez, "Cellular IP." draft-valko-cellulariip-00.txt, Nov. 1998.

[38] R. Ramjee, T. L. Porta, S. Thuel, K. Varadhan, and L. Salgarelli, "IP micro-mobility support using HAWAII." INTERENT-DRAFT draft-ietf-mobileip-hawaii-00.txt, June 1999.

[39] C. L. Tan, S. Pink, and K. M. Lye, "A fast handoff scheme for wireless networks," in *Proceedings of 2nd ACM International Workshop on Wireless, Mobile Multimedia*, (Seattle, Washington), Aug. 1999.

[40] J. Naylon, D. Gilmurray, J. Porter, and A. Hopper, "Low-latency handover in a wireless ATM LAN," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 909–921, Aug. 1998.

[41] A. Acharya, S. Biswas, L. French, J. Li, and D. Raychaudhuri, "Handoff and location management in mobile ATM networks," in *3rd International Workshop on Mobile Multimedia Communications (MoMuC-3)*, (Princeton, New Jersey), Sept. 1996.

[42] K. Keeton, B. A. Mah, S. Seshan, R. H. Katz, and D. Ferrari, "Providing connection-oriented network services to mobile hosts," in *Proceedings of the USENIX Symposium on Mobile and Location-Independent Computing*, (Massachusetts), Aug. 1993.

[43] B. A. Akyol and D. C. Cox, "Rerouting for handoff in a wireless ATM network," *IEEE Personal Communications*, vol. 3, pp. 26–33, Oct. 1996.

[44] R. Yuan, S. K. Biswas, L. J. French, J. Li, and D. Raychaudhuri, "A signaling and control architecture for mobility support in wireless ATM networks," *ACM/Baltzer Mobile Networks and Applicatioins*, vol. 1, Dec. 1996.

[45] B. A. Akyol and D. C. Cox, "Rerouting for handoff in a wireless ATM network," *IEEE Personal Communications*, vol. 3, no. 5, pp. 26–33, 1996.

[46] Y. Bejerano, I. Cidon, and J. Naor, "Efficient handoff rerouting algorithms: A competitive on-line algorithmic approach," in *INFOCOM'2000*, (Israel), 2000.

[47] C.-K. Toh, "Performance evaluation of crossover switch discovery algorithms for wireless ATM lans," in *IEEE Infocom'96, International Conference on Computer Communications*, (San Francisco), Mar. 1996.

[48] C.-K. Toh, "The design and implementation of a hybrid handover protocol for multi-media wireless lans," in *ACM First International Conference on Mobile Computing and Networking (Mobicom'95)*, (Berkeley, California), 1995.

[49] A. Acharya, J. Li, B. Rajagopalan, and D. Raychaudhuri, "Mobility management in wireless ATM networks," in *IEEE Infocom 98*, (San Francisco, CA. USA), Apr. 1998.

[50] A. S. Acampora and M. Naghshineh, "An architecture and methodology for mobile executed handoff in cellular ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. 12, pp. 1365–1375, Oct. 1994.

[51] R. Ghai and S. Singh, "An architecture and communication protocol for picocellular network," *IEEE Personal Communications Magazine*, vol. 1, no. 3, pp. 36–46, 1994.

[52] J. Li and R. Yuan, "Handoff control in wireless ATM networks: An experimental study," in *ICUPC'96 5th International Conference on Universal Personal Communicatioins*, 1996.

[53] M. Cheng, S. Rajagopalan, L. F. Chang, G. P. Pollini, and M. Barton, "PCS mobility support over fixed ATM networks," *IEEE Communications Magazine*, vol. 35, pp. 82–92, Nov. 1997.

[54] B. A. Akyol and D. C. Cox, "Rerouting for handoff in a wireless ATM network," in *ICUPC'96*, (CAmbridge, MA), Sept. 1996.

[55] P. P. Mishra and M. B. Srivastava, "Call establishment and rerouting in mobile computing networks," tech. rep., AT&T Bell Laboratories, Sept. 1994. Technical Memorandum 11384-940906-13TM.

[56] R. Ramjee, T. L. Porta, J. Kurose, and D. Towsley, "Performance evaluation of connection rerouting schemes for ATM-based wireless networks," June 1997.

[57] G. Dommety, M. Veeraraghavan, and M. Singhal, "Route optimization in mobile ATM networks," in *MOBICOM'97*, (Budapest, Hungary), 1997.

[58] ATM Forum, "Private network-network interface specification version 1.0," Mar. 1996. af-pnni-0055.000.

[59] B. A. J. Banh, G. J. Anido, and E. Dutkiewicz, "Handover re-routing schemes for connection oriented services in mobile ATM networks," in *INFOCOM'98*, (San Francisco, USA), 1998.

[60] P. Mishra and M. Srivastava, "Effect of connection rerouting on application performance in mobile networks," *IEEE Transactions on Computers*, vol. 47, p-p. 371–390, Apr. 1998.

[61] W. S. V. Wong, H. C. B. Chan, and V. C. M. Leung, "Performance evaluations of path optimizations schemes for inter-switch handoffs in wireless ATM networks.," in *The fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom'98)*, (Dallas, Texas), 1998.

[62] D. Waitzman, C. Patridge, and S. Deering, "Distance vector multicast routing." RFC1075, Nov. 1988.

[63] J. Moy, "Extension to OSPF." Internet Draft 1584, 1994.

[64] A. Ballardie, P. Francis, and J. Crowcroft, "Core based trees (CBT): An architecture for scalable inter-domain multicast routing," in *SIGCOMM'93*, (San Francisco), pp. 85–95, 1993.

[65] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei, "An architecture for wide-area multicast routing," in *SIGCOMM'94*, pp. 126–135, Aug. 1994.

[66] S. Deering, "RFC 1112: Host extension for IP multicasting," Aug. 1989.

[67] V. Chikarmane, C. L. Williamson, R. B. Bunt, and W. L. Mackrell, "Multicast support for mobile hosts using mobile IP: Design issues and proposed architecture.," *ACM/Baltzer Mobile Network and Applications*, vol. 3, no. 4, pp. 365–379, 1999.

[68] A. Acharya, A. Bakre, and B. R. Badrinath, "IP multicast extensions for mobile internetworking," in *INFOCOM'96*, (San Francisco, USA.), 1996.

