END-TO-END-ACKNOWLEDGED INDIRECT TCP FOR WIRELESS INTERNETWORKED ENVIRONMENTS

by

VICTOR CHIM

B.A.Sc., The University of British Columbia, 1995

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF APPLIED SCIENCE in THE FACULTY OF GRADUATE STUDIES DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

We accept this thesis as conforming to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

December 2000

© Victor Chim, 2000
In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Electrical and Computer Engineering

The University of British Columbia
Vancouver, Canada

Date Dec. 22/2000.
Abstract

TCP is a transport protocol developed for reliable end-to-end communications over the Internet, and is used by many popular Internet applications, including electronic mail, Telnet, FTP and WWW. TCP is designed to perform well over fixed internetworked environments, where packet losses occur primarily due to network congestion. It attempts to alleviate the congestion problem by initiating its congestion control mechanisms when a packet is believed to have lost. Perceived demand for mobile computing means that wireless links and mobile hosts will likely represent a large part of the next generation Internet. Wireless transmission errors and host mobility can introduce significant packet losses in wireless networks. TCP performance degrades on such wireless internetworked environments because the same congestion control mechanisms are invoked in response to these non-congestion related losses.

In this thesis, an alternative transport protocol, known as end-to-end-acknowledged indirect TCP (EI-TCP), is proposed for the wireless internetworked environment. It is based on the indirect TCP (I-TCP) concept, but end-to-end semantics of TCP acknowledgments are maintained. Using computer simulations on a wireless internetworking model, throughput performance of EI-TCP is evaluated and compared against I-TCP and end-to-end TCP for bulk data transfer. Results indicate that by more effective error recovery over the wireless link, EI-TCP and I-TCP perform significantly better than end-to-end TCP. Also, EI-TCP performs slightly better than I-TCP with a more steady data flow as a result of end-to-end flow control. With link layer retransmissions over the wireless link, EI-TCP and I-TCP can boost end-to-end throughput by avoiding competing retransmissions between transport and link layers.
# Table of Contents

Abstract .......................................................... ii

List of Tables ......................................................... vi

List of Figures ......................................................... vii

Acknowledgments ..................................................... ix

Chapter 1  Introduction ........................................... 1
  1.1 Wireless versus Wired ....................................... 3
  1.2 TCP and Wireless Internetworking .......................... 4
  1.3 Related Work ................................................ 5
    1.3.1 TCP Modifications .................................... 5
    1.3.2 Link Layer Recovery ................................. 6
    1.3.3 Split Connection ..................................... 7
    1.3.4 Fast Retransmit Approach ............................ 8
  1.4 Motivation and Scope ....................................... 9
  1.5 Thesis Outline ............................................. 10

Chapter 2  Wireless Internetworking ............................. 11
  2.1 Internetworking Concept .................................... 11
  2.2 TCP/IP Protocol Suite ..................................... 12
  2.3 Overview of TCP ........................................... 14
    2.3.1 Protocol Data Unit ................................... 15
    2.3.2 Connection Establishment and Termination ........ 17
    2.3.3 Retransmission Timer and Round Trip Timing ...... 19
    2.3.4 Karn’s Algorithm and Timer Backoff ................ 20
### Chapter 3: Indirect Transport Protocols

3.1 Indirect Protocol Model .................................................. 28

3.2 Indirect TCP (I-TCP) ......................................................... 29

3.3 End-to-End-Acknowledged Indirect TCP (EI-TCP) ................. 31

3.4 Wireless Transport Protocols ............................................. 33

3.4.1 Wireless TCP without Link Layer Retransmissions ............ 33

3.4.2 Wireless TCP with Link Layer Retransmissions ............... 34

### Chapter 4: Design of Simulation Models

4.1 Overview of OPNET ....................................................... 35

4.1.1 Network Domain ...................................................... 36

4.1.2 Node Domain ......................................................... 36

4.1.3 Process Domain ...................................................... 37

4.2 Model Descriptions ....................................................... 38

4.2.1 Network Models ...................................................... 38

4.2.2 Node and Process Models .......................................... 39

4.2.3 Wireless Link Models ............................................... 44

### Chapter 5: Simulation Results and Discussions

5.1 Simulations Overview ................................................... 48

5.2 Effect of Protocol Architecture ......................................... 49
List of Tables

Table 2.1 TCP header flags.................................................................17
## List of Figures

| Figure 2.1 | Two networks connected by a router ................................................................. | 12 |
| Figure 2.2 | The four layers of the TCP/IP protocol suite .................................................. | 13 |
| Figure 2.3 | Protocol layering at end hosts and routers ..................................................... | 15 |
| Figure 2.4 | TCP header ............................................................................................................. | 16 |
| Figure 2.5 | TCP connection establishment .............................................................................. | 18 |
| Figure 2.6 | TCP connection termination .................................................................................. | 19 |
| Figure 2.7 | Combined slow start and congestion avoidance .................................................. | 22 |
| Figure 2.8 | Cellular network architecture ............................................................................ | 24 |
| Figure 2.9 | 7-cell frequency reuse pattern ......................................................................... | 25 |
| Figure 2.10 | Wireless internetworking architecture ............................................................... | 27 |
| Figure 2.11 | Data path between MH and FH ............................................................................ | 27 |
| Figure 3.1 | Time line for data transmission and acknowledgment with I-TCP ..................... | 31 |
| Figure 3.2 | Time line for data transmission and acknowledgment with EI-TCP .................... | 32 |
| Figure 4.1 | Basic OPNET network model .............................................................................. | 38 |
| Figure 4.2 | FH and MH node model structure ..................................................................... | 40 |
| Figure 4.3 | MSR node model for TCP .................................................................................... | 42 |
| Figure 4.4 | MSR node model for I-TCP and EI-TCP ............................................................... | 43 |
| Figure 4.5 | Two-state Markov model for wireless link ......................................................... | 45 |
| Figure 5.1 | Transport layer throughput comparison for TCP, I-TCP and EI-TCP, with unmodified TCP Reno over the wireless transport connection of I-TCP and EI-TCP ................................................................. | 50 |
| Figure 5.2 | Transport layer throughput comparison for TCP, I-TCP and EI-TCP, with an optimized fast retransmit algorithm over the wireless transport connection of I-TCP and EI-TCP ................................................................. | 53 |
| Figure 5.3 | Transport layer throughput comparison with TCP under different wireless loss distributions | 54 |
| Figure 5.4 | Transport layer throughput comparison with I-TCP under different wireless loss distributions | 56 |
| Figure 5.5 | Transport layer throughput comparison with EI-TCP under different wireless loss distributions | 57 |
| Figure 5.6 | Transport layer throughput comparison for TCP, I-TCP and EI-TCP, with link layer retransmissions over the wireless link of $P_{BG} = 1.0$ | 58 |
| Figure 5.7 | Transport layer throughput comparison for TCP, I-TCP and EI-TCP, with link layer retransmissions over the wireless link of $P_{BG} = 0.7$ | 60 |
| Figure 5.8 | Transport layer throughput comparison for TCP, I-TCP and EI-TCP, with link layer retransmissions over the wireless link of $P_{BG} = 0.4$ | 63 |
Acknowledgments

I am grateful to my research supervisor, Prof. Victor Leung, for his guidance and support throughout the course of my research work. I would also like to thank for the financial support to this work from Motorola Canada Ltd. and NSERC IOR Grant 180452, and the support of OPNET Technologies, Inc. (formerly MIL 3 Inc.) by providing licences to the OPNET simulator. Finally, I would like to express my appreciation for the love and encouragement from my family and friends.
Chapter 1  Introduction

Nowadays, most of us are accustomed to the convenience provided by mobile communications. Over the last decade, mobile telephony was one of the fastest growing communication services. The first generation cellular phone system in North America, known as the Advanced Mobile Phone System (AMPS), began commercial service in the early 1980s. Since then, the use of cellular phones has grown tremendously all over the world. With recent developments in the computing field, we perceive similar prospects for mobile data services.

In recent years, the nature of computing is greatly influenced by two technological advances. The first is the increase in portability of computing devices. Improved technologies in electronic circuits and components allow portable computers to be built with substantial reduction in both size and cost. At the same time, many portable computers now possess computing power comparable to desktop machines. As a result, portable computing devices, including laptop/notebook computers, palmtops and personal digital assistants (PDAs), have become increasingly popular. They now represent the fastest growing segment of the computer market. Another advance is the widespread use of the Internet. Since its beginning in the late 1960s, the Internet has grown from a four-node research network into a network connecting millions of computers that share all kinds of practical information. In addition to the traditional electronic mail, news, Telnet and FTP services, applications like archie, Gopher and World Wide Web (WWW) allow more efficient information exchange across the Internet. In particular, the success of the WWW, which now accounts for about 75% of Internet traffic, contributes significantly to the explosive growth of the Internet in the last five years, making the Net accessible to the general public rather than just technical people within computing-
related fields.

