Mobile IPv6 based multiple network interfaces management for ubiquitous communications

Huang, Haijie

2006

Huang, H. (2006). Mobile IPv6 based multiple network interfaces management for ubiquitous communications. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/2657>

<https://doi.org/10.32657/10356/2657>

Nanyang Technological University

Downloaded on 26 Apr 2023 17:25:27 SGT

Mobile IPv6 Based Multiple Network Interfaces Management for Ubiquitous Communications

Huang Haijie

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Master of Engineering

2006

TK
6570
• M6
17874
2006

Acknowledgments

This project would not have reached this stage without the help of my supervisor, Dr. Cai Jianfei. I am grateful to his suggestions that point me to the right direction, his encouragement that keeps me motivated, and his timely support in providing necessary hardware equipments. I would also like to express my appreciation to Dr. Andreas Jurgen Kassler and Dr. Gao Deyun for sharing their valuable thoughts with me.

I would like to extend my thanks to Ms. Chua Poo Hua and Ms. Oh Hwee May in Center for Multimedia and Network Technology, who kindly provide me with all the convenience I need.

Abbreviations

ACK : Acknowledgment
AP : Access Point
BA : Binding Acknowledgment
BU : Binding Update
CN : Correspondent Node
CoA : Care-of Address
CWND : Congestion Window
GPRS : General Packet Radio Service
HA : Home Agent
HoA : Home Address
LAN : Local Area Network
MN : Mobile Node
NAT : Network Address Translation
PAN : Personal Area Network
PDA : Personal Digital Assistant
RTO : Retransmission Timeout
RTT : Round-trip Time
SRTT : Smoothed Round-trip Time
WLAN : Wireless Local Area Network
WWAN : Wireless Wide Area Network

Contents

Acknowledgments	i
Abbreviations	ii
List of Figures	vi
List of Tables	viii
Abstract	ix
1 Introduction	1
1.1 Heterogeneous Wireless Network Environment	1
1.2 Network Handoff	4
1.2.1 Vertical Handoff and Horizontal Handoff	5
1.2.2 Hard Handoff and Soft Handoff	5
1.2.3 Link-layer Handoff and Network-layer Handoff	6
1.3 Mobile IPv6	7
1.3.1 Components of Mobile IPv6	8
1.3.2 Mobile IPv6 Communications	10
1.3.3 Mobile IPv6 for Linux	14
1.3.4 Why Choosing Mobile IPv6	15
1.4 Project Scope	17
1.5 Thesis Organization	19

2	Vertical Handoff Management System	20
2.1	Literature Review	20
2.2	Design of Vertical Handoff Management System	22
2.2.1	Goals	22
2.2.2	Architectural Design	23
2.3	Implementation and Experimental Results	28
2.3.1	Testbed Setup	28
2.3.2	Handoff from Bluetooth to GPRS	32
2.3.3	Handoff from GPRS to Bluetooth	33
2.4	Summary	34
3	Load Sharing in Multi-homed System	35
3.1	Overview of Multi-homed Environment	35
3.1.1	Per-host Distribution	37
3.1.2	Per-flow Distribution	39
3.1.3	Per-packet Distribution	40
3.2	Literature Review	42
3.3	Proposed Load Sharing System	44
3.3.1	Mapping Data Flows to Multiple Interfaces	45
3.3.2	Load Sharing Algorithm	50
3.4	Simulation Results	54
3.4.1	Buffer Requirement	55
3.4.2	Packet Delay	56
3.5	Summary	57
4	Improving TCP Performance during Soft Vertical Handoff	59
4.1	Background	59
4.2	Overview of TCP Characteristics	60
4.3	TCP Behaviors under Soft Vertical Handoff	62

4.3.1	Handoff from Slow Link to Fast Link	63
4.3.2	Handoff from Fast Link to Slow Link	64
4.4	Literature Review	65
4.5	Network-Layer Solutions to Spurious TCP Timeout	67
4.5.1	Scheme 1: Fast ACK	69
4.5.2	Scheme 2: Slow ACK	74
4.5.3	Scheme 3: ACK Delaying	74
4.6	Summary	90
5	Summary	92
	References	94
	Publications	99

List of Figures

1.1	Hard Handoff between Non-overlapping Networks	5
1.2	Soft Handoff between Overlapping Networks	6
1.3	Mobile IPv6 Components	8
1.4	Data Sent from MN to CN	11
1.5	BU Sent from MN to HA	12
1.6	Data from CN to MN with Binding Cache Entry Present	13
1.7	Data from CN to HoA of MN	14
1.8	Intercepted Packet Tunneled to MN by HA	15
1.9	Mobile IPv6 and Wireless Overlay Networks	16
1.10	Protocol Layers	17
2.1	Architectural Design	24
2.2	Performance of Handoff from Bluetooth to GPRS	32
2.3	Performance of Handoff from GPRS to Bluetooth	34
3.1	Per-host Traffic Distribution	38
3.2	Per-flow Traffic Distribution	39
3.3	Packet Scheduler	50
3.4	Simulation Setup	54
3.5	Re-sequencing Buffer Requirement	56
3.6	Average Packet Delay	57
4.1	Packet Sequence in Normal MIPv6 Vertical Handoff	68

4.2 RTT Changes in Two Step 70

4.3 Simulation Setup 71

4.4 Comparison of CWND between Normal Handoff and Fast ACK 72

4.5 Comparison of RTO between Normal Handoff and Fast ACK 73

4.6 Comparison of CWND between Normal Handoff and Slow ACK 75

4.7 Comparison of RTO between Normal Handoff and Slow ACK 76

4.8 Change of SRTT upon a Step Increase in RTT 77

4.9 Function $s(n)$ 79

4.10 Change of SRTT with Linear Increment of RTT 80

4.11 Function $p(n)$ 82

4.12 Change of SRTT with Quadratic Increment of RTT 83

4.13 Function $q(n)$ 85

4.14 Comparison of RTX between Normal Handoff and ACK Delaying 86

4.15 Percentage of Timeout vs Average RTT Difference (oldRTT=40ms) 87

4.16 Percentage of Timeout vs Average RTT Difference (oldRTT=900ms) 88

List of Tables

3.1	Example of Host-CoA Mapping Table	38
3.2	Example of Flow-CoA Mapping Table on MN	40
3.3	Example of Flow-CoA Mapping Table on CN	40
3.4	Option Feild of Flow-CoA Mapping in BU	49

Abstract

There are various link-layer technologies coexisting in the market that support wireless connections, such as Bluetooth, IEEE 802.11 and GPRS. These technologies are diverse in terms of bandwidth, coverage area, cost, power consumption, etc. Certain wireless networks are preferred over the others under certain circumstances. It is common that a mobile device is equipped with multiple wireless network interfaces, and connects to different networks at different time and locations in order to achieve ubiquitous communications. It is also possible for the mobile device to connect to multiple wireless networks simultaneously to enjoy aggregate bandwidth. Such kind of communications can be achieved by utilizing network mobility protocols, such as Mobile IP or Mobile IPv6.

However, there is a number of issues that need to be concerned in multi-path communications, such as unbalanced traffic load among different network paths and out-of-sequence packet reception. A proper load sharing mechanism is necessary for ensuring efficient packet distribution to multiple network interfaces. Another problem we aim to tackle in the scope of heterogeneous network communications is that inter-system hand-off usually causes abrupt change of network bandwidth and propagation delay, which will degrade the performance of end-to-end transport protocols such as TCP. It is possible to apply network-layer schemes to minimize TCP performance degradation without modifying TCP protocol.

This M.Eng program includes the following three projects. 1) Design and develop an intelligent system that can automatically manage multiple network interfaces on the

mobile device. The system makes handoff decisions at appropriate time based on a cost function. It is also able to predict the loss of network connection according to detected signal strength. Real world experimental results demonstrate that the system can handle vertical handoff smoothly without breaking on-going TCP connections. 2) Design policies and algorithm for a load sharing system that utilizes multiple network interfaces simultaneously to achieve bandwidth aggregation. The system includes a weighted-round-robin channel selector for balancing data traffic among different network links, and a Jump-Ahead packet selector for reducing the probability of out-of-sequence packet reception. 3) Resolve spurious TCP timeout problem using network-layer approaches to improve TCP performance during soft vertical handoff. Three schemes have been proposed, namely, Fast ACK, Slow ACK and ACK Delaying. These schemes can help TCP to gradually adjust its RTO estimation during vertical handoff so that the probability of spurious timeout is reduced.

Chapter 1

Introduction

Internet technology is growing dramatically since the last decades of the 20th century. Nowadays, almost every business equipped with desktop computers has access to the Internet. PCs connected to local area networks (LANs) or to their Internet service providers (ISPs) via fixed line are still the dominant terminals in the Internet community. However, the number of mobile users is increasing rapidly, mobile communication technology is attracting more and more research interests. One of the hottest areas of growth in the computer industry is within portable computing devices [1]. Examples are laptops, personal digital assistants (PDAs), hand phones, etc. These devices can be connected to the Internet using existing and emerging wireless network technologies.

1.1 Heterogeneous Wireless Network Environment

The traditional TCP/IP infrastructure aims at fixed network terminals, whose points of attachment to the Internet are not supposed to change frequently. For a home PC, it normally requires only one Ethernet card or a modem, and it can easily enjoy high-speed global Internet services. In the world of wireless communication, mobile devices are usually moving, changing point of attachment when leaving the previous access network and

CHAPTER 1. INTRODUCTION

entering the coverage of a new network. One of the major challenges in mobile computing is that there is no single wireless technology that can provide both global connectivity and high-speed data transmission. Hence, a variety of wireless technologies coexist in the market, with each technology meeting specific requirements of mobile users. Current wireless technologies vary widely in terms of bandwidth, delays, coverage area, power consumption etc. Some wireless networks provide wide-range access, but very low speed; some cover a short-range area, but provide high-bandwidth services. Different networks can coexist to compromise the different characteristics of each other, and establish a heterogeneous wireless environment. It is common that a mobile device is equipped with more than one network interfaces. For examples, some latest laptop models, such as HP Compaq nc4200 [2], come with internal Wi-Fi and Bluetooth interfaces. Popular PDAs like iPAQ hw6500 [3] have built-in Bluetooth and GPRS connectivity. Additional network interfaces can also be installed using PCMCIA cards. With such equipments, a mobile device can connect to different access networks at different locations to ensure seamless connectivity, or can even access multiple networks simultaneously to achieve higher performance.

Based on the coverage range, wireless networks can be categorized into three classes: *wireless wide area network* (WWAN), *wireless local area network* (WLAN) and *personal area network* (PAN). General Packet Radio Service (GPRS) is an example of WWAN, which is a cellular network that covers wide area but provides limited network bandwidth. IEEE 802.11b is the most popular standard for WLAN, which covers medium range and affords high bandwidth data communications. Bluetooth belongs to PAN, which is a short-range low-power wireless technology providing only in-room coverage. In our project, the mobile terminal is a laptop computer that has all the above three wireless interfaces installed.

GPRS is a wireless data system based on the GSM architecture. It is a value-added service that allows information to be sent or received across a mobile telephone network.

CHAPTER 1. INTRODUCTION

Laptops or PDAs can connect to GPRS network via specific PCMCIA network card, or via GPRS-enabled hand phone with Infrared or Bluetooth connection. The cell diameters of GPRS network can exceed $10Km$, with data rate in the range of $20 - 170Kbps$. GPRS provides the widest coverage area, but the lowest data rate compared with the other two type of technologies, and users will usually be charged for the amount of data they send or receive.[4]

IEEE 802.11b is a popular standard for WLAN. It is classified as a high-power, medium-range, high-bandwidth access technology. It is the wireless equivalent of the well-known IEEE 802.3 Ethernet protocol. IEEE 802.11b is also known as wireless Ethernet and Wi-Fi, which transmits in an unlicensed radio spectrum at $2.4GHz$ and provides a maximum bandwidth of $11Mbps$. This standard enables two operational modes. One is the *infrastructure mode*, where all communications go through a central base station, known as *access point* (AP). AP is a dedicated equipment, which normally has at least one wireless interface that connects with mobile terminals, and one wired interface that connects to a LAN. The AP acts as a gateway with a typical coverage range of about $100m$, which routes packets between mobile nodes and the LAN or the Internet. One or more mobile nodes connected to an AP are called a basic service set (BSS) and several BSSs connected together through an Ethernet link forms an extended service set (ESS). Another operational mode is *ad-hoc mode*, where there is no AP for central control, and all mobile nodes communicate directly if they can hear each other's signal.[5, 6]

Bluetooth is a short-range, low-power wireless technology designed for cable replacement. Unlike Infrared, Bluetooth does not have line-of-sight restriction for wireless connection. Bluetooth is designed to operate in the $2.4GHz$ radio band, providing data rate of up to $721kbps$. The typical range of connection is about $10m$, depending on the power levels of the participating devices. The Bluetooth system provides both point-to-point and point-to-multipoint connections. In point-to-multipoint connection, one device acts

CHAPTER 1. INTRODUCTION

as a master, other devices act as slaves and share the same channel. The master and slaves together form a piconet. Bluetooth is a multi-layered protocol. The bottom layer is the Baseband, lying on top of which are Link Manager Protocol (LMP), Logical Link Control and Adaptation Protocol (L2CAP). L2CAP provides connection-oriented and connectionless data services to upper protocol layers. At a higher layer, the Bluetooth Network Encapsulation Protocol (BNEP) is used by the Bluetooth PAN profile. This layer encapsulates packets from various networking protocols and transports them directly over L2CAP layer. Bluetooth specifies several profiles, each of which is suitable for specific applications. PAN is one of such profiles, which offers IP support over Bluetooth, comparable to IEEE 802.11b WLAN. In an infrastructure PAN system, the master of a piconet acts as network access point (NAP), which is a bridge between a LAN and wireless clients. The slaves of the piconet act as PAN users (PANUs), which connect to the NAP and gain access to the LAN and the backbone Internet. This is identical to a Wi-Fi infrastructure, except that a Bluetooth piconet can only support up to seven slave devices.[7, 8]

1.2 Network Handoff

Consider a user with a PDA equipped with Bluetooth, Wi-Fi and GPRS network interfaces. When she is in her office, she connects the PDA to a Bluetooth NAP to enjoy free-of-charge and low-power-consuming network service. Later, she goes out for an appointment with her client. When she is out of her office, Bluetooth signal is weakened, so the PDA changes to use the Wi-Fi interface to retain network connection. When she is traveling in her car, the WLAN in her company becomes out of range, the GPRS interface is then activated to keep the device connected to the Internet. When she reaches her destination and meets her client in a coffee cafe, Wi-Fi signal from the hotspot is discovered; the PDA then switches back to the Wi-Fi interface to enjoy the high-bandwidth

service. If interface switching is done seamlessly, the network sessions of the running applications on the PDA will not be broken, and the user will not experience any noticeable disruption from the network applications.

1.2.1 Vertical Handoff and Horizontal Handoff

In the above scenario, the mobile device switches among access networks with different link-layer technologies. Such operation is termed as *vertical handoff* or *vertical handover* [9]. If the mobile device is equipped with only one wireless interface, it may move from one cell to another within the same wireless system. Such handoff is called *horizontal handoff*.

1.2.2 Hard Handoff and Soft Handoff

Handoff can also be classified into *hard handoff* and *soft handoff*. A handoff is hard if the mobile device can be associated with only one AP at a time. That is, the mobile device has to disconnect from its previous AP before connecting to a new one. This may occur when the coverage of the old network and new network do not overlap, as illustrated in Figure 1.1.

Figure 1.1: Hard Handoff between Non-overlapping Networks

CHAPTER 1. INTRODUCTION

When MN is moving from Network 1 to Network 2, there is a short connection break at location B. The MN has to de-associate itself with the AP in Network 1 before a connection to Network 2 is established. A hard handoff may cause packet loss, because once the MN lost connection to Network 1, the packets arriving at the old AP cannot reach the mobile device any more.