[69] A. Acharya and B. R. Badrinath, "A framework for delivering multicast messages in networks with mobile hosts." ACM/Baltzer Journal of Wireless Networks, 1997. To Appear.

[70] V. Chikarmane, R. Bunt, and C. Williamson, "Mobile IP-based multicast as a service for mobile hosts," in *Proceedings of the 2nd IEEE International Conference on Services in Distributed and Networks Environments*, 1995.

[71] Y. Wang and W. Chen, "Supporting IP multicast for mobile hosts." Submitted for Journal Review, 1999. http://www.seas.smu.edu/ wchen/mobile.html.

[72] T. G. Harrison, C. L. Williamson, W. L. Mackrell, and R. B. Bunt, "Mobile multicast (MoM) protocol: Multicast support for mobile hosts," in *ACM International Conference on Mobile Computing and Networking (Mobicom)*, Sept. 1997.

[73] C. R. Lin and K.-M. Wang, "Mobile multicast support in IP networks," in *INFOCOM'2000*, 2000.

[74] T.-E. Kim and V. Bharghavan, "A multicast routing algorithm for mobile computing environments," in *Proceedings of IEEE Wireless Communications and Networking Conference*, Sept. 1999.

[75] ATM Forum, "ATM user-network interface (UNI) signaling specification, version 3.0," Sept. 1993. Englewood Cliffs, NJ: Prentice Hall.

[76] S. Komandur and D. Mosse, "SPAM: A data forwarding model for multipoint-to-multipoint connection support in ATM networks," in *6th International Conference on Computer and Communications Networks (ICCCN'97)*, (Las Vegas, USA), 1997.

[77] S. Komandur and D. Mosse, "Performance comparison of CRAM, SEAM and SPAM multipoint VC schemes for ATM networks," in *7th International Conference on Computer and Communication Networks (ICCCN'98)*, (Louisiana, USA), 1998.

[78] G. J. Armitage, "IP multicasting over ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. 15, pp. 445–457, Apr. 1997.

[79] R. R. Talpade, G. J. Armitage, and M. H. Ammar, "Experience with architectures for supporting IP multicast over ATM," in *Procs. IEEE ATM'96 Workshop*, 1996.

[80] R. R. Talpade and M. H. Ammar, "Multicast server architectures for supporting IP multicast over ATM," in *Proceedings IEEE ATM'96 Workshop*, 1996.

[81] M. Grossglauser and K. K. Ramakrishnan, "SEAM:scalable and efficient ATM multicast," in *Proceedings of INFOCOM'97*, (Kobe, Japan), Apr. 1997.

[82] ATM Forum, "ATM user-network interface (UNI) signaling specification, version 4.0." af-sig-0061.0001, July 1996.

[83] T. Tedijanto, S. Komandur, R. Cherukuri, and A. G. Malis, "Support for LIJ in PNNI." ATM Forum Contribution (atm96-1401R3), 1997. Montreal, Canada.

[84] C.-K. Toh, "A unifying methodology for handovers of heterogenous connections in wireless ATM networks," *ACM Computer Communications Review*, vol. 40, Jan. 1997.

[85] P. Manzoni and D. Ghosal, "Impact of mobility on TCP/IP: An integrated performance study," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 858–867, June 1995.

[86] S. Seshan, H. Balakrishnan, and R. H. Katz, "Handoffs in cellular wireless networks: The Daedalus implementation and experience," *Kluwer International Journal on Wireless Communication Systems*, 1996. To Appear.

[87] R. Caceres and L. Iftode, "Improving the performance of reliable transport protocols in mobile computing environments," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 850–857, June 1995.

[88] W. R. Stevens, *TCP/IP Ilustrated, Volume 1.* Addison-Wesley, 1994.

[89] V. Jacobson, "Congestion avoidance and control," *Computer Communication Review*, vol. 18, no. 4, pp. 314–329, 1988.

[90] A. Fladenmuller and R. D. Silva, "The effect of mobile IP handoffs on the performance of TCP," *Mobile Networks and Applications*, vol. 4, pp. 131–135, May 1999.

[91] B. S. Bakshi, P. Krishna, N. H. Vaidya, and D. K. Pradhan, "Improving performance of TCP over wireless networks," in *17th International Conference on Distributed Computing Systems.*, (Baltimore), May 1997.

[92] N. K. G. Samaraweera and G. Fairhurst, "Reinforcement of TCP error recovery for wireless communication," *ACM SIGCOMM Computer Communication Review*, vol. 28, no. 2, pp. 30–39, 1998.

[93] A. Bakre and B. R. Badrinath, "Handoff and system support for indirect TCP/IP," in *2nd USENIX Symposium on Mobile and Location-Independent Computing*, Apr. 1994.

[94] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP performance over wireless networks," in *Proceedings of the first Annual International Conference on Mobile Computing and Networking (Mobicom '95)*, pp. 2–11, Nov. 1995.

[95] K. Brown, R. Gopalakrishnan, and S. Singh, "Transport state handoff in mobile cellular networks," in *Proceedings of the 6th International Conference on Computer Communications and Networks*, 1997.

[96] K.-Y. Wang and S. K. Tripathi, "Mobile-end transport protocol: An alternative to tcp/ip over wireless links," in *IEEE INFOCOM'98*, vol. 3, p. 1046, 98.

[97] K. Brown and S. Singh, "M-TCP: TCP for mobile cellular networks," *ACM SIGCOMM Computer Communications Review*, vol. 27, no. 5, 1997.

[98] B. Maruthi, A. K. Somani, and M. Azizoglu, "Interference robust TCP," in *Proceedings of the 29th Annual IEEE Fault Tolerant Computing Symposium*, (Madison, Wisconsin), 1999.

[99] S. Keshav and S. P. Morgan, "SMART retransmission: Performance with overload and random losses," in *Proceedings Infocom'97*, (Kobe, Japan), Apr. 1997.

[100] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transaction on Networking*, Dec. 1997.

[101] B. S. Bakshi, P. Krishna, D. K. Pradhan, and N. H. Vaidya, "Providing seamless communication in mobile wireless networks," in *Proceedings of the 21st Local Computer Networks Conference*, Oct. 1996.

[102] B. S. Bakshi, P. Krishna, K. K. Pradhan, and N. H. Vaidya, "Providing seamless communication in mobile wireless networks," in *Proceedings of the 21st Local Computer Networks Conference*, Oct. 1996.