The two developments provide the driving force for a new discipline of computing known as *mobile computing*. It represents computing on the move, with access to information and services through the Internet, anywhere and anytime. The demand for mobile computing has spurred the development of wireless data networks (WDNs), especially those supporting wide-area mobility. In addition to packet radio data networks such as ARDIS [1, 2] and RAM/Mobitex [3], the CDPD system [4] allows packet data to be carried over the existing AMPS cellular phone system. Also, data services are incorporated into the second generation digital cellular systems such as GSM [5, 6], as well as the North American TDMA [7] and CDMA [8, 9] systems. To support the mobile computing service, a portable computer will have access to a WDN, which in turn is connected to the Internet. We use the term *wireless internetworking* to refer to the connection of mobile computers to the Internet through wireless networks. As mobile computing continues to grow, wireless links and mobile hosts will eventually represent a significant part of the Internet.

*Wireless internetworking* poses many challenges to the existing Internet communication protocols, which are designed to operate over fixed internetworked environments. The challenges are the results of the differences in characteristics of wireless networks and their wireline counterparts, and the failure of current protocols to cope with such differences.

This chapter introduces the issues arising from wireless internetworking and how they affect the performance of TCP, the reliable transport protocol in the TCP/IP protocol suite for data communications over the Internet, in wireless internetworked environments. This chapter
also presents related work in this area. The drawbacks of the proposed solutions and the motivation for the work presented in this thesis are described. A brief outline of the remaining chapters is given in the last section.

1.1 Wireless versus Wired

Wireless networking technologies enable the convenience of tetherless network computing, but at the same time they introduce issues not present in modern wired networking environments. Major performance-related issues are discussed below.

Available bandwidth. Compared with wireline network connectivity, the bandwidth available in a wireless network connection is generally considered a scarce resource. Even at the low end of wireline networking, today’s analog modems are able to offer connection line speed of up to 56 Kbps. In contrast, the transmission of radio data in wide area cellular networks is limited to about 19.2 Kbps in current service offerings, and actual throughput is approximately half of that when network overheads are taken into account [10]. Due to limited wireless bandwidth, the wireless link becomes the throughput bottleneck in a wireless internetworked environment. Efficient use of the wireless link is crucial to achieve good throughput.

Reliability. Data transmission over the wireless medium is much less reliable than that over a wireline transmission link. Transmission of radio waves in free space is subject to propagation effects such as multipath fading, which can cause the transmission quality of a wireless link to become poor from time to time, resulting in greatly increased probability of transmission errors.
Mobility. Mobility support of networked machines is a major advantage of wireless networking. However, mobility can contribute to variations in wireless link reliability, as fading effects are location dependent. In a wide-area cellular wireless network, the network must handle the changes in routing when a mobile machine moves from cell to cell, a process known as handoff. Temporary disruptions in network connectivity may occur as a result.

1.2 TCP and Wireless Internetworking

The Internet does not assume reliability in the underlying networks. To provide reliable data delivery service to network applications, the Transmission Control Protocol (TCP) is developed. This section introduces the problems with TCP in wireless internetworked environments. More detailed description of TCP is given in Chapter 2.

To ensure correct data reception at the receiver, the TCP sender relies on positive acknowledgments from the receiver for data correctly received. If an acknowledgment is not received within a reasonable period of time, the sender assumes the corresponding data packet fails to reach the receiver correctly, and the packet is retransmitted.

Apart from providing reliable data delivery, modern TCP implementations are required to include algorithms for controlling network congestion [11]. The TCP sender uses a packet loss as an indication of network congestion, and reduces the rate of data transmitted over the network to alleviate the congestion problem. The assumption that a packet loss is caused by network congestion is justified in traditional fixed internetworked environments since modern wired networks indeed have negligible transmission errors.

As discussed in the previous section, data transmission over a wireless network is
subject to losses caused by transmission errors and mobility of networked machines. The presence of such non-congestion-related losses violates the assumption made by TCP that packet losses are caused by network congestion. The congestion control mechanisms of TCP, while effective in controlling congestion over fixed internetworked environments, become performance degrading actions in response to non-congestion-related losses in a wireless internetworked environment.

1.3 Related Work

This section summarizes the related work on improving TCP performance over wireless networks. In general, the proposed solutions achieve performance improvements using one of four approaches:

1. Modify TCP to take care of wireless losses.

2. Use link layer recovery mechanisms over the lossy wireless link.

3. Split the end-to-end TCP connection into two subconnections, one over each of the wired and wireless portions of the end-to-end data path.

4. Utilize the TCP fast retransmit mechanism to boost the recovery phase after handoffs.

The following subsections discuss these approaches in details.

1.3.1 TCP Modifications

As discussed in the previous section, TCP does not handle wireless losses
appropriately, resulting in slow recovery of these losses. The goal of modifying TCP is to change its behaviour regarding wireless losses. One attempt [12] is to allow the TCP sender to distinguish wireless losses from congestion-related ones by using a new acknowledgment strategy. Congestion control is applied only to losses identified as caused by congestion. Other schemes make TCP recover faster from wireless losses by using strategies such as negative acknowledgments (NAKs) [13, 14] and selective acknowledgments (SACKs) [15].

Despite performance improvements, TCP modifications are not very desirable. Unless changes are standardized throughout the Internet community, their implementation will be limited, and they may cause interoperability problems. However, standardization procedures are lengthy, and even after changes are standardized, adaptation by the millions of Internet hosts is expected to be a gradual process. As a result, the benefits cannot be realized in the near term.

1.3.2 Link Layer Recovery

The goal of link layer recovery mechanisms is to attempt recovery of wireless losses in the link layer over the wireless link, thereby hiding these losses from the TCP sender. Techniques employed may include forward error correction (FEC) and retransmissions by automatic repeat request (ARQ). Several link layer protocols have been proposed specifically for wireless networks [16, 17, 18, 19, 20].

FEC allows the receiver to detect and correct a packet with errors by including redundant information. However, the redundant information results in extra bandwidth utilization for transmitting the same amount of useful data. During periods of good wireless channel condition, FEC results in a waste of bandwidth. Moreover, there is a certain limit in
the recovery ability of FEC. It cannot recover a packet that is too badly corrupted or not received at all. Ultimate reliability has to be provided by ARQ or TCP.

With ARQ, transmitted packets are buffered, and they are retransmitted if not received correctly at the destination. This results in more efficient bandwidth utilization compared with FEC, since extra bandwidth is consumed only when a packet is retransmitted. However, retransmission causes increased delay and delay variation as seen by the upper layers. Also, there is concern about interaction of the independent retransmission mechanisms at the link and TCP layers. The study in [21] indicates that retransmissions at the two layers may cause conflicts with each other, whereby TCP retransmits packets already retransmitted by the link layer, resulting in degraded performance. An additional concern is associated with link layer retransmission schemes that do not guarantee in-order packet delivery. Duplicate TCP acknowledgments are generated, which can trigger the fast retransmit algorithm of TCP at the sender, causing redundant retransmissions and subsequent window size reduction.

To combat the possible conflict between TCP and link layer retransmissions, a TCP-aware link layer protocol called the snoop protocol is proposed [22]. The protocol works by examining and buffering TCP packets as they pass through the snooping node, and perform local retransmissions of lost TCP packets from the snooping node. Redundant TCP retransmissions, which have been retransmitted locally, are filtered out at the snooping node so they would not consume the precious wireless bandwidth.

1.3.3 Split Connection

The split connection approach attempts to improve performance by splitting the end-to-end TCP connection into two subconnections, one over each of the wired and wireless
portions of the end-to-end data path. This follows the philosophy that an indirect model of client-server interaction be used when communication involves two drastically different kinds of media, in this case, wired and wireless [23]. Indirect TCP (I-TCP) [24, 25] is an alternative transport protocol based on this approach. Other schemes with similar ideas have also been proposed [26, 27, 28].

Under the split connection approach, the wired subconnection behaves like a regular TCP connection, so the protocol remains compatible with the TCP implementations on existing hosts. Therefore, although the transport layer is modified, no changes to existing hosts are required. The wireless subconnection can use a modified version of TCP [25, 27] or even an entirely different transport protocol [29, 30] to accommodate characteristics of the wireless medium.

The major drawback of the split connection approach is the loss of end-to-end semantics of TCP acknowledgments. Packets are separately acknowledged in the wired and wireless subconnections. An acknowledgment received by the sender cannot guarantee correct reception of the corresponding data packet by the ultimate receiver.

1.3.4 Fast Retransmit Approach

Retransmissions in classical TCP [31] are entirely timer-based. The TCP sender retransmits a packet only if an acknowledgment is not received before retransmission timeout. Modern TCP implementations introduce an additional mechanism, known as the fast retransmit algorithm [32], for triggering packet retransmissions. Under the algorithm, the TCP sender retransmits a packet, without waiting for the retransmission timeout, when it receives three duplicate acknowledgments from the receiver, which means the receiver is still
expecting the next in-sequence packet after three additional packets are received.