On the other hand, a soft handoff occurs if the mobile device can communicate with more than one APs during handoff. This can happen when the coverage of the old network and new network overlap with each other, as illustrated in Figure 1.2.

Figure 1.2: Soft Handoff between Overlapping Networks

In this case, handoff is done in the overlap region, where the interface for the old network remains active for receiving packets until the connection with the new network is created. Seamless handoff is possible under such scenario, and packet loss can be avoided, since the MN is always reachable during and after handoff.

1.2.3 Link-layer Handoff and Network-layer Handoff

During horizontal handoff, e.g., moving within IEEE 802.11b WLAN, if the old cell and the new cell belong to the same IP subnet, the mobile device remains using its old IP address. This is called *link-layer handoff*, since only the information of the associated

AP is changed at the link layer. *Network-layer handoff* occurs when MN moves from one IP subnet to another, which requires the change of IP address at the network layer. Vertical handoff usually involves network-layer handoff, since it is unlikely that networks with different access technologies belong to the same IP subnet.

1.3 Mobile IPv6

When a mobile device changes its point of attachment in the Internet, e.g., switching from Wi-Fi to GPRS, the IP address of the mobile device will be changed. The change of IP address causes problem in traditional IP system, especially for connection-oriented protocol like TCP. In TCP, a connection is uniquely identified by four components: source IP address, source port, destination IP address, and destination port. If either of the communicating endpoint changes its IP address, existing TCP sessions will be broken. Here comes the need for a mobility management protocol.

There are various proposals and drafts for mobility management protocols. The most popular and promising ones are *Mobile IP* [10] for traditional IPv4 network, and *Mobile IPv6* [11] for future IPv6 [12, 13] network. Mobile IP solves the mobility problem by allowing a mobile node (MN) to use two IP addresses: *Home Address* (HoA) and *Care-of Address* (CoA). HoA is static, and used to identify connection by upper layer protocols like TCP. When MN moves from its home network to a foreign network (either with same or different link-layer technology), it obtains a CoA from the foreign network and registers the CoA with the *Home Agent* (HA), which is a proxy in MN's home network. Subsequent data packets targeted to the MN will be sent to the home network first, and the HA redirects them to the MN in the foreign network, which forms a path of triangular routing. Mobile IPv6 works on the same principles as Mobile IP, but has more advanced features. Besides the inherited advantages of IPv6 over IPv4, the *route*

CHAPTER 1. INTRODUCTION

optimization scheme in Mobile IPv6 eliminates the problem of triangular routing, which greatly improves communications efficiency. A more detailed description of Mobile IPv6 is illustrated in the followings subsections.

1.3.1 Components of Mobile IPv6

Figure 1.3: Mobile IPv6 Components

Here are the explanations of some terminologies used in Mobile IPv6:

- **Home link:** The home link is the link that the mobile node is originally connected to.
- **Home address (HoA):** HoA is the address assigned to the mobile node when it is attached to the home link and through which the mobile node is always reachable, regardless of the mobile node's current location. Packets addressed to the mobile node's HoA are delivered to the home link using normal IPv6 routing processes. If the mobile node is attached to the home link, IPv6 mobility processes are not used

CHAPTER 1. INTRODUCTION

and communication occurs normally. If the mobile node is away from home, IPv6 mobility processes are used to deliver the packets to its current point of attachment on the IPv6 Internet. Because the mobile node is always assigned a HoA, it always has a virtual connection to the home link. This relationship is shown in Figure 1.3 as the Virtual Mobile Node.

- **Home agent (HA):** The HA is a router on the home link that maintains awareness of the mobile nodes on its home link and the addresses that the mobile nodes are currently using. If the mobile node is on the home link, the HA acts as a normal IPv6 router, forwarding packets addressed to the mobile node. If the mobile node is away from home, the HA tunnels data to the mobile node's current location on the IPv6 Internet.
- **Mobile node (MN):** The MN is an IPv6 node that can change links, and therefore addresses, and maintain reachability using its HoA. A MN is aware of its HoA and the address on its current link, and indicates them to its HA and the communicating parties.
- **Foreign link:** A foreign link is a link that the MN is attached to when it is away from the home link.
- **Care-of address (CoA):** A CoA is an address used by a MN when it is attached to a foreign link. The association of MN's CoA with its HoA is known as a *binding*. Correspondent nodes and HAs keep information of bindings in binding caches.
- **Correspondent node (CN):** A CN is a node with which the MN is communicating. The CN must be IPv6 enabled in order to maintain communication when MN is away from home.

1.3.2 Mobile IPv6 Communications

1.3.2.1 Mobile Node on Home Link

When MN is on its home link, it communicates with the CN in a normal way. It sends packets to the CN using its HoA as source address and the CN's address as destination address. No mobility management is required.

1.3.2.2 Mobile Node on Foreign Link

When MN is on a foreign link, a CoA will be assigned to it. The CN and HA need to keep a binding cache to store the {HoA, CoA} association of the MN. The MN requests to create or update the binding cache entry on the CN and HA by sending *Binding Update* (BU) packets to them, or by embedding the BU Option in the IPv6 header of data packets. The CN and HA send back *Binding Acknowledgments* (BAs) to MN when binding cache entries are successfully updated.

In brief, when the MN on a foreign link communicates with the CN, it sends data packet to the CN using its CoA as source address, includes its HoA in the Home Address Option field of the IPv6 header, and set the Binding Update Option field of the header. When the packet is received at the destination, the CN first processes the header of the packets. Since the Binding Update Option is present, the CN creates or updates the binding cache entry that reflects the new {HoA, CoA} association of the MN. Before passing the payload to the upper layer protocol, it logically replaces the source address of the packet with the address in the Home Address Option. As far as the upper layer protocols are concerned, the packet was sent from the MN's HoA. Hence, for TCP communication, the on-going sessions are still valid, since the {address, port} pairs haven't been changed. The message sent from MN to CN is illustrated in Figure 1.4.

The MN also needs to send BU packets to its HA so that the HA is aware of the MN's current CoA. This is illustrated in Figure 1.5.

Figure 1.4: Data Sent from MN to CN

When the CN sends a data packet to the MN, the upper layer protocol uses the MN's HoA as destination and passes the packet to IPv6 layer. IPv6 first searches the binding cache according to the destination's HoA. Communication process differs depending on whether a matching entry is found.

- **Case 1:** Binding cache entry exists. If an entry is found to match the MN's HoA, it replaces the destination address with the MN's CoA and puts the MN's HoA in the Routing Header field of the packet. When the MN receives the packet, it logically replaces the CoA with the HoA as the destination address in the IPv6 header. When the packet is passed to upper layer protocol, it appears to have been sent to the MN's HoA. The message sent from CN to MN is shown in Figure 1.6.

Figure 1.5: BU Sent from MN to HA

- **Case 2:** Binding cache entry does not exist. If no matching entry is found, it indicates that the CN is not aware of the MN's CoA. This case happens when the CN initiates the communication with the MN while no BUs from the MN were previously received; or when the binding cache entry on the CN expired. Since the MN's CoA is not available, the packet is sent to the MN's HoA. While addressed to the MN's HoA (represented by the Virtual Mobile Node in Figure 1.3), the packet is routed to the MN's home link according to the prefix of the HoA. This is shown in Figure 1.7.

When the packet is routed to the home link, the HA, which has a binding cache entry for the MN, intercepts the packet and forwards it to the MN by encapsulating the IPv6 packet with an additional IPv6 header. This is known as IPv6-over-IPv6

CHAPTER 1. INTRODUCTION

Figure 1.6: Data from CN to MN with Binding Cache Entry Present

tunneling (see Figure 1.8).

In the tunneled packet, there are two layers of IPv6 headers. The outer IPv6 header indicates that the packet was sent from the HA to the MN, while the encapsulated packet was the actual packet sent from the CN to the MN's HoA, which is indicated by the inner IPv6 header. When the MN receives the packet, it knows that the CN is not aware of its current CoA, so it schedules a BU that will be sent to the CN later. The IPv6 layer removes the outer IPv6 header and passes the packet to upper layer protocol. In the upper layer protocol's perspective, the data was addressed from the CN's address to the MN's HoA. Later when the MN sends the next data packet to the CN, it includes the Binding Update Option in the header so that the CN will update its binding cache. After the binding cache on the CN is

Figure 1.7: Data from CN to HoA of MN

updated, subsequent packets can be sent directly from the CN to the MN's CoA, without the interaction of HA.

In summary, the HoA is used as a host identifier of the MN, whereas the CoA is used as a location identifier of the MN. Transport layer protocols are only aware of MN's HoA, and they may not know that MN has changed its network location. CoA is only processed at the IPv6 layer, and it is totally transparent to the upper layers.

Figure 1.9 illustrates a typical scenario where the MN is connected to wireless overlay networks with Mobile IPv6 entities.

1.3.3 Mobile IPv6 for Linux

Mobile IPv6 for Linux (MIPL) is an open source implementation of Mobile IPv6 developed at Helsinki University of Technology [14]. MIPL is implemented as a kernel module.

Figure 1.8: Intercepted Packet Tunneled to MN by HA

It is distributed as a patch against the Linux kernel and a set of user space tools. User can configure the network node for CN, MN or HA functionalities. The protocol layers with MIPL is shown in Figure 1.10.

MIPL does not require any modification to IPv6 applications, and it is independent of link layer access technology.

1.3.4 Why Choosing Mobile IPv6

Besides Mobile IPv6, there are other protocols for communication mobility support. Compared with those protocols, Mobile IPv6 has the following advantages:

- **Route optimization.** Most mobility management protocols require proxy-based

Figure 1.9: Mobile IPv6 and Wireless Overlay Networks

communications. Each packet must be sent to a proxy before being transmitted to the actual destination. In Mobile IPv6, route optimization enables direct communications between MN and CN. Packets only need to go through HA in certain cases.

- **Promising to become standard.** Most of the mobility management protocols are still Internet drafts. Mobile IPv6 is promising to become the future standard for communications mobility, since it is one of the components of IPv6 protocol. IPv6 being the next generation Internet layer protocol, there is little doubt that it will eventually replace IPv4 in the future.
- **Available implementation.** Some of the mobility management protocols are still paper work, without actual implementation available. Some protocols have been implemented, but the packages are not easily obtainable (e.g., some are not

Application Layer	Applications	
Transport Layer	UDP/RTP/TCP	
Network Layer	Mobile IPv6	FreeS/WAN IPSec
	Linux IPv6 Stack	
Link Layer	802.11/Bluetooth/GPRS	

Figure 1.10: Protocol Layers

open source), or the development is not continuously being supported. However, Mobile IPv6 has become a module in some IPv6 enabled operating systems, such as Windows Server 2003, and the open-source Linux. Mobile IPv6 is even available in the Familiar Linux distribution [15], which is an embedded Linux operation system developed for Compaq iPAQ PDA. Mobile IPv6 for Linux is still an active project under continuous development. With the module ready, it greatly eases the system development process.

1.4 Project Scope

The M.Eng program includes the following three projects:

- **Vertical Handoff Management System.** The objective of this project is to design and develop a multiple network interfaces management system to facilitate seamless vertical handoff among different link layer technologies. The system is responsible for collecting information on the number of available networks and the performance of each network, making decision on selecting the most suitable interface at any time, and triggering handoff when necessary.

CHAPTER 1. INTRODUCTION

The system is implemented on Linux operating system, with MIPL patched kernel. Real network measurements are conducted in test bed to evaluate the performance of network handoff.

- **Load Sharing in Multi-homed System.** The bandwidth provided by a single wireless network is normally small and may not be sufficient to meet the user applications' requirements. If the MN is equipped with multiple network interfaces, then it is possible to utilize all the available access networks simultaneously to achieve a larger aggregate bandwidth. When data traffic is to be sent and received through multiple networks, a load sharing system is required to control which packet should be transmitted via which network interface. This project is to define policies for mapping data flows to multiple interfaces, and design algorithm to properly distribute the packets to multiple network paths. The system should ensure balanced traffic distribution among different network interfaces, and at the same time minimize the probability of out-of-sequence packet reception.
- **Improving TCP Performance during Soft Vertical Handoff.** Although Mobile IPv6 can ensure that on-going TCP sessions will not get broken in case of vertical handoff, TCP performance will still get affected due to sudden change of network characteristics, such as available bandwidth and propagation delay. In this project, we identify the problems that TCP may encounter during a soft vertical handoff, and we propose a number of network-layer schemes to resolve the TCP timeout problem caused by increased network round-trip delay when handing over from a fast link to a slow link.

1.5 Thesis Organization

In the remaining parts of the thesis, Chapter 2 presents the design and implementation of the vertical handoff management system. Chapter 3 discusses the problem issues of simultaneous access to multiple network interfaces, and proposes a packet distribution mechanism to achieve bandwidth aggregation. Chapter 4 describes TCP performance issues during vertical handoff, and presents our proposed schemes to resolve the problem of spurious timeout. Finally, Chapter 5 summarizes the thesis.

Chapter 2

Vertical Handoff Management System

2.1 Literature Review

The concept of vertical handoff is first defined in [9]. The authors also define the architecture of wireless overlay networks, where different networks with various ranges of coverage form a hierarchical wireless network environment. Handoff from a smaller coverage network (in lower overlay) to a wider coverage network (in higher overlay) is called upward vertical handoff. Downward vertical handoff is in the other way round. Upward handoff is initiated when a certain number of beacons from the current network are not received. Downward handoff is initiated when several beacons are heard from the lower overlay network. In their proposed system, the CoA of MN is a multicast address. A group of APs are selected to listen to the CoA's multicast group. One primary AP is forwarding packets to MN, other listening APs maintain buffers MN's packets. When MN is undergoing network handoff, the multiple listening APs keep receiving packets which are targeted to MN, and they send the buffers to MN so that the probability of

CHAPTER 2. VERTICAL HANDOFF MANAGEMENT SYSTEM

MN missing any data during handoff is reduced. Several enhancement mechanisms for reducing handoff latency are also proposed in the paper, such as fast beaconing, packet double casting and header double casting. Their system heavily relies on infrastructure, and it needs multicast support, requiring multiple APs to listen and buffer packets for MN, which increases complexity for practical deployment.

The authors in [16] propose the P-Handoff protocol to complement vertical handoff. The protocol deals with ad-hoc links in peer-to-peer basis, where no infrastructure is required. HOPOVER [17] is a protocol designed to reduce the gaps in packet flows during handoff. This protocol aims at real-time applications that require resource reservation (e.g., via RSVP). HOPOVER defines a scheme to pre-reserve resource in the new path prior to handoff initiation. Before handoff, MN sends Handoff-Prepare messages to a group of APs in the target network so that the target APs can reserve resource and buffer packets for the MN. When handoff is triggered, MN sends a Handoff message to a selected AP, the new AP then start forwarding packets to MN, and the other APs release resource for MN. Additional handoff-preparation steps are required in this protocol, it may fail if handoff is urgent or non-predictable.

There are a number of researches that address the problem of when the MN should initiate handoff. The authors in [18, 19] propose an algorithm for predicting handoff from IEEE 802.11b WLAN to GPRS. When MN moves to the edge of WLAN, it takes samples of beacon signal strength from AP and compares them with a threshold. During a predefined “dwell time” period, if all consecutive samples are below the threshold, then MN initiates handoff to GPRS. The authors in [20] propose a similar algorithm that uses a threshold to classify weak beacon signals. If a predefined number of consecutive signal strengths are lower than the threshold, handoff will be triggered. The authors in [21] propose a mobility management system to accomplish seamless handoff between WWAN and WLAN. Handoff decision relies on Fast Fourier Transform (FFT) based

decay detection. When MN is on the edge of WLAN, and the signal strength is below a threshold $S1$, the system calculates the FFT fundamental term of the most recent N samples of signal strength $x(n)$. If the imaginary part of the fundamental term is negative, then the signal sequence $x(n)$ is statistically decreasing, and handoff will be performed when signal strength goes below another threshold $S2$. The system also introduces the MAC Layer Sensing mechanism that collects the Network Allocation Vector (NAV) in the MAC layer to estimate network conditions such as available bandwidth and access delay. The information is then used for network interface selection. The authors in [22] propose to use fuzzy logic to make handoff decision based on signal strength, however, the computation complexity is too high for resource limited mobile devices.

