TCP Performance with Vertical Handoff

Laila Daniel

Helsinki, November 2008

Licentiate Thesis C-2008-221

University of Helsinki
Department of Computer Science
P. O. Box 68 (Gustaf Hällströmin katu 2b)
FIN-00014 University of Helsinki, FINLAND
TCP Performance with Vertical Handoff

Laila Daniel
Department of Computer Science, University of Helsinki
Licentiate Thesis C-2008-221
November 2008
116 pages

Abstract. With the proliferation of wireless and mobile devices equipped with multiple radio interfaces to connect to the Internet, vertical handoff involving different wireless access technologies will enable users to get the best of connectivity and service quality during the lifetime of a TCP connection. A vertical handoff may introduce an abrupt, significant change in the access link characteristics and as a result the end-to-end path characteristics such as the bandwidth and the round-trip time (RTT) of a TCP connection may change considerably. TCP may take several RTTs to adapt to these changes in path characteristics and during this interval there may be packet losses and/or inefficient utilization of the available bandwidth. In this thesis we study the behaviour and performance of TCP in the presence of a vertical handoff. We identify the different handoff scenarios that adversely affect TCP performance. We propose several enhancements to the TCP sender algorithm that are specific to the different handoff scenarios to adapt TCP better to a vertical handoff. Our algorithms are conservative in nature and make use of cross-layer information obtained from the lower layers regarding the characteristics of the access links involved in a handoff. We evaluate the proposed algorithms by extensive simulation of the various handoff scenarios involving access links with a wide range of bandwidth and delay. We show that the proposed algorithms are effective in improving the TCP behaviour in various handoff scenarios and do not adversely affect the performance of TCP in the absence of cross-layer information.

Key Words: TCP, Vertical Handoff, Wireless Access Networks, Cross-layer Notifications, Performance Analysis
Acknowledgements

I would like to express my deep gratitude to my supervisors Professor (late) Kimmo Raatikainen and Lecturer Markku Kojo for guiding my research and giving me invaluable support. Kimmo taught several courses that helped me to get a broad understanding of the field of mobile and wireless networking and gave me much encouragement in my research work. He gave me valuable comments on the initial draft of the thesis and it is very unfortunate that I could not share the completion of my thesis with him. The countless discussions I had with Markku on a day-to-day basis contributed much to my understanding of my research topic and I am very grateful to him for all the advice and help he gave me. Markku also meticulously read several drafts of the thesis and gave numerous suggestions. I would like to thank the thesis reviewers Professor Jussi Kangasharju and Professor Jukka Manner for their suggestions and comments which helped in improving the content and the style of the thesis.

I am grateful to both the current and the former members of our project team, Ilpo Järvinen, Lauri Hyttinen, Aaron Yi Ding, Seppo Häätinen, Aki Nyriinen and Davide Astuti for creating a cordial and pleasant working environment. Special thanks to Ilpo for many lively discussions and for providing many evaluation scripts. I would also like to thank Pasi Sarolahti and Jouni Korhonen for many discussions and for the joint work which resulted in a publication.

It is a pleasure to thank Tiina Niklander for the help and advice she readily gave me whenever I needed. Her enthusiasm and friendliness lifted up my spirits on numerous occasions when I despaired about the progress of the research work. Tiina gave valuable comments which helped in organizing the thesis. I would like to thank Dr. Timo Alanko for helping me to get a research position here. My thanks to Satu Eloanta for her kindness and friendship. My thanks to my former officemate Oriana Riva for the laughter and fun we shared which brought light moments during many long days spent in office.

My thanks to the system administrators of the Computer Science Department especially Petri Kutvonen, Pekka Niklander and Ville Hautakangas for help with the computing facilities. I would like to thank the office staff of the Department for their help, especially Päivi Karimäki-Suvanto, our former office manager.

I would like to acknowledge the financial support received from the MERCoNe and WISEciti projects and I also wish to express my thanks to the partners of the projects for fruitful discussions.

To my family and friends, I have no words to express my gratitude for their love, support and prayers.
Contents

1 Introduction ........................................ 5
   1.1 Research Area ........................................ 5
   1.2 Description of the Research Work ................. 6
   1.3 Contributions ........................................ 6
   1.4 The Organization of the Thesis .................. 7

2 Wireless Access Networks ......................... 9
   2.1 Wireless Local Area Network (WLAN) ........... 9
       2.1.1 IEEE 802.11 WLAN and its Variants ......... 10
       2.1.2 IEEE 802.11 MAC Protocols ................. 11
   2.2 General Packet Radio Service (GPRS) .......... 11
       2.2.1 Global System for Mobile communications (GSM) 12
       2.2.2 GPRS Architecture .......................... 13
       2.2.3 GPRS Protocol Stack ......................... 13
   2.3 Universal Mobile Telecommunication System (UMTS) 14
   2.4 Worldwide Interoperability for Microwave Access (WiMAX) 15
   2.5 Mobile Broadband Wireless Access (MBWA) - Mobile-Fi 16
   2.6 Summary of Wireless Access Networks .......... 16
   2.7 Wireless Overlay Networks ..................... 17
   2.8 Summary ........................................... 18

3 Mobility on the Internet ......................... 21
   3.1 Mobility Management ............................ 21
   3.2 Subnetwork Layer Mobility Support ............ 22
   3.3 IP Layer Mobility Support ...................... 22
       3.3.1 Global Mobility Protocols: Mobile IP (MIP) 23
           Mobile IPv4 (MIPv4) .......................... 23
           Mobile IPv6 (MIPv6) .......................... 23
       3.3.2 Mobile IP and Handoff Execution ............ 25
           Fast Handoff Techniques ...................... 26
       3.3.3 Local Mobility Protocols .................... 27
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.4</td>
<td>Proxy Mobile IPv6 (Proxy MIPv6)</td>
<td>30</td>
</tr>
<tr>
<td>3.3.5</td>
<td>NEtwork Mobility (NEMO)</td>
<td>31</td>
</tr>
<tr>
<td>3.4</td>
<td>Host Identity Sublayer (Layer 3.5) Mobility Support</td>
<td>31</td>
</tr>
<tr>
<td>3.5</td>
<td>Transport Layer Mobility Support</td>
<td>32</td>
</tr>
<tr>
<td>3.5.1</td>
<td>TCP Migrate</td>
<td>32</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Stream Control Transmission Protocol (SCTP)</td>
<td>33</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Datagram Congestion Control Protocol (DCCP)</td>
<td>33</td>
</tr>
<tr>
<td>3.6</td>
<td>Application Layer Mobility Support</td>
<td>33</td>
</tr>
<tr>
<td>3.7</td>
<td>Summary</td>
<td>34</td>
</tr>
<tr>
<td>4</td>
<td>Transmission Control Protocol (TCP) over Wireless /Mobile Networks</td>
<td>35</td>
</tr>
<tr>
<td>4.1</td>
<td>TCP Congestion Control Algorithms</td>
<td>35</td>
</tr>
<tr>
<td>4.1.1</td>
<td>TCP Tahoe</td>
<td>37</td>
</tr>
<tr>
<td>4.1.2</td>
<td>TCP Reno</td>
<td>37</td>
</tr>
<tr>
<td>4.1.3</td>
<td>TCP NewReno</td>
<td>37</td>
</tr>
<tr>
<td>4.1.4</td>
<td>TCP Selective Acknowledgements (SACK)</td>
<td>38</td>
</tr>
<tr>
<td>4.2</td>
<td>TCP over Wireless and Mobile Networks</td>
<td>38</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Problems with Wireless and Mobile Networks</td>
<td>39</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Enhancing TCP over Wireless and Mobile Networks</td>
<td>39</td>
</tr>
<tr>
<td>4.3</td>
<td>Summary</td>
<td>41</td>
</tr>
<tr>
<td>5</td>
<td>TCP and Vertical Handoff</td>
<td>43</td>
</tr>
<tr>
<td>5.1</td>
<td>Vertical Handoff</td>
<td>43</td>
</tr>
<tr>
<td>5.2</td>
<td>Problems of TCP in Vertical Handoff</td>
<td>44</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Spurious Retransmission Timeouts (RTO)</td>
<td>45</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Packet Reordering</td>
<td>45</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Packet Bursts</td>
<td>45</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Packet Losses</td>
<td>46</td>
</tr>
<tr>
<td>5.2.5</td>
<td>Unused Connection Time</td>
<td>46</td>
</tr>
<tr>
<td>5.2.6</td>
<td>Inability to Adapt to Increased Capacity</td>
<td>46</td>
</tr>
<tr>
<td>5.3</td>
<td>Related Work</td>
<td>46</td>
</tr>
<tr>
<td>5.4</td>
<td>Summary</td>
<td>51</td>
</tr>
</tbody>
</table>
6 Experimental Study of TCP Behaviour with Vertical Handoff

6.1 Simulation Model .................................................. 53
6.2 Motivation for the Organization of the Experiments ............ 55
6.3 Analysis of the Effect of Changes in Bandwidth and Delay .... 57
   6.3.1 Changes in Delay ........................................... 57
      Handoff from a low-delay link to a high-delay link ............ 57
      Handoff from a high-delay link to a low-delay link ............ 59
   6.3.2 Changes in Bandwidth ....................................... 60
      Handoff from a high-bandwidth link to a low-bandwidth link ... 60
      Handoff from a low-bandwidth link to a high-bandwidth link ... 61
   6.3.3 Changes in Bandwidth and Delay for a Fixed Bandwidth Delay Product (BDP) .................................................. 63
      Handoff from a high-bandwidth /low-delay link to a low-bandwidth /high-delay link ............................................. 63
      Handoff from a low-bandwidth /high-delay link to a high-bandwidth /low-delay link ............................................. 64
6.4 Summary .................................................................. 65

7 Cross-Layer Enhanced TCP Algorithms ............................ 67

7.1 Cross-layer Notifications ........................................... 67
7.2 Discussion of the Proposed TCP Enhancements ................. 69
   7.2.1 TCP Enhancements to Avoid Spurious RTOs ............... 69
   7.2.2 TCP Enhancements to Minimize Congestion-related Packet Losses .................................................. 76
   7.2.3 TCP Enhancements to Reduce the Effect of Disconnection .................................................. 76
   7.2.4 TCP Enhancements for a Fast Convergence of RTO .......... 78
7.3 Performance Comparison: Regular TCP vs. Enhanced TCP .... 80
   7.3.1 Handoff between Same Bandwidth and Different Delay Links .................................................. 80
      Handoff from a low-delay link to a high-delay link ............ 80
      Handoff from a high-delay link to a low-delay link ............ 83
   7.3.2 Handoff between Same Delay, Different Bandwidth Links .................................................. 86
      Handoff from a high-bandwidth link to a low-bandwidth link ... 86
      Handoff from a low-bandwidth link to a high-bandwidth link ... 89
   7.3.3 Handoff between Links of the Same Bandwidth Delay Product (BDP) with Different Bandwidth and Delay .................................................................................................................. 89
Handoff from a high-bandwidth/low-delay link to a low-bandwidth/high-delay link .................................................. 89

Handoff from a low-bandwidth/high-delay link to a high-bandwidth/low-delay link .................................................. 91

7.4 Comparison of our algorithms with the algorithms proposed in the literature .................................................. 91

7.5 Summary .................................................................................................................................................. 93

8 Conclusions and Future Work ................................................................................................................. 99
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2G/3G</td>
<td>Second/ Third Generation Cellular Technology</td>
</tr>
<tr>
<td>3GPP</td>
<td>Third Generation Partnership Project</td>
</tr>
<tr>
<td>802.xx</td>
<td>Wireless LAN Standards developed by IEEE</td>
</tr>
<tr>
<td>ACK</td>
<td>Acknowledgment</td>
</tr>
<tr>
<td>AP</td>
<td>Access Point</td>
</tr>
<tr>
<td>AR</td>
<td>Access Router</td>
</tr>
<tr>
<td>ARQ</td>
<td>Automatic Repeat reQuest</td>
</tr>
<tr>
<td>BBM</td>
<td>Break-before-make</td>
</tr>
<tr>
<td>BDP</td>
<td>Bandwidth Delay Product</td>
</tr>
<tr>
<td>BER</td>
<td>Bit-Error Rate</td>
</tr>
<tr>
<td>BS</td>
<td>Base Station</td>
</tr>
<tr>
<td>BSC</td>
<td>Base Station Controller</td>
</tr>
<tr>
<td>BSS</td>
<td>Base Station Subsystem</td>
</tr>
<tr>
<td>BTS</td>
<td>Base Transceiver Stations</td>
</tr>
<tr>
<td>BU</td>
<td>Binding Update</td>
</tr>
<tr>
<td>CDMA</td>
<td>Code Division Multiple Access</td>
</tr>
<tr>
<td>CIP</td>
<td>Cellular IP</td>
</tr>
<tr>
<td>CoA</td>
<td>Care-of Address</td>
</tr>
<tr>
<td>CTS</td>
<td>Clear to Send</td>
</tr>
<tr>
<td>DAD</td>
<td>Duplicate Address Detection</td>
</tr>
<tr>
<td>DCCP</td>
<td>Datagram Congestion Control Protocol</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Control Protocol</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name Service</td>
</tr>
<tr>
<td>DRR</td>
<td>Domain Root Router</td>
</tr>
<tr>
<td>DS</td>
<td>Distribution System</td>
</tr>
<tr>
<td>DSSS</td>
<td>Direct Sequence Spread Spectrum</td>
</tr>
<tr>
<td>ECN</td>
<td>Explicit Congestion Notification</td>
</tr>
<tr>
<td>EDGE</td>
<td>Enhanced Data Rates for GSM Evolution</td>
</tr>
<tr>
<td>EIR</td>
<td>Equipment Identity Register</td>
</tr>
<tr>
<td>ESS</td>
<td>Extended Service Set</td>
</tr>
<tr>
<td>ETSI</td>
<td>European Telecommunications Standards Institute</td>
</tr>
<tr>
<td>FA</td>
<td>Foreign Agent</td>
</tr>
<tr>
<td>FBack</td>
<td>Fast Binding Acknowledgement</td>
</tr>
<tr>
<td>FBU</td>
<td>Fast Binding Update</td>
</tr>
<tr>
<td>FEC</td>
<td>Forward Error Correction</td>
</tr>
</tbody>
</table>
FHSS          Frequency Hopping Spread Spectrum
FNA           Fast Neighbour Advertisement
FTP           File Transfer Protocol
GGSN          Gateway GPRS Support Node
GMSC          Gateway MSC
GPRS          General Packet Radio Service
GSM           Global System for Mobile Communications
GTP           GPRS Tunnel Protocol
GWCS          General Wireless Communications Service
HA            Home Agent
HACK          Handover ACK
HAWAIH        Handoff-Aware Wireless Access Internet Infrastructure
HI             Handover Initiate/Host Identity
HIP            Host Identity Protocol
HIT            Host Identity Tag
HLR            Home Location Register
HMIPv6         Hierarchical Mobile IPv6
HSCSD         High Speed Circuit Switched Data
HSDDPA        High Speed Downlink Packet Access
HSDSCH        High Speed Downlink Shared Channel
HTML          Hypertext Markup Language
IBSS          Independent Basic Service Set
IEEE          Institute of Electrical and Electronic Engineers
IETF          Internet Engineering Task Force
IP             Internet Protocol
IS95          CDMA Cellular Standard
ISDN          Integrated Services Digital Network
ISO           International Organization for Standardization
LAN           Local Area Network
LCoA          Local Care-of Address
LERS          Localized Enhanced Routing Schemes
LFN           Long Fat Network
LLC           Logical Link Control
LMA           Local Mobility Anchor
LSI           Local Scope Identity
LTN           Long Thin Network
MAC           Medium Access Control
MAG           Mobile Access Gateway
MAP           Mobility Anchor Point
MBB           Make-before-break
MBWA          Mobile Broadband Wireless Access (Mobile-Fi)
MIMO          Multiple Input Multiple Output
MIP           Mobile IP
MIPv4, MIPv6   MIP Version 4, MIP Version 6
MN    Mobile Node
MR    Mobile Router
MS    Mobile Station
MSC   Mobile Switching Center
MSS   Maximum Segment Size
MTU   Maximum Transmission Unit
NAR   New Access Router
NEMO  NEtwork MObility
NetLMM Network-based Localized Mobility Management
OFDM  Orthogonal Frequency Division Multiplexing
OSI   Open Systems Interconnection
PAA   Proxy Agent Architecture
PAR   Previous Access Router
PCF   Point Coordination Function
PDCH  Packet Data Channel
PMP   Point to Multipoint
PrTAdv Proxy Router Advertisement
PSTN  Public Switched Telephone Network
QoS   Quality of Service
RADIUS Remote Authentication Dial In User Service
RAN   Radio Access Network
RCoA  Remote Care-of Address
RFC   Request For Comments
RLP   Radio Link Protocol
RNCH  Radio Network Controller
RTO   Retransmission Timeout
RTP   Real Time Transport Protocol
RTS   Request to Send
RtSolPr Router Solicitation for Proxy Advertisement
RTT   Round-Trip Time
SCTP  Stream Control Transmission Protocol
SGSN  Serving GPRS Support Node
SIP   Session Initiation Protocol
TCP   Transmission Control Protocol
UE    User Equipment
UMTS  Universal Mobile Telecommunications System
UTRAN UMTS Terrestrial Radio Access Network
VLR   Visitor Location Register
WAN   Wide Area Network
WCDMA Wideband CDMA
WiFi  Wireless Fidelity
WiMAX Worldwide Interoperability for Microwave Access
WLAN  Wireless LAN
Chapter 1

Introduction

The vast growth of Internet and mobile telephony in recent years has created the exciting prospect of the inter-weaving of the two technologies leading to the Wireless Internet era [81,92,116]. In this emerging scenario it is expected that a wide range of new mobile services and applications will be available for use in a seamless way regardless of their geographical location. In this context, the services and applications will have the need to take into account the characteristics of the mobile environments to operate satisfactorily [116]. In a typical mobile environment, during the lifetime of a connection, the connectivity of a mobile device may change very much across access networks that have significantly different bandwidth, latency and error characteristics. Mobile devices are often equipped with multiple radio interfaces to connect to access networks using diverse link technologies; for example, a mobile phone with Wireless LAN (WLAN) and Wireless WAN (WWAN) interfaces. This is necessary to support the best of connectivity, service quality, application needs, and user preferences.

Transmission Control Protocol (TCP) [115] is the dominant transport protocol in the Internet that provides reliable, in-order, full-duplex delivery of data to applications. TCP supports a variety of applications including file transfer, e-mail, world-wide web, peer-to-peer file sharing and streaming audio and video [30,123,136,142]. To make TCP cope with the changes in the mobile environment, it is desirable to explore how TCP can make use of the information available about the link characteristics in adapting its behaviour.

1.1 Research Area

Handoff refers to the switching between access networks in a mobile environment. It is categorized into horizontal handoff and vertical handoff depending on whether the handoff is within the same access technology or between different access technologies [97].

Access networks with different link layer technologies vary widely in their characteristics such as link bandwidth, latency, bit-error rate and degree of bandwidth asymmetry. The wireless access link is commonly the last-hop or the first-hop link and is usually the bottleneck link on the end-to-end path. A significant change in the access link characteristics can easily affect the end-to-end path
properties and thereby the behaviour of transport protocols. In the case of TCP, a vertical handoff may result in packet losses, intermittent connectivity, packet reordering and spurious or too late retransmission timeouts (RTOs) causing unnecessary TCP congestion response or inefficient loss recovery affecting TCP performance.\cite{53,56,59,84,85,127,130,131}

When a vertical handoff occurs, the TCP sender adjusts its transmission rate and RTO estimate very slowly to the new end-to-end path as it learns the properties of the new path implicitly by probing it over several round trips. If the TCP layer is explicitly notified about the changes in the path properties, the TCP sender could react in a more timely manner and also more efficiently and possibly avoid false congestion responses. The TCP layer on the mobile node (MN) can be notified locally of the changes in the attached access link characteristics, but the TCP layer on the correspondent node (CN) is completely unaware of the changes. In order to inform the TCP sender on the CN many proposals have been introduced recently for delivering information from the MN to the CN about the path change due to a handoff.\cite{91,110,130,131} While the knowledge of the new access link characteristics cannot be used to reliably determine the new end-to-end path properties, learning about significant changes in the last-hop (first-hop) link characteristics can be used as a useful hint about a potential change in the end-to-end path properties.

1.2 Description of the Research Work

In this thesis we study the effect of vertical handoff on TCP and propose a set of algorithms that can be implemented at the TCP sender both at the CN and MN to improve the TCP performance in the presence of vertical handoff. These algorithms are invoked as a response to an explicit cross-layer indication notifying the TCP sender about the changes in the access link bandwidth and/or delay. As MN knows the occurrence of handoff it can send the handoff notification to the TCP sender at the CN as an explicit cross-layer indication along with the mobility signalling messages or as local cross-layer notifications to the TCP sender at MN. We identify the various handoff scenarios which affect the TCP behaviour. We use the ns-2 simulator \cite{108} to study vertical handoff with various combinations of delay and bandwidth of the last-hop link. We demonstrate that our proposed enhancements are effective in avoiding spurious RTOs, reducing packet losses due to change in the capacity of the links, improving the link utilization immediately after a disconnection and converging to the RTO value of the new end-to-end path quickly. With the proposed algorithms TCP performance is improved in many of the handoff scenarios and in some scenarios the improvement is more than a factor of 2. Our proposed enhancements are conservative in nature and do not adversely affect the TCP performance when the cross-layer notification is unavailable.

1.3 Contributions

The motivation for our research work is to carry out a systematic study of the effect of vertical handoff on TCP behaviour and use this study as a basis for designing algorithms that are both conservative in nature and easy to implement to improve TCP performance with vertical handoff. Though several studies have been reported in the literature \cite{53,56,59,84,85,127,130,131} on the behaviour of TCP in vertical handoff, we are not aware of any systematic study of TCP behaviour in
1.4 THE ORGANIZATION OF THE THESIS

A handoff for a wide range of bandwidth and delay of the access links. As the wireless access networks in the Internet exhibit a wide range of bandwidth, delay, error characteristics and geographical coverage our study of the problem is particularly relevant. Our contributions in this thesis are based on the publications [31–33] from this research work and are described below.

○ Our first research work in this area studied the behaviour of TCP in vertical handoff in the specific setting of GPRS and WLAN access networks and it was published as a conference paper [31]. In this paper we proposed a number of simple enhancements to the TCP sender algorithm that make use of the information on whether a significant change in the access link bandwidth and/or delay due to a handoff has occurred or not. This work was done in collaboration with the coauthor, Markku Kojo. The design of the algorithms, planning of the experiments and the analysis of the results were carried out with Markku Kojo while the ns-2 implementation for the simulation and the simulation experiments were done by the author. The simulation model used in our paper was adapted from the model developed by Pasi Sarolahti, Markku Kojo and the author in the paper [127].

○ Our second paper [32] was devoted to a thorough study of TCP behaviour in vertical handoff between access networks involving a wide range of the access link bandwidth and delay. Our experiments clearly show how the different handoff scenarios affect TCP and identify those instances where TCP is particularly sensitive to handoff. In our view, this study can provide a useful basis for evaluating mechanisms designed to improve TCP performance in a vertical handoff across a wide range of access network characteristics. The planning of the experiments and the analysis of the results were carried out with the coauthor Markku Kojo. The simulation model used here is the same as in our first paper [31]. The ns-2 implementation for the simulation and the simulation experiments were carried out by the author.

○ Based on our study [32], we developed algorithms implemented at the TCP sender to improve TCP performance with vertical handoff for wide ranging changes in access link bandwidth and delay and this work was reported in our paper [33]. Our algorithms make use of the explicit cross-layer information about the access link bandwidth and delay delivered to the TCP sender. We show that the proposed enhancements to the TCP sender algorithm are effective in avoiding spurious RTOs, in reducing packet losses due to changes in the capacity of the access links, in improving the link utilization immediately after a disconnection and in converging quickly to the RTO value of the new end-to-end path. The setup for the experiments is identical to that in [32]. This work was carried out in collaboration with the coauthor, Markku Kojo. Both authors were involved in the design of the algorithms, planning of the experiments and the analysis of the results while the ns-2 implementation for the simulation and the simulation experiments were carried out by the author.

1.4 The Organization of the Thesis

The rest of the thesis describes the research problem and its background followed by a detailed description of our experiments, analysis and results towards its solution. Chapter 2 gives an overview of wireless access networks, the setting for vertical handoff. Chapter 3 presents an overview of mobility in the Internet along with the approaches and trade-offs involved in implementing mobility. Chapter 4 describes the TCP congestion control algorithms and the various enhancements that have
been proposed to adapt TCP to wireless mobile environments. Chapter 5 describes the various problems affecting TCP in vertical handoff and provides a literature survey of the previous research work in this area. Chapter 6 describes our study of the TCP behaviour in vertical handoff for a wide range of bandwidth and delay of the access links. Chapter 7 describes the enhanced TCP algorithms proposed in the thesis to improve TCP performance with vertical handoff and presents a comparison of the enhanced TCP and regular TCP. The thesis concludes with a summary of our study and its results and describes the directions for future research.
Chapter 2

Wireless Access Networks

Wireless access networks are the end-user radio connections to public or private core networks. With the growth of the wireless access to the Internet, these access networks play a significant role in the evolution of the wireless Internet. The versatility of these networks has spanned a wide variety of such access networks. A single mobile device equipped with multiple radio interfaces to these wireless access networks is common today. The heterogeneity of the characteristics of wireless access networks poses a special challenge to adapt TCP to the dynamically changing environment arising from handoffs often involving networks with significantly different characteristics such as bandwidth and propagation delay. The focus of this thesis is to address the problems of TCP in dealing with handoff in a dynamic wireless environment.