The scheme proposed in [33] attempts to boost the recovery phase after handoffs by utilizing the fast retransmit algorithm. After the handoff procedure is completed, the receiver sends three duplicate acknowledgments to the sender, so the first missing packet can be retransmitted immediately. The proposal, however, does not address the error characteristics of the wireless link. Moreover, the scheme was evaluated based on the TCP Tahoe implementation. Its effectiveness is not confirmed with the current de facto standard implementation, TCP Reno, which behaves differently in response to duplicate acknowledgments.

1.4 Motivation and Scope

Many popular Internet applications, including electronic mail, Telnet, FTP and WWW, depend on TCP to provide reliable end-to-end data transfers. The performance of TCP is crucial to the performance seen by these applications. Poor TCP performance in wireless internetworked environments means Internet applications will also perform poorly, a barrier to the ultimate success of mobile computing. As a result, improving TCP performance in wireless internetworked environments is of great research interest. While there are several proposed solutions as discussed in the previous section, they have drawbacks that need to be addressed.

This thesis is an extension to current work on addressing the wireless-related performance impact on TCP. In particular, it addresses the drawbacks of the split connection and link layer recovery approaches with an alternative transport protocol known as end-to-end-acknowledged indirect TCP (EI-TCP). This thesis also gives performance evaluation of
EI-TCP compared against I-TCP and end-to-end TCP by computer simulations on a wireless internetworking model.

1.5 Thesis Outline

In Chapter 2, an overview of the wireless internetworked environment is given. Chapter 3 introduces the concept of indirect transport protocols, and describes the I-TCP and EI-TCP protocols, which are based on the indirect protocol model. Chapter 4 introduces the OPNET simulation program, which is used to build the simulation models and execute the simulations. Models developed using OPNET are also described. Chapter 5 presents and discusses the simulation results. Chapter 6 concludes the thesis with a summary of the findings and provides directions for future research.
Chapter 2  Wireless Internetworking

This chapter presents an overview of the wireless internetworked environment. The concept of internetworking is first introduced. The architecture of the Internet and the main ideas of the TCP/IP protocol suite for internetworking are then given, with an overview of TCP in particular. A wireless network architecture based on the cellular concept is also described in this chapter. The last section presents the system model for wireless internetworking.

2.1 Internetworking Concept

Computer networks are developed to facilitate the sharing of data and resources among computers and to satisfy the communication needs of users. Governments, corporations, universities and other organizations have built networks varying in sizes, areas of geographical coverage and networking technologies. However, in each of these independent networks, the scope is limited to a single organization. To further extend the benefits of computer networking, the ability for computers in different organizations to communicate is highly desirable. Building a universal network from a single networking technology is impractical, because no single technology suffices for all uses. A more viable alternative is to link existing individual networks together. This approach is known as internetworking. By internetworking, computers on different networks can talk to each other.

Since the individual networks may employ different networking technologies, they cannot be directly connected together. An intermediate device is used to connect heterogeneous networks. This is known as a router or gateway. Figure 2.1 illustrates how a router is used to connect two networks.
The router in Figure 2.1 must be able to talk with machines on both networks. Its function is to relay packets from one network to another. A packet from a machine on Net 1 destined for a machine on Net 2 is captured by the router. The router then sends the packet over Net 2 to the destination.

More networks can be added in a similar fashion to make a larger internetwork. The Internet is formed from the internetworking of many individual networks worldwide. As the internetwork topology gets more complex, the routers must do more than simple packet relaying function. They have to cooperate with each other and perform routing decisions to determine the path for a packet from source to destination, which may go through several physical networks and routers.

2.2 TCP/IP Protocol Suite

To accommodate the different technologies employed by the underlying networks, a set of standard protocols is developed for communications over the Internet. These protocols are collectively known as the TCP/IP protocol suite. The protocols are organized in layers, where each layer is responsible for a particular aspect of the communication problem. Each layer
delivers its services to the layer above it, and communicates with its peer at the same layer using one or more protocols of that layer. The TCP/IP protocol suite is structured in four layers, as shown in Figure 2.2.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Telnet, FTP, SMTP, HTTP, etc.</td>
</tr>
<tr>
<td>Transport</td>
<td>TCP, UDP</td>
</tr>
<tr>
<td>Network</td>
<td>IP, ICMP, IGMP</td>
</tr>
<tr>
<td>Link</td>
<td>Ethernet, FDDI, etc.</td>
</tr>
</tbody>
</table>

Figure 2.2 The four layers of the TCP/IP protocol suite

The application layer is the top layer in the TCP/IP protocol suite. It handles the details of specific network applications. Common application protocols include Telnet for remote terminal access, FTP for file transfer, SMTP for electronic mail, and HTTP for accessing WWW documents.

The next layer is the transport layer. It is responsible for the end-to-end flow of data between end hosts. The application layer relies on the services of the transport layer to deliver data to, and receive data from, the remote application. TCP and UDP are two transport protocols in the TCP/IP protocol suite. TCP provides a reliable flow of data between the end hosts, while UDP promises only a best-effort datagram delivery service.
Chapter 2 Wireless Internetworking

The layer below the transport layer is the network layer, sometimes known as the internet layer. It is responsible for the movement of packets around the Internet. The main protocol in this layer is IP, which handles packet routing from source to destination across the network. Other protocols include ICMP for communicating error and control messages, and IGMP for IP multicasting.

The last layer is the link layer, also called the network interface layer. It handles communication over a specific physical network, such as an Ethernet.

The application and transport layer protocols operate end-to-end. Therefore, the two layers exist only at the end hosts. The network layer IP is a hop-by-hop protocol, and is present at every IP router. Each physical network has its own link layer. Figure 2.3 illustrates the protocol operations at the four layers for an FTP session.

With the TCP/IP protocol suite, the underlying architecture and communication technologies of the individual physical networks are hidden below the network layer. From the user's point of view, the Internet is a single virtual packet-switched network connected by IP routers.

2.3 Overview of TCP

TCP is one of the most important protocols within the TCP/IP protocol suite. An overview of the protocol is presented in this section. The IETF standard for TCP is described in [31] and is amended by [34]. However, different implementations exist, some with optional features such as SACK [15]. The current de facto standard is called TCP Reno, as described in [35], which is first implemented in the 4.3BSD Reno release in 1990. It includes all the
algorithms described in [32]. The description in this section is based on TCP Reno.

TCP is an end-to-end, connection-oriented transport protocol. It provides an end-to-end full duplex reliable byte stream service to the application layer. The following subsections describe different aspects of the protocol functionality.

### 2.3.1 Protocol Data Unit

A TCP protocol data unit (PDU) or packet is termed a TCP *segment*, since TCP views user data as a continuous stream that it segments into pieces of convenient sizes for transmission. A segment contains a TCP header and zero or more bytes in the user data stream.
Figure 2.4 shows the format of the TCP header.

<table>
<thead>
<tr>
<th>0</th>
<th>15</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-bit source port</td>
<td>16-bit destination port</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32-bit sequence number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32-bit acknowledgment number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-bit header length</td>
<td>Reserved (6 bits)</td>
<td>6-bit flags</td>
<td>16-bit window size</td>
</tr>
<tr>
<td>16-bit checksum</td>
<td>16-bit urgent pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options (if any)</td>
<td>Padding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data (if any)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The TCP header carries identification and control information. The source port and destination port fields are TCP port numbers, which identify the applications at the end points of the TCP connection. The sequence number field identifies the position in the sender's byte stream that the first byte of data in this segment represents. The acknowledgment number field identifies the next byte expected from the other end of the connection. The header length field specifies the length of the header in 32-bit words. This is necessary because the options field has variable length. Padding bits are applied if necessary to align the header to a 32-bit boundary. A normal header without options is 20 bytes in length. The flags field contains six flags, in the order URG, ACK, PSH, RST, SYN and FIN. Their meanings are listed in Table 2.1.
Table 2.1  TCP header flags

<table>
<thead>
<tr>
<th>Flag</th>
<th>Meaning if set to 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>URG</td>
<td>Urgent pointer field is valid</td>
</tr>
<tr>
<td>ACK</td>
<td>Acknowledgment number is valid</td>
</tr>
<tr>
<td>PSH</td>
<td>Segment is pushed (To be passed to application as soon as possible)</td>
</tr>
<tr>
<td>RST</td>
<td>Reset the connection</td>
</tr>
<tr>
<td>SYN</td>
<td>Synchronize sequence numbers</td>
</tr>
<tr>
<td>FIN</td>
<td>Sender finished sending data</td>
</tr>
</tbody>
</table>

The window size field is used by one end of the connection to advertise the amount of data that it is willing to receive from the other end, for flow control purpose. The checksum field provides integrity check to both the header and data. The urgent pointer field is used together with the URG flag to identify urgent data. When the URG flag is set, the current data is marked urgent, and the urgent pointer contains a positive offset from the sequence number to point to the last byte of urgent data.