2.2 Design of Vertical Handoff Management System

2.2.1 Goals

Followings are the goals that our handoff management system is supposed to achieve.

Seamlessness: When the MN needs to switch from one network to another, the handoff process should cause no interruption on MN's current sessions, and the applications should not be affected. Seamlessness heavily depends on the handoff latency. Although in certain cases, handoff latency cannot be reduced to zero, the system should minimize the undesirable effect to the applications as much as possible.

Transparency: The various link layers should be transparent to the user and applications. Applications need not be worried about handling multiple network interfaces separately. Instead, applications should perceive the system as a single integrated connectivity. All network management operations are hidden from the upper layers, so that no interference from the user and applications are necessary.

CHAPTER 2. VERTICAL HANDOFF MANAGEMENT SYSTEM

Optimal network selection: The system should choose the best network to satisfy the requirements of active applications as well as to meet the various constraints, such as power consumption, cost, etc. Some networks provide high bandwidth but consume considerable power, while some provide low bandwidth but consume less power. The system should find the balance and make the optimal decision to take advantage of the wireless diversity.

2.2.2 Architectural Design

The vertical handoff management system consists of six functional modules and two informational databases (See Figure 2.1). Each component will be described in details in the following subsections.

CHAPTER 2. VERTICAL HANDOFF MANAGEMENT SYSTEM

Figure 2.1: Architectural Design

CHAPTER 2. VERTICAL HANDOFF MANAGEMENT SYSTEM

2.2.2.1 Network Interface Profile Database

Different network interfaces have different characteristics. Each type of interface needs a separate profile, which represents the performance specifications of the interface. A profile consists of several parameters that describe the characteristics of the corresponding type of interface. These parameters include:

- **Power consumption (P).** Amount of battery energy that the device consumes per unit of time when it is activated and operating, normally in terms of mW.
- **Available bandwidth (B).** Current “idle” bandwidth available on the network that the interface is connected to.
- **Maximum capacity (M).** Maximum bandwidth that the corresponding network can provide.
- **Signal strength (S).** Strength of radio signal that the device receives from the network.
- **Packet lost ratio (L).** Percentage of lost packets on the networks. This provides useful information for handoff prediction.
- **Usage charge (C).** Some wireless networks are free for use; some impose usage charge based on the time of connection or the amount of data traffic.

Among these parameters, some are static information, which do not change with time, for example, the power consumption and maximum capacity. Some are dynamic components, which will be updated at runtime, for example, the available bandwidth and signal strength.

The network interface profiles provide essential information for network selection decision and handoff prediction.

CHAPTER 2. VERTICAL HANDOFF MANAGEMENT SYSTEM

2.2.2.2 Policy Database

In the proposed system, interface selection is based on a cost-function policy model. The cost of using a network n at a certain time can be specified by a function as:

$$f(n) = w_b \times \frac{1}{B_n} + w_p \times P_n + w_c \times C_n \quad (2.1)$$

Parameters B_n, P_n, C_n are retrieved from the Network Interface Profile Database. The coefficients w_b, w_p and w_c are the weights, i.e., the degree of importance, associated with the respective parameters. These weights are stored in the Policy Database. These coefficients can be specified by the user, or predefined with default values. Some coefficients are dynamically updated at runtime, for example, w_p , which may increase as the battery energy level is going low.

2.2.2.3 User Control Panel

This is a simple user-space tool for the user to specify certain parameters in Network Interface Profile Database and Policy Database. For example, the user can specify the usage charge (C) in the profile database, and w_b, w_p, w_c in Policy Database. This configuration is a one-time operation, so that user involvement is minimal. After the initial specification, interface management is automatic and no user interaction is necessary.

2.2.2.4 System Monitor

This module measures the system performance in various aspects. It consists of the following sub-modules:

- **Power monitor.** It monitors the battery power level of the MN. Power consumption is a critical issue for mobile devices. When the power is running low, the network selection policy may need adjustment. This adjustment can be reflected

by changing w_p in the Policy Database based on the current battery status. When battery energy is running low, w_p will be increased so that the power consumption of a network device has a greater contribution to its cost function, and thus becomes the dominant component that determines the network selection.

- **Network traffic monitor.** This sub-module monitors the data rate (R) of network traffic needed by the running applications. This value can be used to dynamically update the parameter w_b in the Policy Database. When the data rate required by the applications is high, w_b will be increased so that the bandwidth of the network becomes the major consideration when selecting network interface.
- **Available bandwidth monitor.** There are various methods of measuring the available bandwidth of a network. Some requires sending probing messages, so that bandwidth can be estimated based on round-trip time and inter-arrival variance [23]. Alternatively, the MAC layer sensing mechanism [21] estimates bandwidth based on link layer information, and no probing packets are injected to the network. The bandwidth measurement result is used to update parameter B in the profile database.

2.2.2.5 Network Discoverer

This module discovers the available networks around the MN. Basically, the module periodically turns on the idle network devices, searches for APs and requests for router advertisements. If new networks are detected, the Interface Manager module will be informed to make a handoff decision. If handoff is desired, the preferred interface will be used as the active interface, while the previous interface is turned off. If no handoff is necessary, the non-current devices are turned off; the collected network information (such as the address of APs and routers) is stored in database for later use, in case handoff is required in the future.

2.2.2.6 Handoff Predictor

Handoff prediction is especially important when the MN is moving. If the MN does not perform handoff before it moves out of the coverage area of currently connected network, the connection will be broken. The Handoff Predictor module is to detect the tendency of connection loss in the current network, and inform the Interface Manager module to perform handoff in advance. Such a tendency can be detected based on various hints from the current network status. For example, handoff is desired if current signal strength (S) is continuously dropping, or the packet lost ratio (L) exceeds a threshold value.

2.2.2.7 Interface Manager

This is the central module for making decision on interface selection and performing handoff operation. Decision is mainly based on the parameters retrieved from Network Interface Profile Database and Policy Database. The cost function (Eq. (2.1)) previously described is used to calculate the cost of using a network n . The lower the cost function value $f(n)$, the better network n is. If the cost value of a new network n is lower than the cost value of current network by a threshold percentage (say, 5%), the system should handoff to network n . Handoff can also be triggered by the Handoff Predictor module. When a handoff command is received from the predictor module, the Interface Manager module should perform handoff even if the current network has the lowest cost value, since the current network may become unreliable soon.

2.3 Implementation and Experimental Results

2.3.1 Testbed Setup

The system is based on MIPL v1.1 for Linux kernel 2.4.26 [24]. In system implementation, the MIPL module is modified to allow more control on interface selection and handoff

CHAPTER 2. VERTICAL HANDOFF MANAGEMENT SYSTEM

management. The Interface Manager is implemented as a Linux kernel module, so that it can interface with the kernel space functions exported by MIPL module. The Network Interface Profile Database and Policy Database are stored as text files for simplicity. The Interface Manager kernel module also acts as a driver for a character device, so that the user-space tool can pass the information in the databases to the kernel space via the Linux “ioctl” mechanism. The Network Discoverer component is currently implemented as a Linux shell script so that it can manage network interface status and events using user space commands provided by various Linux utilities.

To simplify the experiment, the Mobile IPv6 testbed is setup using two laptop PCs (Laptop 1 and Laptop 2). Laptop 1 acts as the following entities in Mobile IPv6 communications:

- HA: Essential entity in Mobile IPv6 for processing registration messages from MN.
- CN: The communication end point talking with MN.
- Router: Broadcasting IPv6 Router Advertisements to MN for network discovery and handoff operation.
- AP: The laptop is equipped with both Bluetooth and IEEE 802.11b network interfaces so that it can act as AP to establish connections with various network interfaces on MN.

Laptop 1 also has a wired connection to the campus LAN, which can accept packets from the Internet. This is necessary when the MN is using GPRS to communicate with Laptop 1.

Laptop 2 acts as MN with built-in Bluetooth and IEEE 802.11b interfaces. An external GPRS PCMCIA network card is also installed on the laptop.

CHAPTER 2. VERTICAL HANDOFF MANAGEMENT SYSTEM

To use the Bluetooth or IEEE 802.11b interface for communication, the MN simply connects to Laptop 1 directly via the respective interface, since Laptop 1 is the AP for both access networks. The use of GPRS adds complexity to the communications. Using GPRS, MN needs to dial-up to the external ISP (outside campus) via the GPRS card. When connection is established, packets from MN are sent to the GPRS AP, then routed to the Internet and finally arrive at the Ethernet card of the CN (i.e., Laptop 1). Packets from CN to MN go the other way round. There are several problems with such a communication scenario:

- The GPRS router and the Internet is IPv4 based, to transmit IPv6 packets through the IPv4 backbone, IPv6-to-IPv4 tunneling mechanism is required.
- There are firewall protections in both the GPRS network and the campus LAN, only some common types of packets, e.g., TCP and UDP, can go through the firewall. In the experiment, the standard IPv6-to-IPv4 tunneling packets from MN or CN failed to arrive at the other end-point. The packets are filtered by the firewall in GPRS network or the one in campus LAN.
- The MN does not have a public IP address, it can only obtain a private IP address from the DHCP server of the GPRS network. That is, Network Address Translation (NAT) policy is applied in the GPRS network. Every time when the MN sends a packet, the NAT server creates a dynamic mapping between the {private address, private port} to a {public address, public port}, and the address fields in IP headers of the outbound packets are modified accordingly before routed to the CN. If the outbound packet belong to a TCP connection, the mapping will expire when the connection close. In the case of UDP, the mapping will expire after a timeout period. Under such NAT-controlled situation, inbound connection from CN to MN will be blocked, since the creation of port mapping can only be initiated by an

CHAPTER 2. VERTICAL HANDOFF MANAGEMENT SYSTEM

outbound packet. In other words, the MN can initiate a connection with CN, but not the other way round. This limits the communication flexibility between MN and CN.

To tackle these problems, a UDP-tunneling scheme is developed. In this scheme, the MN first sends an IPv4 UDP packet to CN to initiate the tunnel; this practically triggers the NAT server to open a public port for the MN. From the UDP packet, the CN gets aware of the public IP address and the port number of the tunnel endpoint. After tunnel establishment, the endpoints can transmit IPv6 packets through the tunnel in both directions. Instead of using normal IPv6-to-IPv4 tunneling, the IPv6 packets are encapsulated in IPv4 UDP datagrams so that they can pass through the firewalls in GPRS network and campus LAN. The datagrams are decapsulated by the tunnel daemon at receiver and the original IPv6 packets are passed to the IPv6 stack in the Linux kernel. When the tunnel is idle (no IPv6 communication), the MN needs to periodically send dummy UDP packets to prevent the NAT port mapping from expiring, and thus keeping the tunnel active, so that CN can initiate IPv6 connection with MN at any time.

Such a tunneling scheme is also implemented on iPAQ running Linux kernel version 2.4.19-rmk6-pxa1-hh35. However, since the GPRS PCMCIA card does not work on iPAQ, the PDA uses Infrared to connect with a GPRS-enabled hand phone, and the hand phone acts as a modem to dial-up to the GPRS ISP.

Experiments of vertical handoff among GPRS, IEEE 802.11b and Bluetooth have been conducted using such a testbed. The following subsections present the results of handoff from Bluetooth to GPRS and from GPRS to Bluetooth. The Bluetooth devices used in the experiments can support data rate of up to 433.9 kbps. The GPRS service operator provides network data rate of up to 85.6 kbps.

2.3.2 Handoff from Bluetooth to GPRS

In the first experiment, CN and MN first pair up their Bluetooth interfaces to establish a PAN connection. After that, CN initiates a TCP session with MN and starts sending data through the Bluetooth connection. During the TCP session, we slowly move the MN away from the CN, so that the signal strength detected by the MN drops below threshold (which is set to 50) and the handoff management system automatically initiates handoff to GPRS. On MN, network activities are constantly monitored using Ethereal [25] packet sniffer and analyzer. Figure 2.2 shows the results of the received TCP sequence numbers versus time in vertical handoff from Bluetooth to GPRS.

Figure 2.2: Performance of Handoff from Bluetooth to GPRS

In the figure, the slope of the graph reflects the throughput of data transmission. Vertical handoff occurs at time 7.42 during an IPv6 TCP session, where CN is the sender

and MN is the receiver. The throughput declines sharply after switching to the low-speed GPRS network. There is a short idle period right after the handoff occurrence. The duration of the period represents the handoff latency, which is the time between the reception of the last packet from Bluetooth network and the first packet from GPRS network.

2.3.3 Handoff from GPRS to Bluetooth

Figure 2.3 shows the results of vertical handoff from GPRS to Bluetooth. In this experiment, the Bluetooth and Wi-Fi devices of the AP are initially off, so that the MN can only use GPRS for communications. During a TCP session, we turn on the Bluetooth device of the AP. The handoff management system on the MN triggers vertical handoff upon detection of Bluetooth network since Bluetooth is given a higher priority.

The data throughput greatly improves after handoff. However, a few packets are out of sequence right after switching to Bluetooth. The out-of-sequence reception is due to the bandwidth and delay disparity between the two different access networks. When the MN receives the first packet from the fast-speed Bluetooth network, there are still a few earlier packets being transmitted on the long-delay GPRS link. There are two groups of out-of-sequence packets: group A and group C as denoted in Figure 2.3. Packets in group A are the retransmitted packets sent through Bluetooth, which are enforced by TCP congestion control. Retransmission of packets is caused by the early arrival of group C packets, which are sent on the fast link and result in multiple duplicate acknowledgments sent by the receiver. Packets in group B are the last few packets transmitted on the GPRS network. Actually these packets are discarded by the receiver, since the same sequence numbers have already been retransmitted (as group A packets) on Bluetooth before they arrive through GPRS.

Figure 2.3: Performance of Handoff from GPRS to Bluetooth

2.4 Summary

In this chapter, we present the design of our vertical handoff management system. The system is responsible for making handoff decision based on a cost function value and the prediction of possible connection loss. The value of cost function is the weighted sum of available bandwidth, power consumption and usage charge. The prediction of connection loss is mainly based on the signal strength detected on the current network. The system is implemented on Linux platform. Testbed has been setup to conduct vertical handoff between different wireless networks, such as GPRS, IEEE 802.11b and Bluetooth. Experimental results demonstrate that the system can manage vertical handoff successfully without breaking on-going TCP sessions.

Chapter 3

Load Sharing in Multi-homed System

3.1 Overview of Multi-homed Environment

The previous chapter focuses on vertical handoff from one access network to another, keeping only one interface active most of the time. Only one CoA is registered with HA and CN, and network traffic is directed to the single CoA after the handoff procedures are completed.

In some situations, it may be desirable to activate multiple network interfaces simultaneously to achieve better Quality of Service (QoS). The MN can maintain connectivity with multiple access networks, send different packets through different interfaces, and receive traffic from multiple network channels. In such a case, the MN becomes *multi-homed*. Splitting data traffic over multiple network links is known as *load sharing* or *packet striping* [26, 27, 28].

A multi-homed MN may need to have multiple CoAs bound to a single HoA, each CoA is associated with one network interface. A typical benefit of using multiple interfaces

CHAPTER 3. LOAD SHARING IN MULTI-HOMED SYSTEM

simultaneously is bandwidth aggregation. For example, if the access network for interface *iface1* can provide bandwidth $bw1$, and the network for *iface2* can provide bandwidth $bw2$, then it is possible to achieve an aggregate bandwidth of $bw1 + bw2$ when both *iface1* and *iface2* are active. Such a bandwidth increase is especially noticeable when a single access network cannot provide sufficient resource to meet the requirement of network applications on MN.