This chapter gives an overview of the common wireless access network technologies currently in use. The initial sections of this chapter describe Wireless LAN, GPRS/EGPRS, UMTS, WiMAX and Mobile-Fi respectively and we round off this discussion with a comparison of their characteristics pertaining to their impact on TCP performance. The final section of the chapter briefly describes the architectures enabling the integration of the different access networks. As our study includes the range of bandwidth and delay of these specific access networks described in this chapter, our proposed solutions are relevant to these wireless access networks.

2.1 Wireless Local Area Network (WLAN)

IEEE 802.11 WLAN [15,65] has emerged as a popular wireless technology which provides broadband access within a small geographical region (a typical radius of tens of meters). The IEEE 802.11 WLAN standard was introduced in the year 1997, revised in 1999 and reaffirmed in 2003. It aims at “providing wireless connectivity to automatic machinery, equipment or stations that require rapid deployment, which may be portable or hand-held, or which may be mounted on moving vehicles within local area” [65].

The basic form of IEEE 802.11 WLAN is called a Basic Service Set (BSS). There are two forms of BSS, infrastructure BSS and independent BSS. The infrastructure BSS shown in Figure 2.1 consists of an Access Point (AP) and a number of stations associated with an AP. Each AP acts as a
Figure 2.1: 802.11 Infrastructure mode WLAN. (Adapted from [15])

bridge between the wireline infrastructure and the wireless link with its mobile nodes (MNs). In the independent BSS, several mobile nodes can communicate simultaneously with one another in an adhoc mode with a direct path between two communicating nodes.

Access points are the base stations in a wireless network. They use radio frequencies to transmit and receive information to enable the communication of wireless devices. A single AP covers a small geographical area and multiple APs are used to cover a wide geographical area or enterprise networks. Usually a wireline Ethernet is used to connect the multiple APs and such a system of AP networks is called a Distribution System (DS). APs can also be connected via wireless links. A set of BSSs along with a DS is called an Extended Service Set (ESS). An ESS is identified by its service-set identification (SSID). SSID is often referred to as the access network name.

The IEEE 802.11 standard covers the physical and MAC-sublayer of the ISO/OSI reference model [75]. The logical link control (LLC) is defined in the IEEE 802.2 LAN standard. This allows existing protocols to run over IEEE 802.11. The IEEE 802.2 LLC is also used in IEEE 802.3 (Ethernet) and IEEE 802.5 (Token Ring) LANs. So the existing protocols such as TCP/IP can be implemented over WLANs just as in a wired Ethernet.

2.1.1 IEEE 802.11 WLAN and its Variants

The IEEE 802.11 defines three standards for the physical layer: Infra Red (IR), Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS). All these three standards support bit rates of 1 to 2 Mbps. Both FHSS and DSSS operate at 2.4 GHz.

In the year 1999 IEEE defined two high-rate extensions of IEEE 802.11. The IEEE 802.11a [61] extension uses Orthogonal Frequency Division Multiplexing (OFDM) and provides data rates ranging from 6 Mbps to 54 Mbps. It operates in the 5 GHz frequency band. Forward Error Correction (FEC) is used in IEEE 802.11a for error correction to improve the transmission quality.

The IEEE 802.11b [62] extension which operates at 2.4 GHz uses DSSS technology with two different modulation techniques to produce data rates of either 1 Mbps or 2 Mbps. Using a modulation technique called Complementary Code Keying (CCK) with DSSS, IEEE 802.11b could provide
data rates of 5.5 Mbps and 11 Mbps. In this extension it is possible for a high data rate network to slow the rate down to 1 or 2 Mbps. There is no in-built FEC scheme in this standard.

IEEE 802.11g [66] introduced in the year 2003 as an extension to IEEE 802.11b supports data rates up to 54 Mbps in the 2.4 GHz band. IEEE 802.11g is backward compatible with IEEE 802.11b but not compatible with IEEE 802.11a. The backward compatibility of 802.11g is considered to be a disadvantage as an AP running at the high rate of 802.11g will lower its rate to that of IEEE 802.11b when any IEEE 802.11b device is connected to it. Another standard IEEE 802.11h [69] is an enhancement of IEEE 802.11a and it is aimed at providing a data rate up to 100 Mbps in the 5 GHz band.

The still emerging standard IEEE 802.11n Draft 3.2 [73] has released in November 2007) proposes a high data rate of at least 100 Mbps. This data rate is achieved by using Multiple Input Multiple Output (MIMO) antenna and adaptive OFDM technologies.

### 2.1.2 IEEE 802.11 MAC Protocols

The IEEE 802.11 defines two MAC layer protocols, namely Distributed Contention Function (DCF) and Point Coordination Function (PCF). The DCF is a contention protocol based on Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA). It uses small Request to Send / Clear to Send (RTS/CTS) packets to reserve the medium and thereby avoids the needless collision of packets. The sender station sends a short RTS frame before the transmission of each data frame and if the receiver is ready to receive the frame, it sends a short CTS frame. The sender transmits the data frame after it gets the CTS frame. All other stations defer their transmissions when they hear any RTS, CTS or data frame and update their timers. Such a station refrains from transmission until its timer reaches zero. As DCF is a random access scheme its performance suffers under heavy loads. PCF is a synchronous service which uses a polling-based contention-free access scheme. PCF uses a centralized polling scheme, where an AP is the Point Coordinator (PC). It has the drawback of being a centralized scheme.

There are three MAC layer extensions of IEEE 802.11. The IEEE 802.11e [71] introduced in 2005 provides QoS support for a variety of multimedia services over 802.11a, 802.11b and 802.11g by defining classes based on service differentiation. The IEEE 802.11f defines an Inter-AP protocol to allow stations to roam between multi-vendor APs. IEEE 802.11i [68] proposes enhanced security and authentication mechanisms for the 802.11 MAC protocols. The specifications of 802.11a, b, e, g, h and i merged into a single document called IEEE 802.11-2007 [72] in March 2007.

### 2.2 General Packet Radio Service(GPRS)

The first-generation cellular systems, commonly referred to as 1G systems were introduced in the 1980s and used analog radio signals. The 2G systems which emerged around the 1990s used digital encoding of the voice signals and they were based on different radio technologies such as frequency-, time- and code division-multiple access. GSM (Europe), IS-95 (USA) and PDC (Japan) are examples of 2G systems. 2G systems use circuit-switching and had a transmission rate of 10 -20 Kbps. GSM system is the most widely deployed digital cellular network. In 1997, an enhanced
version of GSM, *GSM Phase 2+*, introduced *General Packet Radio Service* (GPRS), a packet-switched extension of GSM [17,24]. GSM combined with GPRS is often known as 2.5 G systems.

### 2.2.1 Global System for Mobile communications (GSM)

*Global System for Mobile communications* (GSM) architecture is illustrated in Figure 2.2. In GSM network [117] the mobile nodes connect to the *Base Transceiver Stations* (BTS) in their immediate vicinity. Each BTS caters to the mobile nodes in a relatively small geographical area around it called a *cell*. Several BTSs are gathered under a *Base Station Controller* (BSC). BSC handles the access to the medium, radio resource scheduling and data transfer from/to the mobile node. Many BSCs are connected to a Mobile Switching Centre (MSC). Thus the combined traffic from various mobile nodes are routed through an MSC. The *Gateway MSC* (GMSC) handles the connections to a fixed network such as a *Public Switched Telephone Network* (PSTN) or *Integrated Services Digital Network* (ISDN). Several databases are used for call control and network management. These databases are the *Home Location Register* (HLR), the *Visitor Location Register* (VLR), the *Authentication Centre* (AUC) and the *Equipment Identity Register* (EIR). HLR stores the permanent (such as user’s current location) of all the users registered with a network operator. VLR contains data of those users who are currently served. AUC stores the authentication information for a user and the equipment data is stored in the EIR register.

Initially GSM had a data rate of 9600 bps for a single user. Later with the introduction of *High-Speed Circuit Switched Data* (HSCSD) it has become possible to have data rates up to 57.6 Kbps. Since GSM uses circuit switching, a dial-up connection with a modem is needed to transmit TCP/IP packets over the GSM network.

![GSM/GPRS System Architecture](image-url)

*Figure 2.2: GSM/GPRS system architecture. (Adapted from [17])*
2.2.2 GPRS Architecture

Figure 2.2 illustrates the GPRS system architecture showing the main network nodes and the interfaces between them. The use of packet switching in GPRS allows an efficient utilization of the resources of the network which is vital for bursty traffic applications such as web browsing. The addition of two principal nodes, Serving GPRS Support node (SGSN) and Gateway GPRS Support Node (GGSN), support the enhancements to the existing GSM network to provide the upgraded service. SGSN handles the routing of packet-switched data to and from the mobile node, user mobility, logical link management besides the authentication and the accounting functions. The GGSN provides the connectivity to an external packet network such as Internet and X.25. GGSN converts the GPRS packets to the Packet Data Protocol (PDP) format before sending them to the external data network and also converts the external PDP address to the GSM address of the destination user which it sends to the corresponding SGSN. Furthermore GGSN also performs billing and authentication.

2.2.3 GPRS Protocol Stack

The GPRS protocol stack follows a layered architecture up to layer 3 of the ISO/OSI reference model and makes a distinction between the signalling protocols and transmission protocols. The main functions of the signalling protocols are connection establishment and tear down, routing and mobility management. The transmission protocols deal with the transmission of user data and the associated control functions like flow control and error control.

The GPRS Tunnel Protocol (GTP) exists between SGSN and GGSN. GTP takes care of the routing of protocol data units (PDUs) from external network to the SGSN serving the MN and also performs the routing of the outgoing PDUs to the proper GGSN. Subnetwork Dependent Convergence Protocol (SNDCP) lies between SGSN and MN. It provides functions such as encapsulation of network layer data to the data link layer and multiplexing of several network layer messages to a single virtual logical link at the data link layer.

The data link layer is split into two sublayers, namely the Radio Link Control/Medium Access Control (RLC/MAC) (the lower sublayer) and the Logical Link Control (LLC) (the upper sublayer) [42, 43]. The RLC protocol is specific to a radio technology while the LLC protocol is independent of the wireless interface characteristics. Both LLC and RLC can operate in the acknowledged and unacknowledged modes. The LLC protocol controls the data transfer between MN and SGSN and in the acknowledged mode LLC recovers using Automatic Repeat reQuest (ARQ). The functionality of the LLC protocol is based on LAP-D with enhanced features like point-to-multipoint transmission. The RLC protocol operates between MN and BSC in both uplink and downlink directions. Each LLC frame is segmented into several RLC data blocks of fixed size. Each RLC block occupies a fixed number of slots depending on the channel coding schemes. When RLC functions in the acknowledged mode, it provides a selective retransmission of erroneous blocks.

The MAC layer is primarily responsible for the efficient sharing of the common radio channel among different MNs. It operates on a slotted-ALOHA-based reservation protocol [122]. The MAC layer performs contention resolution, efficient multiplexing of data and QoS control.
GPRS can provide a data rate up to 80 Kbps. GPRS was enhanced with a channel modulation scheme called \textit{Eight Phase Shift Keying (8PSK)} that triples the transmission rates available in GPRS. This enhanced GPRS is called EGPRS or \textit{Enhanced Data Rate for GSM Evolution (EDGE)} [132]. EGPRS provides a transmission rate ranging from 128 Kbps to 473 Kbps (theoretical maximum) [102].

### 2.3 Universal Mobile Telecommunication System (UMTS)

Next in the line of cellular network evolution is the third-generation (3G) systems which aim at providing universal connectivity and global mobility with a wide range of services including telephony, messaging and Internet access. UMTS is a third-generation cellular network. UMTS is being developed by \textit{Third Generation Partnership Project (3GPP)} [1].

![UMTS Architecture](image)

\textbf{Figure 2.3:} UMTS architecture. Adapted from [60]

The UMTS [2–6] network consists of three interactive domains: \textit{Core Network} (CN), \textit{UMTS Terrestrial Radio Access Network} (UTRAN) and \textit{User Equipment} (UE). The UMTS architecture with network elements and interfaces is shown in Figure 2.3. UMTS also supports GSM systems.

The UMTS Core Network is based on the GSM Phase 2+ network with GPRS and it provides both circuit switching and packet switching. The circuit-switched domain consists of MSC, VLR and GMSC. The SGSN and GGSN are the main elements in the packet-switched domain. EIR, HLR, VLR and AUC are shared by both circuit-switched and packet-switched domains. The Core Network is responsible for switching, routing and transit of user data to external networks such as PSTN, ISDN or Internet.
UTRAN provides the air interface access of the User Equipment. UTRAN consists of Node B and Radio Network Controller (RNC). Node B is equivalent to BTS in GSM and these nodes are connected to the RNC. UE consists of Mobile Equipment (ME) and UMTS Subscriber Identity Module (USIM). ME handles the radio functionalities of a mobile node and USIM is a chip card which contains the subscriber information and the keys used for data encryption.

UTRAN air interface is based on Wideband CDMA (WCDMA) technology and it can provide a maximum date rate of 2 Mbps [54]. WCDMA uses the RLC protocol [8] which is a selective repeat sliding window ARQ. In the RLC protocol the maximum number of retransmissions is a configurable parameter i.e., the persistence level in the RLC protocol can be high or low. Besides ARQ, WCDMA uses FEC for error correction. The use of ARQ and FEC schemes will increase the link delay.

UMTS High Speed Down Link Packet Access (UMTS/HSDPA) [9] is an enhanced version of a packet-based data service with data transmission capability up to 8-10 Mbps with a link delay around 100 ms. It uses WCDMA technology for the downlink in the 5 MHz band. With a multiple antenna support through Multiple-Input Multiple-Output (MIMO) systems the data transmission rate can be increased up to 20 Mbps. HSDPA uses hybrid ARQ [135] for error recovery which increases the reliability of the link.

### 2.4 Worldwide Interoperability for Microwave Access (WiMAX)

IEEE 802.16 [39, 63, 70] is an emerging standard for global broadband wireless access supporting fixed, nomadic, portable and mobile operations. IEEE 802.16 aims at providing the coverage of Wireless Metropolitan Area Networks (WMAN) under both line-of-sight and non-line-of-sight radio conditions at comparable or higher bandwidth of WLAN.

Worldwide Interoperability for Microwave Access (WiMAX) Forum [146] is a non-profit organization formed in June 2001 to promote conformance and interoperability of the IEEE 802.16 standard.
According to WiMAX forum, WiMAX is a standards-based technology enabling the delivery of last-mile wireless broadband access as an alternative to wired broadband like cable and DSL [147]. WiMAX supports data rates from 1 Mbps to 40 Mbps and has a latency of less than 100 ms.

The IEEE 802.16 standard was first published in 2001 and was followed by its amendments a/b/c to address issues related to radio spectrum, inter-operability and QoS issues. The IEEE 802.16-2004 [67] replaces the original standard and the a/b/c amendments and it describes a standard for fixed access networks. IEEE 802.16e standard [70] introduced in 2005 addresses the issue of mobility. Mobile WiMAX [148,149] is compatible with the IEEE 802.16e standard.

The WiMAX architecture is shown in Figure 2.4. WiMAX supports two topologies: Point to Multipoint (PMP) and mesh topology. In PMP, the subscriber station (SS) is connected to a WiMAX base station (BS). In mesh topology, an SS can communicate through another SS and it is possible to reach a third SS without installing a new BS. Figure 2.4 illustrates the fixed access and nomadic/mobile modes of WiMAX. The fixed access network is the basic service model in WiMAX where the access is provided through a fixed antenna like a satellite television subscriber station. The WiMAX terminal includes an outdoor unit and an antenna. This mode does not support portability and handoff between APs. The nomadic mode allows a terminal to access the operator’s network from different APs. The portable mode of operation allows handoff at walking speed. The WiMAX terminal is integrated with a portable device, for instance, the terminal integrated with the PCMCIA card of a laptop computer. In all the above three modes WiMAX can provide a data rate of 40 Mbps per channel within a cell radius of 10 kilometers. The fully mobile mode allows seamless connectivity in a mobile environment and it aims at providing low latency, low packet loss and real-time handoffs between APs at vehicular speeds of 120 Km/h or higher. It can support 15 Mbps of bandwidth with a latency of less than 100 ms.

In contrast to IEEE 802.11, IEEE 802.16 MAC is connection-oriented. All services, including inherently connectionless services are mapped to a connection. This provides a mechanism for requesting bandwidth, associating QoS and traffic parameters. The WiMAX physical layer uses scalable OFDM and multiple antenna support through MIMO to support high data rate capability.

2.5 Mobile Broadband Wireless Access (MBWA) - Mobile-Fi

Mobile Broadband Wireless Access (MBWA) introduced in the year 2002, also known as Mobile-Fi, is IEEE 802.20 in its technical specifications. It envisages an efficient packet-based air interface operating in licensed bands below 3.5 GHz optimized for the transport of IP-based services [64]. It promises a ubiquitous, always-on and inter-operable multi-vendor mobile broadband wireless access network. Mobile-Fi can deliver broadband Internet access with peak data rates of 1 Mbps. It can support vehicular mobility classes up to 250 Km/h.

2.6 Summary of Wireless Access Networks

The Table 2.1 provides a summary of the characteristics of the various wireless access networks that have been described in this chapter. We can observe from the table the wide variation in data rate, propagation delay and mobility of the various access networks.
### Table 2.1: Comparison of the Wireless Access Network Technologies

<table>
<thead>
<tr>
<th>Access Network</th>
<th>Standard</th>
<th>Data rate</th>
<th>Propagation Delay (one way)</th>
<th>Mobility</th>
</tr>
</thead>
<tbody>
<tr>
<td>WLAN</td>
<td>IEEE 802.11b</td>
<td>1,2,5,5,11 Mbps</td>
<td>1-10 ms</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>IEEE 802.11a</td>
<td>Up to 54 Mbps</td>
<td>1-10 ms</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>IEEE 802.11g</td>
<td>Up to 54 Mbps</td>
<td>1-10 ms</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>IEEE 802.11h</td>
<td>Up to 100 Mbps</td>
<td>1-10 ms</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>IEEE 802.11n</td>
<td>Up to 540 Mbps</td>
<td>1-10 ms</td>
<td>Low</td>
</tr>
<tr>
<td>2G</td>
<td>GSM</td>
<td>9.6/57.6 Kbps</td>
<td>350 ms</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>GPRS</td>
<td>171.2 Kbps</td>
<td>350 ms</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>EDGE/EGPRS</td>
<td>236 Kbps</td>
<td>300 ms</td>
<td>High</td>
</tr>
<tr>
<td>3G</td>
<td>UMTS/WCDMA</td>
<td>Up to 2 Mbps</td>
<td>150 ms</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>HSDPA</td>
<td>Up to 20 Mbps</td>
<td>50 ms</td>
<td>High</td>
</tr>
<tr>
<td>WiMax</td>
<td>IEEE 802.16</td>
<td>1 to 40 Mbps</td>
<td>50 ms</td>
<td>High</td>
</tr>
<tr>
<td>Mobile-Fi</td>
<td>IEEE 802.20</td>
<td>1 Mbps (peak)</td>
<td>50 ms</td>
<td>High</td>
</tr>
</tbody>
</table>

### 2.7 Wireless Overlay Networks

As can be seen from the Table 2.1, no single wireless access network is capable of providing high bandwidth, low-latency, and wide-area connectivity to a large number of users simultaneously. For example, the cellular networks such as GPRS and UMTS provide a wide-area connectivity with low data rates to users with high mobility. On the other hand, WLAN offers much higher data rates to users with low mobility. A mobile node equipped with radio interfaces to WLAN and GPRS/UMTS allows a user to combine the benefits of high data rate of WLAN whenever available with the low data rate of GPRS/UMTS always-on connectivity. An architecture which provides an integration of a variety of wireless access networks with diverse characteristics by combining them in a hierarchical order ranging from low-delay/high-bandwidth/low-range access networks to high-delay/low-bandwidth/wide-range is called a wireless overlay network [137]. A wireless overlay network allows a mobile user to roam in the network by seamlessly connecting to the appropriate wireless access network with little disruption in connectivity and in a manner transparent to the application.

We now describe an architecture which provides the internetworking of WLAN and GPRS/UMTS (3G) access networks. There are two main internetworking approaches for integrating WLAN and GPRS/UMTS [7, 10, 25], namely tightly-coupled architecture and loosely-coupled architecture as shown in Figure 2.3. In a tightly coupled architecture, a WLAN appears as another 3G access network to the 3G core network providing a seamless service continuation across 3G and WLAN. The WLAN gateway hides the details of the WLAN network from the 3G core network and implements the 3G protocol functionalities such as mobility management and authentication needed for a 3G access network. The WLAN gateway is connected to the SGSN and the WLAN uses the same authentication and billing procedures as the 3G network. As the WLAN is connected to a 3G core network, it requires that the same operator must own the WLAN and the 3G parts of the
network. The 3G network elements are to be reconfigured to accommodate the increased load from the WLAN.

In the loosely-coupled architecture, the WLAN is connected to the Internet through the WLAN gateway and it does not have any direct connection to the 3G core network or to any 3G network elements. The WLAN can have its own mobility management, authentication and billing procedures. In this approach the data paths for the WLAN and 3G are different and the high speed data of WLAN will not be injected to the 3G network.

2.8 Summary

Wireless access networks are the end user radio connections to support mobility in the Internet. In this chapter we described some of the common wireless access technologies such as WLAN, GPRS, UMTS, WiMax and MBWA that have evolved in the recent past. We presented a brief overview of the architecture of these networks along with a description of their characteristics such as bandwidth, propagation delay and geographical coverage. A comparative view of the characteristics of the different wireless access networks is given in the Table 2.1. It shows the wide disparity in their data rate, latency and mobility.

We also described the integration of wireless access networks in a wireless overlay network architecture with particular reference to the the approaches for internetworking of WLAN and GPRS/UMTS.
With the proliferation of wireless access to the Internet, wireless access technologies play a basic role in the Internet infrastructure. The widely varying characteristics of these networks such as the available bandwidth to an MN (from kilobits per second to megabits per second) and the link latency (from a few milliseconds to a few hundreds of milliseconds) coupled with the dynamic changes in path characteristics due to mobility make the problem of how well transport protocols such as TCP which learn the path characteristics by measurements obtained from probing the network can efficiently adapt their behaviour to the changes important and it is the focus of the research work described in the thesis.
Chapter 3

Mobility on the Internet

This chapter describes mobility on the Internet. It gives an overview of the approaches to implement mobility at different layers of the Internet protocol suite and discusses the advantages and disadvantages of each of the implementations by studying a representative protocol of each layer.

3.1 Mobility Management

A multi-access wireless environment increases nomadicity in the Internet and allows a mobile node (MN), to send and receive data irrespective of its location. The entity in communication with MN is called the correspondent node (CN) and it can be either static or mobile. In the Internet, any device is identified by its IP address. Also the network layer IP address is used for routing purposes i.e., IP address is associated with a fixed network location. This dual role played by the IP address necessitates the reconfiguring of the IP address when a device moves from its original location (home network) to another location (foreign network) in order to have a seamless communication [18]. As TCP uses IP address and port numbers to identify a session, reconfiguring the IP address as the device changes its location should be transparent to TCP and to higher layers.

Mobility on the Internet giving emphasis to the wireless environment encompasses both location management and handoff management. Location management deals with the IP address reconfiguration and the associated signalling when the device changes its point of attachment to the network.

Handoff is another aspect of mobility where an active MN changes its point of attachment to the network [97]. Handoff management should aim at reducing the disconnection period and the packet losses during handoff. Handoff can also be classified into two types based on the connectivity to the old access router during the handoff, namely break-before-make (BBM) handoff and make-before-break (MBB) handoff. In a break-before-make handoff, the MN’s connection to the old access router breaks before it connects to the new access router, causing disruption in connectivity often resulting in packet losses. In a make-before-break handoff, the connection to the old access router is torn down only after the connection to the new access router is operable. So during a make-before-break handoff, MN can communicate with both the access routers, the old and the new and there might not be any packet losses [97].
When an MN moves to a new access point (or to some other aspect of the radio channel depending on the wireless technology used, for example, another Radio Network Controller (RNC) in UMTS belonging to the same subnet, it performs a Layer-2 (data link layer) handoff, $L_2$-handoff. L2-Handoff involves the discovery of the available access point and the layer-2 authentication and is transparent to the IP level routing. When an MN moves to a new access point belonging to a different subnet, it performs a Layer-3 (network layer) handoff, $L_3$-handoff. In addition to L2-Handoff, L3-handoff involves the discovery of a new network address in the new subnet and registration and authentication of the new address with the MN's home network.