2.3.2 Connection Establishment and Termination

A TCP connection has to be established before data can be exchanged. To establish a connection, TCP uses a three-way handshake. The procedure is illustrated in Figure 2.5.

Figure 2.5 shows how Host A and Host B establish a TCP connection. Host A requests a connection by sending a segment with the SYN flag set (SYN segment) and an initial sequence number $x$. Upon receiving the SYN segment, Host B responds with its own SYN segment and its own initial sequence number $y$. The segment also acknowledges the SYN segment from Host A, with the ACK flag set and an acknowledgment number of $x+1$. Finally, Host A acknowledges the SYN+ACK segment from Host B with an ACK segment specifying
acknowledgment number $y+1$, completing the three-way handshake.

During connection establishment, each side usually announces its maximum segment size (MSS) to the other. The MSS is the largest TCP segment that a host expects to receive. MSS announcements are accomplished with an MSS option in the SYN segment.

When the exchange of data is completed, a TCP connection has to be closed. TCP connections are closed by a modified three-way handshake, which takes four segments to complete. This is illustrated in Figure 2.6.

In Figure 2.6, Host A sends a FIN segment after it has sent the last byte of data. When Host B receives this FIN segment, it acknowledges the segment with an ACK segment. Meanwhile, Host B can still send data to Host A. When Host B is done, it sends its FIN...
segment to Host A, which then acknowledges the segment with another ACK.

### 2.3.3 Retransmission Timer and Round Trip Timing

To provide reliable service, TCP uses the mechanism of positive acknowledgment and retransmission. All data sent has to be acknowledged by the receiver. To reduce the number of ACK segments, TCP acknowledgments are cumulative and are piggybacked in data segments in the reverse data flow. The acknowledgment number specifies the next byte expected from the sender, and acknowledges receipt of all previous data in the stream. When the sender is expecting an acknowledgment, it sets a retransmission timer. When the timer expires before an acknowledgment is received, retransmission occurs, starting with the first byte of unacknowledged data.
The retransmission timeout (RTO) is dynamically adjusted, adapting to the current network condition. To calculate the RTO, round trip time (RTT) measurements are taken on a TCP connection. An RTT sample measures the time from sending a byte of data to receiving the acknowledgment for that byte. Each time an RTT measurement $m$ is made, the RTO $rto$ is recalculated from the new values for the smoothed RTT $a$ and the smoothed mean RTT deviation $d$ according to the following calculations.

\[
Err = m - a \tag{2.1}
\]
\[
a \leftarrow a + g \cdot Err \tag{2.2}
\]
\[
d \leftarrow d + h \cdot (|Err| - d) \tag{2.3}
\]
\[
rto \leftarrow a + 4d \tag{2.4}
\]

The values $g$ and $h$ represent the gains for the RTT and the deviation respectively. Values of $g = 0.125$ and $h = 0.25$ are normally used [11].

### 2.3.4 Karn's Algorithm and Timer Backoff

An ambiguity problem with RTT measurements occurs when a TCP segment is retransmitted. When an acknowledgment for a retransmitted segment arrives, the sender cannot determine whether to associate the acknowledgment with the original transmission or the retransmission when taking the RTT sample. Blindly assuming either one will result in inaccurate RTT measurement. Karn's algorithm [36] addresses this issue. It specifies that the smoothed RTT cannot be updated for retransmitted segments due to acknowledgment ambiguity. In addition, an exponential backoff strategy is used for the retransmission timer,
which means the RTO value is doubled for every retransmission. The backed off timer is also used for subsequent transmission until a valid RTT sample is obtained.

2.3.5 Congestion Control

TCP employs two algorithms to control network congestion. They are known as slow start and congestion avoidance [11]. TCP employs the sliding window protocol for end-to-end flow control, where the window size is determined by the receiver’s window advertisement in the TCP header’s window size field. The congestion control algorithms introduce another window, the congestion window (cwnd), to the sender’s buffer. The actual sender window is the minimum of the receiver’s advertised window and the congestion window.

Slow start is used when a TCP connection is first started or has been idle for some time. Its purpose is to clock the transmission of data segments by ACK segments received to avoid injecting a large amount of data into the network at a time that will likely congest an intermediate router. The congestion window is initially set to one MSS. Each time an ACK segment is received, the congestion window is increased by one MSS. This algorithm allows the congestion window to be increased exponentially, doubling with each RTT.

Congestion avoidance is used when network congestion is detected. TCP assumes network congestion when a segment is believed to be lost, indicated by either a retransmission timeout or the receipt of three duplicate acknowledgments. The algorithm restricts the growth of the congestion window to a linear rate, compared with slow start’s exponential increase, to reduce the amount of data in the network. Specifically, \( cwnd \) is increased by \( \frac{MSS}{cwnd} \) of an MSS for each ACK segment received. This is additive increase, since \( cwnd \) increases by at most one MSS per round trip.
To implement slow start and congestion avoidance together, another variable called the *slow start threshold* \((ssthresh)\) is used. Initially, slow start is performed, until congestion is detected. Then, \(ssthresh\) is set to one half of the current window size (minimum of \(cwnd\) and receiver's advertised window). For a timeout, retransmission is initiated with a slow start. When \(cwnd\) becomes larger than \(ssthresh\), congestion avoidance takes over. Figure 2.7 illustrates this.

![Combined slow start and congestion avoidance](image)

**Figure 2.7** Combined slow start and congestion avoidance

### 2.3.6 Fast Retransmit and Fast Recovery Algorithms

The fast retransmit and fast recovery algorithms were proposed in [37] to give TCP an alternative, more efficient means of recovering from lost segments. The algorithms work by utilizing the duplicate ACKs generated by a TCP receiver in response to out-of-order segments received. A duplicate ACK is an ACK segment giving the same acknowledgment number as a previous ACK. Since TCP acknowledgments are cumulative, an ACK cannot
cover an out-of-order segment, so the ACK for the out-of-order segment is a duplicate ACK.

The fast retransmit algorithm states that the TCP sender performs retransmission upon receiving three duplicate ACKs, without waiting for a retransmission timeout. Since a duplicate ACK can be the result of a reordering of segments, retransmission does not occur until three duplicate ACKs are received in a row, which is a strong indication that a segment is lost.

The fast recovery algorithm follows fast retransmit. It allows the lost segment to be retransmitted without going into a slow start. Using fast recovery, each additional duplicate ACK is interpreted as another out-of-order segment being received, and the end-to-end data pipe is freed by one segment. The sender's window is adjusted accordingly, and new data is sent if allowed by the new window size.

Fast retransmit and fast recovery are implemented together as follows:

1. When the third duplicate ACK is received, set $ssthresh$ to one half of $cwnd$. Then set $cwnd$ to $ssthresh$ plus 3 times MSS. Retransmit the lost segment.

2. For each additional duplicate ACK, increase $cwnd$ by one MSS. Send a new segment if allowed by the new value of $cwnd$.

3. When a new ACK arrives, which acknowledges new data, set $cwnd$ to $ssthresh$ and continue with congestion avoidance.

2.4 The Cellular Wireless Network

A wireless data network (WDN) provides data communication services to mobile users
through the wireless medium. Each user carries a portable device, such as a laptop computer, which is equipped with the necessary wireless communication equipment for accessing the WDN. We refer to each of these devices as a mobile host (MH).

Since the RF spectrum allocated to a particular wireless network is limited, the cellular model is usually used to make efficient use of the available bandwidth. In a cellular network, the geographical area covered by the network is divided into smaller areas called cells. Each cell has a base station (BS), which handles communications with mobile hosts (MHs) within the cell over the air interface. All BSs are connected to a mobile switching centre (MSC), which is the nerve centre of the cellular network. It handles routing, mobility management, registration and authentication for MHs. The MSC is also a gateway to other networks or MSCs. The cellular network architecture is shown in Figure 2.8.

![Cellular network architecture](image)

Figure 2.8 Cellular network architecture

In each cell, a subset of the RF channels available to the network is used. Different
channels are used in adjacent cells to avoid interference. With proper control of transmission power, the same RF channels can be used in two non-adjacent cells that are separated by a distance large enough such that co-channel interference is limited to an acceptable level. This concept of frequency reuse allows greater system capacity compared to sharing all channels over the whole service area. An n-cell frequency reuse scheme means RF channels are reused every n cells. Figure 2.9 illustrates the commonly used 7-cell frequency reuse scheme.

In Figure 2.9, each cell is given a number from 1 to 7. The available spectrum is divided into seven sets of channels. Each cell is assigned one of the seven sets according to its number. Cells with the same number are assigned the same channels, while those with different numbers are assigned different channels.