However, performance gain does not come for free. It is obvious that keeping multiple interfaces turned on at the same time results in greater power consumption, which shortens the battery lifetime of the MN. Moreover, keeping connected to multiple access networks induces greater usage charges if those networks do not provide free services. Whether to activate single interface or multiple interfaces depends on various factors:

- QoS characteristics of the available access networks
- Network data rate of currently running applications
- Power consumption of network interfaces
- Current battery status of mobile device
- Usage charge of the available access networks
- Billing budget affordable by MN user

The decision to switch between single interface mode and multi-homed mode can be made by the MN user, or automatically driven by policies which take the above factors into consideration.

For the MN, an outgoing packet is directed to a certain network interface by specifying the interface's CoA as the source address in the IPv6 header. If the MN specifies different source addresses for different packets, the packets will be sent through different access networks, in this way, load sharing for outbound traffic is achieved.

CHAPTER 3. LOAD SHARING IN MULTI-HOMED SYSTEM

To achieve load sharing for MN's inbound traffic, CNs need to specify different CoA as destination address for different packets, so that the packets will be routed to MN's different access networks accordingly. This will only be possible if multiple CoAs are registered with CNs.

In multi-homed environment, data traffic needs to be carefully distributed to different network interfaces, otherwise the overall performance may be even worse than using a single interface. Different network links have different bandwidth and delay. If data packets are not distributed properly, it may result in one network link being congested while other link being under-utilized. Another problem in load sharing is out-of-sequence packet reception due to unequal delays of different links.

There are various schemes for traffic distribution over multiple interfaces. These schemes basically fall into three major categories: per-host distribution, per-flow distribution and per-packet distribution.

3.1.1 Per-host Distribution

In this type of schemes, one CN is associated with only one interface of the MN. When communicating with multiple CNs, different CN may be associated with different network interface. For example, MN can send a BU to *CN1* to register *CoA1* with it, and send a different BU to register *CoA2* with *CN2*, etc. When there are several flows running between MN and *CN1*, all these flows will be directed to one network interface, which is bound to *CoA1*. Similarly, all flows to *CN2* will be routed through *CoA2*. Figure 3.1 illustrates such a scenario.

The Mobile IPv6 stack on the MN needs to be modified to have multiple CoAs bound to a single HoA, and a mapping between CNs and CoAs must be maintained. The number of table entries equals to the number of CNs. Table 3.1 is an example of such a mapping table on the MN.

Figure 3.1: Per-host Traffic Distribution

Destination Address	Address Mask	CoA
2000:a3bd:e287::	64	2000:1111::c1ee:b308
2000:3333::	64	2000:3333::a4b2:3e91
2000:3a56::	64	2000:3333::a4b2:3e91
2000:5e34:7d24:19e2::202	128	2000:2222::2e43:97e2
Default	0	2000:1111::c1ee:b308

Table 3.1: Example of Host-CoA Mapping Table

The Mobile IPv6 stack of the CN requires no modification, since the MN registers only one CoA with a single CN. From the CN’s point of view, the MN is using only one active interface.

In such schemes, bandwidth aggregation can only be achieved if MN is communicating with multiple CNs. If MN is communicating with only one CN, then multiple interfaces are not in use simultaneously, hence there is no performance gain.

3.1.2 Per-flow Distribution

Traffic distribution with higher granularity aims at flow level. Per-host distribution schemes map different CNs to different network interfaces, whereas per-flow distribution schemes map different flows to different network interfaces. When there are multiple flows between MN and a single CN, these flows can be routed through multiple interfaces. Figure 3.2 illustrates such a scenario.

Figure 3.2: Per-flow Traffic Distribution

In such schemes, flow characteristics are taken into consideration during traffic distribution. Different flows may have different priority and different QoS requirements. For each flow, a proper interface is selected, whose network characteristics match the flow characteristics. For example, in video streaming applications, if adaptive multi-layered encoding is used, video streams are separated into base layer and enhancement layers. With support for multi-homing, base layer stream can be sent through the most reliable access network since it is more essential for decoding, whereas enhancement layers can be sent through less reliable access networks due to lower priority. Another example is

CHAPTER 3. LOAD SHARING IN MULTI-HOMED SYSTEM

video conferencing application, where network traffic is separated into two parts: voice and video. Voice data requires lower data rate, so it can be sent through narrow-band network, such as GPRS. Video flow requires much higher bandwidth, and it needs to be routed through faster network like IEEE 802.11b WLAN.

The Mobile IPv6 stack of both MN and CN need modifications. The mapping between flows and CoAs must be stored in table on both MN and CN. Flows can be identified by either {destination address, port number, protocol} triplet or the flow label in IPv6 header. Table 3.2 and 3.3 are examples of mapping tables on MN and CN.

Flow	CoA
Addr=2000:e3d2::3e44:23d1 Port=80 Proto=TCP	2000:1111::c1ee:b308
Addr=2000:e3d2::3e44:23d1 Port=2048 Proto=UDP	2000:3333::a4b2:3e91
Flow label=1	2000:2222::2e43:97e2
Default	2000:1111::c1ee:b308

Table 3.2: Example of Flow-CoA Mapping Table on MN

HoA	Flow	CoA
2000::202	Port=32178 Proto=TCP	2000:1111::c1ee:b308
2000::202	Port=24819 Proto=UDP	2000:3333::a4b2:3e91
2002:1e3a:b2dd::e112:6ac3	Flow label=4	2000:4698:a7d1::2e4e:a827

Table 3.3: Example of Flow-CoA Mapping Table on CN

3.1.3 Per-packet Distribution

Per-packet distribution schemes have the highest granularity level among the three categories. In these schemes, even packets of a single flow can be sent or received via different network interfaces. Since traffic is distributed on packet level, packet scheduling

CHAPTER 3. LOAD SHARING IN MULTI-HOMED SYSTEM

and multiplexing algorithms are employed to place outgoing packets on multiple network channels in appropriate order.

One of the advantages for distributing traffic at packet level is that if the packets within a same flow have different degree of importance, they can be directed to different network interfaces based on their priorities. For example, when sending an MPEG video stream, I-frames, B-frames and P-frames are transmitted in sequence. I-frames are anchor frames, which are most important for decoding, so they should be sent through the most reliable access network. B-frames and P-frames are predicted frames, which have lower priority, hence they can be sent through a less reliable network.

MN may specify the portion of traffic that should be handled by each interface, so that different amount of packets are assigned to different interfaces based on various characteristics of access networks. How to schedule packets on multiple network interfaces efficiently and effectively is not an easy task. Per-packet distribution is the most flexible, but most complicated scheme.

If packet distribution is not carried out in a proper manner, the overall performance of using multiple network interfaces may be even worse than using a single network interface. One of the major problems for transmitting packets through multiple access networks is out-of-sequence packet reception due to disparity of network bandwidth and propagation delay. For example, a flow contains four packets, packet 1 is sent through interface *iface1* first, which has long propagation delay; packets 2, 3 and 4 are sent through interface *iface2* next, which has much shorter delay than *iface1*. Although these packets are transmitted in FIFO order at the sender side, the receiver will receive packets 2, 3 and 4 before getting packet 1 due to different transmission delays.

For connection oriented transport protocol, like TCP, the delayed packets being transmitted on slow channel will keep the earlier-arrived packets waiting in receiver's buffer until the late packets arrive to fill up the gaps. The out-of-sequence delivery may cause

unnecessary congestion control actions in TCP, like shrinkage of window size, slow start, etc. For connectionless communication, e.g., multimedia streaming via RTP, the late packets on slow channel may be dropped, which results in degraded audio or video quality.

3.2 Literature Review

There are a number of protocols and architectures proposed to support multi-homed communications. Some of them require mobility awareness from user applications, and multiple network connections are managed at user space. Some make enhancement on transport layer to address the problems faced by conventional transport protocol like TCP in multi-homed environment. Some focus on network layer to define mechanisms for striping and routing data packets to multiple network interfaces. Transport layer and network layer solutions can be transparent to user space applications, so that legacy applications can benefit from multi-homed mobility without modification, hence they are more generic approaches.

The common goal of transport layer techniques is to avoid improper congestion control in multi-path transmission. In [29], an end-to-end transport layer protocol called pTCP is proposed to decouple congestion control and reliability management. In their design, a virtual pipe called TCP-v is opened for each interface connection, and each TCP-v maintains its own congestion window. The wrapper pTCP is responsible for assigning data packets to appropriate TCP-v based on congestion status. R-MTP is proposed in [30], which is able to adapt to changes in bandwidth of each access network by changing the fraction of flow to be sent to each interface. In addition, R-MTP uses the packet round-trip time and packet inter-arrival time to distinguish between congestion losses and transmission errors. A similar protocol MMTP is proposed in [31], which is governed by

CHAPTER 3. LOAD SHARING IN MULTI-HOMED SYSTEM

two mechanisms: a set of rate control protocols associated with each outgoing channel, and a scheduling algorithm that places incoming packets on the appropriate channel. SCTP is introduced in [32, 33], which is a stream control transmission protocol operating on top of the network layer where both endpoints may have multiple IP addresses. A host has one primary address and may have zero or more alternative addresses. The primary address is used by peer hosts for normal data transmission, while the alternative addresses are used for retransmitted packets to improve reliability. In SCTP, multi-homing is used only for redundancy in the presence of network failures, whereas load sharing and bandwidth aggregation are not supported.

Network layer techniques for multi-homed mobility mostly involved algorithms for packet striping, which assign packets to appropriate channels in proper order. The authors in [27] proposes an architecture for packet striping protocols based on the ideas of fair load sharing and logical FIFO reception. In this architecture, fair load sharing algorithm is transformed from fair queuing algorithm. At the sender side, a causal fair queuing algorithm is employed to schedule the outgoing packets from multiple input queues into a single output queue; packets from the single queue are then assigned to multiple network interfaces via load sharing algorithm. At the receiver side, the same causal fair queuing algorithm is used to accept packets from multiple channels in order and place them onto a single queue, which will later be processed by upper layer protocols. The limitation of such architecture is that the sender and receiver must use the same causal fair queuing algorithm, and they must maintain the same state by periodically sending marker packets for synchronization, which is not flexible enough. The authors in [34] introduce another scheme for using multiple network interfaces, which employs earliest-delivery-first algorithm to select the network interface for a given packet that can minimize the expected arrival time of the packet. The authors in [28] propose a packet striping algorithm, which applies advanced sending at the sender side to limit the

effects of different delays on channels, however, the bandwidth disparity of channels are not taken into consideration, and the traffic may not be fairly shared among the multiple channels.

Most works focus on supporting multiple interfaces in transmission, i.e., how to send out packets through multiple access networks at the MN side. However, there are fewer works that address the problem of using multiple interfaces for reception, i.e., how the CN gets aware of the multi-homed status of the MN, and send packets to MN's multiple interfaces.

3.3 Proposed Load Sharing System

In this research, we propose a network-layer load sharing system that can utilize multiple network interfaces in both reception and transmission.

MN is a non-stationary host, usually with limited CPU resource and battery power. In most cases, MN acts as a client in network communications, retrieving data from servers. In other words, MN is usually the data receiver, while CN (the server) is the sender. In such scenarios, most of the network traffic goes from CN to MN, instead of the other way round. Since the amount of traffic sent by the MN is far less than the amount from CN, how the MN distributes its packets through multiple network interfaces may not have much effect on the aggregate bandwidth gain. What makes a difference is how the CN directs its network flows to multiple access networks that the MN is connected to. This is the problem we aim to address in this research.

The principle of our design is that both MN and CN should be multi-home aware. A data flow targeted to MN can be received via a single interface, or a set of selected interfaces. Hence it is necessary to establish a mapping between data flows and network interfaces. The policies of mapping different data flows to different access networks are

applied on MN based on the network characteristics. The mappings are then sent to CN so that CN becomes aware of the multi-homed status of MN, and CN can distribute its data flows to different network interfaces of MN according to the mappings. Section. 3.3.1 describes how this can be achieved.

A load sharing algorithm is proposed to determine how the CN sends its network traffic to the MN's multiple interfaces, so that CN can distribute packets among different channels in a fair way, and at the same time minimize the chance of out-of-sequence packet reception at MN. The details of the algorithm are described in Section. 3.3.2.

3.3.1 Mapping Data Flows to Multiple Interfaces

Selecting a suitable set of network interfaces to receive packets of a certain data flow can be based on various factors. In our design, we use a simple policy that the mapping is based on data rate requirement, since the major goal of our scheme is to achieve bandwidth aggregation in resource-limited wireless network environment.

At MN, when a new flow is being initiated, the multi-homed system first selects one or more available network interfaces whose bandwidth combination meets the data rate requirement of the flow. There are three possible cases in interface selection:

- There is one interface whose available bandwidth alone can meet the flow's requirement. In such a case, the flow will be mapped to that single interface.
- There are several interfaces, each of which can satisfy the flow's data rate requirement alone. Among the qualified interfaces, the one with the lowest bandwidth will be selected for carrying the traffic of the flow, so that the higher-speed interfaces are reserved for data flows that are more bandwidth-hungry.

CHAPTER 3. LOAD SHARING IN MULTI-HOMED SYSTEM

- No interface meets the bandwidth requirement alone. When the expected data rate of the flow exceeds the available bandwidth of any single interface, a set of interfaces will be selected, whose aggregate bandwidth can meet the requirement.

The following algorithm describes how to select the set of suitable interfaces for a given data flow.

CHAPTER 3. LOAD SHARING IN MULTI-HOMED SYSTEM

Symbols description:

M : The set of interfaces that are mapped to the data flow

A : The set of available interfaces that are not yet selected into M

BW_i : Available bandwidth that can be provided by interface i

DR : Data rate requirement of the given flow

s : Unsatisfied data rate requirement

Algorithm:

$M \leftarrow \phi$;

$s \leftarrow DR$;

while $A \neq \phi$ and $s > 0$ **do**

if $\exists k \in A, BW_k \geq s$ **then**

$j = \operatorname{argmin}_{k \in A} \{BW_k \mid BW_k \geq s\}$;

else

$j = \operatorname{argmax}_{i \in A} BW_i$;

end if

$M \leftarrow M \cup \{j\}$;

$A \leftarrow A \setminus \{j\}$;

$s \leftarrow s - BW_j$;

end while

CHAPTER 3. LOAD SHARING IN MULTI-HOMED SYSTEM

Finally, the interfaces in M are selected for carrying data traffic of the given flow. Such interface selection policy can ensure that a single flow will be mapped to a minimum number of interfaces.

For example, suppose that the MN has four network interfaces: $iface1$, $iface2$, $iface3$ and $iface4$. The currently affordable bandwidths of these interfaces are $1Mbps$, $2Mbps$, $5Mbps$ and $7Mbps$ respectively. If a new flow with bandwidth requirement of $4Mbps$ is being initiated, it will be mapped to $iface3$, since it is the slowest interface that can satisfy the requirement alone. For a flow with data rate of $8Mbps$, the set $\{iface4, iface1\}$ ($8Mbps$ in total) will be selected, instead of $\{iface1, iface2, iface3\}$ which requires one more interface than the minimum set.

The mappings between data flows and interfaces are recalculated if the network conditions change, e.g., an existing interface get disconnected, or a new interface is plugged into the MN. Upon such events, the mapping will be dynamically updated based on the new network conditions.

How to obtain information about the bandwidth requirement of a data flow is up to implementation. Such information might be available if the application explicitly request QoS from the network, e.g., using RSVP for resource reservation. Bandwidth requirement can then be extracted from the application's QoS specification. If the application does not explicitly specify its QoS requirement, we assume that a bandwidth management module on the MN can dynamically monitor the data rates of network flows and provide sufficient information for calculating the mappings between flows and network interfaces.

When the mapping for a flow is established, the MN is ready to distribute its outgoing packets of the flow onto the set of selected interfaces. However, if the role of MN is the data receiver, it is necessary for the MN to "instruct" the sender to distribute the packets to the MN's selected channels. This can be done by embedding the flow-to-interface mapping into a Mobile IPv6 BU packet and send it to the CN. Authors in [35] propose

Option Type	Option Length	Lifetime	Reserved
CoA			
Flow1 ID		Flow2 ID	
Flow3 ID		Flow4 ID	
.....			