[103] T. Goff, J. Moronski, D. S. Phatak, and V. Gupta, "Freeze-TCP: A true end-to-end enhancement mechanism for mobile environments," in *INFOCOM*, (Israel), 2000.

[104] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, "A survey of active network research," *IEEE Communications Magazine*, vol. 35, pp. 80–86, Jan. 1997.

[105] D. L. Tennenhouse, S. J. Garland, L. Shrira, and M. Kaashoek, "From internet to activenet," 1996. http://www.tns.lcs.mit.edu/publications/rfc96/rfc96.html.

[106] D. Katz, "IP router alert option." RFC 2113, Feb. 1997.

[107] D. S. Alexander, M. Shaw, S. M. Nettles, and J. M. Smith, "Active bridging," in *Proceedings of SIGCOMM '97*, 1997.

[108] S. Bhattacharjee, K. L. Calvert, and E. Zegura, "An architecture for active networking," in *High Performance Networking (HPN'97)*, Apr. 1997.

[109] D. D. Clark and D. L. Tennenhouse, "Architectural considerations for a new generation of protocols," in *Proceedings of SIGCOMM 90*, 1990.

[110] A. T. Campbell, I. Katzela, K. Miki, and J. Vicente, "Open signaling for ATM, internet and mobile networks (OpenSig'98)," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 1, pp. 97–109, 1999.

[111] A. T. Campbell, H. G. D. Meer, M. E. Kounavis, K. Miki, J. B. Vicente, and D. Villela, "A survey of programmable networks," *ACM SIGCOMM Computer Communications Review*, vol. 29, no. 2, 1999.

[112] A. T. Campbell, M. E. Kounavis, D. A. Villela, J. B. Vicente, H. G. D. Meer, K. Miki, and K. S. Kalaichelvan, "Spawning networks," *IEEE Network*, pp. 16–29, 1999. July/August.

[113] M. J. Wooldridge and N. R. Jennings, "Agent theories, architectures, and languages: A survey," in *Proceedings ECAI-Workshop on Agent Theories, Architectures and Languages* (M. J. Wooldridge and N. Jennings, eds.), pp. 1–39, 1994.

[114] A. B. Kulkarni and G. J. Minden, "Active networking services for wired/wireless networks," in *IEEE INFOCOM'99*, 1999.

[115] O. Angin, A. T. Campbell, M. E. Kounavis, and R. R. F. Liao, "An open programmable mobile network: Design, implementation and evaluation," in *Proc. 8th Intl Workshop on Network and Operating System Support for Digital Audio and Video (Nossdav'98)*, (Cambridge, England), July 1998.

[116] V. Bose, M. Ismert, W. Wellborn, and J. Guttag, "Virtual radios," *IEEE Journal on Selected Areas in Communications Special Issue on Software Radios*, 1998.

[117] M. E. Kounavis, A. T. Campbell, G. Ito, and G. Bianchi, "Accelerating service creation and deployment in mobile networks," in *IEEE Open Architecture 2000*, (Israel), 2000.

[118] Y. Yemini and S. da Silva, "Towards programmable networks," in *IFIP/IEEE International Workshop on Distributed System: Operations and Management*, 1996.

[119] G. Goldszmidt and Y. Yemini, "Distributed management by delegation," in *Proceedings of the 15th International Conference on Distributed Systems*, June 1995.

[120] B. Schwartz, A. W. Jackson, W. T. Strayer, W. Zhou, R. D. Rockwell, and C. Patridge, "Smart packets for active networks," in *IEEE OPENARCH'99*, 1999.

[121] D. Raz and Y. Shavitt, "An active network approach to efficient network management," in *First International Working Conference on Active Networks (I-WAN'99)*, (Berlin, Germany), 1999.

[122] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A simple network management protocol (snmp)." RFC 1157, May 1990.

[123] T. Faber, "Acc: Using active networking to enhance feedback congestion congestion control mechanisms," May 1998.

[124] S. Bhattacharjee, "On active networking and congestion," Tech. Rep. GIT-CC-96-02, Georgia Institute of Technology, Atlanta, Georgia, 1996.

[125] S. Bhattacharjee and M. W. McKinnon, "Performance of application-specific buffering schemes for active networks," tech. rep., Gerogia Institute of Technology, Atlanta, GA, 1998. GIT-CC-98-17.

[126] B. Williamson and C. Farrell, "Active congestion control," in *IEEE Globecom*, (Sydney), 1998.

[127] B. J. Williamson, C. A. Farrell, and D. W. Reid, "An active approach to consolidating TCP and ABR flow control," in *IEEE 8th International Conference on Computer Communications and Networks (ICCCN'99)*, (Boston, USA), 1999.

[128] R. Wittmann and M. Zitterbart, "Active multicasting for heterogenous groups," in *Proceedings of the fourth International conference on Broadband communications*, (Stuggart, Germany), 1998.

[129] S. Bhattacharjee, K. L. Calvert, and E. W. Zegura, "Network support for multicast video distribution," tech. rep., College of Computing, Georgia Tech., Atlanta, GA, 1998. GIT-CC-98/16.

[130] E. Amir, S. McCanne, and H. Zhang, "An application level video gateway," in *Proceedings ACM Multimedia*, (San Francisco), 1995.

[131] L.-W. Lehman, S. J. Garland, and D. L. Tennenhouse, "Active reliable multicast," in *IEEE INFOCOM'98*, (San Francisco), 1998.

[132] R. Wittmann and M. Zitterbart, "Amnet: Active multicasting network," in *Proceedings of International Conference on Communications (ICC'98)*, 1998.

[133] W. Lau, *On Active Networks and the Receiver Heterogeneity Problem In Multicast Session.* Honours thesis, Curtin University of Technology, Western Australia, 1998.

[134] M. Baldi, G. P. Picco, and F. Risso, "Designing a video conference system for active networks," in *Proceedings of 2nd International Workshop on Mobile Agents*, (Stuggart , Germany), 1998.

[135] R. Keller, S. Choi, D. Decasper, M. Dasen, G. Fankhauser, and B. Plattner, "An active router architecture for multicast video distribution," in *INFOCOM'2000*, (Israel), 2000.

[136] D. Decasper, Z. Dittia, G. Parulkar, and B. Plattner, "Router plugins: A software package architecture for next generation routers," in *SIGCOMM'98*, 1998.