Mobility protocols can be broadly classified based on the layer at which they operate [16]. This classification leads to mobility protocols based on a subnetwork layer (e.g., GPRS [24], network layer (e.g., Mobile IP [113]), transport layer (e.g., TCP-Migrate [133]) and application layer (e.g., SIP [55]). These mobility protocols are briefly described in the following sections.

### 3.2 Subnetwork Layer Mobility Support

Subnetwork mobility is an $L_2$-handoff mechanism and it is transparent to the layers above the link layer. In this mobility scheme, mobility affects the MN and the adjacent subnetwork without any change to the routing to the correspondent node (CN). The disadvantage of this scheme is that it is confined to a single subnetwork. Wireless mobile systems support subnetwork mobility. Each wireless access technology has its own $L_2$-handoff mechanism.

In GPRS and UMTS networks, MNs are connected to RNC and when the serving RNC (the RNC presently attached to the MN) changes and the new RNC is within the same scope of the serving GPRS support node (SGSN), the GPRS tunneling protocol (GTP) redirects the packets between the SGSN and RNC. SGSN handles inter-RNC mobility and GGSN handles inter-SGSN mobility.

The $L_2$-handoff mechanism in WLAN occurs when an MN changes its access point (AP) within an access network. The signals associated with the $L_2$-handoff in WLAN are shown in the $L_2$-handoff part in Figure 4.2. The $L_2$-handoff in WLAN consists of three phases: the scanning phase, the authentication phase and the association phase. There are two types of scanning to select a new AP, namely passive scanning and active scanning. Passive scanning is a power-saving mechanism where MN waits for the beacon from an AP whereas in the active scanning MN sends a $Probe Request$ message to determine whether an AP is operational or not. Upon receiving the $Probe Request$ message, an AP replies with a $Probe Response$ message. The MN associates with the new AP after a successful authentication and this completes the $L_2$-handoff.

### 3.3 IP Layer Mobility Support

IP layer mobility protocols provide a transparent network layer to the upper layers. In the Internet, an IP address is used to identify a host and perform routing to the host. This dual functionality of the IP address creates a problem when the host changes its IP address due to mobility. Network layer mobility protocols take care of the IP address change due to mobility. IP mobility management can be divided into *global mobility management* and *local mobility management*. Global mobility
management takes care of end-to-end routing of packets when MN moves between access networks. Mobile IP (MIP) is a global mobility management protocol. Local mobility management deals with IP mobility when it is confined to a single access network. Hierarchical Mobile IP (HMIP) [134], Cellular IP (CIP) [14] and Handoff-Aware Wireless Access Internet Infrastructure (HAWAI) [119] are examples of local mobility management protocols.

3.3.1 Global Mobility Protocols: Mobile IP (MIP)

Mobile IP is a network layer mobility protocol standardized by the Internet Engineering Task Force (IETF). Mobile IP is a simple scalable protocol. Based on the IP versions IPv4 and IPv6, Mobile IP has two versions, Mobile IPv4 [113] and Mobile IPv6 [79].

Mobile IPv4 (MIPv4)

Mobile IPv4 [113] is an extension to IPv4 which enables an MN to move from one network to another in the Internet. Each MN is identified by a permanent address called the home address and is associated with a home network, a network having a network prefix matching that of MN’s home address. The home agent HA is the router supporting mobility in the home network. When MN moves to another network called the visited network, the foreign agent (FA) is the router which provides routing services to the MN.

The mobility agents, HA and FA advertise their presence using Agent Advertisement messages. An MN can determine whether it is in the home network or in a visited network based on the agent advertisement it receives. When an MN is in the home network, conventional methods are used to route packets addressed to it. In the visited network MN obtains a temporary address, called care-of address (CoA) and informs the HA about this address. The care-of address must be an address to which packets can be delivered via conventional IP routing. There are two types of care-of address: co-located care-of address and foreign agent care-of address. Co-located care-of address is an address acquired by an MN as a local IP address through some external means which the MN associates with one of its own network interfaces. Foreign agent care-of address is the address of the FA.

When the MN is in the visited network, the packets sent by the CN to the MN are intercepted by the HA and forwarded through an IP tunnel to the care-of address. Figure 3.1 illustrates routing of packets when an MN moves to a visited network. Here the care-of address is the address of the FA and the FA directly delivers the packets to MN over the local link. When an MN sends the packets to a CN, it can either directly send packets to the CN or the FA can forward the packets to the CN as shown in Figure 3.1. This routing of packets to an MN from a CN through an HA results in triangle routing which is inefficient when the MN is close to the CN. Another problem with triangular routing is that packets from CN to MN experience greater delay than the packets from MN to CN. This path asymmetry may affect TCP congestion control.

Mobile IPv6 (MIPv6)

Mobile IPv6 takes advantage of the IPv6 [35] protocol enhancements over IPv4 [114] such as optimal header format [35], addressing architecture [57], neighbour discovery mechanisms [107], stateless
address autoconfiguration [143], security and QoS support. When an MN is in the home network, the packets destined to the MN are routed using conventional Internet routing mechanisms. The MN detects that it has moved to a visited network by analyzing the Router Advertisement it receives from the access routers (AR). It can request an AR to send a Router Advertisement by sending a Router Solicitation message. The MN can form the care-of address by stateless or stateful address autoconfiguration. In the stateless address configuration, the MN generates a new care-of address by combining the link interface address and the subnet prefix associated with the link interface. The subnet prefix can be obtained from the Router Advertisement. In the stateful autoconfiguration, the MN obtains interface addresses and/or configuration information and parameters from a server. A Duplicate Address Detection (DAD) algorithm is performed on all addresses, independent of whether they are obtained via stateless or stateful autoconfiguration to verify that they are unique on a given link. So there is no need for any special routers like FA to provide an address to the MN in the visited network.

The association between the home address and the CoA is called binding. MN does a binding registration by sending a binding update message to the HA and the HA replies with a binding acknowledgement message. There are two possible modes of communication between an MN and a CN. The first mode is similar to that of Mobile IPv4, i.e., the packets from CN are routed through the HA and tunnelled to the MN and this triangle routing is inefficient. In the second mode, MN directly communicates with CN using route optimization. Here if the CN is MIPv6 compatible then MN registers its current binding with the CN so that the CN can directly send the packets to the MN. Route optimization not only reduces the delay in packet delivery but also reduces the impact of any possible failure of the HA and the network path to or from it.

Mobile IPv6 uses Neighbour Discovery Protocol to determine the link level addresses of the nodes residing in the same network while Mobile IPv4 uses the Address Resolution Protocol (ARP) which is link dependent. The Mobile IPv6 protocol inherits the security and authentication features of IPv6.
3.3.2 Mobile IP and Handoff Execution

A handoff process involves two steps: a handoff *decision phase* and a *handoff execution phase*. At first either the MN or the network or perhaps both together have to decide when to perform the handoff. This process is called the handoff decision process. The handoff execution process comes after the decision process. The need for an L3-handoff arises when an MN moves to a new access network. L3-handoff involves an L2-handoff and a network layer handoff as the MN changes its link level attachment as well as its IP address. Figure 3.2 shows the signalling associated with an L3-handoff between WLAN access networks.

The delay arising from a handoff can be due to three main components: *detection period, address configuration interval* and *network registration time*. These components can be explained in a handoff scenario where Mobile IP is the underlying mobility protocol. Detection time is the time taken by the MN from discovering (using link-layer beacons or using Probe Requests) that it is under the coverage of a new access point to the time it receives the Router Advertisement from the new access router. Address configuration time is the interval from the time an MN receives the Router Advertisement to the time it takes to update its routing table and assign a new CoA. The network registration time is the time taken by an MN from sending a binding update to the HA as well as to the CN until the receipt of the first packet from the CN. This is so because Mobile IPv6 does not specify (as it is optional) that MN should wait for the binding acknowledgement from the CN. As the connection to the new router will be operational after the handoff delay, this period can be taken as the disconnection period for a break-before-make handoff.

During a handoff, when an MN moves from one access point to another while maintaining its
connectivity, Mobile IP turns out to be inefficient because of the increased delay, packet loss and signalling overhead and this affects the real-time applications over wireless networks. This handoff latency is due to the link-switching delay and the mobile IP protocol operations such as movement detection and the new address configuration. In Mobile IPv4 this delay is due to the round-trip time incurred in the registration message sent to the HA and its reply to the FA. The signalling delay increases as the distance between the home network and the visited network increases. In mobile IPv6, the registration delay with the HA can be eliminated with route optimization but when an MN moves while communicating with a distant CN the binding updates to the CN and the binding acknowledgements introduce significant delays. Next we present some of the methods that have been proposed to reduce the handoff delays.

![Diagram of Handoff Process]

**Figure 3.3: Signals in Fast Handoff. (Adapted from RFC 4086 [89])**

**Fast Handoff Techniques**

RFC 4881, ‘Low-Latency Handoffs in Mobile IPv4’ [40] proposes methods to reduce the built-in handoff delays in Mobile IPv4. In MIPv4, an MN can only communicate with a connected FA and this implies that the MN can start the registration with the new FA (nFA) only after the L2-handoff is over and the MN can send or receive any packets only after the registration process with the nFA. RFC 4881 also presents techniques to reduce these delays by allowing an MN to communicate with the nFA while still connected to the old FA (oFA) and to send and receive packets while the registration process is going on.
3.3. IP LAYER MOBILITY SUPPORT

RFC 4068, 'Fast handoffs for Mobile IPv6' [89], discusses a protocol to reduce the handoff latency in Mobile IPv6 by reducing the movement detection latency, the new CoA address configuration latency and the binding update latency. Figure 3.3 shows the protocol signalling for fast handoff. An MN, after discovering the APs using some link-specific methods, can send a Router Solicitation for Proxy Advertisement (RtSolPr) to its current access router named in the Figure 3.3 as Previous Access Router (PAR) to get the subnet-specific information regarding the APs. The PAR responds with a Proxy Router Advertisement (PrRtAdv) specifying the AP-ID and AR subnet address of one or more APs.

The MN formulates a new CoA of the New Access Router (NAR) using the information in the PrRtAdv message before moving to the NAR. The MN cannot use the new CoA with the CN until the binding update is completed. In order to reduce the binding update latency, the MN sends a Fast Binding Update (FBU) message to its PAR to set up a tunnel between the previous CoA and the new CoA once the new CoA is formulated. PAR sends a Handover Initiate (HI) to the NAR to see if the new CoA is acceptable to the NAR. If it is acceptable NAR responds with a Handover ACK (HACK) and PAR sends a Fast Binding Acknowledgement (FBack) to the MN. The PAR starts tunneling the packets addressed to the previous CoA to the new CoA. The tunnel remains active until the binding update is completed. MN should reverse tunnel the packets destined to the CN to the PAR and PAR will forward these packets to CN until the binding update is completed. The MN sends Fast Neighbour Advertisement (FNA) to the NAR to inform that the MN is reachable through the new subnet and NAR can deliver the tunneled packets to the new CoA as soon as the MN is attached to the new subnet.

3.3.3 Local Mobility Protocols

Local mobility protocols allow mobility within one administrative domain and they are designed to provide fast and seamless handoff. Basically there are two classes of local mobility protocols, namely Proxy Agent Architectures (PAA) and Localized Enhanced Routing Schemes (LERS) [36]. In PAA there is a hierarchy of FAs between the MN and its HA. When an MN moves, it registers with the nearest FA rather than with its HA thereby reducing the signalling delay. Hierarchical Mobile IPv6 [134] is an example of a PAA. In LERS location management and routing support are integrated with the routing protocol for a localized area. Cellular IP (CIP) [14] and Handoff-Aware Wireless Access Internet Infrastructure (HAWAII) [119] are examples in this category.

Hierarchical Mobile IPv6 (HMIPv6)

HMIPv6 [28, 134] is built on MIPv6 in which the global mobility is managed by MIPv6 while the local mobility is managed locally so as to reduce the amount of signalling and the handoff delay. Figure 3.4 illustrates the HMIPv6 domain. In a visited network, an MN is registered to a Mobility Anchor Point (MAP) which acts as a local HA. MN discovers the address of the MAP via router advertisements. It configures two CoAs: one is Local CoA (LCoA), its link local address and Regional CoA (RCoA) which is usually the address of the MAP. The MN creates a binding between its HA and the RCoA and also it registers with the MAP to create a binding between the RCoA and LCoA. All packets from HA or CN to the MN with RCoA, MAP tunnels to LCoA and the MN
decapsulates and process them. When a handoff occurs in the local domain, MN does not have to inform its new address to HA but only have to register the new LCoA to the MAP thereby reducing the amount of signalling and the handoff delay.

Cellular IP (CIP)

In CIP [14] local mobility and fast handoff are integrated with routing in an access network. Figure 3.5 shows a Cellular IP access network. Mobile IP is used for global mobility between access networks. An access network is connected to the Internet through a gateway (GW). IP address of the GW acts as the CoA of the MN. Within an access network the MN is identified by its HA. The GW periodically sends a beacon packet, depending on which BS forms the uplink to GW so that packets from MN reach the GW. Each BS maintains a routing cache and each entry in it binds MN’s HA with the interface through which the MN can be reached. In order to minimize the messages for location update, the location information of an MN is refreshed in the routing cache by reversing the path in the packets which the MN sends at regular intervals. The packets to the MN reach the GW first and then routed hop-by-hop to the BS to which MN is currently attached using the cached information in the intermediate BSs.

CIP supports both hard handoff and semisoft handoff. In hard handoff, MN initiates a handoff based on the received signal strength from different BSs. MN sends a route update packet to the new BS and this route update message creates a routing cache mapping in all the intermediate BSs enroute to the GW as shown in Figure 3.5 configuring the downlink route to the new BS. Handoff delay is the the time between the new route update message from MN to the arrival of the first packet through the new route. Although hard handoff is fast, it may result in packet losses. In semisoft handoff, MN performs a handoff after a delay during which the new downlink route is configured. During this time MN receives packets through the new and the old paths. Semisoft handoff minimizes the packet loss during handoff.
3.3. IP LAYER MOBILITY SUPPORT

Handoff-Aware Wireless Access Internet Infrastructure (HAWAII)

HAWAII [119] proposes a separate routing protocol to handle mobility in an access network and uses Mobile IP for mobility between access networks. Every access network is identified by a gateway called Domain Root Router (DRR) and it handles all mobility-related issues in that domain. Packets destined to an MN will first reach the DRR based on the subnet address and from there are forwarded to the MN over special dynamically established paths. Inside and access network HAWAII uses path setup messages and update messages to establish and update host-based routing entries for the MN in selective routers so that the packets which arrive at the DRR can reach the MN.

When an MN powers up in its home network, it dynamically gets an IP address and packets to MN are routed using typical IP routing. When the MN moves to a visited network, MN acquires a co-located CoA and sends a path setup message to the new DRR. Upon reception of an ACK from the DRR it registers its CoA with its HA. The packets from the CN will be intercepted by the HA and tunneled to the MN. It is only when the MN moves to another access network that it has to inform the HA about the change in the co-located CoA. HAWAII can also use the route optimization in Mobile IPv6 and forward the packets from CN to MN directly. HAWAII defines four path setup schemes for handoff between access points with each scheme offering a different handoff latency and a packet loss rate.
3.3.4 Proxy Mobile IPv6 (Proxy MIPv6)

Proxy Mobile IP [51] protocol is a network based mobility protocol in which a proxy mobility agent in the network does the mobility signalling with the HA on behalf of an MN attached to the network. This protocol supports mobility without the involvement of the MN in the mobility signalling between the MN and the HA and it extends the Mobile IPv6 [79] signalling and the functionalities of the HA. The reuse of the mature Mobile IPv6 protocol and the ability to use a common home agent as the mobility agent for all IPv6 MNs are the main advantages of this protocol.

The main entities in the Proxy MIP protocol are the Local Mobility Anchor (LMA) and Mobile Access Gateway (MAG). LMA acts as the HA for the MN while MAG functions as an access router to the MN when it is attached to its access link. MAG tracks the MN's movement to and from the access link and is responsible for sending binding registrations to LMA. MAG performs the mobility management signalling on behalf of the MN. For an MN to connect to a proxy MIP network, it first contacts a MAG which performs the authentication and determine if the MN is authorized for network mobility management service. If the MN is eligible for mobility management service, the network will provide an address configuration which includes the Mobile Node’s Home Network Prefix (MN-HNP), the prefix assigned to the link between the MN and the MAG, default router address etc. As far as the MN is concerned the entire network appears as a single link to it and the network ensures that the MN will not detect any changes in its layer-3 attachment even when it changes its point of attachment to the network.

![Diagram of Proxy Mobile IPv6](image)

Figure 3.6: Proxy Mobile IPv6: Attachment and Handoff of MN. (Adapted from [51])
When an MN moves from one MAG to another, the new MAG sends a registration message to the serving LMA with MN’s ID and its own ID which the LMA acknowledges. LMA also sends a deregistration message to the old MAG to complete the handoff. The new MAG sends a Proxy Binding Update message to the MN’s LMA which LMA in turn sends a Proxy Binding Acknowledgment message including the mobile node’s home network prefix(es) and sets up a bidirectional tunnel between the LMA and the MAG. LMA receives any packet from the CN and tunnels it to the MAG and the MAG forwards these packets to the MN. MAG receives the packets from MN to the CN and tunnels to MN’s LMA which forwards them to the CN. The Figure 3.6 shows the signalling when an MN is attached to the network and also during a handoff.

### 3.3.5 NEtwork Mobility (NEMO)

Mobile IPv4 and Mobile IPv6 support terminal mobility while NEMO Basic Support Protocol (BSP) [145] has been developed for supporting network mobility. NEMO BSP is an extension of Mobile IPv6 and provides session continuity for every node in the mobile network as the network moves. The mobile network is connected to the Internet through a Mobile Router (MR) and the MR has a home address obtained from its home network. When the MR moves to a different network it gets a CoA and it sends a binding update to the HA. All data to the mobile network is first forwarded to the HA which tunnels this data to the MR and then the MR sends the data to the particular MN in the mobile network. Similarly the data from the MN is first sent to the MR which tunnels this data to the HA and then the HA finally delivers to the CN.

### 3.4 Host Identity Sublayer (Layer 3.5) Mobility Support

Combining both the host identity and location identity with the IP address creates a problem in mobility scenarios when the host has to change its IP address. Host Identity sublayer (layer 3.5) located between the network layer and the transport layer can differentiate the host identity from the location identity. The host identifier provided by the Host Identity sublayer can be used as the invariant host name whereas the IP address can be used to manage mobility.

Host Identity Protocol (HIP) [104,106,141] is a new namespace protocol from IETF which differentiates the host identity from location identity. HIP operates at the Host Identity sublayer. The IP address remains as a location identifier while HIP provides the host identifier. In HIP the upper layer sockets are bound to Host Identity (HI) rather than to IP address. So the IP address change will not affect the upper layers. There is a dynamic binding between the HI and the IP address.

In HIP, hosts are identified by a globally unique public key or a public key pair. HI can be represented by its 128-bit (SHA-1) hash called Host Identity Tag (HIT) or 32-bit Local Scope Identity (LSI). When an MN changes its IP address, it sends an UPDATE message with a Readress (REA) packet which contains the current address of the MN to the CN. On receiving the UPDATE message, the CN validates the address and updates the local binding between the HIP association and the address of MN. In order to verify the MN’s address, the CN replies the MN with an UPDATE ACK message with a nonce set in the ECHO_REQUEST parameter of the UPDATE message. Upon receiving the UPDATE ACK, the MN sends back the ECHO_RESPONSE and this completes the mobility.
procedures and the CN starts using the new IP address. HIP supports both IPv4 and IPv6 addresses and there is no need to change the current routing methods. HIP enhances mobility and increases security.

As HIP introduces an architectural change, every host that implements HIP has to make changes in the IP stack to implement the HIP layer. Even though HIP improves security, its use of cryptographic namespace presents problems to MNs with limited CPU capabilities and power constraints [83].

### 3.5 Transport Layer Mobility Support

In transport layer mobility protocols mobility is handled at the end host. In this approach, the host name, for example, a *fully qualified domain name* (FQDN), can be the invariant name rather than the IP address. Transport layer mobility protocols require an external location manager for obtaining a new address in the new network to which an MN has moved. To this end the transport layer has to interact with the lower layers in the protocol stack. A lower layer protocol like *Dynamic Host Configuration Protocol* (DHCP) can take care of obtaining a new address and reconfiguring the host for the new network. A higher layer protocol like dynamic DNS can be used to maintain reachability for the new connection. Then the transport layer needs to implement a dynamic rebinding of the connection’s IP address.

On the upside, the transport layer mobility protocols inherently have route optimization and thus avoid triangular routing. When mobility is handled at the transport layer there is no need to deploy HAs. These protocols can provide seamless mobility if the MN has multiple interfaces. Another reason implementing mobility at the transport layer is that being the lowest end-to-end layer the transport layer should be aware of the path changes to have seamless mobility. This is due to the fact that a transport layer protocol which takes care of congestion control should be aware of the path changes as the protocol behaviour depends on the end-to-end path characteristics [37]. On the other hand mobility protocols at the transport layer need the support of other layers for location management. Each individual transport protocol needs to implement binding updates and it will be costly if authentication is needed.

#### 3.5.1 TCP Migrate

The *Migrate* [133] framework, by adhering to the standard TCP semantics and using a TCP option *Migrate-permitted* achieves end-to-end mobility. Location management is done by using DNS on a per-session basis. In standard TCP a connection is established by exchanging *synchronization* (SYN) messages. In the Migrate framework, at the time of connection establishment a host should initially negotiate a *Migrate-permitted* option with a token in the initial SYN segment. If CN permits the migrate option it should send a *Migrate-permitted* option with SYN/ACK and the connection is established as in TCP. When the host moves to a new address, it sends a new *Migrate* SYN packet with the new IP address. The token in the SYN packet identifies that this segment is a part of the previously established connection and the CN can now resynchronize the connection with the MN at the new end point. Migrate has the advantage of being an end-to-end mobility protocol but it
needs changes to the TCP layer at the end hosts irrespective of whether they are mobile or static. As DNS is used for location management the effectiveness of the protocol depends much on the secure DNS updates.

### 3.5.2 Stream Control Transmission Protocol (SCTP)

SCTP [140] developed by IETF is a general purpose transport layer protocol which has features like *partial reliability* and *multihoming*. It is possible to configure the reliability level of SCTP from unreliable services to fully reliable services. The multihoming feature allows a host to have multiple network addresses. This feature enables an SCTP session to be established over multiple interfaces identified by multiple addresses. Between two SCTP hosts there is a primary path using primary addresses and potentially many secondary paths involving secondary addresses. This type of session is known as *association* in SCTP. The ADDIPn [139] extension of SCTP allows SCTP endhosts to add or delete an IP address and change the primary address during a session. In a mobile scenario, an MN initiates an SCTP association with a CN using a primary address. When MN moves to a new location, it obtains a new address from a new access router and informs the CN that it is going to use this new IP address as its primary address. Then the CN adds this new IP address and makes this association the primary path and deletes the old IP address. SCTP with ADDIP extension is called *mobile SCTP* (mSCTP) [121].

In order to take full advantage of SCTP as a mobility protocol, the CN should have multihoming feature, i.e., it should use multiple addresses. Since most of the servers in the Internet have a single address, multihoming feature cannot be used in implementing SCTP as a mobility protocol when both end hosts are moving.

### 3.5.3 Datagram Congestion Control Protocol (DCCP)

DCCP [87, 88] is a congestion-controlled unreliable transport protocol. The current DCCP specified in RFC 4340 [87] does not support mobility. However, the features of DCCP such as tolerance to packet reordering, duplication and loss along with its capability to define a new set of congestion control parameters to the new path to which MN moves make it a good candidate for an application level mobility protocol [86].

*Generalized connections* enable DCCP to have mobility and multihoming. A generalized connection combines one or more transport connections, called component connections into a single application-level entity. Mobility is achieved when a host attaches a new component connection and deletes the old component connection. Multihoming is implemented by maintaining multiple component connections with different endpoint addresses.

### 3.6 Application Layer Mobility Support

Application layer mobility protocols are end-to-end protocols. An MN is identified by an application level identifier and when the MN moves to a visited network, the application level mobility protocol binds MN’s identifier to the new CoA.
Session Initiation Protocol (SIP) \cite{SIP1,SIP2} is an application layer protocol for establishing interactive multimedia sessions. SIP can handle terminal, session, personal and service mobility. We are considering only the terminal mobility here as it is the mobility which allows an MN to move between subnets while maintaining connectivity. SIP can establish two types of terminal mobility, namely \textit{pre-call mobility} and \textit{mid-call mobility}. In pre-call mobility SIP can establish a session at the start of a new connection when an MN has moved to a new location. The MN registers the new IP address with the \textit{redirect server} at its home network. For midcall mobility, where the MN acquires a new IP address in the middle of a session, it has to inform the CN regarding the address change. It takes a one-way delay to update the new address after the MN recognizes the change in IP address. Furthermore, the MN has to inform the redirect server at its home. The redirect server acts like HA in the case of mobile IP. SIP cannot provide terminal mobility if applications are to maintain the TCP connections across subnet changes. However, typical applications that use SIP are based on UDP rather than TCP.