Table 3.4: Option Feild of Flow-CoA Mapping in BU

some Mobile IPv6 extensions to register multiple CoAs to be bound to a single HoA. In our project, we propose a different extension such that the information of data flows mapped to each CoA are also embedded in the BU packets. The format of our proposed BU option is specified in Table 3.4.

In the table, the *Lifetime* field specifies the duration within which the mapping will be valid. If the value of *Lifetime* field is 0, the mapping should be deregistered from the CN. A *Flow ID* consists of a source port (CN's port), destination port (MN's port), protocol and flow label. These four sub-fields together identify a single data flow.

Each option field specifies mapping information for one CoA. If multiple CoA mapping information is to be registered, they are stored in separate option fields. Multiple options may contain a same *Flow ID*, which indicates that the specific flow is mapped to multiple CoAs due to high bandwidth requirement.

When CN receives BU containing such type of options, it updates or deletes the specified mapping information in its local cache accordingly. When a packet is to be sent to the MN, CN first searches the cache for a match of *Flow ID*. If a *Flow ID* matching the headers of the packet is found, and if such a *Flow ID* is mapped to a single CoA, then the particular CoA is used as the destination address for this packet. If the flow is mapped to multiple CoAs, a load sharing algorithm (Section 3.3.2) is employed to assign the packets of that flow to different CoAs. If no mapping information for the data flow is found, the packets will be sent to a default CoA of the MN.

3.3.2 Load Sharing Algorithm

At CN, the outgoing packets of a single data flow whose destination is a multi-homed MN form an input queue to a packet scheduler. When the data flow is mapped to multiple CoAs of the destination MN (according to the BU information), each of the associated CoAs can be considered as a virtual output channel (assuming that the CN itself has only one network interface). The packet scheduler is responsible for placing each packet in the input queue of the data flow to an appropriate output channel in a proper order (See Figure 3.3).

Figure 3.3: Packet Scheduler

The proposed packet scheduler involves two functions: $C(s)$ and $P(s)$, where $C(s)$ is the *channel selector* function, $P(s)$ is the *packet selector* function and s is the state variable. When a packet from an input queue is to be sent, $C(s)$ is applied first to choose a channel c for transmitting the next packet, $P(s)$ then picks a packet p from the input queue based on the selected c . The packet p is then put on channel c for transmission. The detailed functionalities of $C(s)$ and $P(s)$ are described as follows.

3.3.2.1 Channel Selector

$C(s)$ selects output channel in a weighted-round-robin manner, which has two advantages:

CHAPTER 3. LOAD SHARING IN MULTI-HOMED SYSTEM

- The computation complexity is low, so that it is simple to implement and does not require much overhead.
- The amount of network traffic handled by each channel is proportional to its available bandwidth, which results in a well balanced packet distribution.

The latter point is further illustrated as follows. The number of times a channel is selected within a single round is determined by the weight associated with that channel. The weights are calculated based on the ratio of different channels' bandwidth. For example, suppose there are three channels: $c1$, $c2$ and $c3$, and their available bandwidths are $200Kbps$, $400Kbps$ and $800Kbps$ respectively, then the weights are $1 : 2 : 4$. In this case, for a single round, $c1$ is selected once, $c2$ is selected twice and $c3$ is selected four times. The order of channel selection is: $c1$, $c2$, $c2$, $c3$, $c3$, $c3$, $c3$. After a complete round, the same order is repeated starting from $c1$. In such a manner, fast channels will transmit more packets to utilize their high bandwidth, whereas slow channel will handle less traffic to avoid congestion.

In practical implementation, applying weighted-round-robin channel selection mainly requires a pointer and a counter. The pointer remembers the last assigned channel, the counter stores the number of packets that has been sent through the last assigned channel in the current round. When the counter reaches the weight of the associated channel, the pointer then proceeds to the next channel. Since such mechanism is straight forward, the overhead of the operations is not significant.

3.3.2.2 Packet Selector

After a channel c is selected, The packet selector $P(s)$ will pick a packet from the input queue to be assigned to the channel. It is not desirable to always pick the first packet at the head of queue, since in-sequence packet transmission at the sender may result in

CHAPTER 3. LOAD SHARING IN MULTI-HOMED SYSTEM

out-of-sequence reception at the receiver due to disparity of network characteristics of the multiple channels. Instead of picking the first packet in the queue, $P(s)$ actually skips a number of packets (say, k) from the head and select the $(k + 1)th$ packet for next transmission. We call it “Jump-Ahead” packet scheduling, since the $(k + 1)th$ packet effectively jumps ahead in the input queue and gets transmitted in advance.

The number of packets to skip depends on the selected channel c . The principle is that the skipped packets are expected to be sent through faster channels and arrived at the receiver earlier than the currently selected packet. The following example briefly describes how to determine the number of packets to skip.

Suppose that a flow is mapped to channel $c1$ and $c2$, where $c2$ is faster than $c1$. The bandwidths and propagation delays for the two channels are bw_1, d_1 and bw_2, d_2 respectively. Assume both channels are currently empty, and all packets have the same size m . Channel $c1$ is selected by $C(s)$ for transmitting the next packet. If the $(k + 1)th$ packet is to be sent through $c1$, then the leading k packets are supposed to be sent through $c2$ later. The time (t_1) taken for the $(k + 1)th$ packet to arrive at receiver via $c1$ is the sum of transmission time ($\frac{m}{bw_1}$) and the link propagation delay:

$$t_1 = \frac{m}{bw_1} + d_1$$

The time (t_2) taken for k packets to finish transmission via $c2$ is:

$$t_2 = \frac{k \cdot m}{bw_2} + d_2$$

To ensure in-sequence reception, $t_2 = t_1$ should be satisfied, that is:

$$\frac{k \cdot m}{bw_2} + d_2 = \frac{m}{bw_1} + d_1$$

Hence,

$$k = \frac{bw_2(d_1 - d_2)}{m} + \frac{bw_2}{bw_1} \quad (3.1)$$

CHAPTER 3. LOAD SHARING IN MULTI-HOMED SYSTEM

In general, for a selected channel c , if there are n channels that are faster than c , then the number of packets (k_i) to be skipped due to each of the faster channels is calculated separately using the formula identical to Eq. (3.1). The final result of k equals to the sum of all the n individual k_i . Hence, the value of k is generally calculated as:

$$k = \sum_{i \in F} \left[\frac{bw_i(d_c - d_i)}{m} + \frac{bw_i}{bw_c} \right] \quad (3.2)$$

where F is the set of channels that are faster than channel c .

If the calculated k exceeds the current length of queue, the packet at the tail will be selected. Since the formula for calculating k is not complicated, it will not cause much overhead for re-sequencing the outgoing packets on CN.

The advantage of the proposed load sharing scheme is that both the bandwidth and delay disparities of different network channels are taken into consideration to mitigate the undesirable effects of multi-path data transmission. The weighted-round-robin style channel selector $C(s)$ compensates the bandwidth disparity by dividing the amount of traffic on different channels according to bandwidth ratio. The packet selector $P(s)$ compensates delay disparity by enforcing out-of-sequence transmission at sender side to improve the chance of in-sequence reception at receiver side.

The load sharing scheme described above is based on the assumption that the MN has multiple network interfaces and the CN has only one. If the CN also has more than one interface, things become more complicated. When sending out a packet to the MN, the CN not only needs to select the destination CoA, it also needs to select the source CoA for the packet, i.e., from which local interface the packet should be sent out. The possible number of virtual channels for distributing network traffic increases to $x \times y$, where x and y are the number of network interfaces on CN and MN respectively. This special case is out of the scope of this research.

3.4 Simulation Results

We conduct a series of simulations based on *ns2* [36] to evaluate the performance of our proposed load sharing algorithm. The scenario setup consists of two nodes. One is the MN that has three network interfaces, and the other is the CN that has one network interface. Assuming that Mobile IPv6 route optimization is effective, the CN can directly send data to MN without interception of HA. The CN can send packet to MN via three different paths, each of which corresponds to one of the MN's multiple interfaces. The bandwidths and propagation delays associated with the different paths are specified as:

- path1: 100Kbps, 200ms
- path2: 1Mbps, 20ms
- path3: 2Mbps, 5ms

This path setup approximates the network characteristics of GPRS, Bluetooth and IEEE 802.11b respectively. The setup is illustrated in Figure 3.4.

Figure 3.4: Simulation Setup

The load sharing module is implemented on CN. In this simulation, CN sends a continuous stream of fixed-sized packets to the MN. The load sharing module decides which outgoing packet to be sent through which network path. The data rate of source

CHAPTER 3. LOAD SHARING IN MULTI-HOMED SYSTEM

traffic is exponentially distributed, with the average of 3.05Mbps , which is close to the limit of the three links' aggregate bandwidth. The simulation scripts can be obtained from [37].

The performance of the following three load sharing algorithms are evaluated:

- Weighted-Round-Robin with Jump-Ahead (WRR-JA), i.e., our proposed algorithm.
- Weighted-Round-Robin (WRR): output channels are selected in WRR manner, but outgoing packets from the queue are transmitted in FIFO order.
- Round-Robin with Jump-Ahead (RR-JA): output channels are selected alternately in simple RR manner while packets are transmitted out of sequence with Jump-Ahead packet scheduling.

These algorithms are compared based on two metrics: buffer requirement and average delay.

3.4.1 Buffer Requirement

In our simulation, when the receiver receives out-of-sequence packets at transport layer, it stores the packets in buffer and waits for the preceding packets to arrive. The packets in the buffer are passed to upper layer when the gaps are filled up, so that user applications experience in-sequence packet reception. The buffer size requirement depends on the amount of out-of-sequence packets received. The greater amount of out-of-sequence packets, the larger buffer is required. Figure 3.5 shows the simulation result of the buffer requirement under different packet sizes used in source traffic.

RR-JA scheme displays the worst performance due to unfair packet distribution. The weakest link is heavily congested with an average drop ratio of 29.81%, which results in a high volume of out-of-sequence packets. WRR and WRR-JA perform much better than

Figure 3.5: Re-sequencing Buffer Requirement

RR-JA due to fair traffic distribution. WRR-JA outperforms WRR by an average of 13.76% reduction in buffer requirement. This shows that Jump-Ahead packet scheduling effectively increases the chance of in-sequence packet reception.

3.4.2 Packet Delay

Here, packet delay refers to the delay experienced by the upper-layer application, which includes link transmission time, propagation delay, and queuing time in re-sequencing buffer. The measured packet delay for using different packet sizes in source traffic is illustrated in Figure 3.6.

The long delay in RR-JA scheme is mainly caused by congestion on slow path. Again,

Figure 3.6: Average Packet Delay

WRR-JA performs the best, with 74.88% reduction in average delay comparing against RR-JA, and 11.59% reduction against WRR. A shorter delay against WRR is the result of shorter queuing time in re-sequencing buffer at the receiver.

3.5 Summary

In this chapter, we have described a multi-homed system that supports load sharing among multiple network interfaces so that the MN can enjoy the aggregate bandwidth. The system is based on Mobile IPv6, which is extended to support multi-homed communications.

In particular, we have designed the policy for mapping network flows to appropriate

CHAPTER 3. LOAD SHARING IN MULTI-HOMED SYSTEM

interfaces based on the data rates of the flows and network characteristics associated with different interfaces. The system allows the MN to establish the mappings from network flows to interfaces based on the policy and send them to CN, so that CN can transmit data to the MN's multiple interfaces accordingly.

We have also proposed a load sharing algorithm to distribute packets to multiple links by incorporating a weighted-round-robin channel selector and Jump-Ahead packet selector. The proposed channel selector is able to balance the loads among the channels so that slow links are not congested and fast links are reasonably utilized, which compensates the bandwidth disparity. The proposed packet selector, selecting packets from the transmission queue in a controlled out-of-sequence manner, is able to compensate the delay disparity and improve the probability of in-sequence reception at the receiver side. Simulation results demonstrate that our load sharing algorithm is effective in reducing both the buffer requirement and packet delay.

Chapter 4

Improving TCP Performance during Soft Vertical Handoff

4.1 Background

With the help of Mobile IPv6, on-going TCP sessions can remain active when MN hands over from one network to another. However, only providing an unbroken TCP connection during handoff is not good enough. Many studies [38, 39, 40, 41, 42] show that the change of link characteristics such as bandwidth and propagation delay during vertical handoff raises problems on the performance of TCP and also TCP-friendly protocols. Some undesirable phenomenon, such as packet reordering and increased round trip delay, may erroneously trigger TCP congestion-related actions and thus cause degradation of TCP performance.

In this chapter, we are looking for network-layer approaches to improve TCP performance during vertical handoff without modifying TCP protocol. We argue that the network layer is closer to the link layer in the OSI stack, and based on Mobile IPv6, the network layer is aware of the link-layer change during vertical handoff. Thus, it might be

more appropriate to use network-layer approaches than using transport-layer approaches in terms of complexity and performance. By using network-layer approaches, vertical handoff becomes transparent to the transport layer and transport-layer protocols can focus on dealing with end-to-end packet delivery without caring about the underlying network changes. In particular, we consider two soft vertical handoff scenarios, i.e., from slow link to fast link and from fast link to slow link. Especially, we focus on the TCP timeout problem caused by increased round-trip time (RTT) when handing over from fast link to slow link, and we propose three network-layer schemes to solve this problem.

This chapter is organized as follows. Section 4.2 gives a brief overview of TCP characteristics. Section 4.3 describes the problems TCP may encounter during vertical handoff. Section 4.4 gives a literature review on existing research work on TCP mobility. Section 4.5 presents our proposed schemes for solving TCP timeout problem due to increased round-trip delay. Finally, Section 4.6 concludes the chapter.

4.2 Overview of TCP Characteristics

With the help of Mobile IPv6, on-going TCP sessions do not get broken upon vertical handoff. Particularly, during soft handoff, packet loss due to change of route can be avoided, since MN is reachable on both the new network and old network at handoff time. However, the change of network characteristics may still have undesirable effects on TCP performance. This section gives a brief overview of TCP characteristics, and we will discuss how TCP gets affected by vertical handoff in the next section.

TCP is a connection-oriented and reliable end-to-end transport protocol. It employs sliding-window based flow control and congestion control mechanisms to avoid the network routers and data receivers being overloaded, and at the same time to make the network links reasonably utilized.

CHAPTER 4. IMPROVING TCP PERFORMANCE DURING SOFT VERTICAL HANDOFF

The stream of data is divided into TCP packets and delivered to the receiver. The receiver must send an *acknowledgment* (ACK) back to the sender upon receiving each or every second data packet. When an ACK is received, the sender calculates the RTT of the acknowledged packet, and estimates the *retransmission timeout* (RTO) for future data packets. Normally, TCP updates the value of RTO once per RTT, and RTO has a lower bound of 1 second [43].

The maximum number of packets that a TCP sender can send in a burst before receiving new ACK is limited by the *congestion window* (CWND). The throughput of TCP is highly associated with the size of CWND, since the larger the CWND, the more packets the sender can send per RTT.

At the time a TCP session is newly established, TCP sender enters the *slow start* phase, and CWND is set to 1. At the TCP sender side, every successfully received ACK triggers CWND to be increased by 1. When CWND reaches the *slow start threshold* (SSThresh), TCP enters the *congestion avoidance* phase, where CWND is increased by 1 only when an entire window of packets have been acknowledged.

The size of CWND will shrink in the following two situations:

- No ACK is received within the period of RTO. In this case, TCP goes back to slow start phase by reducing CWND to 1 and reducing SSThresh to half of the previous CWND.
- Duplicate ACKs (DupACKs) are received. When a TCP receiver receives an out-of-sequence packet, it will re-acknowledge the last received in-order packet. When the number of duplicate ACKs exceeds the threshold *DupThresh*, congestion control actions will be triggered. Different implementations of TCP react differently upon such event. For TCP Tahoe, it enters slow start phase by shrinking CWND to 1. For TCP Reno, it enters *fast recovery* phase by halving both CWND and SSThresh, which results in a smaller scale of throughput reduction.