A mobile user is free to move around. When a user moves out of the coverage area of a
cell and into another cell, the cellular network should maintain network connectivity to the MH by updating its location information at the MSC, so that packets can be routed to the MH through the new BS. This process is called handoff. Cellular handoffs can be either soft or hard. Soft handoff is performed only in networks with overlapping cells. During a soft handoff, the MH is simultaneously communicating with both the old and new BSs. In a hard handoff, communication with the old BS is dropped while routing is being updated. Hard handoff is normally used, because it does not require cells to be overlapping and less network processing is needed. However, it results in the momentary loss of network connectivity to MHs.

2.5 System Model for Wireless Internetworking

The concepts of internetworking, the Internet and cellular WDNs are introduced earlier in this chapter. This section summarizes all these concepts and presents the system model for wireless internetworking.

As described earlier, individual networks are connected to the Internet through routers. In the case of a cellular WDN, an MSC is connected to a mobility support router (MSR), which in turn is connected to the Internet. This is shown in Figure 2.10.

From Figure 2.10, we can view the wireless internetworking architecture as an extension of the existing wireline Internet to the wireless domain through MSRs.

In a communication session between an MH and a fixed host (FH) on the wireline Internet, the data path consists of a single-hop wireless section between the MH and the MSR, and a wireline section between the MSR and the FH through the wireline Internet, as shown in
Figure 2.10 Wireless internetworking architecture

Figure 2.11. The model shown in Figure 2.11 is used as the foundation for discussions in the following chapters.

Figure 2.11 Data path between MH and FH
Chapter 3  Indirect Transport Protocols

This chapter introduces the indirect protocol model, and its specific application to the transport layer. The original indirect TCP (I-TCP) protocol, as well as the modifications made in the proposed end-to-end-acknowledged indirect TCP (EI-TCP), are described in this chapter. Proposed wireless transport protocols for wireless links with and without a link layer retransmission scheme are also described.

3.1 Indirect Protocol Model

In the previous chapter, the layered model of the TCP/IP protocol suite is described. As discussed there, the application and transport layers deal with end-to-end issues in Internet communications. These layers exist only at the end hosts. The protocol entities at these layers are said to interact directly with each other end-to-end, without protocol processing by the intermediate routers. This is illustrated in Figure 2.3. The design of this model is based on the principle of hiding network-specific characteristics from the transport and application layers, and the protocols at these upper layers are able to perform regardless of the underlying network architecture and protocol implementation. In reality, however, since an upper layer depends on the service provided by a lower layer, the performance at the lower layer will affect the performance seen at the upper layer. Complex interactions between protocol layers may be involved.

The indirect protocol model proposed in [23] challenges the principle of making network characteristics transparent above the network/internet layer, due to the possibility of interactions between layers. It argues that by having knowledge of underlying network characteristics, transport and application protocols may be better tuned to the specific network
environment. In cases where drastically different kinds of media are involved in the end-to-end communication path, the end host protocol entities should interact *indirectly*. This means the end-to-end interaction at these layers should be split into separate interactions, one for each kind of medium, and each using a possibly different protocol specific to the kind of medium involved. The wireless internetworked environment is an example where the indirect protocol model is applicable, since drastically different wireless and wired media are involved.

With an indirect transport or application protocol, one or more of the intermediate routers has to perform more than its original IP routing task. In a conventional router, the highest protocol layer involved is the network/internet layer. To support an indirect transport or application layer protocol, processing at a higher layer is required at a splitting point router, which acts as the mediator for the two separate transport or application interactions.

### 3.2 Indirect TCP (I-TCP)

Based on the indirect protocol model, I-TCP is developed as an alternative to TCP as the transport protocol for wireless internetworking. The basic idea of I-TCP is to split the end-to-end TCP connection into two separate transport connections at the MSR.

As discussed in Chapter 1, wireless links present characteristics that are not seen in fixed networks, and these characteristics challenge the performance of TCP in wireless internetworked environments. I-TCP offers the following advantages over end-to-end TCP:

1. It separates the flow control, congestion control and loss recovery in the wireless connection from the wired connection. Losses due to transmission errors and
handoffs on the wireless connection do not affect the operation of the wired connection.

2. The wireless connection can employ a different TCP implementation, or even an entirely different transport protocol, which can better cope with the characteristics of the wireless link, without causing interoperability problem with the TCP on the fixed host (FH).

3. A TCP connection end point at the MSR, which is physically closer to the mobile host (MH), allows faster reaction to losses on the wireless link, especially when the wired connection is over a path with significant delay.

When an MH initiates an I-TCP connection with an FH, the connection request is sent to the MSR. The MSR then requests a regular TCP connection with the FH on behalf of the MH, using the MH address and port number as the connection end point on the MSR. Once the wired connection is established, the MSR completes the I-TCP connection by establishing the wireless transport connection with the MH.

When a TCP data segment from the FH is sent through an I-TCP connection to the MH, the segment is received and acknowledged by the MSR over the regular TCP connection in the wired data path. The MSR then forwards the segment over the wireless transport connection to the MH. Figure 3.1 illustrates this.

Similarly, a data segment from the MH to the FH is received and acknowledged by the MSR over the wireless transport connection. The MSR subsequently sends it over the regular TCP connection to the FH.
3.3 End-to-End-Acknowledged Indirect TCP (EI-TCP)

Although I-TCP works without modifications to TCP implementations on existing hosts, the end-to-end semantics of TCP acknowledgments are violated. In TCP, an acknowledgment indicates correct reception of the corresponding data by TCP at the receiving end. In the FH-to-MH data flow, an acknowledgment received at the FH should indicate correct reception of the corresponding data at the MH. This is not true with I-TCP. In I-TCP, the wired and wireless transport connections operate independently of each other. In particular, data are independently acknowledged on each connection. As shown in Figure 3.1, the MSR will send an ACK segment (segment <2>) to FH right after receiving data X (segment <1>). This ACK can only indicate correct reception by the MSR and not the MH. The MH may eventually not receive data X due to transmission errors or equipment failures.
EI-TCP is a modification to I-TCP to maintain the end-to-end semantics of TCP acknowledgments. The idea in EI-TCP is to delay segment 2 in Figure 3.1 until an acknowledgment from the MH for data X (segment 4) is received by the MSR. The revised time line diagram is shown in Figure 3.2.

![Figure 3.2](image)

Figure 3.2 Time line for data transmission and acknowledgment with EI-TCP

Delayed acknowledgments in EI-TCP are for new data only. For redundant data segments transmitted by the FH that have been acknowledged previously by the MH, the MSR will not forward them to the MH. They are immediately acknowledged and discarded.

In EI-TCP, data received by the MSR but pending acknowledgment from the MH may result in a timeout and retransmission at the FH. When the MSR receives the retransmitted segment from the FH, it will discard it instead of forwarding it again to the MH. This prevents
FH retransmissions from interfering with the retransmission scheme of the wireless transport protocol implemented over the wireless link.

3.4 Wireless Transport Protocols

As mentioned earlier in this chapter, the wireless transport connection in I-TCP and EI-TCP does not need to be compatible with TCP on existing hosts. Therefore, we are free to modify TCP for use as the wireless transport protocol so that it can better handle characteristics of the wireless network. In the original proposal of I-TCP, the wireless transport connection uses TCP with a slight modification to reset the congestion-related parameters \((cwnd\) and \(ssthresh\)) and to initiate slow start after cellular handoffs. This modification, however, only addresses the handoff issue.

In this section, two other proposed modifications for wireless TCP are described. The objective of these modifications is to improve performance in the presence of wireless transmission errors, whether or not a link layer retransmission scheme is employed.

3.4.1 Wireless TCP without Link Layer Retransmissions

Without a link layer retransmission scheme, the transport layer is solely responsible for retransmitting lost data. As discussed in Chapter 2, modern TCP uses both a retransmission timer and the fast retransmit algorithm to trigger retransmission. Normally, the fast retransmit algorithm is faster in detecting a lost segment and triggering retransmission. Fast retransmit requires three duplicate acknowledgments to be received before retransmission occurs, to account for the possibility of duplicate acknowledgments caused by out-of-order segment delivery, which can happen as a result of the connectionless IP protocol.
According to the system model in Chapter 2, the path from the MSR to an MH is only a single IP hop. Therefore, out-of-order segment delivery is not possible, and there is no need to wait for three duplicate acknowledgments to detect a lost segment. For wireless TCP, proposed modification is made to the fast retransmit algorithm so that retransmission occurs at the first duplicate acknowledgment received. This allows even faster detection and recovery of losses. Also, more fast retransmits can be invoked instead of lengthier retransmission timeouts, since only one additional segment has to be received to invoke fast retransmit, instead of three.