[137] S. McCanned, M. Vetterli, and V. Jacobson, "Low-complexity video coding for receiver-driven layered multicast," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 6, p. 983, 1997.

[138] V. C. Van, "A defence against address spoofing using active networks," m. s. thesis, Department of Electrical Engineering and Computer Science, MIT, May 1997.

[139] M. Hicks and A. D. Keromytis, "A secure PLAN," in *1st International Working Conference on Active Networks*, (Berlin, Germany), 1999.

[140] U. Legedza and J. Guttag, "Using network-level support to improve cache routing," in *3rd International WWW Caching Workshop*, (Manchester, England), 1998.

[141] U. Legedza, D. Wetherall, and J. Guttag, "Improving the performance of distributed applications using active networks," in *INFOCOM'98*, (San Francisco,USA.), 1998.

[142] S. Bhattacherjee, K. L. Calvert, and E. W. Zegura, "Congestion control and caching in CANES," in *ICC'98*, 1998.

[143] N. F. Maxemchuk, "Active networks in telephony," in *IEEE OPENARCH'99*, 1999.

[144] J. Zhang and M. Ott, "ANSWER: Information routing based on active networks," internal report, 98-r-089, NEC C&C Research Laboratories, Princeton, NJ, 1998.

[145] D. J. Wetherall and D. L. Tennenhouse, "The ACTIVE IP option," in *Proceedings of the 7th SIGOPS European Workshop,*, (Connemara, Ireland), 1996.

[146] D. J. Wetherall, J. Guttag, and D. L. Tennenhouse, "ANTS: A toolkit for building and dynamically deploying network protocols," in *IEEE OPENARCH'98*, (San Francisco), Apr. 1998.

[147] Sun Microsystems, "The java language: A white paper." White Paper, 1994.

[148] R. Rivest, "The MD5 message-digest algorithm." RFC 1321, Apr. 1992.

[149] A. Fernando, B. Kummerfeld, A. Fekete, and M. Hitchens, "A new dynamic architecture for an active network," in *IEEE Open Architecture 2000*, (Israel), 2000.

[150] D. Decasper and B. Plattner, "DAN:distributed code caching for active networks," in *INFOCOM'98*, 1998.

[151] E. L. Nygren, S. J. Garland, and M. F. Kaashoek, "PAN:a high performance active network node supporting multiple mobile code systems," in *IEEE OPENARCH'99*, Mar. 1999.

[152] A. B. Kulkarni, G. J. Minden, R. Hill, Y. Wijata, A. Gopinath, and S. Sheth, "Implementation of a prototype active network," in *1st IEEE Conference on Open Architectures and Network Programming*, Apr. 1998.

[153] A. Banchs, W. Effelsberg, C. Tschudin, and V. Turau, "Multicasting multimedia streams with active networks," in *IEEE Proceedings of LCN*, (Boston), Oct. 1998.

[154] D. S. Alexander, B. Braden, C. A. Gunter, A. W. Jackson, A. D. Keromytis, G. J. Minden, and D. Wetherall, "Active network encapsulation protocol (ANEP)," July 1997. http://www.cis.upenn.edu/ switchware/ANEP/docs/ANEP.txt.

[155] M. Hicks, J. T. Moore, and D. S. Alexander, "PLANet: An active internetwork," in *IEEE INFOCOM'99*, 1999.

[156] X. Leroy, "The caml light system (release 0.71)," July 1996. http://maya.cicrp.jussieu.fr/man-caml/.

[157] M. Hicks, P. Kakkar, J. T. Moore, C. A. Gunter, and S. Nettles, "PLAN:a packet language for active networks," in *Procs. of the 3rd ACM SIGPLAN International Conference on Functional Programming Language*, pp. 86–93, 1998.

[158] J. H. Hartman, U. Manber, L. L. Peterson, and T. A. Proebsting, "Liquid software: A new paradigm for networked systems," tech. rep., Department of Computer Science, University of Arizona, 1996. Technical Report 96-11.

[159] J. H. Hartman, L. L. Peterson, A. Bavier, P. A. Bigot, P. Bridges, B. Montz, R. Piltz, T. A. Proebsting, and O. Spatscheck, "Joust: A platform for liquid software," *IEEE Networks*, July 1998.

[160] D. M. Murphy, "Building an active node on the internet," m. s. thesis, Department of Electrical Engineering and Computer Science, MIT, May 1997.

[161] S. Merugu, S. Bhattacharjee, E. Zegura, and K. Calvert, "Bownman: A node OS for active networks," in *INFOCOM'2000*, (Israel), 2000.

[162] S. Merugu, S. Bhattacharjee, Y. Chae, M. Sanders, K. Calvert, and E. Zegura, "Bowman and CANEs: Implementation of an active network," in *37th Annual Allerton Conference on Communication, Control and Computing.*, (Moticeilo, Illinois), Sept. 1999.

[163] A. Boulis, P. Lettieri, and M. B. Srivastava, "Active basestations and nodes for a mobile environment," in *Proceedings of the first ACM International Workshop on Wireless Mobile Multimedia (WOWMOM'98)*, (Dallas, Texas), Oct. 1998.

[164] S. Eichert, J. Kaplan, J. M. Smith, W. Arbaugh, and D. Fisher, "The SANE O/S." http://www.cis.upenn.edu/s̃witchware/sane_os, 2000.

[165] D. S. Alexander, W. A. Arbaugh, A. D. Keromytis, and J. M. Smith, "A secure active architecture." To Appear in IEEE Network, Special Issue on Active and Controllable Networks, 1998.

[166] N. C. Hutchinson and L. L. Peterson, "The x-kernel: An archictecture for implementing network protocols," *IEEE Transactions on Software Engineering*, vol. 17, p. 64, Jan. 1991.

[167] D. A. Wallach, D. R. Engler, and M. F. Kaashoek, "Ashs: Application-specific handlers for high-performance messaging," in *Proceedings of ACM SIGCOMM'96*, Aug. 1996.

[168] D. C. Feldmeier, A. J. McAuley, J. M. Smith, D. S. Bakin, W. S. Marcus, and T. M. Raleigh, "Protocol boosters," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 3, pp. 437–443, 1998.

[169] X. Inc, "Xbind broadband source code." http://www.xbind.com.

[170] E. Emir, S. McCanne, and R. Katz, "An active service framework and its application to real-time multimedia transcoding," in *SIGCOMM'98*, 1998.