TCP assumes that the packet loss in both of the above situations is caused by network congestion, and therefore it shrinks CWND to slow down its transmission rate. However, this assumption is not always true, and reducing CWND may not be appropriate. In the next section, we will see that under soft vertical handoff between different access networks, timeout and DupACKs may occur in some cases that are not related to congestion loss, and CWND shrinkage unnecessarily hinders TCP performance. More details about TCP congestion control can be found in [44].

4.3 TCP Behaviors under Soft Vertical Handoff

TCP was developed mainly under stationary network environment, it works well in network communications using fixed cables. However, many studies [39, 40, 42] have shown that some of the TCP characteristics do not fit well into the scenes of wireless and mobile communications, where hosts are connected to radio access networks and sometimes need to switch from one network to another due to user movement. In this section, we are particularly interested in examining TCP behaviors under soft vertical handoff, so that we can isolate the problems of performance degradation due to change of network characteristics from those due to handoff packet loss.

We assume the CN located in the Internet is the TCP sender and the MN is the TCP receiver that is going through vertical handoff under wireless overlay networks.

As mentioned earlier, vertical handoff occurs when MN changes from one network to another with different access technologies. If the two networks have large bandwidth and delay disparities, it poses a big challenge for a transport-layer protocol to adapt itself to the new network in a proper and efficient way. In general, TCP behaviors in vertical handoff can be classified into two scenarios, i.e., from a low-bandwidth high-delay network to a high-bandwidth low-delay network, and in the other way round. In the following, we will study each scenario in details.

4.3.1 Handoff from Slow Link to Fast Link

The major problem that TCP may encounter during a vertical handoff from a high-delay (slow) network to a low-delay (fast) network is out-of-sequence packet reception, which will cause multiple DupACKs. When the TCP sender receives a DupThresh number of DupACKs, it will reduce its CWND. The typical value of DupThresh in current TCP implementations is 3.

For example, during a vertical handoff from a slow link S to a fast link F , the last sequence number sent via link S is k , and the subsequent packets, $k+1$, $k+2$ and $k+3$, are sent via the new link F . The packets $k+1$, $k+2$ and $k+3$ may arrive before the packet k since they go through the fast link F with shorter delay. Upon reception of each of the out-of-sequence packets, the receiver generates a DupACK requesting for packet k to be sent. The TCP sender will then receive three DupACKs, and hence it will retransmit the packet k and reduce its CWND. We can see that the congestion control under such situation is not appropriate since the packet k is not lost.

There are some researches focusing on this out-of-sequence problem. In [40], the authors analyze under which situations a handover may falsely trigger the TCP congestion control action. In [39], the authors investigate the behaviors of various TCP versions, including TCP Tahoe, TCP Reno, TCP NewReno and TCP SACK. Their simulation results show that the handoff from a slow link to a fast link does cause severe reduction of TCP CWND. To minimize the impact of packet reordering on TCP, the authors in [42] propose to use dynamic value of DupThresh. The reason is that the value of DupThresh limits the number of DupACKs that TCP can receive before retransmission. In other words, dynamically adjusting DupThresh changes the degree of tolerance against reordered packets.

A possible network-layer solution to eliminate the packet reordering problem is that when the TCP sender starts sending packets to the fast link, the IP layer temporarily

withholds these new packets for a while until the previous packets on the old slow link have finished transmission. However, withholding packets will reduce the network throughput. Alternatively, we can withhold those new packets at the IP layer of the TCP receiver, which will not reduce the throughput but reordering is needed before passing the packets to the TCP layer.

4.3.2 Handoff from Fast Link to Slow Link

The packet reordering problem normally does not happen in the case of handoff from a low-delay link to a high-delay link. It is because earlier packets are sent via fast link and later packets are sent via slow link, packets usually arrive in order (with the assumption that there is no packet reordering caused by intermediate network routing). However, there is another problem that TCP may encounter during a handoff from a fast link to a slow link.

Before handoff, TCP has a relatively small value of RTO due to short RTTs in the fast link. During handoff, the first packet transmitted via the slow link may be acknowledged long after RTO elapses since the current RTO only reflects the estimated timeout in the previous fast link but not in the slow link that the MN is switching to. Due to RTO expiry, TCP will retransmit the packet and enter slow start phase by reducing CWND to 1. In this case, timeout is not caused by packet loss, it is simply due to late arrival of ACK. Such phenomenon is called *spurious timeout*. Entering slow start is not appropriate since the packet is not lost and the ACK may still be in-flight on the slow link. Retransmission in this case is a waste of network resource, and resetting CWND is not necessary.

4.4 Literature Review

In the literature, some works have been done to tackle the problems of transport layer mobility. I-TCP [45] proposes to partition the TCP connection between CN and MN into two parts: CN to base station (BS), and BS to MN. BS is responsible for caching data from the sender, delivering the data to the receiver, and sending back ACKs to the sender. When MN hands over to another network, the connection states are migrated from the old BS to the new BS so that established TCP connections are still alive. M-TCP [46] also proposes to split up the connection between CN and MN as in I-TCP. When BS detects a packet loss, it sends an ACK to the sender with zero-size window advertisement, so that the sender stops data transmission without dropping its congestion window. BS then retransmits the loss packet to MN and relays ACKs back to the sender when MN has successfully received the data. This provides a simple mechanism to distinguish random packet loss in wireless network from congestion loss in wired network. Both of these schemes need the help of BS to handle TCP connections, which require large data buffer and high processing power on the BS. The migration of data buffer and connection state between old BS and new BS during handoff usually involves considerable overhead.

As an end-to-end solution, the authors in [47] propose the Freeze-TCP scheme. In this scheme, when the receiver is about to initiate a network handoff, it sends a zero window advertisement (ZWA) to the sender. Upon receiving ZWA, the sender freezes its retransmission timer and stops data transmission. After handoff is completed, the receiver sends three DupACKs with non-zero receiver window advertisement. The sender then resumes data transmission with previous CWND size and RTO value. Since no data is transmitted during handoff, it effectively eliminates handoff packet loss. This scheme increases TCP throughput by preventing the sender from shrinking its CWND and backing off RTO. TCP-Probing [48] introduces a probing mechanism into the standard TCP. When a data packet is detected lost, the sender enters a probe cycle. During the probe

CHAPTER 4. IMPROVING TCP PERFORMANCE DURING SOFT VERTICAL HANDOFF

cycle, data transmission is suspended and only probe segments are sent. The sender continuously measures the probe RTTs and determines the level of congestion. After the probe cycle is finished, the sender performs congestion control operations according to the detected nature of packet loss. If real congestion is detected, the sender enters slow start or fast recovery as in standard TCP. If no congestion is detected in the probe cycle, the sender enters Immediate Recovery state, where neither CWND nor RTO is adjusted. The authors in [49] propose a TCP scheme that also takes vertical handoff into consideration. In their scheme, TCP uses an optional field in the header to identify the handoff situation. When handoff is impending, TCP at the receiver is notified about the nature of handoff by link layer or IP layer. The receiver then sends an ACK to the sender, indicating in the optional field that whether the handoff is horizontal or vertical. The sender freezes data transmission during handoff. After handoff is finished, the receiver sends another ACK notifying the sender to resume transmission. Upon receiving the ACK, the sender resumes sending data and adjusts its congestion control state based on the type of handoff. For horizontal handoff, CWND, slow start threshold and RTO are all unchanged. For vertical handoff, the sender enters slow start state where CWND drops to 1 and it starts to reestimate the capacity of the new network. All of the above schemes have the common feature: the sender halts data transmission during network handoff. In hard handoff, packet loss due to handoff is usually unavoidable, stopping transmission in this case can avoid downward adjustment of CWND caused by packet loss. However, in soft handoff, packet loss can be eliminated since MN is always reachable. Freezing data transmission in this case would unnecessarily reduce TCP performance.

Eifel algorithm is proposed in [50] to tackle the problem of spurious timeout. The algorithm requires that timestamps should be included in TCP header of every packet. The algorithm can detect spurious timeout based on the timestamps information. Once spurious timeout is detected, congestion control state will be restored to what it was

prior to timeout occurrence. Eifel algorithm is simple and effective, however, adding timestamps to TCP header of every packet is quite an overhead.

4.5 Network-Layer Solutions to Spurious TCP Timeout

In this section, we try to explore the capability of network layer in coping with spurious timeout problem. Considering the scenario of soft handoff, our schemes allow TCP packets to be continuously delivered without pausing data transmission. In particular, the network-layer schemes are completely transparent to TCP, no modifications on TCP protocol are required.

In normal Mobile IPv6 handoff from fast link to slow link (with route optimization), packets are acknowledged via the same link through which the packets were transmitted. That is, fast-link packets are acknowledged via fast link, and slow-link packets are acknowledged via slow link. This can be explained by Figure 4.1. The figure illustrates the sequence of packet transmission before and after a handoff from Link F (the fast link) to Link S (the slow link), where CN is the sender and MN is the receiver. Before handoff, both data and ACKs are sent via Link F. At time t_0 , MN performs a vertical handoff to Link S and sends a BU packet to CN via the new link so that the CN gets aware of the new CoA of the MN. CN receives the BU message at time t_1 and responds with a binding acknowledgment (BA). From this time onward, CN starts sending data to the MN's new CoA via Link S. At time t_2 , MN receives the BA, and hence it starts using the slow link to send ACKs to CN. According to such packet transmission sequence, we can see that the MN usually acknowledges a data packet via the same link from which the data was received, since it receives BA after the reception of the last fast-link data packet and before the arrival of the first slow-link packet.

CHAPTER 4. IMPROVING TCP PERFORMANCE DURING SOFT VERTICAL HANDOFF

Figure 4.1: Packet Sequence in Normal MIPv6 Vertical Handoff

Under normal handoff, suppose the sequence number of the last data packet sent via the fast link is k , and the first data packet to be sent via the slow link is $k + 1$. Then, the observed RTT of the packet k is equal to the sum of the transmission time for the data packet k and the transmission time for the corresponding ACK, i.e., $2D_f$, where D_f is the averaged one-way delay of the fast link. Similarly, the RTT of the packet $k + 1$ is equal to $2D_s$, where D_s is the averaged one-way delay on the slow link. Hence, the difference between the two RTTs is:

$$RTT(k + 1) - RTT(k) = 2(D_s - D_f).$$

The spurious timeout problem during vertical handoff is caused by the large difference in RTTs of the old fast link and the new slow link, which the estimated RTO fails to tolerate. To solve this problem, we propose three different network-layer schemes, namely, *Fast ACK*, *Slow ACK* and *ACK Delaying*. The basic idea of these schemes is to shorten the difference between the observed RTTs of the two links during the handoff period, so that it allows TCP to adapt its RTO to the new link condition without causing timeout.

In practice, it is possible for MN to estimate the round-trip delay of the old link and new link by calculating the time between sending BUs and receiving BAs. Once it detects that the RTT of the new link is significantly longer than that of the old link, the network layer can apply the following schemes to avoid spurious TCP timeout.

4.5.1 Scheme 1: Fast ACK

In the Fast ACK scheme, we propose to route the ACKs of the first few slow-link packets ($k + 1, k + 2, k + 3 \dots$) to the old fast link to reduce the gap between old RTT and new RTT. Following is the sequence of operations on CN and MN when applying this scheme.

- (1) Before handoff, CN sends data packets to MN via fast link, MN sends ACKs to CN through the same link.
- (2) MN initiates soft handoff from fast link to slow link, CN gets updated about MN's new CoA.
- (3) CN starts sending data packets to MN's new CoA via slow link.
- (4) When MN receives the first n (predefined value) packets from the slow link, it still sends ACKs via the old fast link.
- (5) When the $(n + 1)th$ packet from the slow link is received, MN starts sending ACKs via the slow link, and disconnects from the old fast link.

Under such scheme, the RTT for packet $k + 1$ (the first packet which is sent via slow link but acknowledged via fast link) becomes $D_s + D_f$, and hence

$$RTT'(k + 1) - RTT(k) = D_s - D_f.$$

That is, the gap is halved compared with that in the normal handoff scenario. The reduction of RTT for the first few slow-link packets may be able to avoid TCP timeout.

The RTTs observed by TCP roughly follows the pattern shown in Figure 4.2. There

Figure 4.2: RTT Changes in Two Step

is a time period within which TCP observes the “in-between” values of RTTs, which helps TCP to adapt its RTO before timeout occurs. We call this period as *adaptation time*.

The Fast ACK scheme requires the receiver to continue using the old fast link for a short while after the sender is already sending data on the new slow link. One may argue

CHAPTER 4. IMPROVING TCP PERFORMANCE DURING SOFT VERTICAL HANDOFF

that handoff is usually triggered by the downturn of the network conditions in the old link, which means that the transmission of ACKs on the old fast link during handoff may be unreliable. In fact, even if a few ACKs on the old fast link are lost, TCP performance may not get affected due to the feature of *accumulative acknowledgment*. That is, when an ACK is lost, as long as some later ACKs successfully arrive, TCP sender will assume that the earlier packets are also acknowledged and will move its sliding window forward accordingly. Hence, the unreliability of the old fast link may not cause a real problem.

In order to verify the effectiveness of this scheme, we conducted simulations based on ns2. The simulation setup is shown in Figure 4.3. The upper link in Figure 4.3 is the fast link with bandwidth of $2Mbps$ and propagation delay of $20ms$. The bottom link is the slow link with bandwidth of $800Kbps$ and propagation delay of $600ms$. Exponential on-off background traffic is applied to both links to make the simulation dynamic. The mean burst time and mean idle time of the random traffics are set to $50ms$. The average burst rate of the random traffic on the fast link is $500Kbps$, and the burst rate on the slow link is $200Kbps$.

Figure 4.3: Simulation Setup

The TCP session starts on the fast link, with data sent from CN to MN. In normal handoff scenario, both CN and MN switch to the slow link at the 20th second. In Fast

CHAPTER 4. IMPROVING TCP PERFORMANCE DURING SOFT VERTICAL HANDOFF

ACK scheme, MN switches to the slow link at the 21th second. In other words, in our proposed scheme, between the 20th and the 21 second, TCP data are sent via slow link, but acknowledged via fast link. Simulation scripts are available on [37].

The comparisons of TCP performance are shown in Figure 4.4, and Figure 4.5.

Figure 4.4: Comparison of CWND between Normal Handoff and Fast ACK

Figure 4.4 compares the changes of CWND between the normal handoff and Fast ACK scheme. We can see that in the normal handoff, TCP experiences timeout and CWND is reset to 1 at around the 21th second due to the late arrived ACK for the first packet transmitted on the slow link. In the Fast ACK scheme, no timeout occurs since the ACK arrives in time via the fast link. Elimination of spurious timeout helps TCP to retain its CWND at a higher value, which results in a faster throughput than the normal handoff.

CHAPTER 4. IMPROVING TCP PERFORMANCE DURING SOFT VERTICAL HANDOFF

Figure 4.5: Comparison of RTO between Normal Handoff and Fast ACK

Figure 4.5 shows the changes of RTO. Before handoff, the value of RTO is determined by the 1-second lower bound due to short RTTs on the fast link. In normal handoff, RTO doubles to the value of 2 at around the 21th second due to timeout occurrence. In the Fast ACK scheme, RTO starts rising before timeout occurs when the ACKs for the first few slow-link packets are received. The sudden change of propagation delay results in large variance between RTT samples, it causes RTO to rise to a value that is much higher than the observed RTTs. After handoff is completed, RTO becomes stabilized and gradually converges to the RTTs of the new link. We can see that in the Fast ACK scheme, TCP adapts its RTO earlier than the normal scheme to avoid spurious timeout, and RTO converges to the new RTTs at a faster speed since TCP doesn't enter the slow start phase.