Application layer mobility protocols are free from the suboptimal routing problems of the network layer mobility protocols and the protocol stack modifications required for transport layer mobility protocols. The disadvantages associated with mobility management at the application layer are the delays involved in application layer processing of longer protocol messages and the need for support from lower layers to detect the network change.

### 3.7 Summary

The dual role of the IP address in identifying a host in the Internet as well as in routing packets to this host creates the need for reconfiguring the IP address when a host changes its location in the Internet. This is the problem of mobility management and in a multi-access wireless environment it has two components, namely, location management and handoff management. In this chapter, we presented an overview of the mobility protocols implemented at the different layers of the Internet protocol suite: subnetwork layer (e.g., GPRS), network layer (e.g., Mobile IP), layer 3.5 (e.g., HIP), transport layer (e.g., TCP-Migrate, SCTP, DCCP) and application layer (e.g., SIP). We also described in detail the handoff execution in Mobile IP and the different schemes used for its implementation.

We described how the choice of the implementation of mobility at a particular layer involves trade-offs among issues related to bandwidth, delay, asymmetry in routing and security schemes. While there are advantages and disadvantages in implementing mobility and handoff at the different layers, the role of cross-layer information in mediating between the different layers to better support mobility and handoff and in improving protocol efficiency becomes an important issue and it is discussed in Chapter \[4\].
Chapter 4

Transmission Control Protocol (TCP) over Wireless /Mobile Networks

In this chapter we present an overview of the TCP congestion control mechanisms. TCP was originally designed for wired networks and many enhancements have been developed over the years to adapt it to the wired-cum-wireless environment of the Internet today. In Section 4.1 we present a brief description of the TCP congestion control algorithms along with their four common variants: Tahoe, Reno, NewReno and SACK. Section 4.2 deals with the problems of TCP in wireless/mobile environments along with the proposals that have been made to mitigate these problems.

4.1 TCP Congestion Control Algorithms

TCP belongs to a generic family of sliding window protocols that provides a reliable, in-order, full-duplex, byte-stream delivery of data to the applications [115]. The basic unit of data transfer, known as a segment, is a contiguous sequence of bytes. Each byte is identified by a 32-bit sequence number. Each segment can be of size less than or equal to the maximum segment size (MSS) which is negotiated by the TCP sender and the TCP receiver at the beginning of a connection. The TCP receiver sends a cumulative acknowledgement for byte k to the TCP sender which indicates that all the bytes with lower segment number than k have been received successfully and that it is expecting a segment starting from byte k. The delayed acknowledgement mechanism [23] allows a TCP receiver to refrain from sending an acknowledgement (ACK) for every segment received, usually acknowledging every second segment, which results in sending fewer ACKs thereby saving bandwidth and processing. When the TCP receiver receives an out-of-order segment it will send a duplicate ACK (dunpack) for the last received in-order-segment.

The purpose of the TCP congestion control algorithm is to share the network resources in an efficient manner. An efficient use of the resources implies the utilization of the link bandwidth to near-capacity and to keep the queues in the router buffers small. TCP uses a window-based congestion control algorithm which regulates the number of unacknowledged (i.e., outstanding) bytes of the TCP sender in the network. In order to determine the amount of outstanding data, two variables at the TCP sender are defined for a TCP connection: the congestion window($cwnd$) and
the receiver advertised window (\textit{rwnd}). The size of the \textit{cwnd} determines the maximum amount of data the TCP sender can send before getting a cumulative ACK and the size of the \textit{rwnd} specifies most recently advertised receiver window. So the maximum outstanding data in a connection should be the minimum of \textit{cwnd} and \textit{rwnd}.

TCP congestion control mechanism is built from four basic algorithms: \textit{slow start}, \textit{congestion avoidance}, \textit{fast retransmit}, and \textit{fast recovery} \cite{13, 76}. These algorithms are described below.

- \textit{Slow Start}: This algorithm is used at the beginning of a TCP connection and the purpose of this algorithm is to probe the capacity of the end-to-end path between the TCP sender and receiver. The TCP sender starts the transmission by sending an \textit{initial window} of segments and the \textit{cwnd} is initially set to this value. The upper bound for the initial window is \textit{min}(4 \times \text{SMSS, max}(2 \times \text{SMSS, 4380 bytes})) \cite{12}. Each of these segments should be of size less than or equal to the Sender Maximum Segment Size (SMSS). The TCP sender increases the \textit{cwnd} by one segment for each ACK that acknowledges new data segment until the slow start ends or a loss is detected. A variable called \textit{slow start threshold} (\textit{ssthresh}) determines when to end the slow start phase and to begin the congestion avoidance phase. When \textit{cwnd} exceeds the \textit{ssthresh}, the slow start phase ends and congestion avoidance starts. Initially \textit{ssthresh} is set to a high value and it is reduced when a packet loss occurs. As TCP originated in the context of wired Internet, it made the justifiable assumption that any packet loss is due to congestion in the network.

- \textit{Congestion Avoidance}: In this phase TCP increases the \textit{cwnd} by one SMSS for a window of ACKs, i.e., by one SMSS per RTT which is slower than in the slow start phase. Congestion avoidance ends when a packet loss is detected and TCP enters either slow start or fast retransmit depending on how the packet loss is detected. For each segment sent, the TCP sender waits for the ACK for a certain duration called \textit{Retransmission Timeout} (RTO) and retransmits the segment if the ACK has not arrived before the expiry of the RTO timer. TCP interprets a packet loss detected by an RTO as an indication of severe congestion in the network. By way of its response, TCP retransmits the lost packet, reduces the sending rate by setting the \textit{cwnd} to one MSS and the \textit{ssthresh} to half of the number of outstanding segments in the network (“FlightSize”), and reenters slow start.

Another indication of packet loss is the reception of duplicate ACKs (\textit{dupacks}). \textit{Dupacks} are generated due to out-of-order reception of segments at the receiver arising from loss, reordering or replication of packets in the network. When the TCP sender receives a \textit{dupthresh} number (usually 3) of \textit{dupacks} it interprets this as an indication of packet loss and sets \textit{cwnd} and \textit{ssthresh} to \textit{min}(\textit{FlightSize}/2, 2 \times \text{SMSS}) and triggers the fast retransmit algorithm.

- \textit{Fast Re transmit and Fast Recovery}: Upon reception of three \textit{dupacks}, TCP sender retransmits the missing segment immediately. This is called the fast retransmit algorithm and it is intended to recover from a packet loss faster than the recovery using an RTO. Based on the TCP behaviour after the fast retransmit, there are four popular variants of TCP, namely TCP Tahoe \cite{76}, TCP Reno, \cite{138} TCP NewReno \cite{46, 47} and TCP SACK \cite{98} and we describe them in the following subsections.
4.1. TCP CONGESTION CONTROL ALGORITHMS

4.1.1 TCP Tahoe

TCP Tahoe is described in the seminal paper by V. Jacobson [76]. A TCP connection begins in the slow start phase and continues in the congestion avoidance phase. When TCP Tahoe receives three dropacks, it infers packet loss and fast retransmits the missing segment. TCP Tahoe sets the ssthresh to half the FlightSize and the cwnd is set to one MSS as in the case of an RTO recovery and enters slow start. As a result, the pipe (which refers to the TCP sender’s estimate of the number of outstanding segments) is emptied fully and all the outstanding segments are retransmitted. This is especially inefficient in high bandwidth-delay product (BDP) links.

4.1.2 TCP Reno

TCP Reno [78] is similar to TCP Tahoe except for the difference in behaviour after a fast retransmit. After a fast retransmit, TCP Reno sets the cwnd to half the FlightSize and enters fast recovery whereas TCP Tahoe sets the cwnd to one and enters slow start. In fast recovery, TCP Reno inflates the cwnd by three segments (i.e., ssthresh +3) assuming that three segments have already left the network as TCP has received three dropacks. For each additional dropack that arrives, cwnd is incremented by one segment and a new segment is sent if the new cwnd and the receiver window allow it. This implies that the sender effectively waits for half a window of dropacks before it can send a new segment. Upon reception of the first ACK that acknowledges the new data, the cwnd is deflated back to ssthresh. This ends the fast recovery and TCP continues in congestion avoidance phase. The advantage of TCP Reno is that instead of flushing the pipe completely as in TCP Tahoe, it reduces the pipe by half and the arrival of each dropack is taken as an indication that a packet has left the network.

TCP Reno performs well when a single packet is dropped from a window. Compared to TCP Tahoe, Reno significantly improves the performance for single packet drop but RTO recovery may be needed for recovering multiple packet losses in a window.

4.1.3 TCP NewReno

NewReno [46,47,58] enhances the fast recovery algorithm of TCP Reno. At the time the recovery starts, it keeps a variable recover to denote the highest sequence number transmitted. NewReno introduces the concept of partial ACK which acknowledges a retransmitted segment but not all the segments up to recover. The behaviour of NewReno is similar to Reno except for its behaviour in partial ACKs. In fast recovery, upon the receipt of partial ACK, NewReno retransmits the first unacknowledged segment. NewReno can recover from multiple packet losses by retransmitting one lost packet per RTT until it gets an ACK that covers recover. After the fast recovery cwnd is deflated and the congestion avoidance is resumed. If there are multiple packet losses in a window TCP performance degrades especially in the case of long-delay networks as NewReno can retransmit only one lost packet per RTT.
4.1.4 TCP Selective Acknowledgements (SACK)

The use of Selective Acknowledgements (SACK) significantly improves TCP's loss recovery mechanism especially when there are multiple losses in a single window [98]. In TCP SACK, the TCP receiver along with an ACK can send the information regarding up to four non-contiguous segments received beyond the first missing segment. Based on this information TCP sender can retransmit the missing segments. TCP SACK is one of the most widely used TCP versions [109].

In a strict sense SACK is an option for TCP, so this option can be used with any of the TCP versions. At the time of opening a TCP connection both the sender and the receiver should agree upon using the SACK option. The TCP sender sends a 2-byte TCP SACK permitted option with the SYN segment. In the SACK option the receiver should inform the sender about the non-contiguous blocks of data it has received and queued. The number of SACK blocks can be up to 4. The left edge of a block specifies the first byte of the block successfully received and the right edge specifies the sequence number of the byte immediately following the last byte in the block. The first SACK block specifies the contiguous block of data containing the segment which triggered this ACK [48], unless that segment advances the window. In this manner the ACK with the SACK option reflects the most recent change in the data receiver’s buffer queue. The receiver should generate the SACK option for all ACKs which do not acknowledge the highest sequence number in the receiver’s queue. The receiver should send an ACK with the SACK block for every valid segment that arrives containing any new data which generates dupacks. The sender implements a scoreboard structure to keep track of the segments that have been received and retransmits the missing segments. When missing segments are received, the receiver acknowledges them and advances the cumulative ACK.

A conservative SACK-based loss recovery algorithm for TCP is given in [21]. This algorithm is similar to Reno, enters fast retransmit on the arrival of three dupacks and differentiates between a full ACK and a partial ACK as in NewReno. During the Fast recovery SACK maintains a variable called pipe which represents the number of outstanding segments in the path. TCP sender sends new segments or retransmits lost segments only if the pipe is less than cwnd. A new segment is transmitted only if there are no lost segments to be retransmitted. The pipe is incremented by one when the sender sends a segment and it is decremented by one when the sender receives a dupack with a SACK option that acknowledges a new out-of-order segment. When an incoming cumulative ACK covers the recover i.e., if the ACK acknowledges all the segments that were outstanding at the beginning of fast recovery, TCP sender exits from loss recovery. If the incoming ACK does not cover the recover, pipe is decremented and the scoreboard is updated based on the SACK information. If a retransmitted packet is lost, SACK uses the conventional retransmit timer to detect the loss, retransmit the packet and then enters slow start.

A comparison of Tahoe, Reno and SACK algorithms based on an extensive simulations study can be found in [44]. The results show that the SACK algorithm has the best overall performance in many scenarios.

4.2 TCP over Wireless and Mobile Networks

With the increasing number of wireless/mobile devices such as laptops, PDAs and data-enabled mobile phones that are connected to the Internet, it is apparent that the wireless networks also
should support TCP as efficiently as wired networks. The properties of wireless networks such as high bit-error rate (BER), bursty traffic due a mixed voice and data and disconnections due to handoffs and interferences have an adverse effect on TCP. So wireless and mobile networks pose particular difficulties to TCP. We briefly outline these problems here [41, 74].

4.2.1 Problems with Wireless and Mobile Networks

1) Non-congestion-related losses:
   - In wireless and mobile networks, besides congestion-related losses, frames can be lost due to link errors and disconnections. The BER of wireless links leads to the loss of packets and ACKs. In addition, bit errors are bursty as they occur in clusters due to wireless channel characteristics such as multipath transmission and signal fading due to interference.
   - Link outage/disconnections can happen due to handoffs and interference. Disconnection can occur if an MN moves out of the reach of a base station or the radio signals are blocked by buildings or similar objects.

One of the basic assumption in TCP design is that all packet losses are due to congestion. When packet losses occur due to link errors or due to disconnections, TCP unnecessarily invokes the congestion control measures and reduces the sending rate.

2) Variation in link latency: The link level error recovery techniques such as Automatic Repeat reQuest (ARQ) and Forward Error Correction (FEC) appear as delay jitter to TCP, thereby increasing the RTO value. Link level recovery from a link outage/disconnection results in a long delay spike which increases the latency of the link and Spurious RTOs can occur as a result.

3) Variation in available bandwidth: The available bandwidth in a wireless network is affected by the location of the MN and the number of active users. There may be a lot of variation in bandwidth even within a cell and this may affect the TCP behaviour as its determines TCP’s sending rate.

4) Abrupt changes in bandwidth and delay: Due to a vertical handoff there may be abrupt and significant changes in bandwidth and delay of the end-to-end path affecting TCP performance. This problem is studied in detail in this thesis and algorithms to improve TCP performance are given in the following chapters.

4.2.2 Enhancing TCP over Wireless and Mobile Networks

RFC 3481, 'TCP over Second (2.5G) and Third (3G) Generation Wireless Networks' [74], gives an overview of 2.5G and 3G wireless networks, describes the problems that TCP encounters in these networks and also recommends enhancements to adapt TCP to the specific characteristics of the wireless and mobile networks. It mostly recommends the IETF standard track mechanisms. The main recommendations therein for TCP over 2.5G and 3G networks are the following:

- Choosing an appropriate window size: Based on the bandwidth-delay product (BDP), the 2.5G networks can be classified as Long Thin Networks (LTNs) [103] and 3G networks can be classified as Long Fat Networks (LFNs) [22]. Both LTNs and LFNs need a large window of outstanding data to have good TCP performance. In order to have the full advantage of
SACK [98] for loss recovery, both LFNs and LTNs may need to maintain twice the window of outstanding packets for full utilization of the available bandwidth. The window scale option [22] that enables TCP to have a receiver window larger than 64 KB should be used when the end-to-end BDP is larger than 32 KB.

- **Choosing the initial window:** TCP over 2.5G and 3G networks should set the initial congestion window to the \( \min(4 \times \text{MSS}, 4380 \text{bytes}) \) [12]. This increased initial window is beneficial especially for short transfers over wireless networks as the entire data transfer can be completed within the slow start phase of TCP itself.

- **Enabling limited transmit:** A TCP connection with a small window may not be able to generate three dupacks to trigger the fast retransmit algorithm. So in the case of LTNs with a window of less than four segments, fast retransmit algorithm cannot be triggered which leads to an RTO recovery. The limited transmit algorithm [11] allows TCP to send a new data segment in response to the first two dupacks thereby avoiding the RTO recovery. The implementation of the limited transmit algorithm is recommended for TCP over 2.5G and 3G networks.

- **Choice of maximum transmission unit:** 3G networks can use a high Maximum Transmission Unit (MTU) value, such as 1500 bytes, the MTU for IP packets on Ethernet. The MTU on 2.5G networks may be selected as per the recommendations of RFC3150, “End-to-end Performance Implications of Slow Links” [34]. TCP over 2.5G and 3G networks should implement path MTU discovery [100, 101] to determine the MTU size of the end-to-end path.

- **Enabling SACK option:** As the wireless links have high BDP and high packet loss rate, it is recommended to implement the SACK option [98] which helps to recover multiple losses in a single window. SACK is robust when compared to Tahoe and Reno [44].

- **Supporting ECN:** TCP over wireless links should support Explicit Congestion Notification (ECN) [118]. ECN provides indication of the congestion in the network and TCP can adapt its sending rate.

- **Enabling ARQ and FEC schemes:** Error detection and correction schemes such as ARQ and FEC are used in wireless link layer protocols to reduce the packet losses over the wireless links considerably but they introduce additional latency and jitter in the IP traffic. Further delays are introduced due to handoff. These delays can cause spurious RTOs that trigger unnecessary TCP congestion control measures. Algorithms like Eifel [94, 95], F-RTO [126] are recommended to detect spurious RTOs and to improve the performance of TCP over wireless links.

- **Combating packet reordering and packet duplication:** Packet reordering and packet duplication are present in wireless networks due to handoffs and preemption of data packets by circuit-switched voice may adversely affect the performance of TCP. It is desirable to make TCP robust against these events. DSACK information [19, 20, 48] can be used to detect packet reordering. The Eifel algorithm [94, 95] proposes another solution to this problem. These algorithms can be used to improve the performance of TCP over wireless links.

- **Enabling TCP timestamps option:** The use of the TCP timestamps option [22] is recommended in RFC 3481 as it allows better RTT estimation which can help to avoid spurious RTOs. Usually the RTT samples are collected once per window [111]. TCP connections with large windows as in 2.5G and 3G networks may benefit from frequent RTT samples with timestamps as they reflect the change in network conditions due to delay or handoff. TCP timestamps incur an additional overhead of 12 bytes for every segment and also hinder the functioning of the current TCP header compression scheme [77].
○ **Use of TCP header compression schemes**: The TCP header compression scheme described in RFC 1144, 'Compressing TCP/IP Headers for Low-Speed Serial Links' [77] does not perform well in the presence of losses [82, 96]. As the header compression algorithms send only the changes in the headers of consecutive segments, the TCP receiver will not be able to decompress the headers even if there is a single segment loss. It is recommended to disable the header compression algorithm given in RFC 1144 for wireless networks with BER as high as $10^{-3}$. Robust Header Compression Scheme [26] aims to achieve header compression in wireless links with RTT varying from 100 ms to 200 ms and link error rates of $10^{-4}$ to $10^{-3}$.

○ **Recommendations from RFC 2988**: Also recommended is the compliance with the recommendations of RFC 2988, 'Computing TCP’s Retransmission Timer', in computing the values of the variables associated with TCP retransmission timer [111]. This RFC recommends an initial RTO value of 3 seconds and a minimum RTO ($\text{minrto}$) value of 1 second. If the RTO value computed is smaller than $\text{minrto}$, the RTO timer is set to the predefined $\text{minrto}$ value. In many implementations the $\text{minrto}$ value is taken as 200 ms [120] as it speeds up the loss recovery. Also recommended is that the retransmission timer be restarted when an ACK is received.

### 4.3 Summary

TCP is the transport protocol for reliable data transfer in the Internet which is used by a variety of applications. In this chapter we described the TCP congestion control algorithm along with its four common variants namely, Tahoe, Reno, NewReno and SACK. TCP was originally designed for wired networks and TCP congestion control algorithm makes the assumption that all packet losses are due to congestion in the network. In the wireless and mobile environments this assumption no longer holds due to the high error rate, link outages, disconnections and handoffs. As a result the congestion control response of TCP in wireless networks can be unnecessary and inefficient. We described the various problems that TCP encounters in wireless and mobile networks. We also gave an outline of the various proposals recommended in RFC 3481 to mitigate these problems in 2.5G and 3G wireless networks.

As TCP protocol behaviour depends on the end-to-end path characterizes which it learns through indirect measurements, it may take TCP several RTTs to adapt to the abrupt changes in path characteristics due to handoffs. In later chapters we discuss the behaviour of TCP in the presence of vertical handoff and present our algorithms to improve its behaviour using cross-layer information.
Chapter 5

TCP and Vertical Handoff

In this chapter we provide the necessary background to the problem of TCP in vertical handoff. Section 5.1 provides an introduction to vertical handoff. Section 5.2 describes the problems of TCP due to a vertical handoff. Section 5.3 presents a review of the literature and the proposed solutions to improve the performance of TCP with vertical handoff.

5.1 Vertical Handoff

The concept of vertical handoff was proposed by Stemn and Katz as a means by which a mobile node (MN) can roam among multiple wireless networks in a manner that is transparent to applications with minimum disruption in connectivity [137]. As the ideal characteristics of low latency, high bandwidth and wide area connectivity are not simultaneously achieved by any single wireless technology, Stemn and Katz proposed a wireless overlay structure in which there is an ordering among the wireless networks with a high-bandwidth, low-coverage network at one end and a low-bandwidth, high-coverage network at the other end. The goal is to provide transparent IP mobility and to achieve seamless handoff between the networks in the wireless overlay architecture.

In a multi-access mobile environment, during the lifetime of a connection, the connectivity to an MN may change across access networks of different orders of bandwidth, latency and error characteristics. MNs are often equipped with multiple radio interfaces to connect to access networks using diverse link technologies in order to support the best of connectivity, services quality, application needs and user preferences. Handoff is the process by which an active MN switches between access networks. As explained in Section 3.1 a handoff between access networks consists of an L2-handoff which enables the MN to associate with a new access point and an L3-handoff which provides the MN with a new IP address in the new access network. Handoff can be classified as horizontal handoff (intrasystem) and vertical handoff (intersystem).

Horizontal handoff involves an MN moving between access points of the same type of access networks that use the same link layer technology, for example, from UMTS to UMTS or from WLAN to WLAN. The need for a horizontal handoff arises, for example, when the signal strength of the serving BS deteriorates below a certain threshold value.
A vertical handoff arises when an MN moves out of the serving network to another access network of a different link technology, for example, handoff between WLAN and UMTS. In a vertical handoff, if the MN is equipped with separate interface cards to different wireless technologies, it is possible to decouple the L3 handoff from the L2-handoff.

Like in general handoff, vertical handoff can be of two types based on the connectivity to the old access router during the handoff, namely, break-before-make (BBM) and make-before-break (MBB) [97]. In a break-before-make handoff, the MN’s connection to the old access router breaks before the handoff completes, thereby causing disruption in connectivity which often results in packet losses. In the worst case, an entire window of TCP segments may be lost. By contrast, in a make-before-break handoff, an MN can have connection to more than one access router at the same time and the MN ends its connection to the old access router only after establishing the connection to the new access router, thereby avoiding packet losses due to a handoff. Vertical handoff occurs often between different access networks, so the IP addresses change due to handoff. Make-before-break handoff decouples an L3-handoff from an L2-handoff and can be performed independently as both the old and the new interfaces can send and receive packets simultaneously. Even when an MN is equipped with more than one radio interface, a break-before-make handoff can occur due to the following reasons: (i) battery issue, there may not be enough power from the battery to have two radios on simultaneously (ii) no overlapping coverage area.

5.2 Problems of TCP in Vertical Handoff

A vertical handoff involves switching to a new access network whose link characteristics differ significantly from those of the old access network. For example if we take a WLAN bandwidth as 5 Mbps and one way propagation delay as 10 ms and the corresponding figures for EGPRS as 200 Kbps and 300 ms, the ratio of change in bandwidth after a vertical handoff from WLAN to EGPRS is 25:1 whereas the ratio of change in delay is 1:30. So there is an order of magnitude change in bandwidth and delay after a vertical handoff from WLAN to EGPRS. Though the mobility management protocols such as Mobile IP [79, 113] provide a transparent mobility to the transport layer, the order of magnitude change in bandwidth and delay in the last-hop or first-hop link characteristics due to a vertical handoff may change the end-to-end path characteristics and it may take TCP several RTTs to converge to the new values determined by the new path. Since immediately after a vertical handoff TCP still relies on the cwnd and RTT of the old path, it may unnecessarily invoke congestion control actions resulting in performance degradation.

TCP is known to converge slowly to the new network conditions after a vertical handoff [38]. TCP can only increase the sending rate by one segment in the congestion avoidance phase. If the new link after a handoff has high capacity, TCP will take many RTTs to achieve a high sending rate. Similarly after a handoff to a low capacity path, when packet losses occur, TCP starts halving the cwnd. The new path capacity may still be lower or higher than half of the old path capacity resulting in under(or over) utilization of the new path. A significant change in RTT can give rise to spurious retransmission timeouts (RTO), delayed timeout recovery and packet reordering. As a result a vertical handoff may incur packet losses, intermittent connectivity, packet reordering and spurious retransmission timeouts (RTOs) resulting in either unnecessary TCP congestion response
or inefficient loss recovery that sacrifice TCP performance \cite{53, 56, 59, 84, 85, 127, 130, 131}. Next we describe the various problems of TCP due to a vertical handoff.