3.4.2 Wireless TCP with Link Layer Retransmissions

With a link layer retransmission scheme over the wireless link, the link layer can recover some lost data without the need for retransmission at the transport layer. With a link layer that guarantees delivery, the transport layer can rely solely on the link layer to recover data over the wireless link. This approach not only relieves the transport protocol from the task of loss recovery, but also avoids the undesirable effect of competing retransmissions at the two layers. Therefore, for wireless TCP over a reliable link layer, the proposed modification is to remove the retransmission timer, so that TCP does not perform any timeout retransmissions. However, TCP data segments are still acknowledged, and in EI-TCP, the MSR still waits for the acknowledgment from the MH before sending an acknowledgment to the FH.
Chapter 4  Design of Simulation Models

In the previous chapter, the I-TCP and EI-TCP protocols, associated with proposed protocols for the wireless transport connection, are presented as alternatives to TCP for wireless internetworked environments. In order to evaluate the indirect transport protocols against TCP and the impact of the EI-TCP modification over I-TCP for end-to-end acknowledgments, computer simulations were performed using OPNET, a software package for modelling and simulating computer and data communication networks as well as distributed computing systems. In this chapter, a brief overview of OPNET is given, followed by descriptions of the OPNET models built to accomplish the desired simulations.

4.1 Overview of OPNET

OPNET is a comprehensive network engineering software package, with tools for modelling, simulating and analyzing performance in communication networks and distributed computing systems. Users can model a system with the modelling tools, simulate its behaviour with the Simulation Kernel, and make performance measurements and analysis with the analysis tool.

Models are developed using object-oriented, graphics-based modelling tools. The user constructs a model from a set of objects applicable to the current modelling domain, using a graphical user interface. The behaviour of each object is specified by its own model, which is either predefined within OPNET or defined by the user at the next level of the hierarchical modelling domains. Objects can also contain user-configurable parameters, which are known as object attributes.
Three hierarchical modelling domains are defined in OPNET. These are known as the network, node and process domains, which are described below.

4.1.1 Network Domain

The network domain is the highest level in the hierarchy. It is where network models are specified. A network model is a representation of the system as a whole. It is made up of nodes and communication links, which connect the nodes and allow packets to be passed between them.

Nodes are entities in which packets are created, destroyed and processed. A node is defined by its node model, which is specified in the node domain.

Communication links are full-duplex paths for carrying packets between nodes. Three types of communication links are defined in OPNET: point-to-point, bus and radio. Point-to-point links connect a single source node to a single destination node. Bus links connect a fixed set of nodes to each other. Radio links allow all nodes with radio transceivers to potentially connect to each other, based on a dynamic evaluation. Packet transmission over a link is modelled by a set of sequential packet processing routines. These routines are called pipeline stages in OPNET. Each pipeline stage models a particular aspect of link behaviour. Pipeline stage processing is based on packet and link attributes. Default pipeline procedures are supplied by OPNET for each stage in all link types, but the user can override the default behaviour with user-defined pipeline procedures.

4.1.2 Node Domain

The node domain is the second one in the hierarchy following the network domain. It is
where node models are specified. A node model is used to define the behaviour of a node in a network model.

A node model is constructed from modules and packet streams, which indicate the paths of packet flow between the modules of the node. The most common modules are processors, queues, transmitters and receivers. Processors are general-purpose modules for packet processing. A process model, specified in the process domain, defines the behaviour of a processor. Queues are special processors with packet-queueing capability. A process model is also used to specify the behaviour of a queue module. Transmitters and receivers represent interfaces to communication links. Each link that a node connects to should have a corresponding pair of transmitter and receiver.

4.1.3 Process Domain

The process domain is the lowest level in the hierarchy. It is where process models are specified. A process model defines the behaviour of a processor or queue module in a node model.

A process model is specified by a finite state machine. Each state contains software code, written in C, for handling events that occur at the particular state. An event may also trigger a conditional or unconditional state transition, either before or after event processing. An event can be a packet arrival from one of the incoming packet streams to the processor or queue, or an interrupt scheduled by the current processor or queue, other processors or queues in the node, or the Simulation Kernel. An interrupt may be accompanied by an interface control information (ICI) block, which contains parameters to be passed with the interrupt.
Chapter 4 Design of Simulation Models

OPNET provides standard process models for common communication protocols, such as Ethernet, X.25, frame relay, ATM, and various protocols in the TCP/IP suite.

4.2 Model Descriptions

In this section, the OPNET models developed to simulate the various transport protocols in a wireless internetworked environment are described. Network models are first presented, followed by node and process models, and finally link models.

4.2.1 Network Models

The OPNET network models developed are based on the wireless internetworking system model presented in Chapter 2. The basic model is shown in Figure 4.1.

![Basic OPNET network model](image)

Figure 4.1 Basic OPNET network model

The basic network model consists of three nodes, representing the fixed host (FH), mobility support router (MSR) and mobile host (MH).

The FH and MSR are connected by a point-to-point link that represents the wireline communication path between the FH and the MSR over the Internet. It uses the standard
OPNET point-to-point link model with a data rate of 100 Kbps and propagation delay of 500 ms. As we are primarily interested in the impact of wireless losses on protocol performance, the wireline link is modelled as lossless.

Another point-to-point link connects the MSR and MH. It models the wireless link between the two nodes. Although OPNET provides a sophisticated radio link model, the simpler point-to-point link model is sufficient, since we are only dealing with a single MSR and a single MH. The data rate for the wireless link is specified at 10 Kbps and propagation delay at 10 ms. This is the average throughput and delay seen by the user in a CDPD-based cellular WDN, according to measurements made in an experimental WDN [10]. Specialized error and delay models are used on the wireless link. These are described in a later subsection.

Various network models are derived from the basic model, based on different models used for the MSR and the wireless link. The different network models are for simulation of different transport protocols (TCP, I-TCP and EI-TCP) and, in each case, the presence and absence of a link layer retransmission (LLR) scheme over the wireless link.

4.2.2 Node and Process Models

As shown in Figure 4.1, the basic network model consists of three nodes: FH, MSR, and MH. Both FH and MH use the same node models for all network models. There are three node models for MSR to accommodate the three transport protocols (TCP, I-TCP and EI-TCP).

4.2.2.1 FH and MH

The FH and MH have the same node model structure, which is shown in Figure 4.2.
According to Figure 4.2, the FH/MH node model consists of six modules. They are laid out in a similar manner to the protocol layers in the TCP/IP protocol stack.

The *app* processor corresponds to an application in the application layer of the TCP/IP stack. The only difference between the FH and MH node models is the process model for the *app* processor. In the FH, the process model for *app* implements a simple packet source. It issues a passive open command to the *tcp* processor to listen for a connection. When a connection request is received and accepted, the *app* process will generate fixed-size packets continuously and pass the packets to the *tcp* processor for sending over the newly established connection. The *app* process will try to maintain a full transmit buffer in the *tcp* processor, so
that throughput would not be constrained by the lack of data packets to send.

In the MH, the process model for *app* implements a client requesting data packets from a remote server. It issues an active open command to the *tcp* processor to request a connection to the FH. Once a connection is established, the *app* process will continuously request for data packets. It will try to maintain an outstanding request at any time, so the source will not stop sending due to the lack of requests for data packets.

The *tcp* processor corresponds to the TCP module of the transport layer. Its process model is the standard OPNET TCP process model, with the addition of the fast retransmit and fast recovery algorithms to conform to the TCP Reno implementation.

The *ip-encap* processor and *ip* queue correspond to the IP layer. When a TCP segment arrives from the *tcp* processor, the *ip-encap* processor encapsulates it in an IP datagram before passing it to the *ip* queue. Similarly, when an IP datagram arrives from the *ip* queue, the *ip-encap* processor decapsulates it and forwards the decapsulated TCP segment to the *tcp* processor. The *ip* queue handles the queuing and routing of IP datagrams coming from the network (through the *rcv* module) and the *ip-encap* processor. Both the *ip-encap* processor and *ip* queue use the respective standard OPNET process models.

The *xmt* and *rcv* modules form a standard OPNET transmitter-receiver pair, representing the interface to the communication link.

### 4.2.2.2 MSR

Three node models are developed for the MSR. They correspond to the three transport protocols (TCP, I-TCP and EI-TCP) simulated.
Figure 4.3 shows the MSR node model used for TCP simulations.

The MSR node model for TCP simulations represents a standard IP router with two network interfaces. The *ip* queue is the same as the one in the FH and MH nodes. It is connected to two transmitter-receiver pairs, representing the interfaces to the wireline and wireless links.

MSR node models for I-TCP and EI-TCP simulations share the same structure, as shown in Figure 4.4.

The MSR model for I-TCP and EI-TCP differs from the one for TCP by the addition of three processors, namely *ip-encap*, *tcp* and *itcp-proxy*.