4.5.2 Scheme 2: Slow ACK

The previous scheme reduces the chance of TCP timeout problem by shortening the RTTs of the first few packets transmitted on the new slow link. Our proposed second scheme tries to reduce the gap between old RTT and new RTT by enlarging the RTTs of the last few packets which are transmitted on the old fast link. This can be achieved by sending the ACKs of the last few fast-link packets via the new slow link, while it requires the receiver to switch to the new link in advance. Scheme 2 is called *Slow ACK* scheme.

The last few fast-link packets which are acknowledged via the slow link experience the RTTs of $D_f + D_s$. If these enlarged RTTs do not exceed the previous RTO, they will help TCP to adjust its RTO so that TCP can adapt to the new network condition.

We have also conducted simulations for Slow ACK scheme, and similar results as those of Fast ACK scheme can be observed (See Figure 4.6 and Figure 4.7).

4.5.3 Scheme 3: ACK Delaying

The third scheme is called *ACK Delaying*, which gradually increases the RTTs for the last few fast-link packets by withholding the ACKs for those packets at the IP layer of the receiver (assuming that MN is the receiver) and sending them out to the sender after a certain delay. Alternatively, if MN is the data sender, it can withhold the received ACKs at IP layer for a certain period before passing them up to TCP layer. Unlike Fast ACK and Slow ACK schemes, in ACK Delaying scheme, the fast-link packets are still acknowledged via the fast link, and the slow-link packets are acknowledged via the slow link, the same as that in the normal handoff. ACK Delaying scheme simply introduces the delay of TCP ACKs at the IP layer of MN. This scheme is more flexible than the previous two schemes, because we can purposely design the delay at the IP layer to be gradually increased and thus TCP sender will observe a smooth transition in RTT during handoff, instead of a large step change of RTT as in the previous schemes.

CHAPTER 4. IMPROVING TCP PERFORMANCE DURING SOFT VERTICAL HANDOFF

Figure 4.6: Comparison of CWND between Normal Handoff and Slow ACK

In such a scheme, two important issues need to be considered:

- When should IP layer start delaying TCP ACKs?
- What should be the proper periods that TCP ACKs need to be delayed?

We will answer these questions by analyzing how TCP updates its RTO value when RTT changes. TCP calculates RTT upon every received ACK, and uses the Jacobson's

CHAPTER 4. IMPROVING TCP PERFORMANCE DURING SOFT VERTICAL HANDOFF

Figure 4.7: Comparison of RTO between Normal Handoff and Slow ACK

algorithm [44] to estimate the RTO:

$$SRTT(k+1) = (1-g) \times SRTT(k) + g \times RTT(k+1) \quad (4.1)$$

$$SERR(k+1) = RTT(k+1) - SRTT(k)$$

$$SDEV(k+1) = (1-h) \times SDEV(k) + h \times |SERR(k+1)|$$

$$RTX(k+1) = SRTT(k+1) + f \times SDEV(k+1) \quad (4.2)$$

$$RTO(K+1) = MAX(1, RTX(k+1)) \quad (4.3)$$

where $SRTT$, $SERR$ and $SDEV$ are the exponentially smoothed estimates of RTT, error term and deviation, respectively, and g, h and f are tunable parameters. In current TCP implementations, $g = 0.125$, $h = 0.25$ and $f = 4$.

CHAPTER 4. IMPROVING TCP PERFORMANCE DURING SOFT VERTICAL HANDOFF

From Eq. (4.2) and (4.3), we can see that RTO is finally the weighted sum of SRTT and SDEV, with a lower bound of 1 second. In our analysis, we focus on the SRTT term.

Figure 4.8 illustrates how SRTT changes when the sampled RTTs increase from 20ms to 600ms. (The figure is purely based on the equations, not based on simulation.)

Figure 4.8: Change of SRTT upon a Step Increase in RTT

Eq. (4.1) shows that the latest RTT sample contributes to SRTT with the weight of g . Suppose that $RTT(k)$ is the last RTT sample on the old link, and $RTT(k + 1)$ to $RTT(k + n)$ are the first n samples on the new link. Eq. (4.1) can be recursively extended

as:

$$\begin{aligned}
 SRTT(k+n) &= (1-g)^n \cdot SRTT(k) + (1-g)^{n-1}g \cdot RTT(k+1) + \cdots \\
 &\quad + (1-g)g \cdot RTT(k+n-1) + g \cdot RTT(k+n) \\
 &= (1-g)^n \cdot SRTT(k) + g \sum_{i=1}^n (1-g)^{n-i} RTT(k+i) \quad (4.4)
 \end{aligned}$$

To simplify the problem, we assume $SRTT(k) = oldRTT$, $RTT(k+1) = RTT(k+2) = \cdots = RTT(k+n) = newRTT$. Then Eq. (4.4) can be written as:

$$\begin{aligned}
 SRTT(k+n) &= (1-g)^n \cdot oldRTT \\
 &\quad + g \sum_{i=1}^n (1-g)^{n-i} \cdot newRTT \\
 &= (1-g)^n \cdot oldRTT + [1 - (1-g)^n] \cdot newRTT \quad (4.5)
 \end{aligned}$$

In the above equation, the weight of newRTT is $s(n) = 1 - (1-g)^n$. With $g = 0.125$, Eq. (4.5) effectively becomes:

$$SRTT(k+n) = 0.875^n \cdot oldRTT + (1 - 0.875^n) \cdot newRTT$$

and the weight of newRTT becomes:

$$s(n) = 1 - 0.875^n$$

The function $s(n)$ is shown in Figure 4.9.

The value of $s(n)$ is greater than 0.8 when n is larger than 13. In other words, after a step increase from oldRTT to newRTT, the TCP sender should observe at least 13 samples of newRTT in order for the value of newRTT to contribute more than 80% to the calculation of SRTT.

In the above scenario, RTT increases in a single step. It is the fastest way for SRTT to grow toward newRTT, however, such an abrupt change in RTT may cause spurious timeout. If the observed RTT samples increase in a gradual manner, it helps to reduce

CHAPTER 4. IMPROVING TCP PERFORMANCE DURING SOFT VERTICAL HANDOFF

Figure 4.9: Function $s(n)$

the probability of timeout. In this way, SRTT takes a longer time to approach the value of newRTT.

In the following subsections, we consider two schemes for gradual RTT increment:

- Linear increment
- Quadratic increment

For each scheme, we examine how SRTT grows when RTT is gradually increasing, and we can determine how long it will take SRTT to become close to the value of newRTT. When the value of SRTT is close enough to newRTT, we can safely say that the estimated RTO is suitable to be used in the new network environment. And only at that time, we should perform vertical handoff to the slow link.

4.5.3.1 Linear Increment on RTT

In linear increment scheme, the delay of TCP ACKs introduced by IP layer is a linear function, so that the RTTs observed by TCP gradually increase from $oldRTT$ to $newRTT$ in n steps, with a uniform step size of $\frac{newRTT - oldRTT}{n}$. Figure 4.10 shows the RTTs observed by TCP and how SRTT is updated in this scheme. (Again, this is based on calculation, not simulation.)

Figure 4.10: Change of SRTT with Linear Increment of RTT

For n -step linear increment,

$$RTT(k+i) = oldRTT + \frac{newRTT - oldRTT}{n} \cdot i \quad (4.6)$$

Substitute Eq. (4.6) into Eq. (4.4), we have:

$$\begin{aligned}
SRTT(k+n) &= (1-g)^n \cdot oldRTT \\
&\quad + g \sum_{i=1}^n (1-g)^{n-i} (oldRTT + \frac{newRTT - oldRTT}{n} \cdot i) \\
&= [(1-g)^n + g \sum_{i=1}^{n-1} (1-g)^i] oldRTT \\
&\quad + g \cdot \frac{newRTT - oldRTT}{n} \sum_{i=1}^n (1-g)^{n-i} \cdot i \\
&= oldRTT \\
&\quad + g \cdot \frac{newRTT - oldRTT}{n} \sum_{i=1}^n (1-g)^{n-i} \cdot i \\
&= oldRTT \\
&\quad + \{1 - \frac{1-g}{n \cdot g} [1 - (1-g)^n]\} (newRTT - oldRTT) \\
&= \frac{1-g}{n \cdot g} [1 - (1-g)^n] \cdot oldRTT \\
&\quad + \{1 - \frac{1-g}{n \cdot g} [1 - (1-g)^n]\} \cdot newRTT
\end{aligned} \tag{4.7}$$

The weight of newRTT is:

$$p(n) = 1 - \frac{1-g}{n \cdot g} [1 - (1-g)^n] \tag{4.8}$$

Figure 4.11 shows the function $p(n)$ with $g = 0.125$.

The value of $p(n)$ is greater than 0.8 when n is larger than 35. That is to say, when RTT is linearly increased, in order for the value of newRTT to contribute more than 80% to the calculation of SRTT, the number of intermediate steps should be greater than 35.

4.5.3.2 Quadratic Increment on RTT

Instead of using linear increment, the ACK delay introduced by IP layer can follow a quadratic curve, which produces an even smoother transition from oldRTT to newRTT. In a quadratic increment scheme, the ACK delay can be expressed as the function:

$$D(x) = ax^2$$

CHAPTER 4. IMPROVING TCP PERFORMANCE DURING SOFT VERTICAL HANDOFF

Figure 4.11: Function $p(n)$

If RTT is to be increased from $oldRTT$ to $newRTT$ in n steps, then

$$D(n) = an^2 = newRTT - oldRTT$$

Hence

$$a = \frac{newRTT - oldRTT}{n^2}$$

We can rewrite $D(x)$ as:

$$D(x) = \frac{newRTT - oldRTT}{n^2} \cdot x^2 \quad (4.9)$$

The step size of RTT increment is:

$$\frac{dD(x)}{dx} = \frac{2(newRTT - oldRTT)}{n^2} \cdot x$$

CHAPTER 4. IMPROVING TCP PERFORMANCE DURING SOFT VERTICAL HANDOFF

For earlier RTT samples, the step size is small, and for later RTT samples, the step size grows larger. This kind of smooth transition has a higher probability of eliminating spurious timeout compared with the linear scheme. Figure 4.12 illustrates the quadratic increment of RTT and the resulting SRTT pattern.

Figure 4.12: Change of SRTT with Quadratic Increment of RTT

For n -step increment,

$$\begin{aligned} RTT(k+i) &= oldRTT + D(i) \\ &= oldRTT + \frac{newRTT - oldRTT}{n^2} \cdot i^2 \end{aligned} \quad (4.10)$$

Substitute Eq. (4.10) into Eq. (4.4), we have:

$$\begin{aligned} SRTT(k+n) &= (1-g)^n \cdot oldRTT \\ &\quad + g \sum_{i=1}^n (1-g)^{n-i} \left(oldRTT + \frac{newRTT - oldRTT}{n^2} \cdot i^2 \right) \\ &= [(1-g)^n + g \sum_{i=1}^{n-1} (1-g)^i] oldRTT \\ &\quad + g \cdot \frac{newRTT - oldRTT}{n^2} \sum_{i=1}^n (1-g)^{n-i} \cdot i^2 \\ &= oldRTT \\ &\quad + g \cdot \frac{newRTT - oldRTT}{n^2} \sum_{i=1}^n (1-g)^{n-i} \cdot i^2 \\ &= oldRTT \\ &\quad + \left\{ 1 - \frac{(2-g)(1-g)[(1-g)^n - 1] + 2g \cdot n(1-g)}{(g \cdot n)^2} \right\} \\ &\quad \cdot (newRTT - oldRTT) \\ &= [1 - q(n)] \cdot oldRTT \\ &\quad + q(n) \cdot newRTT \end{aligned} \quad (4.11)$$

where

$$q(n) = 1 - \frac{(2-g)(1-g)[(1-g)^n - 1] + 2g \cdot n(1-g)}{(g \cdot n)^2} \quad (4.12)$$

which is the weight of newRTT.

Figure 4.13 shows the function $q(n)$.

The value of $q(n)$ is greater than 0.8 when n is larger than 62. In other words, when RTT is quadratically increased, in order for the value of newRTT to contribute more than 80% to the calculation of SRTT, the number of intermediate steps should be greater than 62.

Figure 4.13: Function $q(n)$

4.5.3.3 Simulation Results

Simulations have been carried out using similar setup as in Figure 4.3. Both the linear increment scheme and quadratic increment scheme estimate the RTTs of the fast link and slow link by sending probing messages, and start delaying TCP ACKs from fast link at the 20th second. In linear scheme, MN hands over to slow link after $35 \times oldRTT$. In quadratic scheme, MN performs handoff after $62 \times oldRTT$.

In Figure 4.14, the values of RTX (See Eq. (4.2)) have been plotted instead of RTO, since RTX is purely calculated based on RTT and RTT variance, without the 1-second lower bound, hence it shows more details on how TCP gets adapted.

Again, in normal handoff, RTX backs off due to timer expiry. In ACK Delaying schemes, we can observe smooth transitions of RTX values. Compared with the linear

CHAPTER 4. IMPROVING TCP PERFORMANCE DURING SOFT VERTICAL HANDOFF

Figure 4.14: Comparison of RTX between Normal Handoff and ACK Delaying

increment scheme, the quadratic scheme takes a longer time for RTX to reach the peak, but it allows RTT to change less aggressively, especially at the beginning of adaptation.

The effectiveness of the schemes depends on the difference between the average round-trip delays of the fast link and slow link, since the step size of increment is based on the estimated $(newRTT - oldRTT)$. The larger the RTT difference, the more likely that timeout will occur. Hence, in the next experiment, we try to evaluate the effectiveness of the ACK Delaying schemes against the RTT difference. In the second experiment, we run the simulation upon increasing values of average RTT difference. The average one-way propagation delay of the fast link is set to $20ms$, while the average delay of the slow link starts from $120ms$, and increased by $100ms$ at each loop. For each different value of slow link delay, we run the simulation 100 times, and record the percentage of

CHAPTER 4. IMPROVING TCP PERFORMANCE DURING SOFT VERTICAL HANDOFF

timeout occurrence. In each run, random background traffic as specified in Section 4.5.1 is applied on both links, and vertical handoff is triggered at random time. The results of this experiment is presented in Figure 4.15.

Figure 4.15: Percentage of Timeout vs Average RTT Difference (oldRTT=40ms)

Since the average fast link RTT is small, RTO on the fast link is determined by the 1-second lower bound, instead of the calculated RTX. For normal handoff, once the slow link RTT exceeds 1 second, timeout occurs. The linear increment scheme effectively avoids spurious timeout when the average RTT difference is below 3 seconds. However, when the difference is larger than 4 seconds, the scheme does not perform well, since the step size of increment is too large. The quadratic increment scheme performs much better than the linear scheme. It manages to eliminate TCP timeout in most of the cases.

In the third experiment, the setup is similar to the second one, but the average fast

CHAPTER 4. IMPROVING TCP PERFORMANCE DURING SOFT VERTICAL HANDOFF

link delay is increased to $450ms$. That is, the mean RTT of the fast link is around $900ms$. In this case, the RTO on the fast link is not dominated by the 1-second lower bound. Figure 4.16 shows the simulation results.

Figure 4.16: Percentage of Timeout vs Average RTT Difference (oldRTT=900ms)

Comparing Figure 4.16 with Figure 4.15, we can see that both the linear and quadratic scheme can tolerate a higher RTT difference. The quadratic scheme can even avoid spurious timeout when the RTT difference is as high as 10 seconds. This is mainly due to the enlarged average RTT of the fast link. With a larger fast link RTT, the ratio of RTT difference against oldRTT becomes smaller. In other words, when handing over from the fast link to the slow link, the change of RTT becomes less significant, hence the ACK Delaying schemes can perform better.