### 5.2.1 Spurious Retransmission Timeouts (RTO)

Spurious RTOs are the unnecessary retransmission timeouts caused by a delayed or lost ACK. As a result of a spurious RTO, the TCP sender retransmits packets unnecessarily and decreases the sending rate by reducing the \textit{cwnd} to one MSS and \textit{ssthresh} to \textit{FlightSize}/2. Bandwidth is wasted due to unnecessary retransmissions whereas \textit{cwnd} reduction results in under-utilization of the link.

Spurious RTOs occur when a make-before-break handoff occurs from a low-delay to a high-delay link \cite{56, 59}. After the handoff, the ACKs will be delayed due to the high propagation delay of the new high-delay link. Due to the small RTO value calculated on the basis of the old path, the TCP retransmission timer expires before the arrival of the ACKs through the new link. This spurious RTO will cause unnecessary retransmission of packets and reduction in \textit{cwnd} and \textit{ssthresh}. Unnecessary retransmissions waste the bandwidth and reduction in \textit{cwnd} and \textit{ssthresh} reduce the sending rate resulting in performance degradation.

### 5.2.2 Packet Reordering

Packet reordering occurs when a make-before-break handoff takes place from a high-delay link to a low-delay link \cite{56}. During a make-before-break handoff, an MN can receive packets through the old link as well as the new link. The packets with high sequence numbers sent after the handoff through the new link arrive at the TCP receiver earlier than the packets sent before the handoff through the old link. Reordering occurs at the receiver as the sequence number of the packets arriving through the new link is greater than the expected sequence number in the TCP variable \textit{rev.nxt}. As a consequence of this reordering, the TCP receiver sends duplicate acknowledgments (\textit{dupacks}) over the new link. When the TCP sender gets three \textit{dupacks}, it triggers fast retransmission and fast recovery algorithms and as a result the \textit{cwnd} and the \textit{ssthresh} are reduced. As the \textit{dupacks} arise, not due to congestion but due to reordering, the retransmissions are unnecessary. The \textit{cwnd} reduction is undesirable if the BDP of the new path is larger than that of the old path.

### 5.2.3 Packet Bursts

Packet burst can occur after a reordering event \cite{56}. Consider a situation where both the old high-delay link and the new low-delay link are active and in-order packets are received through the old link and out-of-order packets through the new link. This situation arises in a make-before-break handoff from a high-delay link to a low-delay link. When the last in-order packet arrives at the receiver, it advances the \textit{rev.nxt} TCP variable by the amount of packets received through the new link the receiver sends a cumulative ACK through the new link. Upon receiving this ACK, the TCP sender sends a burst of packets through the new link which may cause severe packet drops leading to retransmission timeouts if the new link’s transmission queue does not have the capacity to handle it.
5.2.4 Packet Losses

In addition to congestion-related packet losses there may be packet losses due to disconnection as well. If the BDP of the new link is less than that of the old link and the TCP sender continues to inject packets to the new link at the same high rate as old link, it may result in buffer overflow of the new link and consequent packet losses [53,56]. A significant decrease in BDP may lead to an RTO recovery. With a break-before-make handoff connectivity is lost for some period of time and resumes after the handoff completes. This disconnection causes packet losses.

5.2.5 Unused Connection Time

A break-before-make handoff is likely to result in unused connection time [130]. During the disconnection period the RTO timer may expire several times, each time doubling the RTO value [111]. When the connectivity is resumed, the TCP sender needs to wait until the RTO timer expires again before attempting another retransmission. This unused connection time delays the start of the recovery of lost packets.

5.2.6 Inability to Adapt to Increased Capacity

A vertical handoff to an increased BDP path results in underutilization due to the inherent inability of TCP to adapt to the high BDP available [56]. If the handoff occurs during the congestion avoidance phase, increasing the cwnd by one in one RTT will take several RTTs for TCP to fully utilize the increased new link capacity.

5.3 Related Work

The earlier proposals in improving TCP with vertical handoff can be categorized as sender-based algorithms and receiver-based algorithms. In the sender-based algorithms [53,93,127,131,144] the TCP sender algorithm is modified while in the receiver-based algorithms [49,50,99], the TCP receiver algorithm is modified to cope with a vertical handoff. The following paragraphs describe the various proposals that have been made in the literature.

Goff et al. [49] propose Freeze TCP which is one of the early proposals of an end-to-end approach to enhance the TCP performance in a break-before-make handoff. Freeze TCP assumes that an MN can detect an imminent handoff and the TCP receiver at the MN can advertise a zero window size, preventing the TCP sender at the CN from sending new packets and from reducing the cwnd. As soon as the connection is operational again, the TCP receiver will send three dupacks as suggested in [27] so that the TCP sender invokes the fast retransmit and fast recovery algorithms to avoid the problem of unused connection time described in Section 5.2.5. Freeze TCP was designed for handling disconnections and reducing packet losses during a horizontal handoff. But in a vertical handoff scenario, keeping the cwnd of the old path is not advisable as the new path may have a different capacity. Besides, packet losses can occur after a handoff due to decrease in BDP, packet bursts etc which may not be prevented by merely reducing the TCP receiver window to zero.
Hansmann and Frank [56], in their study of the effect of handoff on TCP performance, identify packet reordering, segment burst due to delay difference and BDP change as the main problems of TCP due to a vertical handoff. They suggest a nodupack scheme for packet reordering which suppresses the transmission of dupacks during a handoff. They also propose to reduce the congestion window (cwnd) to overcome the packet losses due to decrease in BDP. However reducing only the cwnd may make TCP more aggressive by taking it to slow-star. Based on simulation experiments they show that the nodupack scheme improves the TCP performance by not invoking congestion actions in response to reordering. The cwnd reduction scheme combined with the nodupack scheme takes care of the situations where the TCP window reduction is necessary. They show that the cwnd reduction scheme alone is not able to reduce the packet losses due to segment bursts. 

Kim and Copeland [84] propose a scheme for seamless handoff in which they introduce a Handoff Option (HO) in the TCP header to identify the beginning and end of a handoff. The HO field can have values for no handoff, horizontal handoff and vertical handoff. When a handoff occurs, the TCP receiver sends an ACK with HO field set to a value indicating a vertical or horizontal handoff. The receiver sends an ACK with the HO field “00” to indicate the completion of a handoff. In the case of a horizontal handoff, the retransmission timer is stopped and the cwnd and ssthresh values are saved. No data is transferred during the handoff. When the handoff is complete, the timer is restarted and the cwnd and ssthresh values are restored. In vertical handoff, the TCP sender stops the retransmission timer and holds the data transfer. After the handoff is over, the ssthresh is set to a high value and cwnd is set to one initiating the Slow Start phase gain. When there is no handoff, the TCP algorithm used is Reno. Using simulation the authors show that their scheme avoids packet losses during a handoff and that the TCP is able to reach a stable condition rapidly since it slow starts and finds the capacity of the new network in a vertical handoff. In [85] Kim and Copeland point out that a sudden change in RTT due to a handoff affects the TCP performance. They show that resetting the TCP retransmission timer after a handoff from a high-delay network to a low-delay network improves the performance of TCP Reno.

Chakravorty et al. [29] study the performance of TCP with vertical handoff between GPRS and WLAN using a real test bed. They find that the latency difference between GPRS and WLAN links often causes TCP to timeout and contributes to the degradation in TCP performance. Another interesting result of their study is that large buffering in GPRS aggravates the performance of TCP as it inflates the RTT and RTO values. They also propose a number of network layer optimizations to reduce the handoff delay.

Gurtov and Korhonen [53] report a comparative study of the effect of vertical handoff on TCP and TCP-Friendly Rate Control (TFRC). They note that the self-clocking property of TCP helps to adapt to the new link rapidly in a handoff. However, when a timeout is needed to recover a packet, TCP may lose its self-clocking property and the connection may break for more than 10 seconds in the case of GPRS to WLAN handoff. This high break period is due to the high latency and queuing delay in GPRS. With measurements and simulations they show that TFRC has serious difficulties in adapting to the network after handoff. When there are competing TCP flows, it may take several minutes before TFRC gets a fair share of bandwidth after a handoff. The authors propose overbuffering to improve the handoff performance of TCP and TFRC. In this scheme, the buffer size of all the links are configured to the maximum of the BDP of any link in the end-to-end path. Though overbuffering helps TCP to have a smooth handoff between links of different BDPs, this scheme is not easy to implement as the bandwidth and delay of all the links in the path should
be known beforehand. Overbuffering the low BDP links increases the queuing delay which affects interactive applications. Overbuffering also inflates the RTO which delays the RTO recovery.

Huang and Cai [59] describe the problems of TCP with make-before-break vertical handoff from a slow link to a fast link and vice versa. They propose three schemes called fast response, slow response and ack delaying to solve the problem of spurious RTOs caused by increase in RTT after a handoff. The basic idea behind these schemes is to reduce the difference in RTT between the old path and the new path so that TCP can gradually adapt the RTO of the new path. In the fast response scheme, the old fast link is used for a short period to send the ACKs for the first few packets from the new slow link. The reduction in RTT for the first few slow link packets will help to avoid the spurious RTO. In the slow response scheme, the difference between the old RTT and the new RTT is reduced by increasing the RTTs of the last few packets that are transmitted through the old fast link. This is achieved by sending the ACKs for the last few packets through the new slow link which requires the receiver to switch to the new link in advance. In these schemes the RTT is changed in two steps: first from $2 \times D_f$ to $D_f + D_s$ and later to $2 \times D_s$ where $D_f$ and $D_s$ denote the one way propagation delay of fast link and slow link respectively. The RTT should remain in the intermediate step for a certain period so that the RTO value should be higher than the RTO of the fast link to avoid spurious RTO. The choice of a proper value of this period is a design issue. Even if the proposed schemes are for make-before-break handoff, these schemes may have practical problems if the old link is not available just after handoff and the new link cannot be used just before handoff. In the ACK delaying scheme, the ACKs of a few packets that are last transmitted through fast link are delayed at the IP layer and they are passed to the TCP sender after a delay. Even though this scheme is easier to implement than their other two schemes, a proper value of the delay is needed to get good results.

Schütz et al [130] identify that TCP performance degradation in a mobile environment characterized by frequent change of address of an MN and intermittent connectivity is mainly due to three reasons: (1) the change of IP address (2) transport layer timeouts and (3) the transport layer retransmission behaviour. They propose Host Identity Protocol (HIP) [104,106,141] as a solution to the IP address changes due to mobility. TCP in its normal operation will abort a connection if the disconnection period is more than the user timeout and this will be problematic for scenarios with frequent and extended disconnections. TCP does not restrict the value for the user timeout [115] but in many TCP implementations user timeout is set to a few minutes. They propose a user timeout option in which an MN can specify the value of the user timeout to prevent the connection from being aborted during the disconnection period. Also proposed is a TCP retransmission trigger which causes TCP to attempt a retransmission when the connectivity is restored to avoid the unused connection time and this method is shown to be useful for paths with intermittent connectivity. A drawback of this scheme is that there can be unnecessary retransmissions in the case of a make-before-break handoff. This drawback can be avoided using their proposal [131] to retransmit the first unacknowledged packet only if the TCP is in RTO recovery when the connectivity is restored.

Quick-Start [45,124] employs in-band signaling to set the initial congestion window to a higher value than the default if all the routers along the communication path approve the Quick-Start Request for higher sending rate. Sarolahti et al [127] show that a variant of the Quick-Start algorithm can be applied after a vertical handoff to determine the capacity of the new path. An explicit cross-layer handoff notification is employed to trigger the Quick-Start algorithm when the handoff completes. In the original Quick-Start algorithm, only the cwnd is set based on the Quick-Start Response but
after a vertical handoff $ssthresh$ and $cwnd$ are made equal to the Quick-Start Response. Simulation results show that TCP performance is improved by reducing the packet losses due to buffer overflow as Quick-Start is able to estimate the path capacity after a handoff.

DSACK [48] is an extension of SACK in which the receiver reports to the sender that a duplicate segment has been received. When DSACK is used, the first block of the SACK option will be the DSACK block that triggered the ACK. Each duplicate segment is reported in a single ACK packet. The use of DSACK to detect unnecessary retransmissions and undo the unnecessary congestion control actions is presented in [20]. When DSACK information is received, the congestion control measures that have been taken already are undone only if all the segments in a particular window have been duplicated. DSACK can also be used to detect packet reordering and to undo the consequent congestion control actions [19]. In a vertical handoff, as the path characteristics may change after the handoff, TCP may not be able to know how long it has to wait to confirm that all the retransmitted packets in a particular window have been retransmitted unnecessarily due to packet reordering and to undo the congestion control actions that have been taken already. In addition, when there is a significant change in delay and bandwidth of the paths involved in a handoff, restoring the old $cwnd$ and $ssthresh$ values may adversely affect the performance of TCP if the BDP of the new path is smaller than the BDP of the old path.

The TCP-Eifel detection algorithm [95] uses the TCP timestamps option [22] to detect spurious retransmissions. The TCP-Eifel response algorithm [94] describes the methods to undo the unnecessary congestion control measures taken during the RTO recovery. Eifel detection algorithm provides a faster detection of spurious RTOs compared to DSACK but for every packet there is an overhead of 12 bytes because of the use of timestamps.

Forward-Retransmission Timeout (F-RTO) [126] is a TCP sender-only algorithm that helps to detect spurious RTOs. Unlike TCP-Eifel it does not require any TCP options to operate. The F-RTO algorithm retransmits the first unacknowledged segment as a response to an RTO. By monitoring the incoming acknowledgements, the algorithm determines whether or not the timeout was spurious and decides whether to send new segments or to retransmit unacknowledged segments. The F-RTO algorithm helps to avoid unnecessary retransmissions and congestion control actions and thereby improves the TCP performance.

In vertical handoff scenarios Eifel and F-RTO algorithms are effective in avoiding the unnecessary retransmissions but not in congestion control response as the selection of a proper response is hard without additional information about the new path.

Lin and Chang [93] propose a vertical handoff-aware TCP (VA-TCP) where the TCP sender gets a notification from MN regarding the occurrence of a handoff. During a vertical handoff, if the TCP sender detects that a packet sent over the old access network is lost then it retransmits the missing segment without invoking any congestion control action. After the handoff, the TCP sender estimates the bandwidth and RTT using the packet-pair scheme [80] and sets the $cwnd$ and $ssthresh$ to the BDP calculated using the bandwidth and the RTT estimates. However, if a handoff occurs to a network with a smaller BDP continuing with the old $cwnd$ even for a few RTTs during the packet-pair estimation after a handoff will congest the network leading to an RTO recovery.

Matsushita et al. [99] propose ACK pacing, a receiver based mechanism to improve the performance of TCP with vertical handoffs. When a handoff decision is made, a MN calculates the BDP of
the end-to-end path before and after the handoff. If the BDP of the new path is less than that of the old path the TCP receiver sends duplicate ACKs till the transmission rate is reduced below the bandwidth of the new wireless access link at which point a vertical handoff occurs. After the handoff the ACKs are sent at a rate depending on the bandwidth of the new link. However, the duplicate ACKs will force the sender to unnecessarily retransmit an already received packet. If the BDP of the new path is larger, multiple partial ACKs are sent until the transmission rate increases to the bandwidth of the new link. However, sending more than one ACK for each received packet is not advisable as malicious users can exploit it to increase the sending rate aggressively [13, 128].

Tsukamoto et al. [144] addresses the problem of a large change in bandwidth of the access links before and after a vertical handoff and proposes two schemes to overcome this problem. In the first scheme, the TCP sender goes to slow-start as soon as the interface change is detected. However, going to slow-start may lead to inefficient utilization of the wireless link after a handoff and also may unnecessarily retransmit the segments whose ACKs are delayed in the case of a make-before-break handoff from a low-delay to a high-delay link. In the second scheme known as Bandwidth-Aware scheme, TCP sender goes to slow-start after a vertical handoff and the bandwidth of the new path estimated using a single packet-pair [80]. The ssthresh is set to the BDP calculated using the bandwidth and the RTT of the new path. Simulation results show that Bandwidth-Aware scheme is capable of 80% utilization of the available bandwidth.

Gou et al. [50] propose a receiver-based mechanism to address the problem of abrupt change in link capacity due to a vertical handoff. When an impending handoff is detected, the TCP receiver sends the receiver advertised window (rwnd) based on the BDP of the new network and this rwnd will be effective once the mobility registration is complete. The new rwnd helps the TCP congestion control mechanism to set the appropriate sending rate so that the underutilization or overflow due to a vertical handoff can be prevented. When a vertical handoff occurs from a high BDP network to a low BDP network (for example from WLAN to GPRS), the rwnd is taken as the minimum of the BDP of the GPRS network and the current value of rwnd and the TCP sender will reduce its sending rate thereby preventing the buffer overflow at SGSN. The underutilization of the WLAN network in the GPRS-WLAN handoff is due to the slow draining of the SGSN buffer. In the GPRS-WLAN handoff, the rwnd is increased by two segments so that there will be sufficient dupacks to trigger the fast retransmit of the packets queued in the SGSN buffer through the WLAN network. Afterwards the handoff, rwnd can be increased based on the measured BDP of the WLAN network.

Schütz et al. [131] propose an extension to TCP, TCP Response to Connectivity Change Indications (RLCI) in response to a lower-layer notification called Connectivity Change Indications (CCI). A TCP sender receives the CCI either from its local stack or through a TCP option when there is a change in connectivity. CCI is taken as a signal to TCP to re-probe the network path to find the characteristics of the new path. The TCP sender may use the congestion window of data on the new path and reset the congestion control state, RTT variables and RTO timer as recommended in RFC 2988 [111] for a new connection. The rwnd should not be adjusted when the ACKs for the packets delivered through the old link are received as they do not reflect the current path parameters. TCP timestamps option [22] may be used to distinguish the ACKs transmitted before or after the CCI. If a connection is stalled in an exponential backoff, TCP may retransmit the first unacknowledged segment.
Table 5.1: Summary of the problems to TCP due to vertical handoff and the enhancements to TCP algorithm suggested in the literature

<table>
<thead>
<tr>
<th>Event</th>
<th>Problem</th>
<th>TCP Enhancements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handoff from low delay to</td>
<td>Spurious</td>
<td>Eifel [95], F-RTO [126]</td>
</tr>
<tr>
<td>high delay</td>
<td>RTOs</td>
<td>Reducing the difference between RTTs before and after handoff [59],</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Retransmitting the lost packet without congestion control actions [93]</td>
</tr>
<tr>
<td>Handoff from high delay to</td>
<td>Packet</td>
<td>Eifel [95], DSACK [48]</td>
</tr>
<tr>
<td>low delay</td>
<td>Reordering</td>
<td>Nodupack: avoid sending dupacks [56]</td>
</tr>
<tr>
<td>Handoff from high BDP to</td>
<td>Packet</td>
<td>Freeze TCP [49], Handoff option [84]</td>
</tr>
<tr>
<td>low BDP</td>
<td>Losses</td>
<td>cwnd reduction [56], Overbuffering [53]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Adjusting rwnd [50], ACK pacing [99]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bandwidth-aware scheme [144]</td>
</tr>
<tr>
<td>Handoff from low BDP to</td>
<td>Inability to</td>
<td>Quick-Start to set cwnd and ssthresh [127]</td>
</tr>
<tr>
<td>high BDP</td>
<td>catch up</td>
<td></td>
</tr>
<tr>
<td></td>
<td>with high</td>
<td></td>
</tr>
<tr>
<td></td>
<td>bandwidth</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTOs during disconnection</td>
<td>Unused</td>
<td>Retransmit immediate [130]</td>
</tr>
<tr>
<td></td>
<td>connection</td>
<td></td>
</tr>
<tr>
<td></td>
<td>time</td>
<td></td>
</tr>
</tbody>
</table>

5.4 Summary

Vertical handoff is the process which enables an MN to switch between wireless access networks of different link technologies in a seamless manner to combine the best of connectivity, data rate and user preferences. MNs are equipped with multiple radio interfaces to a variety of wireless access networks whose link characteristics such as bandwidth and propagation delay may vary widely. As the behaviour of TCP is strongly dependent on the end-to-end path characteristics, the abrupt changes in the access link characteristics due to a vertical handoff can result in various problems
such as spurious RTOs, packet reordering, packet bursts, packet losses, unused connection time which affect the efficiency of the TCP connection. In this chapter we gave a background on vertical handoff in the context of a discussion of the problems of TCP due to a vertical handoff. This chapter also gave an overview of the literature dealing with the various proposals that have been made to address these problems along with a tabular summary of these proposals.

The remaining chapters of the thesis present our research work based on a detailed study of the behaviour of TCP in the presence of vertical handoff. In the first part of our study we identify the various problems of TCP in different handoff scenarios. We then describe various algorithms to overcome these problems by using cross-layer information regarding the changes in access link characteristics due to a vertical handoff.
Chapter 6

Experimental Study of TCP Behaviour with Vertical Handoff

In this chapter we discuss our findings on TCP behaviour with vertical handoff based on simulations. The purpose of the simulation is to perform a systematic study of the behaviour of TCP in vertical handoff when the bandwidth and delay of the access links vary over a wide-range. We describe the various problems of TCP in vertical handoff scenarios and some of this discussion has not been reported in the literature previously. We also evaluate the TCP behaviour in the case of handoff between access links of the same BDP but different bandwidth and delay. In Section 6.1 we describe the simulation model used in our experiments. In Section 6.2 we study the effect of the changes in bandwidth and delay on TCP behaviour. This chapter is based on our paper [32].

6.1 Simulation Model

The basic idea behind a simulation model for vertical handoff is to provide multiple paths with different characteristics between a mobile node (MN) and its correspondent node (CN). In the simplest topology as shown in Figure 6.1, MN is connected to the base stations BS1 and BS2 using two wireless access points. These base stations in turn are connected to a router R using two independent fixed links. The CN is a host in the Internet and is connected to the router R using a fixed link in this model. When a handoff occurs, MN switches from BS1 to BS2 or vice versa.

Gurtov and Floyd [52] describe a simple simulation model to study the effect of vertical handoff on transport protocols. In their model, a vertical handoff is simulated by changing the link characteristics. In the ns-2 simulator [108], it is possible to instantly change the link bandwidth, delay and buffer size. When the buffer size is changed, the default ns-2 behaviour is not to discard packets which cannot be accommodated within the size of the new buffer. This models a make-before-break handoff as there is no loss of packets during the handoff. To model a break-before-make handoff where packets may be lost during the handoff, in ns-2 it is possible to apply an error model with 100% (or arbitrarily specified loss percentage) loss on the link. The simulation model used in [53] models a vertical handoff as a step change in the wireless link bandwidth, latency and buffer size and so the same link models the characteristics of both the links involved in the handoff. In our
view there are many drawbacks in this approach as follows. In a make-before-break handoff, MN can continue to receive packets for a while from both the old link and the new link. So modelling the same link to represent the old link and the new link makes it difficult to distinguish the packets arriving through these links and treat them separately if needed. Another problem with this modelling is that the link after a handoff will always have the packets of the old link. It is difficult to simulate a new link which may have an empty buffer. Yet another problem with this model is that it is not possible to represent the background traffic of the old link and the new link separately.

The simulation model we use reflects a vertical handoff more realistically by changing the routes the packets take before and after a handoff rather than by just changing the link characteristics. We use the ns-2 routing features to this end as ns-2 supports routing updates on-the-fly. By changing the route metrics of the two links involved in a handoff, packets can be directed along different links. The route metric of the current route is set to a very low value while that of the unused route is to a very high value. When a handoff notification arrives, the route metric is changed to model a make-before-break handoff. To model a break-before-make handoff, an error model with a packet loss rate of 100% is applied to the old link and after the disconnection period, the route metric
of the new link can be set to a small value. In our model we can treat the packets from different wireless interfaces separately. As the wireless link buffers are different, we can represent background traffic in a link conveniently.

The simulation model used in our experiments is shown in Figure 6.2. The MN is capable of switching between two wireless access interfaces, namely Wir1 and Wir2. Both Wir1 and Wir2 have dedicated base stations BS1 and BS2 that are connected to a common access router R which has a link to the CN. The delay and bandwidth of the fixed links are as shown in Figure 6.2.