[171] P. Sudame and B. R. Badrinath, "Transformer tunnels: A framework for providing route-specific adaptations," in *The USENIX Annual Technical Conference*, June 1998.

[172] B. R. Badrinath and P. Sudame, "Car pooling on the net: Performance and implications," in *ACM SIGCOMM'99*, (Massachusetts, USA), 1999.

[173] J. E. van der Merwe, S. Rooney, I. M. Leslie, and S. A. Crosby, "The tempest: A practical framework for network programmability," *IEEE Network Magazine*, vol. 12, pp. 20–28, June 1998.

[174] D. S. Alexander, K. G. Anagnostakis, W. A. Arbaugh, A. D. Keromytis, and J. M. Smith, "The price of safety in an active networks," 1999. http://www.cis.upenn.edu/s̃witchware.

[175] J. S. Shapiro, S. J. Muir, J. M. Smith, and D. J. Farber, "Operating system support for active networks." Technical Report, 1997.

[176] S. Thibault, C. Consel, and G. Muller, "Safe and efficient active network programming," in *17th IEEE Symposium on Reliable Distributed Systems*, (Indiana, USA), pp. 135–143, 1998.

[177] S. Thibault, J. Marant, and G. Muller, "Adapting distributed applications using extensible networks," in *IEEE Proceedings of ICDCS*, (Austin, Texas), 1999.

[178] P. Menage, "RCANE: A resource controlled framework for active network services," in *First International Working Conference on Active Networks (I-WAN'99)*, (Berlen, Germany), 1999.

[179] J.-H. Yeh, R. Chow, and R. Newman, "A dynamic interdomain communication path setup in active network," in *First International Working Conference on Active Networks (IWAN'99)*, (Berlin, Germany), 1999.

[180] B. Williamson and C. Farrell, "Independent active program representation using ASN.1," *ACM SIGCOMM Computer Communication Review*, vol. 29, Apr. 1999.

[181] ITU-T, "Specification of basic encoding rules for abstract syntax notation one (ASN.1)." Recommendation X.209, International Telecommunication Union, 1999.

[182] D. S. Alexander, B. Braden, C. A. Gunter, A. W. Jackson, A. D. Keromytis, G. J. Minden, and D. Wetherall, "Active network encapsulation protocol (ANEP)." RFC, http://www.cis.upenn.edu/ switchware/ANEP.

[183] R. Caceres and V. N. Padmanabhan, "Fast and scalable handoffs for wireless internetworks," in *ACM/IEEE MOBICOM'96*, (Rye, New York, USA), Nov. 1996.

[184] R. Ludwig and B. Rathonyi, "Link layer enhancements for TCP/IP over GSM," in *IEEE INFOCOM'99*, 1999.

[185] H. Balakrishnan, V. Padmanabhan, S. Seshan, M. Stemm, and R. H. Katz, "TCP behaviour of a busy internet server," in *IEEE INFOCOM'98*, pp. 252–262, 1998.

[186] Network Research Group, "ns-LBNL network simulator." Lawrence Berkeley National Laboratory, http://www-nrg-ee.lbl.gov/ns/.

[187] K. Fall and S. Floyd, "Simulation-based comparisons of tahoe, reno, and SACK TCP," *Computer Communication Review*, vol. 26, no. 3, pp. 5–21, 1996.

[188] J. Moore, "Mobile code security techniques," tech. rep., University of Pennsylvania, May 1998. MS-CIS-98-28.

[189] E. Amir, H. Balakrishnan, S. Seshan, and R. H. Katz, "Efficient TCP over networks with wireless links," in *Proceedings of fifth Workshop on Hot Topics in Operating Systems (HotOS-V)*, (Orcas Island, WA), May 1995.

[190] M. Srivastava and P. P. Mishra, "On quality of service in mobile wireless networks," in *Proceedings of the seventh International workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, (St Louis, Missouri), May 1997.

[191] R. Ghai and S. Singh, "A protocol for seamless communication in a picocellular network," in *IEEE Supercomm/ICC*, May 1994.

[192] MIL3 Inc., "Opnet Modeler 3.5.A. Network Simulation Software," 1997.

[193] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms and Applications.* Prentice-Hall, 1993.

[194] K. Brown and S. Singh, "A network architecture for mobile computing," in *IEEE Infocom'96*, (San Francisco), Mar. 1996.

[195] M. Calderon, M. Sedano, A. Azcorra, and C. Alonso, "Active network support for multicast applications," *IEEE Network*, vol. 12, pp. 46–53, May 1998.

[196] R. Droms, "RFC1542: Dynamic host configuration protocol," Oct. 1993.

[197] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A new resource ReSerVation protocol," *IEEE Network*, Sept. 1993.

[198] C. L. Hedrick, "RFC1058: Routing information protocol," June 1988.

[199] L. Wei and D. Estring, "The trade-offs of multicast trees and algorithms." Internet Draft, Mar. 1995. draft-ietf-idmr-mtree-00.txt.

[200] S. Komandur, J. Crowfort, and D. Mosse, "CRAM: Cell re-labelling at merge-points for ATM multicast," in *IEEE International Conference on ATM (ICAT-M'96)*, (Colmar, France), 1996.

[201] S. Min, H. K. Pung, and L. Wong, "A source-rooted multipoint-to-multipoint multicast in ATM networks," in *Second International Conference on Information, Communications and Signal Processing*, (Singapore), 1999.

[202] H. Mitts, H. Hansen, J. Immonen, and S. Veikkolainen, "Lossless handover for wireless ATM," *Mobile Networks and Applications (MONET)*, vol. 1, no. 3, p-p. 299–312, 1996.

[203] K. W. Chin, M. Kumar, and C. Farrell, "A model for enhancing connection rerouting using active networks," in *The 2nd ACM International Workshop on Modeling and Simulation of Wireless and Mobile Systems (MSWiM'99)*, (Seattle, Washington), Aug. 1999.

[204] N. Golmie, F. Mouveaux, L. Hester, Y. Saintillan, A. Koenig, and D. Su, "The NIST ATM/HFC network simulator." National Institue of Standards and Technology, U.S. Department of Commerce, Dec. 1998. http://w3.antd.nist.gov/Hsntg/prd_atm-sim.html.

[205] B. A. J. Banh, "Handover re-routing schemes for connection oriented services in mobile ATM networks," tech. rep., Switched Networks Research Centre, The Institute for Telecommunications Research, University of Wollongong, Australia, June 1997. http://www.snrc.uow.edu.au.