From all the above simulation results, we can conclude that the ACK Delaying

schemes can effectively avoid spurious timeout. The quadratic scheme can tolerate a higher RTT difference than the linear scheme. Actually, in real network environment, the average RTTs of some commonly used high-delay networks (e.g., cellular network, satellite network) are usually kept below 3 seconds. Under such condition, the linear increment scheme is good enough to resolve the spurious timeout problem in most situations.

4.5.3.4 Discussions

In both of the ACK Delaying schemes, we assume that network layer knows the value of oldRTT and newRTT . Actually, this can be done by sending ICMP echo packets (ping) between MN and CN and calculate the round trip delay, or it can be estimated by the time difference between sending out BU and receiving BA. One may think that if network layer already knows the RTT of the new link, it can directly pass up the information to TCP so that TCP can adjust its RTO accordingly. However, TCP congestion control is only performed on the sender, whereas our proposed network-layer schemes are initiated on MN. In most cases, MN acts as a client in network communications, hence it is the data receiver most of the time. When MN is the receiver, it does not keep the record of history RTT values at TCP layer, and it does not determine the value of RTO. During vertical handoff, network layer of MN knows the change of link characteristics, and ACK Delaying scheme is applied to help the remote TCP sender to adapt its RTO to match the new network conditions.

In ACK Delaying scheme, we determine the time needed for RTO adaptation based on the weight of newRTT ($p(n)$ and $q(n)$) in SRTT calculation. The more increment steps (or intermediate RTT samples), the higher the weight. We believe that 0.8 is a reasonable threshold for the weight. In SRTT calculation, if 20% is contributed by oldRTT and 80% is contributed by newRTT , then SRTT is close enough to newRTT already. RTO is the

weighted sum of SRTT and SDEV. With SRTT close enough to newRTT, it is highly possible that the estimated RTO exceeds newRTT, which in fact avoids TCP timeout. If a lower threshold for the weight is selected, then the adaptation time is shorter, but the probability of spurious timeout is higher.

The linear scheme requires shorter adaptation time, since the increment of RTT is more aggressive. However, if the difference between newRTT and oldRTT is large, the step size may exceed the degree of tolerance of RTO, which cannot eliminate spurious timeout. The quadratic scheme requires longer adaptation time, but the step size is lower than the linear scheme, hence it has a greater probability of eliminating spurious timeout. We propose that if oldRTT is small, and the different between oldRTT and newRTT is large, the quadratic scheme should be chosen, since it produce a smoother RTT transition, and the adaptation time would not be too long with a small oldRTT; If oldRTT is large, and the different between oldRTT and newRTT is small, the linear scheme is preferred, since it takes shorter adaptation time, and the step size would not be too large for small difference between oldRTT and newRTT.

4.6 Summary

In this chapter, we have studied the TCP behaviors during vertical handoff. We found that although with the help of Mobile IPv6 and soft handoff, on-going TCP sessions can remain active and handoff packet loss can be avoided, TCP still faces several performance degradation issues due to the change of network bandwidth and propagation delay.

Particularly, when handing over from a fast link to a slow link, a possible problem that TCP may encounter is that the estimated RTO based on the fast link transmission does not tolerate the long RTT on the slow link, which could cause TCP timeout. This TCP timeout problem can be avoided by three possible schemes: Fast ACK, Slow ACK and

CHAPTER 4. IMPROVING TCP PERFORMANCE DURING SOFT VERTICAL HANDOFF

ACK Delaying. Our schemes are applied at network layer of MN, with no modification on TCP protocol. Simulation results have demonstrated that these schemes can effectively reduce the gap between the TCP experienced RTTs on the old fast link and the new slow link so that TCP can gradually adjust its RTO to fit into the new network environment.

Chapter 5

Summary

In this thesis, we present the research works we have conducted during the two-year period of the M.Eng program. The focus of our research is on communication efficiency in heterogeneous mobile network environment. The variety of wireless link-layer technologies provides more options for mobile users to achieve ubiquitous network communications. However, the characteristics disparities among different types of wireless interfaces also raise several performance issues.

When a MN is equipped with multiple network interfaces, two fundamental questions need to be answered: which interface should be activated at a certain time and when should the MN hands over from the current network to another. Our proposed vertical handoff management system automatically selects the best network interface based on a cost function which takes the available bandwidth, power consumption and usage charge into consideration. Users can define their preferences for interface selection by setting the values of parameters in the policy database. The system also continuously monitors the status of currently connected network and makes handoff decision if the signal strength becomes weak. The proposed system is partly implemented on both laptop PC and handheld PDA. Real network experiments have shown that the system can handle vertical handoff properly without breaking upper layer sessions.

CHAPTER 5. SUMMARY

Wireless networks usually have very limited capacity, which may not be sufficient to satisfy the need of some bandwidth-hungry applications. Our proposed load sharing system allow data to be sent or received via multiple network interfaces simultaneously, so that the effective bandwidth is greater than the bandwidth of any single interface. The system includes the policy for mapping data flows to appropriate interfaces, and the algorithm for scheduling packets to different network channels. The algorithm can ensure balanced traffic loads on different network interfaces, which compensates bandwidth disparity, and at the same time reduce the probability of out-of-sequence packet reception, which compensates delay disparity.

The last problem we study in our research is the TCP performance during vertical handoff. In particular, when handing over from a fast network to a slow network, the sudden increase in round-trip delay may cause spurious TCP timeout. To solve this problem, we proposed three network-layer schemes, namely, Fast ACK, Slow ACK and ACK Delaying. The basic idea behind these schemes is to shorten the gap between old RTT and new RTT perceived by TCP, so that TCP can observe a smoother RTT transition, instead of an abrupt increase in a single step. Fast ACK and Slow ACK schemes simply change the route of sending TCP ACK packets to help TCP to adapt its RTO value before timeout occurs. ACK Delaying scheme introduces certain amount of delay for TCP ACK at the network layer, so that the RTTs observed by TCP are gradually increased in a linear or quadratic manner. The third scheme is more complicated than the other two schemes, but it is more flexible, and it can tolerate larger difference of RTTs between the old network and the new network. These schemes require no modifications on TCP protocol. Simulation results show that the proposed schemes are effective in eliminating spurious timeout under certain conditions.

References

- [1] D. Kotz and R. S. Gray, "Mobile agents and the future of internet," *ACM Operating Systems Review*, vol. 33, no. 3, Aug. 1999.
- [2] Hp compaq nc4200 notebook pc specifications. [Online]. Available: <http://h10010.www1.hp.com/wwpc/us/en/sm/WF06a/321957-64295-89315-321838-f34-447354.html>
- [3] Hp ipaq hw6500 mobile messenger series overview. [Online]. Available: <http://h10010.www1.hp.com/wwpc/us/en/sm/WF05a/215348-64929-215381-314903-f85-501209.html>
- [4] "GSM world," <http://www.gsmworld.com/technology/gprs/index.shtml>.
- [5] J. F. Kurose and K. W. Ross, *Computer Networking - A top-down approach featuring the Internet*, 2nd ed. Boston: Addison Wesley, 2003.
- [6] N. Montavont and T. Noël, "Analysis and evaluation of Mobile IPv6 handovers over wireless LAN," *Mobile Networking and Applications (MONET), special issue on mobile networking through IPv6 or IPv4*, vol. 8, no. 6, Nov. 2003.
- [7] The official bluetooth wireless info site. [Online]. Available: <http://www.bluetooth.com>
- [8] Bluez - official linux bluetooth protocol stack. [Online]. Available: <http://www.bluez.org>
- [9] M. Stemm and R. H. Katz, "Vertical handoff in wireless overlay networks," *Mobile Networks and Applications*, vol. 3, no. 4, 1998.

REFERENCES

- [10] C. E. Perkins. (2002, Aug.) RFC 3344: IP mobility support for IPv4. [Online]. Available: <http://www.ietf.org/rfc/rfc3344.txt>
- [11] D. Johnson, C. Perkins, and J. Arkko. (2004, June) RFC 3775: Mobility support in IPv6. [Online]. Available: <http://www.ietf.org/rfc/rfc3775.txt>
- [12] S. Deering and R. Hinden. (1998, Dec.) RFC 2460: Internet protocol, version 6 (IPv6) specification. [Online]. Available: <http://www.ietf.org/rfc/rfc2460.txt>
- [13] J. Davies, *Understanding IPv6*. Microsoft Corporation, 2003.
- [14] A. J. Tuominen and H. Petander, "MIPL: Mobile IPv6 for Linux in HUT Campus Network MediaPoli," in *Proc. Ottawa Linux Symposium*, Ottawa, Canada, June 2001.
- [15] Open source operating systems for handheld devices. [Online]. Available: <http://www.handhelds.org>
- [16] J. Tourrilhes and C. Carter, "P-Handoff: a protocol for fine grained peer-to-peer vertical handoff," in *The 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 2, Sept. 2002.
- [17] F. Du., L. M. Ni, and A. H. Esfahanian, "HOPOVER: A new handoff protocol for overlay networks," in *International Conference on Communications 2002 (ICC2002)*, June 2002.
- [18] M. Ylianttila, M. Pande, J. Makela, and P. Mahonen, "Optimization scheme for mobile users performing vertical handoffs between IEEE 802.11 and GPRS/EDGE networks," in *Proc. IEEE GLOBECOM*, vol. 6, 2001.
- [19] M. Ylianttila, J. Makela, and P. Mahonen, "Supporting resource allocation with vertical handoffs in multiple radio network environment," in *Proc. The 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 1, Sept. 2002.

REFERENCES

- [20] H. S. Park, S. H. Yoon, T. H. Kim, J. S. Park, M. S. Do, and J. Y. Lee, "Vertical handoff procedure and algorithm between IEEE802.11 WLAN and CDMA cellular network," in *Mobile Communications: 7th CDMA International Conference (CIC)*, Seoul, Korea, Nov. 2002.
- [21] Q. Zhang, C. Guo, Z. Guo, and W. Zhu, "Efficient mobility management for vertical handoff between WWAN and WLAN," *IEEE Communications Magazine*, vol. 41, no. 11, Sept. 2003.
- [22] G. Bianchi, B. M. Nicola, M. L. Chan, M. Holzbock, Y. F. Hu, A. Jahn, and R. E. Sheriff, "Design and validation of QoS aware mobile internet access procedures for heterogeneous networks," *Mobile Networks and Applications*, vol. 8, no. 1, Feb. 2003.
- [23] M. Jain and C. Dovrolis, "End-to-end available bandwidth - measurement methodology, dynamics, and relation with TCP throughput," Technical Report, University of Delaware, Feb. 2002.
- [24] Linux kernel archives. [Online]. Available: <http://www.kernel.org>
- [25] Ethereal: A network protocol analyzer. [Online]. Available: <http://www.ethereal.com>
- [26] C. Brendan, S. Traw, and J. M. Smith, "Striping within the network subsystem," *IEEE Network*, pp. 22-29, Jul/Aug 1995.
- [27] A. Hari, G. Varghese, and G. Parulkar, "An architecture for packet-striping protocols," *ACM Trans. Computer Systems*, vol. 17, no. 4, Nov. 1999.
- [28] F. Jacquet and M. Misson, "A method for increasing throughput based on packet striping," in *Proc. IEEE International Conference on Networking*, France, July 2001.
- [29] H. Y. Hsieh and R. Sivakumar, "A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts," in *Proc. ACM MOBICOM*, Atlanta, GA, USA, Sept. 2002.
- [30] L. Magalhaes and R. Kravets, "Transport level mechanisms for bandwidth aggregation on mobile hosts," in *Proc. IEEE ICNP*, Riverside, CA, USA, Nov. 2001.

REFERENCES

- [31] —, “MMTP multimedia multiplexing transport protocol,” in *The First Workshop on Data Communications in Latin America and the Caribbean, SIGCOMM, LA*, 2001.
- [32] R. Stewart, Q. Xie, and etc. (2000, Oct.) Rfc 2960: Stream control transmission protocol. [Online]. Available: <http://www.ietf.org/rfc/rfc2960.txt>
- [33] S. Fu and M. Atiquzzaman, “SCTP: State of the art in research, products, and technical challenges,” *IEEE Commun. Mag.*, no. 4, pp. 64–76, Apr. 2004.
- [34] K. Chebrolu and R. Rao, “Communication using multiple wireless interfaces,” in *Proc. IEEE WCNC*, Orlando, Mar. 2002.
- [35] R. Wakikawa, K. Uehara, T. Ernst, and K. Nagami. (2004, June) Multiple care-of addresses registration. [Online]. Available: <http://www.ietf.org/internet-drafts/draft-wakikawa-mobileip-multiplecoa-03.txt>
- [36] The network simulator - ns-2. [Online]. Available: <http://www.isi.edu/nsnam/ns/>
- [37] Simulation scripts used in Haijie’s master thesis. [Online]. Available: <http://www.ntu.edu.sg/home5/pg04878518/hhj/index.html>
- [38] R. Chakravorty, P. Vidales, L. Patanapongpibul, K. Subramanian, I. Pratt, and J. Crowcroft. (2003, June) On inter-network handover performance using Mobile IPv6. University of Cambridge Computer Laboratory, Technical Report. [Online]. Available: <http://www.cl.cam.ac.uk/users/rc277/handovers.pdf>
- [39] W. Hansmann, M. Frank, and M. Wolf, “Performance analysis of TCP handover in a wireless/mobile multi-radio environment,” in *Proc. the 27th Annual Conference on Local Computer Networks*, Nov. 2002.
- [40] W. Hansmann and M. Frank, “On things to happen during a TCP handover,” in *Proc. the 28th Annual Conference on Local Computer Networks*, Oct. 2003.
- [41] A. Gurtov and J. Korhonen, “Effect of vertical handovers on performance of TCP-Friendly Rate Control,” *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 8, no. 3, pp. 73–87, July 2004.

REFERENCES

- [42] E. Blanton and M. Allman, "On making TCP more robust to packet reordering," *ACM Computer Communication Review*, vol. 32, no. 1, Jan. 2002.
- [43] V. Paxson and M. Allman. RFC 2988: Computing TCP's retransmission timer. [Online]. Available: <http://www.ietf.org/rfc/rfc2988.txt>
- [44] V. Jacobson, "Congestion avoidance and control," in *Proc. SIGCOMM'88*, 1988.
- [45] A. Bakre and B. Badrinath, "I-TCP: Indirect TCP for mobile hosts," in *Proc. IEEE ICDCS*, 1995, pp. 136–143.
- [46] K. Brown and S. Singh, "M-TCP: TCP for mobile cellular networks," *ACM Computer Communication Review*, vol. 27, no. 5, 1997.
- [47] T. Goff, J. Moronski, D. S. Phatak, and V. Gupta, "Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments," in *Proc. IEEE INFOCOM*, 2000, pp. 1537–1545.
- [48] V. Tsaoussidis and H. Badr, "TCP-Probing: Towards an error control schema with energy and throughput performance gains," in *Proc. the 8th IEEE International Conference on Network Protocols*, 2000.
- [49] S. Kim and J. A. Copeland, "Tcp for seamless vertical handoff in hybrid mobile data networks," in *Proc. GLOBECOM*, 2003.
- [50] R. Ludwig and R. H. Katz, "The Eifel algorithm: making TCP robust against spurious retransmissions," *ACM Computer Communication Review*, no. 1, pp. 30–36, Jan. 2000.

Publications

- [1] H. Huang, J. Cai, A. J. Kassler and C. Fu, "Load-sharing in wireless multi-homed systems" in *Proc. IEEE International Conference on Communications*, May. 2005.
- [2] H. Huang and J. Cai, "Improving TCP performance during soft vertical handoff" in *Proc. 19th International Conference on Advanced Information Networking and Applications*, pp. 329-332, Mar. 2005.
- [3] J. Cai, H. Huang and Z. Ni, "Ubiquitous video streaming through wireless overlay networks" in *Proc. International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, Dec. 2005.
- [4] J. Cai, H. Huang, Z. Ni and C. W. Chen, "Ubiquitous map-image access through wireless overlay networks" in *Proc. SPIE vol.5600*, pp. 250-260, Oct. 2004.