The nodes Wir1 and Wir2 are introduced to correctly model a break-before-make handoff. In ns-2 when an error model is applied to a node, the packets at that node are dropped. Without Wir1/Wir2, if we apply the error model to BS1/BS2 node, only the packets arriving at BS1/BS2 are dropped and the packets in transit from BS1/BS2 to MN will not be dropped. In order to understand the necessity of the nodes Wir1 and Wir2, let us assume the model without these nodes as in Figure 6.1. For simplicity of explanation, let us assume a handoff between EGPRS and WLAN. In Figure 6.1 the wireless access Interface 1 can be taken as an EGPRS link of 200 Kbps bandwidth and 300 ms one-way propagation delay and the wireless access interface 2 can be a WLAN link of 5 Mbps bandwidth and 10 ms propagation delay. Consider a break-before-make handoff from EGPRS to WLAN starting at $t_0$. If we apply an error model with a packet loss rate of 100% to BS1, all packets arriving at BS1 after $t_0$ are dropped. However, all packets in transit between BS1 and MN will arrive at MN subsequently. This means that roughly for 300 ms after $t_0$, packets still arrive at MN. From MN’s point of view, no packets should arrive over the EGPRS link after $t_0$ since at $t_0$, it assumes that the old link is down. To overcome this problem we introduce the nodes Wir1 and Wir2 between the MN and the respective base stations. When we apply the error model with a loss rate of 100% both to BS1 and Wir1 at time $t_0$, the link between BS1 and Wir1 breaks and the packets that have arrived at BS1 and those in transit between BS1 and Wir1 are lost. Thus MN loses the connection to the base station BS1 at the instant $t_0$ itself. The link between MN and Wir1/Wir2 has “infinite buffer” and “no” propagation delay, i.e., Wir1 and Wir2 are local to MN and do not contribute to the delay of MN’s path to the base station.

In order to drop both data as well as ACKs during a break-before-make handoff, the error model is applied in both directions of the old link. For simulating different bandwidth and delay for the wireless access interface, we set different bandwidth and delay for the links Wir1-BS1 and Wir2-BS2.

### 6.2 Motivation for the Organization of the Experiments

In order to study the effects of changes in link bandwidth and delay on TCP behavior we categorize our experiments into the following three classes: (1) handoff between access links which have the same bandwidth but different delay, (2) handoff between access links which have the same delay but different bandwidth, and (3) handoff between access links which have the same bandwidth-delay product (BDP) but bandwidth and delay differ. The choice of the parameters for the bandwidth and delay cover the entire range of values that are of interest in typical handoff scenarios.

In the first set of experiments, keeping the bandwidth of the access links fixed, we simulate the handoff from a low-delay link to a high-delay link. In these experiments we study both the effect of increase in the delay as well as increase in the BDP after a handoff. In the experiments where
the handoff is from high-delay link to a low-delay link where bandwidth of the links remain fixed, we study the effect of decrease in link delay and decrease in link BDP after a handoff. The effect of decrease/ increase in bandwidth and BDP can be studied with the second set of experiments where the link delay is kept constant. In the third set of experiments we study the effect of changing the bandwidth and delay of the access links in opposite ways so that BDP of the access links involved in the handoff remains the same. Our third set of experiments cover the wireless overlay networks where bandwidth and delay may change in the opposite direction while the BDP of the access links remains more or less the same.

We use the TCP SACK algorithm implemented in the ns-2 simulator. The SACK algorithm implemented in the ns-2 network simulator is called the Sack1 algorithm [44] and we use this algorithm as the baseline TCP in our simulation experiments in Chapters 4 and 7. Sack1 is basically the conservative SACK algorithm [21] except for its behaviour towards partial ACKs. When a partial ACK is received, Sack1 decreases the pipe by two because a successful partial ACK means that both the original packet (lost) and the retransmitted packet (arrived at the receiver) have left the network. Sack1 can now send two packets for each partial ACK and as a result Sack1 fast recovery is as fast as slow start.

Our experimental study is based on observations from a single bulk TCP flow from the CN to the MN. By confining attention to a single flow we are able to clearly identify the behaviour of TCP in the presence of a vertical handoff. The TCP packet size is 1500 bytes with the TCP/IP headers included. The router buffer size of each link is set to the BDP of the link if the BDP is greater than five packets; otherwise, it is set to five packets. In our experiments a handoff can occur once in the lifetime of a TCP connection in any of slow start, slow start overshoot, fast retransmit/fast recovery and congestion avoidance phases. In our experiments a 20-second interval is chosen to cover all the phases of a TCP connection and a handoff can occur uniformly in any of the 200 points at 100 ms intervals once in the lifetime of a TCP connection. Each experiment is conducted 200 times corresponding to the handoff occurring at these 200 points in the 20-second interval. The duration of each test run includes the completion of the handoff occurring in the 20-second interval.

Both make-before-break as well as break-before-make handoffs are examined. The disconnection period for break-before-make handoff is taken to be 500 ms. As discussed in Section 6.3.2, the handoff delay which is the disconnection period for break-before-make handoff, consists of the time taken for network discovery, authentication and for sending the binding updates. In our experiments we take the handoff delay to be the minimum time for the binding updates from MN to reach the CN and it corresponds to the the one way propagation delay of the new path after the handoff. In our experiments we choose 500 ms as the disconnection period which is greater than the least one way propagation delay of the access link (300 ms). As discussed in 6.2.5 the increase in disconnection period can further aggravate the problems of TCP due to a vertical handoff and so our analysis is still applicable.

No link errors are modelled in our simulations as we assume that the packet losses are either due to disconnection or congestion. This choice is made as the present study is to isolate the effect of vertical handoff on TCP.

In all the experiments, the parameter bandwidth/delay of either the old access link or the new access link is kept constant and we vary the parameters of the other link involved in the handoff. As we are interested in studying the behaviour of TCP with a vertical handoff we study how TCP behaves
6.3. Analysis of the Effect of Changes in Bandwidth and Delay

In the section we describe the specific experimental setup and the corresponding analysis of the results for the three handoff scenarios namely, change in delay with fixed bandwidth, change in bandwidth with fixed delay, and change in delay and bandwidth for fixed BDP of the access links involved in a handoff.

6.3.1 Changes in Delay

The aim of these experiments is to study the effect on TCP of a change in the access link delay arising from a vertical handoff. We vary the delay of one of the links involved in a handoff while the delay of the other link is kept fixed at 300 ms. The varying link delays are 150 ms, 75 ms, 37 ms, 18 ms 9 ms and 1 ms so that the ratio between the delays of the two links is 2, 4, 8, 16, 32 and 300 respectively. This range is wide enough to accommodate the majority of different access links deployed at present. These experiments are repeated for link bandwidths of 200 Kbps, 1600 Kbps and 6400 Kbps. As the link delay is the varying parameter in all the experiments, we study the behaviour of TCP with handoff from a low-delay link to high-delay link and high-delay link to low-delay link separately.

Handoff from a low-delay link to a high-delay link

Figure 6.3 shows the transfer time for make-before-break and break-before-make handoffs for 6400 Kbps, 1600 Kbps and 200 Kbps links as the delay of the old link varies from 150 ms to 1 ms while the delay of the new link is fixed at 300 ms. The main problem that affects the performance of TCP in a make-before-break handoff from a low-delay link to a high-delay link are the occurrence of spurious RTOs and the unnecessary congestion control actions associated with it. As a result of a spurious RTO, the TCP sender retransmits segments unnecessarily and decreases the sending rate by reducing the cwnd and ssthresh.

Given that the delay increase is significant, the small RTO value based on the measurements over the low-delay path causes the TCP retransmission timer to expire spuriously as the ACKs take the high-delay link after the handoff resulting in a significant increase in RTT. Typically no packet losses occur during a make-before-break handoff. Hence when the TCP sender times out spuriously, it retransmits a full window of segments unnecessarily and continues in congestion avoidance with reduced cwnd (cwnd =1) resulting in performance degradation.
Figure 6.3: Handoff from a low-delay link to a high-delay link. Transfer time for 100 packets after a make-before-break (MBB) and a break-before-make (BMM) handoff with a fixed bandwidth (200 Kbps, 1600 Kbps and 6400 Kbps links) with varying delays for the old link. The delay of the new link is fixed at 300 ms.

We can see in Figure 6.3 that due to the adverse effect of spurious RTOs, the performance penalty with the make-before-break handoff becomes more severe with the increase in the ratio of the delays of the old and the new links. We observe that for 1600 Kbps and 6400 Kbps links with make-before-break handoff, spurious RTOs occur in more than 85% of the handoff points when the delay of the new link is at least eight times the delay of the old link and they occur in less than 20% of the handoff points when this ratio is less than 8. For low-bandwidth links, such as 200 Kbps links, serialization delay reduces the ratio between the old and new link delay thereby reducing the occurrence of spurious RTOs. Our experiments with 200 Kbps links show that the spurious RTOs occur only in 10-40% of the handoff points, increasing with the decrease in the old link delay.

Our experiments show that the reduction in ssthresh due to the occurrence of more than one RTO is a key factor in affecting the TCP performance in a break-before-make handoff. If more than one RTO has occurred, i.e., a retransmission is lost, then the ssthresh value is further reduced and the recovery of the lost packets is carried out mostly in the congestion avoidance phase. However, in a break-before-make handoff, the retransmission is lost due to disconnection and not due to congestion and so making this reduction in ssthresh is unnecessary. The other problems affecting the break-before-make handoff are packet losses and the unused connection time. As explained in Section 5.2.3 the unused connection time delays the start of the recovery.

In Figure 6.3 we can see that for the 6400 Kbps links, the break-before-make handoff from a 75 ms link to a 300 ms link shows a sharp increase in transfer time. Due to the relatively high BDP of the 75 ms link (80 packets) a large number of packets are lost due to disconnection. This typically requires an RTO recovery. The retransmission timer expires once during the disconnection period.
of 500 ms and the TCP sender doubles the RTO value. Another RTO is required to recover the losses and $ssthresh$ is reduced to 2 after this RTO. This reduction in $ssthresh$ is the main reason for the long transfer time. We observe an unused connection time (100 ms to 300 ms) after the handoff completes which increase in transfer time. For the handoff from the 150 ms link, the retransmission timer will not expire during the disconnection period of 500 ms and lost packets are recovered by a single RTO recovery and $ssthresh$ is reduced just once. With smaller link delays (i.e., 9 ms and 1 ms) the link BDP is small and only a small number of packets are lost during the disconnection. For the break-before-make handoff between 200 Kbps links, the link BDP is less than 5 packets, which never results in significant number of packet losses and the transfer time remains roughly the same in all cases.

**Handoff from a high-delay link to a low-delay link**

Figure 6.4 shows the make-before-break and break-before-make handoffs from a 300 ms delay link to a new link with varying link delays when the link bandwidth is 200 Kbps/1600 Kbps. With a make-before-break handoff from a high-delay link to a low-delay link, the main problems of TCP are due to (i) the decrease in the link BDP after a handoff resulting in congestion-related packet losses, (ii) the ACKs arriving through the new link trigger more packets to the low BDP link resulting in packet losses and (iii) the slow convergence of RTO to the new path delay.

![Figure 6.4: Handoff from a high-delay link to a low-delay link with a fixed bandwidth (200 Kbps and 1600 Kbps). Transfer time for 100 packets after a make-before-break (MBB) and break-before-make (BBM) handoff with varying delays of the new link. The delay of the old link is fixed at 300 ms.](image)

In the case of 1600 Kbps links, when a make-before-break handoff occurs from a 300 ms link to a low delay link, many packets are dropped due to the large decrease in BDP which require RTO recovery. For the low link delays (delay from 18 ms down to 1 ms), after a make-before-break
handoff, as the ACKs arrive through the new fast link the sender injects packets quickly to the new low BDP link which further congests the link. The initial packet losses due to the decrease in BDP and the packet losses due to the high sending rate after the handoff cause a series of RTOs. This accounts for the transfer time being nearly the same or greater than the corresponding transfer time for break-before-make handoff when the new link delay decreases from 18 ms to 1 ms. The large difference between the median and the quartiles for the make-before-break handoff when the delay of the new link is either 9 ms or 1 ms link is due to the variable number of RTOs required for the loss recovery. The high buffering in the old high BDP link inflates the RTO value and invoking RTO recovery takes a long time due to the slow convergence of the RTO to the new path values. We observe a similar TCP behaviour for the handoff between 6400 Kbps links. In the case of 200 Kbps links, the decrease in BDP due to handoff is within 5 packets for all the delay values of the new link. As a result the graph for make-before-break handoff with a 200 Kbps link is nearly constant. In a break-before-make handoff from a high delay link to a low-delay link, all packets sent during the disconnection are lost in addition to packets lost due to the decrease in BDP. This accounts for the increase in transfer time of a break-before-make handoff compared to that of a make-before-break handoff in a similar scenario.

Packet reordering is observed when a make-before-break handoff occurs from a high-delay to a low-delay link as packets with higher sequence numbers traversing the new low-delay link arrive at the receiver earlier than the packets sent through the old high-delay link before handoff. A false fast retransmission will be triggered only when at least a dupthresh number (usually 3) of out-of-order segments for each in-order segment are received. As the bandwidth remains the same before and after the handoff, sufficient dupacks may not be generated to trigger a false fast retransmission.

6.3.2 Changes in Bandwidth

In this set of experiments, we vary the bandwidth of the links involved in the handoff while keeping the delay of the links constant. The bandwidth of one of the links is varied while the bandwidth of the other link is kept fixed at 6400 Kbps. The varying link bandwidths are 200 Kbps, 400 Kbps, 800 Kbps, 1600 Kbps and 3200 Kbps. The experiments are conducted for the link delays of 300 ms, 75 ms, 9 ms and 1 ms. As the link bandwidth is the only variable, we study the behaviour of TCP in a handoff from a low-bandwidth link to a high-bandwidth link and from a high-bandwidth link to a low-bandwidth link separately.

Handoff from a high-bandwidth link to a low-bandwidth link

Figure 6.4 shows the transfer time for make-before-break and break-before-make handoffs when the bandwidth of the old link is fixed at 6400 Kbps and the bandwidth of the new link is varied. The link delay values used here are 300 ms and 75 ms. The major problem affecting TCP here is the packet losses due to decrease in BDP. For the break-before-make handoff, the recovery of the lost packets due to disconnection increases the transfer time compared to that of the make-before-break handoff. For the 6400 Kbps /300 ms link, the slow-start overshoot starts around 9.3 seconds resulting in large packet losses due to the high BDP of the link (320 packets) which leads to an RTO recovery. It can be seen in Figure 6.5 that for links with 300 ms delay the BDP decrease is maximum when the
bandwidth decreases from 6400 Kbps to 200 Kbps and the transfer time shows maximum increase. When a handoff from 6400 Kbps to 200 Kbps occurs during the slow-start recovery, TCP needs a series of RTOs to recover the lost packets as there is a significant decrease in the BDP of the new link (from 320 packets to 10 packets). This accounts for the high third quartile value (almost double the median) of the transfer time. The high transfer time decreases with increase in bandwidth of the new link. For the break-before-make handoff, the recovery of the lost packets due to a disconnection increases the transfer time compared to that of the make-before-break handoff. The behaviour of the make-before-break and break-before-make handoffs between 75 ms delay links is similar to that of the corresponding handoffs between 300 ms delay links.

Figure 6.6 gives the transfer time for the handoff between low-delay links. In the make-before-break handoff between low-delay links (9 ms and 1 ms), the increase in serialization delay due to the decrease in bandwidth after a handoff may result in spurious RTOs. The major problems affecting TCP here are the decrease in BDP and spurious RTOs. For higher delay links (i.e., 300 ms link), the serialization delay adds little to the total delay and no spurious RTOs are observed.

**Handoff from a low-bandwidth link to a high-bandwidth link**

Figure 6.7 shows the transfer time for the make-before-break and break-before-make handoffs between 300 ms and 9 ms links when the handoff occurs from a low-bandwidth link to a high bandwidth bandwidth link. The bandwidth of the old link is varied and the new link bandwidth is fixed at 6400 Kbps. The main problem affecting TCP here is its inability to efficiently utilize the high bandwidth

---

**Figure 6.5**: Handoff from a high-bandwidth link to a low-bandwidth link with fixed delay (75 ms, 300 ms). Transfer time for 100 packets after a make-before-break (MBB) and a break-before-make (BBM) handoff between same delay links with varying old link bandwidth. The bandwidth of the old link fixed at 6400 Kbps.
available after a handoff. We can see in Figure 6.6 that the transfer time between 300 ms delay links depends mainly on the bandwidth of the old link even though the new link bandwidth/delay is 6400 Kbps/300 ms in all the cases and the new link has a higher BDP than the old link. For the handoffs occurring during or after the slow-start overshoot, the reduced $cwnd$ and $sthresh$ of the old path decrease the TCP sending rate even though a high BDP link is available after the handoff. In the worst affected case of handoff from a 800 Kbps/300 ms link, the slow-start overshoot starts at about 6.5 seconds and approximately 120 packets are lost. The first lost packet is retransmitted in Fast retransmit but the ACK gets delayed resulting in a spurious RTO. As the link is already congested due to slow-start overshoot, some retransmitted packets are also lost resulting in another RTO expiry. The consequent reduction in $cwnd$ and $sthresh$ values further reduces the sending rate for scenarios where handoffs occur after the slow-start overshoot. This accounts for the high variation in the lower and upper quartiles of the time-taken to transfer 100 packets after a handoff. There is an additional burden to the recovery in break-before-make handoff owing to packet losses due to disconnection.

TCP behaviour for make-before-break handoff between 9 ms delay links is similar to the behaviour we described for 300 ms links. In the case of break-before-make handoff, TCP will be in RTO recovery during the disconnection period and another RTO is required to recover the losses. As explained in Section 6.3.2, the unused connection time and the $sthresh$ reduction are the main problems due to disconnection. The high RTO value of the low-bandwidth links increases the unused connection time resulting in increased transfer time. For 200 Kbps links, the unused connection time is about 700 ms and as the bandwidth increases the unused connection time decreases. As the
Figure 6.7: Handoff from a low-bandwidth link to a high-bandwidth link with fixed delay (9 ms, 300 ms). Transfer time for 100 packets after a make-before-break (MBB) and a break-before-make (BBM) handoff between same delay links with varying old link bandwidth. The bandwidth of the new link fixed at 6400 Kbps.

RTO value gets clamped to the minrto, the unused connection time for higher bandwidths (from 800 Kbps) is 100 ms. As the ssthresh reduction is not significant here as the BDP of the old link for all the bandwidths used in the experiments are less than or equal to 5.

6.3.3 Changes in Bandwidth and Delay for a Fixed Bandwidth Delay Product (BDP)

In this set of experiments, the bandwidth and delay of the links involved in a handoff are different while their BDP remains unchanged. We vary the bandwidth and delay of one of the links involved in the handoff while the bandwidth and delay of the other link are kept fixed. We have two sets of fixed bandwidth/delay values namely, 200 Kbps/300 ms and 6400 Kbps/9 ms. For the varying link, the bandwidth/delay combinations are 200 Kbps/300 ms, 400 Kbps/150 ms, 800 Kbps/75 ms, 1600 Kbps/37 ms, 3200 Kbps/18 ms and 6400 Kbps/9 ms. With these combinations, both old and new access links have a BDP of 10 packets. We perform the experiments for a handoff from a high-bandwidth/low-delay link to a low-bandwidth/high-delay link and vice versa.

Handoff from a high-bandwidth/low-delay link to a low-bandwidth/high-delay link

Figure 6.8 shows that a significant decrease in bandwidth/increase in delay due to a handoff increases the transfer time for both make-before-break and break-before-make handoffs. Here we fix the new link bandwidth/delay at 200 Kbps/300 ms while varying the bandwidth and delay of the old link.
When there is a significant increase in delay after a make-before-break handoff, TCP suffers from spurious RTOs whereas in a break-before-make handoff, unused connection time and the ssthresh reduction are the main problems. Spurious RTOs occur in more than 90% of the handoff points when the ratio of change in delay is at least 8.

When the old link bandwidth is 1600 Kbps or higher, many packets are lost in a make-before-break handoff due to the bursty transmission caused by the arrival of late ACKs at the high rate of the old link resulting in heavy congestion on the new low-bandwidth link. In most of the cases, recovery needs one or more RTOs and the reduction in the sending rate is drastically affected by the reduced ssthresh and cwnd.

**Handoff from a low-bandwidth/high-delay link to a high-bandwidth/low-delay link**

Figure 6.8 shows the performance of make-before-break and break-before-make handoffs from a low-bandwidth/high-delay link to a high-bandwidth/low-delay link when the new link is fixed at 6400 Kbps/9 ms. Packet reordering is a problem affecting TCP when there is a significant reduction in delay after a make-before-break handoff. In the case of break-before-make handoff with a high delay old link (for 200 Kbps/300 ms and 400 Kbps/150 ms links), the high RTO value prolongs the start of the RTO recovery. For links with lower delay (from 75 ms to lower link delays) the increase in transfer time is mainly due to the occurrence of multiple RTOs and the resulting reduction in ssthresh. The maximum unused connection time and the reduction in ssthresh account for the peak
Figure 6.9: Handoff from a low-bandwidth/high-delay link to a high-bandwidth/low-delay link with fixed BDP. Transfer time for 100 packets after a make-before-break (MBB) and a break-before-make (BBM) handoff for the same BDP links with varying bandwidth and delay of the old link and the bandwidth and delay of new link fixed at 6400 Kbps/9 ms.

in the transfer time for the break-before-make handoff from 800 Kbps /75 ms link to 6400 Kbps/9ms link in Figure 6.9

6.4 Summary

In this chapter we described our simulation experiments to study the behaviour of TCP in the presence of handoff. We presented a simulation model which includes an accurate modelling of both make-before-break and break-before-make handoffs. We described our experiments involving both make-before-break and break-before-make handoffs to bring out the specific problems of TCP in the different handoff scenarios and to show the extent to which TCP is affected by the changes in bandwidth and delay. Our experiments are divided into three categories, namely, (1) handoff between links which have the same bandwidth but different delay, (2) handoff between links which have the same delay but different bandwidth, and (3) handoff between links which have the same BDP but bandwidth and delay differ. These experiments enabled us to study the effects of changes in delay and in bandwidth separately and together. The choice of the parameters for the bandwidth and delay cover the entire range of values that are of interest in typical handoff scenarios.

While spurious RTOs are the main problem with the increase in access link delay after a handoff, packet reordering and slow convergence to the new low RTO value are the problems when the handoff occurs to a low-delay access link. The slow convergence to the low RTO value increases the recovery time especially when packet losses occur due to large decrease in BDP after a handoff.
As the disconnection time of a break-before-make handoff increases, multiple RTOs occur and the unused connection time increases. Another serious problem in this scenario is the repeated reduction in \textit{ssthresh} due to multiple RTOs which allows the recovery only in congestion avoidance and that too with very small window. One of the interesting results of the study is that TCP behaviour is affected due to a vertical handoff between same-BDP links. Packet losses occur when the bandwidth of the new access link after a handoff is less than eight times the bandwidth of the old link even when the BDP of the two access links remain the same.

The results obtained in this chapter provide a basis for developing the enhanced algorithms described in Chapter 4 to improve TCP performance in vertical handoff. Further discussion of the TCP behaviour is given in the context of our results obtained for enhanced TCP for different handoff scenarios.
Chapter 7

Cross-Layer Enhanced TCP Algorithms

In this chapter, we propose solutions to mitigate the problems of TCP in the presence of a vertical handoff. These problems have been described in detail in Chapter 6. We also bring out additional details on these problems while we discuss the proposed enhancements to the TCP sender algorithm. Our algorithms described here take a conservative approach in setting the TCP congestion control parameters. The enhancements are invoked upon arrival of a handoff notification from the lower layer. Here we assume that the MN sends the handoff notification to the TCP sender at the CN which includes an estimate of the bandwidth and delay of the old and new access links involved in the handoff. The baseline TCP used in our experiments is the TCP SACK algorithm implemented in ns-2 simulator and we refer to it as regular TCP. Section 7.1 describes the motivation for cross-layer notifications to TCP in vertical handoff along with a summary of the previous work in this area. In Section 7.2, we develop enhancements to the TCP sender algorithm to avoid the various problems of TCP due to a handoff. In Section 7.3, we compare the performances of the regular TCP and enhanced TCP. This chapter is based on our paper [33].

7.1 Cross-layer Notifications

In Chapter 6, we presented an overview of the various mechanisms to support host mobility in IP networks [40, 79, 89, 113, 134, 141]. They include both basic protocol support for IP mobility [79, 113, 141] and various enhancements to reduce packet losses as well as the amount and latency of signalling [40, 89, 134]. All these mobility management mechanisms aim at hiding the host mobility from the layers above IP. However, this is not a viable approach if the implications of the mobility interact with the segment delivery at the transport layer. A vertical handoff that results in significant changes in end-to-end path properties cannot be totally hidden from the transport layer even if the handoff latency is reduced to a minimum and no packets are dropped. The TCP congestion control algorithms enable TCP to adapt to the changes in path characteristics; however, it may be slow and take TCP several RTTs. The TCP response in the immediate vicinity of the handoff may be inefficient due to poor utilization of the available bandwidth or may lead to loss of packets causing congestion response. Therefore it would be advisable to explicitly notify the transport layer of any significant changes in path properties.
Korhonen et al. [91] discuss the motivation and design goals for link characteristic information delivery when the MN moves from one access network to another. The existing mobile protocols do not have the facility to indicate the type of the link that MN is currently attached to. The information regarding the link characteristics is valuable to transport protocols especially in vertical handoff scenarios where the new link characteristics differ widely from that of the old. The requirements of such link characteristics notifications from the MN to both HA and CN should be independent of the mobility protocol used but it should be compatible with the existing mobility protocols like MIP, HIP, SCTP and SIP. The delivery of the link characteristics information should be independent of the underlying link technology and it should not increase the signalling over the wireless links significantly.