In the MSR model for I-TCP, the *ip-encap* and *tcp* processors are the same as the ones in the FH and MH nodes. The *itcp-proxy* processor represents the proxy for I-TCP.
connections. It issues a passive open command to the tcp processor to listen for a connection. When a connection request arrives from the MH over the wireless link, the itcp-proxy processor will issue an active open command to the tcp processor to request a connection to the FH on behalf of the MH. Subsequently, for any packets arriving from the FH over the wired connection, the itcp-proxy will forward the packets to the MH over the wireless connection.

In the MSR model for EI-TCP, slight modifications are made to the tcp and itcp-proxy
processors. The tcp processor is modified so that it does not automatically generate acknowledgments in response to arrived segments. Instead, acknowledgments are generated in response to acknowledge commands from the itcp-proxy processor. Moreover, it would notify the itcp-proxy processor of any new data acknowledged by the MH. The itcp-proxy, on receiving a new data acknowledged indication that new data are acknowledged by the MH, would issue an acknowledge command back to the tcp processor, so an acknowledgment can be sent back to the FH.

4.2.3 Wireless Link Models

Specialized wireless link models are developed to model wireless link behaviour with OPNET's point-to-point link model. The wireless link error model is first presented, followed by modifications made to the standard OPNET point-to-point link pipeline stages to model the error behaviour in both the presence and absence of link layer retransmission mechanism.

The model for wireless error behaviour is based on a two-state Markov model, also known as the Gilbert channel model, based on early work by Gilbert [38]. This is shown in [39] to be an adequately accurate model for fading channels. Figure 4.5 illustrates the model.

As shown in Figure 4.5, the model consists of two states: a Good state (shown as G) and a Bad state (shown as B). In the Good state, all packets transmitted over the wireless link are error-free, whereas in the Bad state, all packet transmissions will fail. Whenever a packet arrives at the link, an evaluation is made to determine if a state transition should occur. Let

\[ P_B = \text{Probability that current state is Bad} \]
Figure 4.5 Two-state Markov model for wireless link

\[ P_G = \text{Probability that current state is Good} = 1 - P_B \]

\[ P_{BG} = \text{Transition probability from Bad to Good} \]

\[ P_{GB} = \text{Transition probability from Good to Bad} \]

Then, we can calculate \( P_B \) as follows:

\[
P_B = P_B \cdot (1 - P_{BG}) + (1 - P_B) \cdot P_{GB} \quad (4.1)
\]

Solving for \( P_B \) in (4.1), we get

\[
P_B = \frac{P_{GB}}{P_{BG} + P_{GB}} \quad (4.2)
\]

This is also the mean packet loss rate (PLR) for packet transmissions over the wireless link.
By rearranging (4.2), we can express $P_{BG}$ as follows:

$$P_{BG} = P_{GB} \frac{1 - P_B}{P_B}$$  \hspace{1cm} (4.3)

PLR and $P_{BG}$ together are used as the attributes to the wireless link error model. PLR defines how "lossy" the link is, and $P_{BG}$ determines the distribution of those losses, as $1/P_{BG}$ gives the mean number of packets in an error burst. For a given $P_{BG}$, the maximum PLR is attained when $P_{GB} = 1$. This means the maximum PLR is given by $1/(1+P_{BG})$.

Without link layer retransmissions, wireless transmission errors are made visible above the link layer. The error pipeline stage in the OPNET link model is modified to determine transmission error using the two-state Markov model instead of the default independent bit error model.

With link layer retransmissions, the link layer is assumed to hide all wireless transmission errors within the link layer. Accordingly, the error pipeline stage is modified to make all packet transmissions error-free. The link layer is modelled to perform standard Go-Back-N retransmissions. When a packet undergoes retransmissions, the upper layers will see an increased transmission delay for the packet. The transmission delay pipeline stage is modified to calculate transmission delay according to the following formula:

$$N \times T + (N - 1) \times 2 \times P$$  \hspace{1cm} (4.4)

In the above formula, $T$ is the transmission delay of each link layer transmission, and $P$ is the propagation delay. $N$ is the number of link layer transmissions required for success. It is
a random variable based on the geometric distribution. The two-state Markov model determines whether each link layer transmission attempt is a success or a failure.
Chapter 5  Simulation Results and Discussions

In the previous chapter, OPNET models used for performing the simulations of interest are described. This chapter presents results from the simulations and discusses the results. An overview of the simulation scenarios and the corresponding objectives are first presented. Each subsequent section deals with the results of a particular scenario.

5.1  Simulations Overview

The general objective of the simulations is to study and compare the performance of TCP, I-TCP and EI-TCP in wireless internetworked environments. Different simulation scenarios reflect the different wireless link characteristics and protocol implementations under investigation. The main performance metric for comparison is transport layer throughput seen at the MH in a bulk data transfer session from the FH to the MH.

The first scenario studies performance in relation to the protocol architecture and the impact of variation in the packet loss rate (PLR). Here, standard TCP is used for both the wireline and wireless side connection of I-TCP and EI-TCP. Performance differences are the results of using indirect (I-TCP and EI-TCP) versus direct (TCP) transport connection and whether end-to-end acknowledgments are used (TCP and EI-TCP) or not (I-TCP).

The second scenario investigates the effect of using an optimized fast retransmit algorithm on the wireless side transport connection in I-TCP and EI-TCP, as discussed in Chapter 3. Results are compared with those from the first scenario, and the differences can be attributed to the different wireless transport connection employed.

The third scenario investigates the impact on performance resulting from the
distribution of wireless packet losses. Here, results for different $P_{BG}$ values with the same PLR are recorded and compared.

The fourth scenario involves the use of link layer retransmissions on the wireless link. Performance of I-TCP and EI-TCP in eliminating possible redundant retransmissions in the transport layer is studied. Non-retransmitting TCP, as described in Chapter 3, is used for the wireless side connection in I-TCP and EI-TCP. Results for variations in both the PLR and $P_{BG}$ are obtained.

In each simulation scenario, ten independent simulations are performed with the same set of simulation parameters but different random seeds. Each simulation run lasts for 20000 seconds in simulation time. The time average result of each run is recorded, and the average of the results from the ten runs in each simulation set is presented. In all simulations, a TCP segment size of 512 bytes is used, and all TCP receive buffers are 4 KB in size.

5.2 Effect of Protocol Architecture

This section presents the results and discussions on the first simulation scenario, which studies the effect of protocol architecture on performance. Simulations of TCP, I-TCP and EI-TCP are performed with different PLR values. The same parameter for wireless loss distribution is used in all cases, based on a $P_{BG}$ value of 1.0. This means the wireless link does not remain in the Bad state for more than one packet transmission. Also, the same TCP Reno implementation is used on the TCP simulations and on both the wireline and wireless transport connections in the I-TCP and EI-TCP simulations.

Figure 5.1 shows the transport layer throughput under various PLR values for the three
transport protocols.

![Transport Layer Throughput Comparison for TCP, I-TCP, and EI-TCP](image)

**Figure 5.1** Transport layer throughput comparison for TCP, I-TCP and EI-TCP, with unmodified TCP Reno over the wireless transport connection of I-TCP and EI-TCP.

As shown in Figure 5.1, the performances for the three protocols are similar. Although the differences in performance are small, EI-TCP is seen to give consistently better performance than the other two transport protocols. I-TCP also has better performance over TCP except in the zero PLR case. Throughput for all protocols drop practically to zero at
around 35% PLR.

The generally better performances seen for EI-TCP and I-TCP over TCP can be explained by the more effective local error recovery over the wireless network under the indirect architecture, compared to end-to-end error recovery in TCP. However, the improvement is still ultimately limited by the inefficiency of TCP Reno over a wireless environment.

An interesting observation is the lower throughput for I-TCP in the zero PLR scenario. When PLR is zero, EI-TCP and TCP have essentially the same behaviour in the simulations. What I-TCP differs from the other two protocols in this case is the mechanism for acknowledgments. EI-TCP and I-TCP both use end-to-end acknowledgments, whereas I-TCP has separate acknowledgments for the wireline and wireless sides. In the network model, the wireline data path between FH and MSR has ten times the data rate of the wireless data path between MSR and MH. Under I-TCP, there is a roughly ten times throughput on the wireline transport connection compared to the wireless transport connection. From time to time, the wireline connection is stalled waiting for data to be sent over the wireless connection and then cleared from the buffers in the MSR. Slow-start is used every time the wireline connection is restarted. As a result, throughput is reduced. The problem is not seen in EI-TCP, since the end-to-end mechanism of acknowledgments means the FH is receiving acknowledgments at the same rate as the wireless data rate, so the FH sending rate is bound to the rate of this ACK clock. This also explains the better performance obtained in EI-TCP over I-TCP.

5.3 Effect of Optimizing the Wireless Transport Protocol

This section presents the results and discussions on the second simulation scenario,
which studies the effect of optimizing the wireless transport protocol on the performances of I-TCP and EI-TCP. All simulation parameters are the same as those in the first scenario discussed in the previous section, with the exception of the wireless transport protocol for I-TCP and EI-TCP. In this scenario, the wireless transport protocol uses an optimized fast retransmit algorithm discussed in Chapter 3.