[206] R. Sivakumar, S.-W. Han, and V. Bharghavan, "A scalable architecture for active networks," in *IEEE Open Archicturecture 2000*, (Israel), 2000.

Appendix A

Parameters Used in the Derivation of Algebraic Expressions

The parameters used in the analysis of the ACR scheme are shown in Table A.1 and A.2.

A.1 Topologies Used in Simulation Studies

We tested our scheme on the mesh topology with 100 switches interconnected randomly. The degree of each switch in the topology varies from one to three. An example of a mesh topology is shown in Figure A.1.

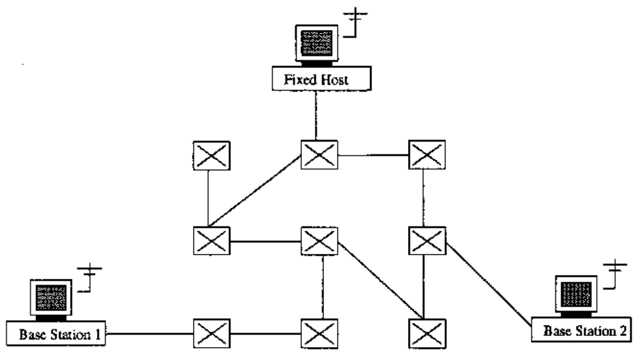


Figure A.1: Mesh Topology.

Variable	Description	Values
M_{sig}	Signalling message size	50 Bytes
M_d	Data packet size or AAL PDU size	8 Kbytes
BW_{sig}	Signalling channel bandwidth	64 Kbytes
BW_d	Data channel bandwidth	1 Mbps
L_w	Propagation delays for the wireless links	3 μs
$L_{bsc-bts}$	Propagation delays for the BSC-to-BTS Links	50 μs
L_{bsc-sw}	Propagation delays for the BSC-to-switch links	50 μs
L_{sw}	Propagation delays for the inter-switch links	50 μs
S_{stp}	Processing time of signalling messages in nodes where only Signal Transfer Point (STP) function is required.	0.3 ms
$S_{sw(setup)}$ $S_{bsc(setup)}$ $S_{bts(setup)}$	Processing time of SETUP message and target BTS determination functions in various type of network nodes.	8 ms 16 ms 32 ms
$S_{sw(disc)}$ $S_{bsc(disc)}$ $S_{bts(disc)}$	Processing time of DISC message in various types of network nodes	4 ms 8 ms 16 ms
$S_{sw(upd-tbl)}$	Time to update translation table	4 ms
$S_{sw(other)}$ $S_{bsc(other)}$ $S_{bts(other)}$	Processing time of signalling messages (other than SETUP, DISC, and fast resource reservation) in various types of network nodes	3 ms 6 ms 12 ms
$S_{sw(sw)}$ $S_{sw(bsc)}$ $S_{sw(bts)}$	Processing time for switching an ATM cell through an ATM switch	10 μs 20 μs 40 μs
$S_{sw(setup-active)}$ $S_{bsc(setup-active)}$ $S_{bts(setup-active)}$	Processing time of SETUP message (with programming) and target BTS determination functions in various type of network nodes.	9.6 ms 19.2 ms 38.4 ms
$N_{old-new}$	Number of switches between the old BSC and the new BSC.	N
N_{cx-old}	Number of switches between from the CX to the old BS.	N
$N_{old-org}$ $N_{new-org}$	Number of switches between BS_1 or BS_2 and the switch connecting to the FH	20 20
S_m	Signalling message processing time in the MS	5 ms

Table A.1: Input parameters.

We consider a tree with four levels. The connectivity of each intermediate node varies from three to eight. Since the BSs are connected randomly, there are different numbers of BSs at different levels of the hierarchy. When the MH migrates from BS_1 to BS_2 where both BSs are in different parts of the tree, during path extension it will reuse nodes from BS_1 to the node common to both these BSs. This common node is the CX. Also in the tree topology a high number (compared to the Mesh topology) of nodes will

Variable	Message Type	Transmitted on..	Value
T_w	Signalling	Wireless links	$(M_{sig}/BW_{sig}) + L_w$
$T_{bsc-bts}$	Signalling	BSC-BTS link	$(M_{sig}/BW_{sig}) + L_{bsc-bts}$
T_{bsc-sw}	Signalling	BSC-Switch links	$(M_{sig}/BW_{sig}) + L_{bsc-sw}$
T_{sw}	Signalling	Switch-Switch links	$(M_{sig}/BW_{sig}) + L_{sw}$
$T_{d(w)}$	Data	Wireless links	$M_d/BW_d + L_w$
$T_{d(bsc-bts)}$	Data	BSC-BTS links	$M_d/BW_d + L_{bsc-bts}$
$T_{d(bsc-sw)}$	Data	BSC-Switch links	$M_d/BW_d + L_{bsc-sw}$
$T_{d(sw)}$	Data	Switch-Switch links	$M_d/BW_d + L_{sw}$

Table A.2: Message transmission times.

be reused when BS_2 sets up a path to BS_1 a number of nodes from the CX to BS_1 will be reused. Figure A.2(a) shows a tree topology with three levels and each node has two child nodes.

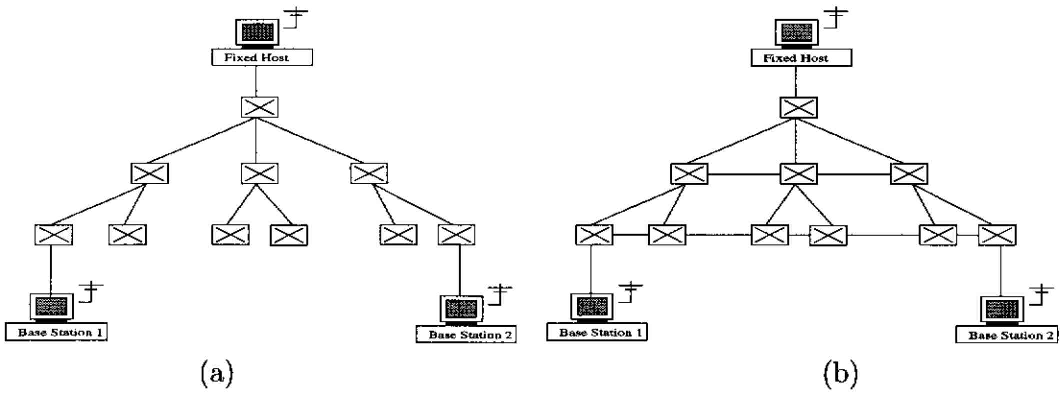


Figure A.2: (a) Tree (b) Redundant Tree Topologies

The redundant tree looks similar to the tree topology with additional linear connection among nodes of each level. An example of this topology is shown in Figure A.2(b). As can be seen the routing does not need to traverse up the tree to reach another node on a different branch. In the simulation our topology consists of four levels each node has three child nodes.