Park et al. [110] introduce a model for delivery of link characteristics information from an MN to both HA and CN when a vertical handoff occurs or when there is a significant change in the link characteristics. This model is applicable to both MIPv4 and MIPv6. A new mobility option called link characteristics information is defined to carry the link information. This option can be included in the binding update message of MIPv6 or with the registration request message of MIPv4. This option includes the information regarding the type of the current link, its bandwidth, latency and other relevant details, if any. On receiving the link characteristic information, the upper layers at the CN, for instance, the transport layer, can take action to shape their traffic to the MN.

Korhonen et al. [90] analyze whether the various mobility protocols are capable of delivering the link and path characteristic information (LPCI) between the communicating end points (i.e., between the MN and the CN). They observe that with MIPv4, it is not possible to exchange control messages between the MN and the CN. With the route optimization feature of MIPv6, it is possible to send the LPCI with the binding update messages. It is possible to send the LPCI as a part of the TCP options. The setting of TCP’s URG flag in the TCP header for the packets with the LPCI data will allow the end nodes to process the options without delay. Regarding the HIP protocol, the LPCI data can be exchanged between the end nodes as a part of the HIP UPDATE packet that is used when an MN moves to a new address. It is possible to send the LPCI information if SCTP or SIP is used for mobility signalling while DCCP is not be a suitable candidate for LPCI delivery.

As described in Section 5.3, Schütz et al. [131] describe a Connectivity Change Indication (CCI) from the lower layers to TCP on a per-connection basis which is generic and link-technology-independent. CCI can either be delivered to the TCP at the local stack or sent as a TCP option to the TCP at the peer.

Cross-layer notifications have been shown to be beneficial to TCP when path characteristics change widely due to a vertical handoff [127, 130]. During a vertical handoff, the TCP layer at the MN can be locally notified of the changes in the path characteristics. This information can be sent to the TCP layer at the CN as a TCP option [131], or along with the mobility registration message such as the binding update message in Mobile IPv6 [79] or along with the UPDATE packet in HIP [105] to be further forwarded to the TCP layer. A TCP sender can interpret the notification from the lower layers as hints about the characteristics of the network path, adjust the congestion control parameters and RTO estimate so as to adapt to the new path in an efficient and timely manner.

In our experiments we assume that the delivery of the notifications is piggybacked in the mobility signalling messages so that they can be delivered to the TCP layer exactly when the handoff completes. We assume that the notifications include an estimate of the bandwidth and delay of
the access links involved in the handoff. The TCP enhancements we propose in this chapter make use of the cross-layer information that is delivered to the TCP at the CN in this manner. In the absence of this information, the behaviour of TCP is not affected by these enhancements and we get the performance of the regular TCP. The TCP enhancements proposed here are specific to the TCP sender and no change is required in the TCP receiver.

7.2 Discussion of the Proposed TCP Enhancements

In this section we describe the TCP enhancements proposed in this thesis. They are TCP sender-based algorithms that address the problems of spurious RTOs, congestion-related packet losses, prolonged disconnection and the slow convergence of RTO arising from a vertical handoff. The proposed algorithms aim at minimizing the effect of these problems in various handoff scenarios and we evaluate the proposals through a simulation study. A detailed comparative analysis of regular TCP behaviour along with the behaviour of the enhanced TCP in the context of the above problems is presented in the following sections.

As the end-to-end path characteristics of a TCP connection can be assumed to be relatively stable over the lifetime of a connection [115], we consider that the changes in the path characteristics are mainly caused by the changes in the access link characteristics due to a vertical handoff. We calculate a set of parameters from the access link characteristics as an approximation to the end-to-end path characteristics and conservatively make use of these quantities in our algorithms.

7.2.1 TCP Enhancements to Avoid Spurious RTOs

A more detailed analysis of the results discussed in Section shows that in a make-before-break handoff spurious RTOs can occur primarily due to the following conditions:

- Case (1): there is a significant increase in link delay after the handoff,
- Case (2): TCP is in fast recovery when the handoff occurs, and
- Case (3): TCP enters fast recovery after the occurrence of the handoff but before the ACKs for all packets sent before the handoff are received.

Next we describe these conditions in detail.

Case (1): In Sections and we observed that spurious RTOs occur in more than 85-90% of the handoff points when the delay of the new link is at least 8 times that of the old link. A typical scenario giving the time-sequence graph of a make-before-break handoff from a 6400 Kbps/9ms link to a 200 Kbps/300 ms link is shown in Figure 7.1(a). We see that handoff occurs at 8.2 seconds after the beginning of a TCP connection, causing a sudden increase in RTT as the ACKs start taking the new high-delay link. A spurious RTO occurs at 8.4 seconds and TCP retransmits the first unacknowledged segment. The late ACKs for the segments sent before the handoff start arriving at 8.5 seconds triggering a retransmission of a full window of 21 segments unnecessarily. As the late ACKs arrive back-to-back roughly at the line rate of the old high-bandwidth link, unnecessary retransmissions are triggered at a rate which far exceeds the capacity of the new link. Therefore the new link is congested soon. The late ACKs for data segments that took the new path after the
Figure 7.1: Comparison of regular TCP (a) and enhanced TCP (b): make-before-break handoff at 8.2 seconds from a 6400Kbps/9ms link to a 200Kbps/300ms link.
handoff start arriving at 8.8 seconds, triggering transmission of new data segments. These segments enter the router queue in front of the new link which is filled with unnecessary retransmissions and experience a long queuing delay in addition to the high delay of the new link. As the RTO estimate converges very slowly to the long RTT of the new path, the total delay for these segments exceeds the current RTO value resulting in another spurious RTO at 9.7 seconds followed by unnecessary retransmission of the current window again.

Case (2): Figure 7.2(a) shows an example where TCP is in fast recovery when a make-before-break handoff occurs from a 800 Kbps/75 ms link to a 200 Kbps/300 ms link. The TCP sender congests the old link and fast retransmits at 5.9 seconds. The retransmitted segment enters a long queue of the bottleneck router. After this a handoff occurs at 6.0 seconds. The retransmitted segment arrives at the TCP receiver after experiencing a relatively long queuing delay and triggers an ACK that takes the new high delay link, adding further delay in the arrival of the ACK. This causes a spurious RTO at 6.3 seconds. This condition arises even when the delay of the new link is only four times the delay of the old link, which alone is not enough to cause a spurious RTO.

Case (3): Here the TCP sender enters fast recovery after a make-before-break handoff. This situation is illustrated in Figure 7.3(a) The handoff is from a 400 Kbps/150 ms link to a 200 Kbps/300 ms link, where the ratio of change in delay is only 2. The sender has already congested the old link after which a handoff occurs at 10.9 seconds. The TCP sender, unaware of the handoff, continues transmitting new segments clocked by the ACKs arriving roughly at the line rate of the old link and thereby quickly filling the router queue in front of the new link. When the third dupack arrives at 11.6 seconds, indicating a packet loss on the old link, the TCP sender fast retransmits the lost segment. However, the queuing delay on the new link delays the delivery of the fast retransmitted segment and spurious RTO occurs at 12.4 seconds.

In addition to the cases discussed above there are scenarios where queuing delay increases due to the increase in link serialization delay resulting in spurious RTOs as the ACKs get delayed. This situation arises in make-before-break handoffs from a high-bandwidth link to a low-bandwidth link when TCP is either in fast recovery before a handoff or TCP enters fast recovery after a handoff. As the effect of serialization delay is conspicuous for low delay links, this accounts for the occurrence of spurious RTOs for a make-before-break handoff from a high-bandwidth link to a low-bandwidth link described in Section 6.3.2 for the same delay link experiments (9 ms and 1 ms).

To avoid the occurrence of spurious RTOs we calculate the following parameters which are used in the enhanced TCP sender algorithm. In a make-before-break handoff where neither the data segment nor its ACK is lost, we may consider the data segment sent just before handoff traversing the old link with its ACK traversing the new link. The minimum RTT of this data segment-ACK pair can be calculated by the following formula taking into account only the access links.

\[ RTT_{old\_newlink} = D_{oldlink} + SD_{pkt_{oldlink}} + D_{newlink} + D_{ack_{newlink}} \]

where -

- \( SD_{pkt_{oldlink}} \) - Serialization delay for a data packet on the old link
- \( SD_{ack_{newlink}} \) - Serialization delay for an ACK on the new link
- \( D_{oldlink} \) - Propagation delay for the old link
- \( D_{newlink} \) - Propagation delay for the new link
Figure 7.2: Comparison of regular TCP (a) and enhanced TCP (b): make-before-break handoff at 6.0 seconds from a 800Kbps/75ms link to a 200Kbps/300ms link.
Figure 7.3: Comparison of regular TCP (a) and enhanced TCP (b): make-before-break handoff at 10.9 seconds from a 400Kbps/150ms link to a 200Kbps/300ms link.
When handoff notification arrives indicating an increase in delay / decrease in bandwidth:
If (TCP not in RTO recovery)
    Save minrto
/* Case 1 and Case 2 */
If ((RTT\text{old\_newlink} > currentRTO) OR
    (TCP is already in Fast Recovery))
    minrto = RTT\text{newlink} + \text{FlightSize}/BW_{\text{old\_link}} + 200\text{ms}
    Update RTO timer
/* Case 3 */
If (TCP enters Fast Recovery before all packets sent
    before handoff are ACKed)
    minrto = RTT\text{newlink} + \text{FlightSize}/BW_{\text{new\link}} + 200\text{ms}
    Update RTO timer
/* Case 1, Case 2 and Case 3 */
If (there is an increase in link delay after handoff)
    When all packets sent before handoff are ACKed
    Initialize RTT variables as for a new connection
    When ACK for a new data segment arrives
    Update the RTT variables
Restore minrto

Figure 7.4: Algorithm A1 to avoid spurious RTOs.

The RTT of the data segment-ACK pair traversing the new link is calculated based on the new access link delays:
$$RTT_{\text{new\link}} = 2 \cdot D_{\text{new\link}} + SD_{\text{pkt\_new\link}} + SD_{\text{ack\_new\link}}$$

The new link BDP is calculated as follows:
$$BDP_{\text{new\link}} = BW_{\text{new\link}} \cdot RTT_{\text{new\link}}$$
where
$$BW_{\text{new\link}}$$ - Bandwidth of the new link

The incRTO algorithm proposed by us in [31] to avoid spurious RTOs maintains the RTO value at 3 seconds until all packets sent before handoff are ACKed. Even though this simple approach was shown to be very effective in avoiding spurious RTOs, we found in our experiments that the high RTO value can delay the RTO recovery in some break-before-make scenarios where TCP will not be in RTO recovery when a handoff notification arrives. Furthermore a fixed value of RTO will not reflect the path RTT. So here we calculate the minrto (minimum RTO value) based on the new access link delay and update the RTO timer immediately so that the new minrto takes into effect. Thus any change in the delay of the end-to-end path will be better reflected in the RTO calculation.

The proposed algorithm to avoid spurious RTOs is given in Figure 7.4. The algorithm takes into account all three cases discussed above and it is invoked on the arrival of a handoff notification indicating an increase in the link delay. Here we calculate the minrto based on the new access
7.2. DISCUSSION OF THE PROPOSED TCP ENHANCEMENTS

link delay and update the RTO timer immediately so that the new \textit{mrnto} comes into effect. Thus any change in the delay of the end-to-end path will be better reflected in the RTO calculation. It, however, takes effect only if the TCP sender is not already in RTO recovery when the notification arrives.

Regarding Case (1) and Case (2), the spurious RTO occurs either due to a significant increase in the link delay or due to queuing delay in the old link. In order to address Case (1), the TCP sender checks if the \textit{RTT} \textit{old \_newlink} (the minimum estimated RTT for a data segment traversing the old link and its ACK taking the new link) is greater than the current RTO. For Case (2) the algorithm checks if the TCP is already in fast recovery when a handoff notification arrives. If the TCP sender is in fast recovery, the queuing delay in the old link may increase the effective RTT for the fast retransmitted segments beyond the current RTO value even though the \textit{RTT} \textit{old \_newlink} is not larger than the current RTO. If any of these conditions is true, we set the \textit{mrnto} to the sum of the \textit{RTT} \textit{newlink}, an estimate for the queuing delay in the old link and a rough estimate of the rest-of-the-path delay (here taken as 200 ms which is the default \textit{mrnto} value used in ns-2 and in many real TCP implementations). Strictly speaking, instead of \textit{RTT} \textit{newlink} we can use \textit{RTT} \textit{old \_newlink} but here we slightly overestimate the new \textit{mrnto} value.

In order to address Case (3) the TCP sender sets the \textit{mrnto} to the sum of the \textit{RTT} \textit{newlink}, a rough estimate for the queuing delay in the new link and a rough estimate of the rest-of-the-path delay.

In addition to setting the \textit{mrnto} value, we update the RTO timer immediately to allow the new \textit{mrnto} value to take effect as soon as possible. When the ACK for all the segments sent before the handoff has been received, we initialize the RTT variables and the RTO timer as recommended for a new connection in RFC 2988 [111] provided there is an increase in the link delay after the handoff. The RTT variables are updated immediately upon arrival of the first valid ACK and the \textit{mrnto} is restored to its default value.

The effectiveness of the algorithm in avoiding spurious RTOs can be seen in Figures 7.1(b) and 7.3(b) against the corresponding cases shown in Figures 7.1(a) and 7.3(a) for regular TCP.

Figures 7.1(a) and 7.1(b) represent the behaviour of TCP with and without the algorithm A1 for Case (1). The ACK for the first outstanding segment arrives after 337 ms but the RTO value of the regular TCP at handoff is only 200 ms resulting in spurious RTO. Algorithm A1 calculates the \textit{mrnto} to be 552 ms, sets RTO to this value, thereby avoiding the occurrence of spurious RTO.

A comparison of Figures 7.2(a) and 7.2(b) shows the effect of algorithm A1 for Case (2). We see that TCP is in fast recovery when the handoff occurs at 6.0 seconds. There are 24 segments outstanding at that time. The arrival of the ACK for the fast retransmitted segment takes 529 ms which is more than the RTO value of the regular TCP (440 ms) at the handoff and the RTO timer expires spuriously before the arrival of the ACK. The use of algorithm A1 results in the \textit{mrnto} value of 1220 ms, thereby avoiding spurious RTOs.

For Case (3), Figure 7.3(a) shows that there are 22 segments outstanding when TCP fast retransmits. The arrival of the ACK for the fast retransmitted packet takes 1.17 seconds which is more than the RTO value of the regular TCP (850 ms) at that point and spurious RTO occurs due to the late arrival of the ACK. Using the algorithm A1, the \textit{mrnto} is calculated to be 2.12 seconds and the resulting larger value of RTO is effective in avoiding the spurious RTO as can be seen in Figure 7.3(b).
7.2.2 TCP Enhancements to Minimize Congestion-related Packet Losses

Congestion-related packet losses may occur due to a handoff occurring from a high-BDP path to a low-BDP path. The BDP of the bottleneck link determines the minimum size of the TCP window that may fully utilize the bottleneck link. The congestion point of the bottleneck link is determined by the BDP of the link, the router queue size in front of the link and the number of packets in flight elsewhere on the end-to-end path.

Figure 7.5(a) shows a make-before-break handoff from 1600 Kbps /75 ms link to 400 Kbps /150 ms link. The old link BDP is 20 packets while the new link BDP is 10 packets. When a handoff occurs at 6.5 seconds TCP continues to inject packets to the new link at the previous rate and several packets starting from sequence number 764 are lost. On the new high-delay link it takes about 2.5 seconds for the TCP sender to recover the lost packets and adapt to the sending rate of the new link.

If the FlightSize at the time of handoff is greater than the buffering capacity of the new link, packet losses due to congestion may occur after the handoff. Therefore we check whether the FlightSize exceeds the estimated buffering capacity of the new link. We assume that the router queue size in front of the access link equals the BDP of the link so that the link has a total buffering capacity of twice the link BDP. The total buffer capacity of the end-to-end path is likely to be larger than this estimate allowing some slack in the estimate. In order to avoid the underutilization of the new access link we reduce the congestion window (cwnd) and the slow-start threshold (ssthresh) to the BDP of the new link. The algorithm for reducing congestion-related packet losses is given in Figure 7.5 and is invoked if TCP sender is not in RTO recovery. A flag, cwnd_reduced, is set to 1 so that cwnd is not reduced further if TCP enters fast recovery to recover lost packets sent before the handoff. This flag is cleared when all packets sent before handoff are ACKed.

A comparison of Figure 7.5(a) and Figure 7.5(b) shows the effect of the algorithm A2. The FlightSize when the handoff notification arrives, is 39 packets whereas the BDP of the new link is only 11 packets. As this FlightSize is larger than twice the BDP of the new link, the algorithm A2 sets the ssthresh and cwnd to the BDP of the new link which effectively avoids the congestion-related losses and allows the TCP sender to smoothly continue data transmission over the new link.

7.2.3 TCP Enhancements to Reduce the Effect of Disconnection

As described in Chapter 6 main problems of TCP in a break-before-make handoff are excessive ssthresh reduction, packet losses and unused connection time. A typical break-before-make scenario shown in Figure 7.7 illustrates these problems. It shows a break-before-make handoff between 6400 Kbps links with link delay changing from 9 ms to 300ms in a handoff. The handoff occurs at 7.0 seconds and there are 20 outstanding packets at that time. All the packets from sequence number 3427 to 3446 except 3428 and 3430 are lost due to the disconnection arising from the break-before-make handoff. Although packet 3426 was received, the TCP receiver did not send the ACK for it immediately because of the delayed ACK feature of TCP. In this scenario, we see that the first retransmission timeout occurs at 7.2 seconds and the retransmission of packet 3426 is lost during the disconnection period of 500 ms. A second retransmission of packet 3426 at 7.6 seconds reaches the receiver. After the first retransmission, the ssthresh is reduced to half of the FlightSize (10 packets)
Figure 7.5: Comparison of regular TCP (a) and enhanced TCP (b): make-before-break handoff at 6.5 seconds from a 1600Kbps/75 ms link to a 400 Kbps 150 ms link.
and after the second retransmission the \textit{ssthresh} is further reduced to 2. The loss recovery carried out in congestion avoidance takes a very long time over the high-delay link even though the BDP of the new link is very high (320 packets). So in this scenario the repeated reduction in \textit{ssthresh} contributes more to the worsening of TCP performance than the unused connection time of 100 ms. Algorithm A3 shown in Figure 7.8 is used to reduce the unused connection time and to avoid the unnecessary \textit{ssthresh} reduction. Upon the first expiry of the retransmission timer for a given TCP segment, the TCP sender reduces \textit{cwnd} and \textit{ssthresh} as usual and then saves the reduced \textit{ssthresh} value. If the TCP sender is already in RTO recovery when the handoff notification arrives, the TCP sender immediately retransmits the first unacknowledged packet and restores the saved value of \textit{ssthresh}. This immediate retransmission reduces the unused connection time and the restoration of the \textit{ssthresh} increases the sending rate thereby improving the TCP performance.

The effect of algorithm A3 in reducing the unused connection time and avoiding \textit{ssthresh} reduction can be seen by comparing Figure 7.8(a) and Figure 7.8(b). The improved performance of algorithm A3 is mostly due to the restored \textit{ssthresh} value. With immediate retransmission in algorithm A3, the retransmission occurs at 7.5 seconds as soon as the connection is restored and the unused connection time of 100 ms in the regular TCP is eliminated. However, the effect of unused connection time becomes more visible for longer disconnection periods that cause RTO timer to backoff to a large value.

7.2.4 TCP Enhancements for a Fast Convergence of RTO

In a handoff from a high-delay link to a low-delay link, the RTO value may be very high compared to the new end-to-end RTT. The RTO may be even higher due to the queuing delay, if the old link has a high BDP. After a handoff, the RTO will converge to the RTT of the low-delay path very slowly. This convergence is exceptionally slow when the RTT variables are updated only once in an RTT [111]. The high RTO value delays the timeout recovery unnecessarily if an RTO recovery is needed relatively soon after a handoff. The algorithm A4 given in Figure 7.9 helps TCP to converge faster to the new RTO value by initializing the RTT variables and updating the RTT variables again immediately when an ACK for a data segment sent over the new path arrives to reflect the end-to-end delay of the new path.

We designate the TCP Sack1 sender algorithm incorporating the enhancements given by algorithms A1, A2, A3 and A4 as the \textit{enhanced TCP}. In Section 7.3 we study the behaviour of the enhanced TCP in various handoff scenarios.
Figure 7.7: Comparison of regular TCP (a) and enhanced TCP (b): break-before-make handoff at 7.0 seconds from a 9 ms link to a 300 ms link with identical bandwidth of 6400 Kbps. Disconnection period is 500ms.
On the first expiration of RTO:
   Reduce cwnd and ssthresh as usual
   Save ssthresh

When handoff notification arrives:
   If (TCP in RTO recovery)
      Retransmit the first unacknowledged packet
   Restore ssthresh
   If there is a significant change in delay
      Initialize RTT variables as for a new connection
   When ACK for new data arrives
      Update RTT variables

Figure 7.8: Algorithm A3 to reduce the unused connection time and to restore the ssthresh.

When handoff notification arrives indicating a decrease in delay:
   If (TCP is not in RTO recovery)
      Wait till the segments sent before
      handoff have been ACKed
      Initialize RTT variables as for a new connection
   When ACK for a new data segment arrives
      Update RTT variables

Figure 7.9: Algorithm A4 for fast convergence of RTO.

7.3 Performance Comparison: Regular TCP vs. Enhanced TCP

In this section we discuss the second set of experiments to evaluate the performance of enhanced TCP and compare its performance to regular TCP. The experimental setup used here is the same as in Chapter 6.

7.3.1 Handoff between Same Bandwidth and Different Delay Links

Handoff from a low-delay link to a high-delay link

We recall from Section 6.3.1 that the key problem affecting TCP performance in a make-before-break handoff from a low-delay to high-delay link is the occurrence of spurious RTOs along with the unnecessary congestion control actions associated with it.

Figure 7.10(a) shows the transfer time for 100 packets after a make-before-break handoff for 6400 Kbps links and 1600 Kbps links for regular TCP and enhanced TCP. The delay of the new link is fixed at 300 ms and the old link delay is varied as in Section 6.3.1. For the handoff between 6400 Kbps links enhanced TCP achieves a reduction in transfer time (median value) of 35-65% by avoiding spurious RTOs. When the ratio of the link delays is less than 8 (for new link delays less than 37 ms), enhanced TCP behaviour is similar to that of regular TCP as the spurious RTOs occur.
Figure 7.10: Handoff from a low-delay link to a high-delay link with fixed link bandwidth (1600Kbps, 6400 Kbps). Transfer time for 100 packets after a make-before-break (MBB) handoff (a) and break-before-make (BBM) handoff (b) between same bandwidth links with varying delays of the old link. The new link delay is fixed at 300 ms. The disconnection period for break-before-make handoff is 500 ms.
Figure 7.11: Handoff from a low-delay link to high-delay link. Transfer time for 100 packets after make-before-break (MBB) and break-before-make (BBM) handoffs between 200 Kbps links with varying delays of the old link. The new link delay is fixed at 300 ms and the disconnection time for break-before-make handoff is 500 ms.

rarely in this situation. For handoff between 1600 Kbps links, enhanced TCP behaviour resembles the case of 6400 Kbps links described above and it achieves a reduction in transfer time up to 55 %.

Figure 7.10(b) shows the transfer time for a break-before-make handoff for bandwidth values of 6400 Kbps and 1600 Kbps. With the disconnection period of 500 ms, the regular TCP will be in timeout recovery in most of the handoff points. Here the enhanced TCP applying algorithm A3 will retransmit the first unacknowledged packet immediately when the link comes up and reduce the ssthresh value only once thereby decreasing the delay in the recovery of lost packets allowing the TCP sender to continue with a larger window. In the case of 6400 Kbps links enhanced TCP reduces the transfer time up to 55 %. The performance improvement of enhanced TCP is very significant (about 55 %) for the handoffs from 75 ms delay link to 300 ms delay link. For a 1 ms delay link, the old link buffer is set to a minimum value of 5 and the link BDP is small. This allows only a slight performance improvement for enhanced TCP over regular TCP as ssthresh will have a low value anyway. For link bandwidth values of 1600 the enhanced TCP reduces the transfer time up to 45 %. The algorithm A3 is invoked if TCP is in RTO recovery when the handoff notification arrives. For the handoff from 150 ms link, TCP will not be in RTO recovery and there is no performance improvement in using enhanced TCP in this case.

Figure 7.11 shows the make-before-break and break-before-make handoff from a low-delay link to a high-delay link where the bandwidth of the links are fixed at 200 Kbps. For low-bandwidth links such as 200 Kbps links, serialization delay reduces the ratio between the old and new link delays which decreases the occurrence of spurious RTOs and enhanced TCP performance is only slightly
better than that of regular TCP. The enhanced TCP reduces the transfer time by about 5-10 % in the case of break-before-make handoff for link bandwidths 200 Kbps except for the handoff from a 150 ms delay link to a 300 ms delay link. The improvement is only minor because the serialization delay of 60 ms for 200 Kbps links adds to the path delay resulting in an increased RTO value and TCP will not be in RTO recovery in most of the handoff points. For the handoff from the 150 ms link to the 300 ms link TCP may not be in timeout recovery when a handoff notification arrives and so the enhanced TCP behaviour is similar is that of regular TCP.