A.2 Analysis of ACR

Here the expressions for handoff latency, disruption time and buffer requirements for the ACR scheme are derived. Equations for other rerouting schemes have already been derived and can be found in [205]. The ACR scheme is then compared with other rerouting schemes that have been proposed in the literature.

Firstly, to analyse the ACR scheme the sequence of signaling and messages generated during handoff including message generated during path rerouting are used. Figure A.3 depicts the signaling and messages involved. Note that the network architecture used is adapted from [59]. The BSCs and MSCs are ATM switches and have switching, routing and buffering capabilities. All handoffs initiated by the MH and hard-handoffs are considered [59].

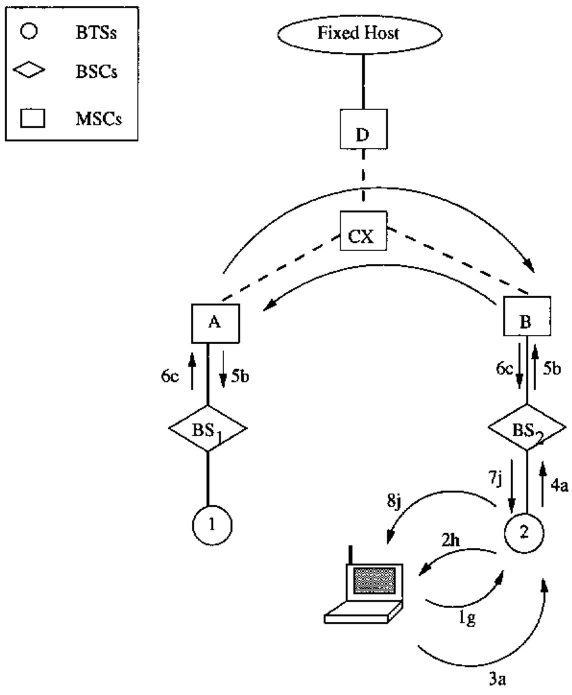


Figure A.3: Active Path Rerouting.

When the MH detects that it is in the vicinity of BS_2, it sends an ACCESS command in order to establish a radio link. The latency incurred is expressed as

$$T_{1g} = T_w + S_{bts(other)} \tag{A.1}$$

which is the sum of latency transmitting the message across the wireless links and the processing time at BS_2. If BS_2 accepts the MH it sends an ACCESS_GRANTED message back to the MH. The delay incurred is given by:

$$T_{2h} = T_w + S_m \tag{A.2}$$

After establishing a radio channel the MH sends a HO_REQUEST to the BS, whose delay is given by,

$$T_{3a} = T_w + S_{bts(other)} \tag{A.3}$$

The HO_REQUEST is forwarded onto the BSC. The latency is calculated as the sum of transmission time from the BTS to BSC and the processing time at BSC,

$$T_{4a} = T_{bsc-bts} + S_{bsc(other)} \tag{A.4}$$

Once the BSC has received the HO_REQUEST command it proceeds to setup up a path to the previous BS while programming all ASs between BS_2 and BS_1. Here the latency of the setup (including programming) at each network node is set to be 20% higher than the normal latencies. These latencies are reflected in the variables $S_{sw(setup-active)}$, $S_{bts(setup-active)}$ and $S_{bsc(setup-active)}$. The latency incurred for the message T_{5b} is calculated as the transmission time of the setup message to the first switch, followed by the transmission and processing time of the message through N number of ASs (including programming) before reaching BS_1,

$$\begin{aligned} T_{5b} &= 2S_{bts(setup)} + N_{old-new}T_{sw} + S_{bts(setup-active)} + \\ &\quad N_{old-new}S_{sw(setup-active)} + 2T_{bsc-sw} + S_{sw(upd-tbl)} \end{aligned} \tag{A.5}$$

A connect message is then returned to BS_2 if the setup was successful. Hence the latency of the connect message is given by:

$$T_{6c} = 2T_{bsc(sw)} + S_{sw(other)}N_{old-new} + T_{sw}N_{old-new} + 2S_{bsc(other)} \tag{A.6}$$

After the path has been setup, a HO_REPLY is generated by the BSC which is then forwarded to the MH. The HO_REPLY has to traverse through BSC, BTS and the wireless link before reaching the MH. The latency from the BSC to BTS is expressed by:

$$T_{7j} = T_{bsc-bts} + S_{bts(other)} \tag{A.7}$$

The later part of the traversal equates to,

$$T_{8j} = T_w + S_m \tag{A.8}$$

In an architecture where BSs have direct links to each other, the following modifications to T_{5b} and T_{6c} are made;

$$T_{5b} = 2S_{bts(setup)} + T_{sw} + 2T_{bsc-sw} + S_{sw(upd-tbl)} \tag{A.9}$$

$$T_{6c} = 2T_{bsc-sw} + 2S_{bsc(other)} \tag{A.10}$$

The equations, (A.6) and (A.6), imply that handoff latency will be at a minimal and dependent on the link connecting BS_1 and BS_2.

A.2.1 Handoff Latency

The handoff latency is calculated from the time the MH detects BS_2 to the receipt of HO_REPLY. Hence the handoff latency is calculated by summing the signaling latencies from T_{1g} to T_{8j}. The handoff latency is computed as follows,

$$\begin{aligned} T_{HO-latency} &= 3T_w + 3S_{bts(other)} + 2S_{bts(setup-active)} + S_{bsc(setup-active)} + \\ &\quad S_{sw(setup-active)}N_{old-new} + 4T_{bsc(sw)} + S_{sw(other)}N_{old-new} + \\ &\quad 2T_{sw}N_{old-new} + S_{bsc(other)} + 2S_{stp} + S_{sw(upd-tbl)} \end{aligned} \tag{A.11}$$

If BSs have links to each other (1-hop scenario) the handoff latency is:

$$\begin{aligned} T_{HO-latency-1-hop} &= 4T_w + 3S_{bts(other)} + 2S_m + 2T_{bsc-bts} + 2S_{bts(setup-active)} + \\ &\quad T_{sw} + 4T_{bsc-sw} + 3S_{bsc(other)} + S_{sw(upd-tbl)} \end{aligned} \tag{A.12}$$

If soft handoff is performed where an extended path has been setup when the MH migrates to BS_2 the handoff expression would be:

$$\begin{aligned} T_{HO-latency-soft} &= T_{1g} + T_{2h} + T_{3a} + T_{7j} + T_{8j} \\ &= 2(S_m + T_{bsc-bts} + 4T_w + 3S_{bts(other)} + S_{bsc(other)} \end{aligned} \quad \text{(A.13)}$$

In Equation A.14 the time to setup the extended path is omitted. In this scenario once BS_2 receives the HO_REQUEST, it returns a HO_REPLY immediately.

A.2.2 Disruption Time

The disruption time is the time in which the MH is not communicating with the corresponding host [59]. Therefore, the disruption time in ACR is the sum of the handoff latency and the time taken for data to reach the MH after the translation table at BS_1 has been updated.

$$T_{disrupt} = T_{HO-latency} + N_{old-new}T_{d(sw)} + T_{d(w)} + T_{d(bsc-sw)} + T_{d(bsc-bts)} + S_m \quad \text{(A.14)}$$

The disruption time for 1-hop scenario is:

$$T_{disrupt-1-hop} = T_{HO-latency-1-hop} + T_{d(bsc-bts)} + T_{d(w)} + S_m \quad \text{(A.15)}$$

In the soft handoff scenario the disruption time is calculated based on the following latencies:

1. Establishing communication link with BS_2. This equates to $T_{1g} + T_{2h} + T_{3a}$.
2. A HO_COMPLETE is then sent to the MH and a HO_DATA_FORWARD message is generated by BS_2 to BS_1 (traversing through $N_{old-new}$ switches). Upon receiving the HO_DATA_FORWARD message BS_1 updates its translation table. The time taken for the MH to receive the first cell forwarded from BS_1 is therefore,
$2(T_{bsc-sw} + N_{old-new}(T_{sw} + S_{sw(other)}) + 2T_{d(bsc-sw)} + N_{old-new}T_{d(sw)} + T_{d(bsc-bts)} + T_{d(w)}$

Hence the disruption time for soft handoff is:

$$\begin{aligned} T_{disrupt-soft} &= 3T_w + S_m + 2_{bts(other)} + N_{old-new}(T_{sw} + S_{sw(other)} + T_{d(sw)} + \\ & T_{d(w)} + T_{d(bsc-bts)} + T_{bsc-bts} + S_{bsc(other)} + 2(T_{bsc-sw} + \\ & T_{d(bsc-sw)}) \end{aligned} \tag{A.16}$$

A.2.3 Buffer Requirements

The amount of buffering required by a path rerouting scheme determines its scalability. In ACR, buffering is required at the BS and MH. We denote Q_{MH} and Q_{BS} as the buffering requirements at the MH and BS respectively. During handoff the MH has to buffer all upstream traffic and the amount of cells buffered is dependent on the handoff latency. On the other hand, the buffering at the BS is dictated by the length of the old path which extends from from the CX passing through BS_1 and then onwards to BS_2.

Q_{MH} equates to $T_{HO-latency}$. This is because buffered cells at the MH can only be transmitted after uplink stream(s) from BS_2 to BS_1 has been setup. Therefore, the expression for Q_{MH} is

$$Q_{MH} = T_{HO-latency} \times BW_d \tag{A.17}$$

Consequently the equation for the 1-hop case would be:

$$Q_{MH} = T_{HO-latency-1-hop} \times BW_d \tag{A.18}$$

and the equation for soft handoff scenarios is:

$$Q_{MH} = T_{HO-latency-soft} \times BW_d \tag{A.19}$$

Q_{BS} is dependent on the length of the path CX-BS_1-BS_2[1]. BS_2 starts buffering after receiving a BUFFER command from the CX. The time needed to drain the old path is the time taken by the marker cell to traverse the old path. Therefore,

$$Q_{BS} = ((N_{cx-old} + N_{old-new})(T_{d(sw)} + S_{sw(sw)}) + T_{d(sw)}) \times BW_d \tag{A.20}$$

[1]refer subsequently as the *old path*

Appendix B

Signaling Messages Used by Proposed Protocols

SETUP	Initiate a connection to corresponding party
CONNECT	Corresponding party accepts connection setup
RELEASE	Connection setup is rejected
HO_REQUEST	Indicates the new MH's location
HO_DATA_FORWARD	Indicate the resumption of cell forwarding
HO_REPLY	Indicate the handoff process is complete
HO_REJECT	The handoff process has been rejected
OPTIMIZE	To begin the optimization process
HO_UPDATE	To update all MH bindings at switches

Table B.1: Signaling messages for ACR

Signaling Message	Description
JOIN_REQUEST	Issue by clients to join the multicast session
JOIN_ACK	Join successful
REG_CONF	Registration message sent to the location directory
CORE_CONNECT	Sent by core routers to MH's current location
HO_UPDATE	as part of the process to determine closest core Informs the previous BS of the MH's new location
HOP_DISCOVERY	A message send by the previous BS to inform all cores to begin core discovery process
OP_DISCOVER	To discover whether optimisation is needed
GRAFT_MESG	To graft a new branch onto the tree
GRAFT_ACK	To signify successful graft operation
PRUNE_MESG	To prune off unwanted branches
STOP_FORWARD	To stop the forwarding of data from previous BS
HO_COMPLETE	Marks the end of the handoff process

Table B.2: Signaling messages for AMTree

Signaling Message	Description
ADD_PARTY	Initiate a join. Issue by PR on behalf of clients
HO_UPDATE	To update all proxy roots of the MH's new location
TREE_UPDATE	To initiate optimization process
TREE_OP_ACK	Sent after successful optimization, signaling a new branch has been formed.
PRUNE_RDY	To inform upstream switch that the branch leading to it, is ready to be pruned.
MIG_CELL	To inform all cores to prune and update to new connections
LIJ	Used by clients to join a given multicast session

Table B.3: Signaling messages for MCoRe