Handoff from a high-delay to a low-delay link

![Graph](image)

Figure 7.12: Handoff from a high-delay link to a low-delay link. Transfer time for 100 packets after a make-before-break (MBB) handoff (a) and break-before-make (BBM) handoff (b) between 1600 Kbps links with varying new link delays and the delay of the old link is fixed at 300 ms.

Figure 7.12 shows a comparison of the transfer time for regular TCP and enhanced TCP after make-before-break and break-before-make handoffs from a high-delay link to a low-delay link when both links have the same bandwidth of 1600 Kbps. This scenario can be considered as a handoff from a high BDP link of 80 packets to a low BDP link where the BDP varies from 40 packets to 1 packet.

As discussed in Section 3.3.1, the main problems of the regular TCP in a make-before-break handoff from a high-delay link to a low-delay link (links of the same bandwidth) are (1) decrease in the link BDP after a handoff resulting in congestion-related packet losses and (2) the slow convergence of RTO to the low delay of the new link.

Algorithms A2 and A4 of enhanced TCP enable it to mitigate these problems. Algorithm A2 enables enhanced TCP to minimize the packet losses by setting the cwnd and ssthresh to the BDP of the new link. Algorithm A4 helps a rapid convergence of RTO to the RTT value of the new path. Figure
shows that up to 85 % reduction in the transfer time is possible with enhanced TCP. When a handoff occurs between links of 6400 Kbps bandwidth, the decrease in BDP due to a change in delay can be a maximum of 320 packets and as a result TCP can experience severe packet losses. In this scenario our algorithms help to reduce the congestion related packet losses and improve the TCP performance as in the case of a handoff occurring from a 1600 Kbps/300 ms link.

To understand the effect of a large decrease in BDP on regular TCP let us consider a specific example shown in Figure 7.13(a). Here the handoff between 1600 Kbps links occurs at 15.0 seconds with the link delay changing from 300 ms to 1 ms, i.e., the link BDP decreases from 80 packets to 1 packet. There are 124 packets outstanding at the time of handoff. Many packets are lost due the very low buffering capacity of the new link (approximately 6 packets). TCP fast retransmits the first unacknowledged packet at 15.62 seconds. TCP is not able to recover all the lost packets with the Fast recovery and an RTO recovery occurs at 17.08 seconds. The new value of ssthresh (61 packets) is still significantly larger than the buffering capacity of the new link. The TCP sender increases cwnd in slow start relatively fast beyond the capacity of the new link, resulting in several packet losses and another RTO recovery is triggered at 20.01 seconds. One more RTO recovery at 25.77 seconds is necessary to recover all the lost packets yielding very poor performance. On the other hand, we can see from Figure 7.13(b) that, immediately after the handoff, the enhanced TCP stops injecting more segments to the network as cwnd is reduced down to the BDP of the new link. This effectively avoids any congestion-related losses after the handoff. Once enough cumulative ACKs have arrived, the TCP sender continues transmitting new segments in congestion avoidance, resulting in an ordinary steady-state behavior with an occasional packet drop and subsequent fast retransmit and cwnd reduction.

For the break-before-make handoffs, we can see from Figure 7.12 that there is about 35-50 % reduction in transfer time except in the case of handoffs from a 300 ms link to links with delays of 150 ms and 75 ms. RTO recovery is essential to recover the lost packets in all the scenarios. When the BDP decrease is at least a factor of eight, setting the ssthresh to half the FlightSize again results in losses in the RTO recovery phase leading to a further RTO. By contrast, the algorithm A2 enables enhanced TCP to set the cwnd and ssthresh to the BDP of the new link avoiding the losses during the RTO recovery. However, for the handoffs to a 150 and 75 ms delay links, setting the ssthresh to BDP of the new link underutilizes the path slightly resulting in about a 10 % increase in the transfer time for enhanced TCP.

Figure 7.14 shows the transfer time for the make-before-break and break-before-make handoffs between 200 Kbps links. For the make-before-break handoffs the enhanced TCP reduces the transfer time by 15 %. It is interesting to note that the upper quartile values of the transfer time for regular TCP is almost double that of the corresponding value for enhanced TCP. This high value is due to the reduction in sending rate by the occurrence of more than one RTO to recover the lost packets. The underutilization of the link causes a slight increase in the transfer time for the enhanced TCP with break-before-make handoffs to moderately high delay links (150 ms, 75ms and 37 ms). Setting the ssthresh to the new link BDP is beneficial in handoffs to small delay links (18 ms, 9 ms and 1 ms). Enhanced TCP has a comparable median and upper quartile values whereas for regular TCP the upper quartile is about 25 % larger than the median due to the occurrence of RTOs.
### Figure 7.13: Comparison of regular TCP (a) and enhanced TCP (b): make-before-break handoff at 15.0 seconds, between 1600Kbps links from 300ms to 1ms.
Figure 7.14: Handoff from a high-delay link to a low-delay link. Transfer time for 100 packets after a make-before-break (MBB) and a break-before-make (BBM) handoffs handoff between 200 Kbps links with varying new link delays and the delay of the old link is fixed at 300 ms.

7.3.2 Handoff between Same Delay, Different Bandwidth Links

Handoff from a high-bandwidth link to a low-bandwidth link

Figure 7.15 compares the time taken by regular TCP and enhanced TCP to transfer 100 packets after a make-before-break handoff for 300 ms and 9 ms delay links.

For the 300 ms delay link with regular TCP, a make-before-break handoff from a high-bandwidth to a low-bandwidth link results in congestion-related packet losses. The enhanced TCP by applying A2 sets sssthresh and cwnd to the BDP of the new link to avoid packet losses due to congestion and this reduces the transfer time (median) by about 25-35%. The Figure 7.15 also shows that there is at least a 25% reduction in the upper quartile values of the transfer time for enhanced TCP. In the case of handoff between links 9 ms delay the slight reduction in transfer time (5%) is due to algorithm A1 used in enhanced TCP to avoid spurious RTOs.

With the break-before-make handoff between 300 ms delay links, enhanced TCP behaves similarly to regular TCP as the retransmission timer will not expire during the disconnection period of 500 ms. In the case of a break-before-make handoff between 9 ms delay links, TCP will be in RTO recovery when the handoff notification arrives and this causes enhanced TCP to invoke the algorithm A3 to immediately retransmit the first unacknowledged packet. As the BDP of the new link is small, the cwnd and sssthresh values are also small and restoring the cwnd and sssthresh values using the algorithm A3 does not bring much performance improvement to enhanced TCP.
Figure 7.15: Handoff from a high-bandwidth link to a low-bandwidth link with fixed link delay of 300 ms (a) and 9 ms (b). Transfer time for 100 packets after a make-before-break (MBB) handoff and a break-before-make (BBM) handoff between same delay links with varying new link bandwidth. The old link bandwidth is fixed at 6400 Kbps.
Figure 7.16: Handoff from a low-bandwidth link to a high-bandwidth link with fixed delay of 300 ms (a) and 9 ms (b). Transfer time for 100 packets after a make-before-break and a break-before-make handoff between same delay links with varying old link bandwidth. The new link bandwidth is fixed at 6400 Kbps.
### 7.3. PERFORMANCE COMPARISON: REGULAR TCP VS. ENHANCED TCP

#### Handoff from a low-bandwidth link to a high-bandwidth link

Figure 7.19 compares the make-before-break and break-before-make handoff performances of regular and enhanced TCPs for 300 ms and 9 ms delay links. The handoff occurs from a low-bandwidth link to a high-bandwidth link. The problem that TCP faces here is its inability to efficiently utilize the high-bandwidth available after a handoff. In the case of a make-before-break handoff from a low-bandwidth to a high-bandwidth link for both link delays, there is no improvement in enhanced TCP performance as our algorithms are conservative in nature and do not attempt to blindly increase the *cwnd* in the absence of adequate information about the end-to-end path [125]. It can be seen in Figure 7.16(b) that for break-before-make handoff between 9 ms links there is 10-40% reduction in transfer time with enhanced TCP due to the algorithm A3. In the case of handoff from a 200 Kbps link to a 6400 Kbps link, the unused connection time is about 700 ms for regular TCP whereas enhanced TCP avoids this delay by applying algorithm A3 and reduces the transfer time by 40%. For higher bandwidths, the reduction in transfer time for the enhanced TCP is only 10% as the unused connection time has the much smaller value of 100 ms. With the break-before-make handoff between 300 ms delay links, enhanced TCP behaves similarly to regular TCP as the retransmission timer will not expire during the disconnection period of 500 ms.

#### 7.3.3 Handoff between Links of the Same Bandwidth Delay Product (BDP) with Different Bandwidth and Delay

In this set of experiments we have two sets of fixed bandwidth/delay values namely, 200 Kbps/300 ms and 6400 Kbps/9 ms.

#### Handoff from a high-bandwidth/low-delay link to a low-bandwidth/high-delay link

Figure 7.17(a) shows the time taken to transfer 100 packets after a make-before-break handoff from a high-bandwidth/low-delay link to a 200 Kbps/300 ms link. The problems arising in a make-before-break handoff are the occurrence of spurious RTOs and the packet losses due to the change in bandwidth. A couple of observations can be made here. Starting from the ratio of 8 (between bandwidth of the new link to that of the old link or between the delay of the old link to the new link, i.e., from 1600 Kbps/37 ms onwards) enhanced TCP yields up to 45% reduction in transfer time (median). With the regular TCP spurious RTOs occur in almost all handoff points while enhanced TCP avoids the spurious RTOs by using the algorithm A1 and reduces the packet losses by using the algorithm A2. Enhanced TCP behaves similarly to the regular TCP when the above ratio is less than 8 (i.e., handoff from 200 Kbps/300 ms to links below 1600 ms/37 ms bandwidth/delay).

We observe that there are scenarios where the algorithm A2 is not invoked though it could be beneficial to apply it. If there is a significant reduction in bandwidth due to a make-before-break handoff, segment burst in the new link can cause packet losses even if the new link capacity is not reached when the handoff completes. Figure 7.18(a) and Figure 7.18(b) present a comparison of regular TCP and enhanced TCP in one such scenario when a make-before-break handoff takes place from a 1600 Kbps/37 ms link to a 200 Kbps/300 ms link at 8.9 seconds. The BDP of both the links is 10 packets. Due to the increase in link delay after the handoff spurious RTO occurs at 9.1 seconds.
and the regular TCP retransmits the first unacknowledged segment. The late ACKs for the segments sent before the handoff, start arriving at 9.2 seconds triggering unnecessary retransmissions. As the late ACKs arrive back-to-back roughly at the line rate of the old high-bandwidth link, unnecessary retransmissions are triggered at a rate which far exceeds the capacity of the new link. Therefore the new link is congested soon and we can see from Figure 7.18(a) that many retransmissions are lost. The late ACKs for data segments (from sequence number 1109 onwards) that took the new path after the handoff start arriving at 9.6 seconds triggering more new packets to be sent to the already congested new link. It is interesting to note that the new packets sent after the handoff are also dropped. As TCP is already in RTO recovery, the duplicate acknowledgements are not taken as an indication of a new instance of congestion [13] and regular TCP needs another RTO at 10.18 seconds to recover the lost packets. The recovery takes more time with the high-delay link and the performance of regular TCP is drastically affected.

We observe in Figure 7.18(b) that the enhanced TCP avoids spurious RTO in the same scenario but incurs the loss of the last new packets sent after the handoff even though these losses are fewer than the losses of regular TCP. The late ACKs for the segments sent before the handoff start arriving back-to-back at 9.2 seconds roughly at the line rate of the old high-bandwidth link triggering 16 new packets which congest the new link. This is because the algorithm A2 is not at all invoked as the FlightSize is less than twice the BDP of the new link and so there is no consequent window reduction. A fast retransmit at 10.7 seconds is needed to recover these losses. The algorithm A5 given in 7.19 is a modification of the algorithm A2 to make it effective when there is significant reduction in bandwidth due to a handoff. As we have noted earlier for make-before-break handoffs between same BDP links that the regular TCP performance becomes poor when the bandwidth and delay change by a factor of 8 or more. So in algorithm A5, if the bandwidth of the old link is at least 8 times the bandwidth of the new link, algorithm A5 sets the cwnd and ssthresh to the BDP of the new link when the FlightSize is greater than 1.5 times the BDP of the new link. Otherwise, as in algorithm A2, the cwnd and ssthresh is set to the BDP of the new link when the FlightSize is greater than 2 times the BDP of the new link.

Figure 7.20 shows the effectiveness of algorithm A5. The window is reduced to the BDP of the new link as the ratio of decrease in bandwidth is 8 (bandwidth/delay of the new link is less than 1600 Kbps/37 ms) even though the FlightSize is smaller than twice the BDP of the link. We can see that there are no packet losses incurred with enhanced TCP when using the algorithm A5. Figure 7.17(a) shows the transfer time taken by the regular TCP and the two versions of the enhanced TCP. The slight reduction in transfer time for the enhanced TCP using the algorithm A5 is due to elimination of packet losses as shown in Figure 7.17.

Figure 7.21 shows the time taken to transfer 100 packets after a break-before-make handoff from varying bandwidth/delay links to a 200 kbps/300 ms link. We can see that immediate retransmission after a break-before-make handoff is beneficial with old link delay 75 ms or less as in these cases the retransmission timer has already expired when the handoff notification arrives. In this case enhanced TCP retransmits the first unacknowledged packet immediately and restores the ssthresh value as in algorithm A2. Enhanced TCP behaves similarly to the regular TCP when the difference in the old and new link delays is small.

When the old link bandwidth/delay is fixed at 6400 Kbps/9 ms we observe that with a make-before-break handoff enhanced TCP performs better as it is able to avoid spurious RTOs with the increase
in delay of the new link. In the break-before-make handoff, immediate retransmission improves the performance of enhanced TCP.

**Handoff from a low-bandwidth/high-delay link to a high-bandwidth/low-delay link**

In this set of experiments we fix the bandwidth and delay of the old link to 200 Kbps /300 ms and the bandwidth and delay of the new link is varied. The main problems that TCP faces with a make-before-break handoff in this scenario are packet reordering and the inability to utilize the high bandwidth available after a handoff. Packet reordering in vertical handoff scenarios is a problem in its own right and we are not addressing it in the present study. [127] describes how Quick-Start can be used to effectively utilize the high bandwidth available after a handoff.

In the case of break-before-make handoff the retransmission timer is not likely to expire during the disconnection period of 500 ms and the behaviour of enhanced TCP is similar to that of regular TCP. If the disconnection period is more than the RTO of the end-to-end path, TCP will be in RTO recovery when the handoff occurs and the algorithm A3 will be invoked resulting in improved TCP performance. Even though we have not conducted the experiments with disconnection time greater than 500 ms, the results for the specific scenario of a break-before-make handoff from EGPRS (200 Kbps/300 ms) to WLAN (5 Mbps/10 ms) reported in our paper [31] show the effectiveness of enhanced TCP when the disconnection period is 4 seconds which is greater than the RTO of the end-to-end path that includes the GPRS access link. On this basis we expect that our algorithms will be effective for long disconnection periods as well.

Figure 7.22 compares the make-before-break and break-before-make handoff performances of regular and enhanced TCPs when handoff occurs from a low-bandwidth/high-delay link to 6400 Kbps/9 ms link. We can see in Figure 7.22(a) that the regular TCP is unable to utilize the high bandwidth available after a make-before-break handoff. We can also see that when a high bandwidth link is available after a handoff, just setting the cwnd to the new link BDP underutilizes the path. This accounts for the slight increase in transfer time in the case of enhanced TCP. In break-before-make handoff with a disconnection period of 500 ms, TCP is in timeout recovery when the handoff notification arrives when the delay of the old link is less than 150 ms and enhanced TCP yields better performance by using algorithm A2. Figure 7.22(b) shows 20-45 % reduction in transfer time with enhanced TCP when the delay of the old link is smaller than 150 ms. For the handoff from 300 ms delay link and 150 ms delay link, the retransmission timer may not expire during the disconnection period and enhanced TCP behaves similarly to regular TCP as the algorithm A3 will not be invoked.

**7.4 Comparison of our algorithms with the algorithms proposed in the literature**

The algorithms given in the literature are summarized in the Table 8.1. A summary of our proposed algorithms is given in Table 7.1. Here we provide a brief comparison of our proposals and the earlier ones.
<table>
<thead>
<tr>
<th>Event</th>
<th>Problem</th>
<th>Enhancements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handoff from low delay to high delay</td>
<td>Spurious RTOs</td>
<td>Increase RTO value based on the bandwidth and delay of the new link</td>
</tr>
<tr>
<td>Handoff from high delay to low delay</td>
<td>Packet Reordering</td>
<td>Future Study</td>
</tr>
<tr>
<td></td>
<td>Slow convergence of RTO</td>
<td>Initialize RTT Variables and update RTO</td>
</tr>
<tr>
<td>Handoff from high BDP to low BDP</td>
<td>Packet Losses</td>
<td>Set the cwnd and ssthresh to the BDP of the new link in case of immediate congestion</td>
</tr>
<tr>
<td>Handoff from low BDP to high BDP</td>
<td>Inability to catch up with high bandwidth</td>
<td>Future Study</td>
</tr>
<tr>
<td>RTOs during disconnection</td>
<td>Unused connection time</td>
<td>Retransmit immediately if TCP is in RTO recovery and restore the old ssthresh and cwnd</td>
</tr>
</tbody>
</table>

- **Spurious RTOs**: Our algorithm A1 determine the minrto based on the bandwidth and delay of the old and the new access links. This helps TCP to set a minimum RTO value so that spurious RTOs can be avoided and at the same time it helps TCP to adapt to the correct RTO value based on the end-to-end path delay. The Eifel algorithm proposed to avoid spurious RTOs needs timestamps for its implementation. In another proposal [59] to avoid spurious RTOs, it is difficult to determine the duration for which the ACKs/packets are to be sent through the new/old link.

- **Congestion-related packet losses**: Our algorithm A2 sets both ssthresh and cwnd to the BDP of the new access link if there is an imminent congestion. If the access link is the bottleneck
link in the end-to-end path, setting the $ssthresh$ and $cwnd$ to the BDP of the link gives a lower boundary to the $cwnd$ and $ssthresh$ with which TCP can operate safely. In the algorithms which suggest [99, 144] measuring the bandwidth and RTT of the end-to-end path to set the $cwnd$, the measurements will not be accurate as ACKs and packets may flow through both the old and the new access links in a make-before-break handoff.

- The unused connection time: The unused connection time and the repeated reduction in $ssthresh$ are the major problems arising due to a break-before-make handoff. Our proposed Algorithm A3 takes into account of both the above problems while the solution proposed in [130, 131] addresses only the problem of unused connection time.

### 7.5 Summary

In this chapter we described solutions to mitigate the problems of TCP in the presence of vertical handoff. We analyzed the problems of TCP in handoff by identifying the various scenarios that lead to these problems and devised algorithms specific to these scenarios to overcome the problems. For example, spurious RTOs occur when there is a significant increase in delay after a handoff. From our experiments we found out that the occurrence of spurious RTOs is dependent on the TCP state as well. Even with a two fold increase in delay due to a handoff, if TCP is in fast recovery, spurious RTOs can occur due to queueing delay. We proposed solutions to the problems of spurious RTOs, congestion-related packet losses, prolonged disconnections and slow convergence to the new RTO which arise in a vertical handoff.

Our proposed enhancements are to the TCP Sack algorithm at the TCP sender and they are invoked when a cross-layer notification arrives from the MN to the TCP sender at the CN. This cross-layer notification includes information about the occurrence of a handoff and an estimate of the bandwidth and delay of the old and the new access links. Our algorithms use these values to devise a bound for the TCP parameters such as $minrto$, $ssthresh$ and $cwnd$. Our algorithms are conservative in the sense that they are not counter-productive in any situation. Our algorithms are relatively simple and are easy to implement. In the absence of cross-layer information, TCP behaviour is virtually unaltered.

We evaluated the proposed algorithms in various scenarios to show that their effectiveness. Our algorithms can yield up to 40% reduction in the transfer time immediately after a handoff. Our results show that TCP has severe performance problems even in handoff between same-BDP links and our algorithms are effective in this scenario.

Table 7.1 provides a summary of the problems of TCP with vertical handoff and the algorithms we proposed made to mitigate these problems. We also presented a comparison of our proposed algorithms with the earlier work in this area.
Figure 7.17: Handoff between Same BDP links. Transfer time (a) and #Dropped packets (b) after a make-before-break handoff between the same BDP links with varying bandwidth and delay of the old link and bandwidth and delay of the new link fixed at 200 kbps/300ms
Figure 7.18: Comparison of regular TCP (a) and enhanced TCP with Algorithm A2 (b): make-before-break handoff at 8.9 seconds from a 1600Kbps/37ms link to a 200Kbps/300ms link.
When handoff notification arrives
If ((TCP not in RTO recovery)
  If ( \(BW_{oldlink} \geq 8 \times BW_{newlink}\))
    If (\(FlightSize > 1.5 \times BDP_{newlink}\))
      \(cwnd\_reduction = 1\)
    Else if (\(FlightSize > 2 \times BDP_{newlink}\))
      \(cwnd\_reduction = 1\)
  If (\(cwnd\_reduction == 1\))
    \(cwnd = \max(2, BDP_{newlink})\)
    \(ssthresh = cwnd\)
    \(cwnd\_reduced = 1\)

Figure 7.19: A5: Modified algorithm to reduce congestion-related packet losses.

Figure 7.20: Enhanced TCP with Algorithm A5: make-before-break handoff at 8.9s from a 1600 kbps/37ms link to a 200 kbps/300ms link
7.5. SUMMARY

Figure 7.21: Transfer time for 100 packets after break-before-make handoff for the same-BDP links with varying bandwidth and delay of the old link and the newlink fixed at 200 kbps/300ms. The disconnection period is 500 ms
Figure 7.22: Transfer time for 100 packets after make-before-break (MBB) handoff (a) and break-before-make (BBM) handoff (b) for the same BDP links with varying bandwidth and delay of the old link and the bandwidth and delay of the new link are 6400 kbps/9 ms. The period of disconnection for break-before-make handoff is 500 ms.
Chapter 8

Conclusions and Future Work

In this thesis, we have studied the effect of vertical handoff on TCP performance. We have analyzed the problems of TCP due to a vertical handoff in various handoff scenarios. On the basis of this analysis, we have proposed enhancements to the TCP sender algorithm to improve TCP performance in vertical handoff. Our algorithms make use of the cross-layer notifications to the TCP sender about the changes in the access link characteristics. The changes in the end-to-end path characteristics due to a vertical handoff are taken as the changes arising from delay, bandwidth and connectivity of the access links involved in the handoff and we have developed algorithms that enhance the TCP sender algorithm to adapt to the vertical handoff efficiently. We have shown that our algorithms effectively address the problems arising from spurious RTOs, packet losses, prolonged disconnection and slow convergence to the new RTO value due to a handoff. The proposed algorithms have been evaluated in both make-before-break and break-before-make handoffs in ns-2 simulator for access links with a wide range of bandwidth and propagation delay of interest in real-world access networks. Our algorithms are shown to yield significant improvement in TCP performance for many vertical handoff scenarios. Our algorithms do not adversely affect the behaviour of TCP in the absence of cross-layer notifications.

The study on the effect of vertical handoff on TCP and the algorithms proposed in this thesis are based on our experiments with a single TCP flow. While the study is interesting in its own right, we think that it is important to examine how the proposed algorithms behave in the presence of multiple TCP flows. Therefore, as further study we plan to conduct experiments to see how the proposed algorithms can be adapted for multiple TCP flows in the presence of a vertical handoff.

Moving to a higher bandwidth environment after a vertical handoff is challenging to TCP as TCP needs to be more aggressive to fully utilize the available bandwidth but this cannot be done safely based on the information about the access link characteristics alone. A TCP sender should probe the new network path, but often it takes relatively long before TCP can adapt to the new path. In the future, we intend to study how TCP can combine the notifications regarding the access link characteristics with the information gathered by probing the new network path and thereby try to converge quickly and safely to the new end-to-end RTT and available network capacity. In addition, it will be interesting to conduct experiments in a real access network environment such as EGPRS, UMTS, WLAN and WiMAX to evaluate the proposed algorithms. Another direction for further work is to develop an analytical model for TCP behaviour in a vertical handoff.
Bibliography


[42] ETSI. Digital cellular telecommunications system (phase 2+); General Packet Radio Service (GPRS); mobile station (MS)-BSS interface; radio link control/medium access control (RLC/MAC) protocol (GSM04.60), July 1999.
[43] ETSI. Digital cellular telecommunications system (phase 2+); General Packet Radio Service (GPRS): mobile station-serving GPRS support node (MS-SGSN) logical link control (LLC) layer specification (GSM04.64), October 1999.