Figure 5.2 shows the transport layer throughput under various PLR values for the three transport protocols in the new scenario.

The TCP curve in Figure 5.2 is exactly the same as the one in Figure 5.1, as there is no change in the simulation conditions. Comparing the I-TCP and EI-TCP curves in the two figures, the ones in Figure 5.2 show greater performance improvements over the TCP curve. Up to 500% increase in transport layer throughput over TCP is obtained by using I-TCP or EI-TCP with an optimized fast retransmit algorithm in the wireless transport protocol. Also, the throughput drops to zero at a higher PLR. This is consistent with the claim made in Chapter 3 that the optimized fast retransmit algorithm allows more effective error recovery over the wireless transport connection, compared to the standard fast retransmit algorithm.

The slightly better performance obtained in EI-TCP over I-TCP is also seen in the new scenario. The advantage of end-to-end acknowledgments, discussed in the previous section, is still valid in the new scenario.

5.4 Effect of Wireless Loss Distribution

This section presents the results and discussions on the third simulation scenario, which studies the effect of wireless loss distribution on the performance of TCP, I-TCP and
EI-TCP. Simulation conditions are based on the ones in the second scenario. In particular, optimized fast retransmit is used on the wireless transport protocol for I-TCP and EI-TCP. The parameter for wireless loss distribution, $P_{BG}$, is varied. A transport layer throughput curve, similar to the ones presented for the previous scenarios, is obtained for a number of different $P_{BG}$ values for each of TCP, I-TCP and EI-TCP.
Figure 5.3 shows the transport layer throughput curves for TCP with $P_{BG}$ values between 0.1 and 1.0.

![Transport layer throughput comparison with TCP under different wireless loss distributions](image)

According to Figure 5.3, significant performance differences are seen for different wireless loss distributions. It shows that TCP throughput depends not only on PLR, but also to
a large extent on the loss distribution. In general, for the same PLR, throughput decreases as \( P_{BG} \) decreases, which means an increase in the average number of lost packets per error burst. This observation is consistent with the results from [40], which demonstrates the significant impact of consecutive packet losses on throughput in a TCP Reno connection. However, as PLR increases, the effect of \( P_{BG} \) differences becomes less significant. According to [40], consecutive packet losses increase the likelihood of a retransmission timeout as opposed to the more efficient error recovery by the fast retransmit and fast recovery algorithms. The likelihood of a retransmission timeout also increases when PLR increases. With a retransmission timeout, TCP Reno performs a Go-Back-N retransmission of unacknowledged segments, so the distribution of losses becomes less significant. This explains why \( P_{BG} \) difference has less effect with increasing PLR.

Figures 5.4 and 5.5 show the corresponding transport layer throughput curves for I-TCP and EI-TCP respectively.

Both Figures 5.4 and 5.5 show the same general trend of decreasing throughput with decreasing \( P_{BG} \), which is consistent with the trend observed in Figure 5.3 for TCP. However, comparing Figures 5.4 and 5.5 with 5.3 shows that I-TCP and EI-TCP tend to sustain better throughput against decreasing \( P_{BG} \) than TCP. Significant throughput drops tend to occur at lower \( P_{BG} \) values for I-TCP and EI-TCP. This shows that I-TCP and EI-TCP with the optimized fast retransmit algorithm over the wireless transport connection is not only more robust against increases in PLR, but also against consecutive losses.
5.5 Effect of Retransmissions at the Link Layer

This section presents the results and discussions on the fourth simulation scenario, which studies the effect of link layer retransmissions (LLRs) on the performance of TCP, I-TCP and EI-TCP. The LLR-enabled link layer is modelled as error-free to the IP layer and above, but with variable transmission delay, as presented in Chapter 4. The wireless transport
connection in I-TCP and EI-TCP has its retransmission mechanism turned off, so the problem of potential redundant retransmissions by both transport and link layers, as mentioned in Chapter 1, is eliminated. However, TCP is subject to this redundant retransmission problem. Throughput comparison is made among the three protocols at different $P_{BG}$ values. For each $P_{BG}$ value, PLR is varied, and a transport layer throughput curve similar to those from the
previous scenarios is obtained for each protocol. Any performance difference between EI-TCP and TCP can be attributed to degradation by redundant TCP retransmissions.

Figure 5.6 shows the transport layer throughput curves with wireless link $P_{BG} = 1.0$.

![Transport layer throughput curves with wireless link $P_{BG} = 1.0$.](image)

Figure 5.6 Transport layer throughput comparison for TCP, I-TCP and EI-TCP, with link layer retransmissions over the wireless link of $P_{BG} = 1.0$.

According to Figure 5.6, except for the lower throughput sustained by I-TCP at 0 to
20% PLR, the three curves essentially overlap one another. The lower throughput for I-TCP at the low end of PLR is consistent with results obtained for previous scenarios, and can again be explained by the effect of non-end-to-end acknowledgments, as discussed earlier in this chapter. Other than that, virtually no performance difference exists among the three protocols. In other words, the problem of redundant retransmissions does not exist.

At $P_{BG} = 1.0$, at most one retransmission is needed before a packet is successfully transmitted. That means the worst case end-to-end delay increase is about one packet transmission delay over the wireless link. The RTT variation allowance built into the TCP RTO calculation is able to account for such limited RTT increase without triggering a retransmission timeout at the TCP source. Without a retransmission, the TCP source can use the increased RTT measurement caused by wireless link LLR to adjust future RTO values, which should be increased due to the larger RTT measurement received. This should further decrease the likelihood of a TCP retransmission timeout and redundant retransmission.

The situation is not the same if $P_{BG}$ is decreased. Figure 5.7 shows the throughput curves for $P_{BG} = 0.7$.

Comparing Figures 5.6 and 5.7, both the I-TCP and EI-TCP curves show no difference between the two figures. However, the TCP curve shows lower throughput between 0 and 40% PLR in Figure 5.7. A maximum of 0.6 Kbps, or about 9% throughput difference is seen between TCP and the indirect transport protocols at roughly 25% PLR.

In the LLR simulation scenario, performance of I-TCP and EI-TCP depends almost entirely on the performance of the underlying wireless link LLR. The GBN LLR mechanism
Figure 5.7  Transport layer throughput comparison for TCP, I-TCP and EI-TCP, with link layer retransmissions over the wireless link of $P_{BG} = 0.7$

employed in the model is insensitive to loss distribution, which explains why I-TCP and EI-TCP performance is not affected by changes in $P_{BG}$.

In the case of TCP, the lower throughput seen for lower $P_{BG}$ shows that redundant TCP retransmissions start to degrade TCP performance when average error bursts get longer. A
longer error burst means that the LLR has to make more retransmission attempts in order to transmit a given packet successfully. That means a longer end-to-end delay is seen by the upper layers, which can result in a TCP retransmission timeout and subsequent redundant retransmission.

However, the performance difference between TCP and the indirect transport protocols does not keep widening as PLR increases. After hitting the maximum throughput difference point at around 25% PLR, the TCP throughput gradually gets back to the level attained by the indirect protocols, and for PLR of 40% and above, this performance difference can no longer be seen. This shows that the redundant retransmission problem is getting less severe as PLR increases beyond a certain threshold value. In other words, the TCP RTO is adjusted to a more appropriate value, such that redundant retransmissions resulting from the increased end-to-end delay caused by LLR are reduced or eliminated.

The observed TCP performance can be explained by Karn's algorithm, described in Chapter 2. When LLR causes a TCP retransmission timeout, Karn's algorithm prohibits the update of the smoothed RTT from RTT measurements made by retransmitted TCP segments. This means the RTT measurement that contains the delay increase caused by LLR cannot be used to update the smoothed RTT, from which the RTO value is calculated. If RTO cannot be updated to reflect the LLR delay increase, more redundant retransmissions can be expected. However, another feature of Karn's algorithm can help to cope with this problem. Karn's algorithm also specifies the use of exponential backoff to the retransmission timer. The backed off timer is used in retransmissions and subsequent transmission of new data, until a valid RTT measurement can be obtained. During the period when the backed off timer is in effect, the
TCP sender can tolerate greater delay variation without causing a retransmission timeout, which means RTT measurements for segments that have experienced LLR delay increase are more likely to be accepted by the TCP sender for updating the smoothed RTT and RTO. When PLR increases beyond a certain value, the higher probability of LLR delay increase during the timer backoff period results in faster RTO adjustment to an optimal value, and redundant retransmissions can be reduced or eliminated sooner. This explains why TCP performance approaches that of the indirect protocols again after hitting some worst performance point.

The effect of further decreasing $P_{BG}$ can be seen in Figure 5.8, which shows the transport layer throughput curves for $P_{BG} = 0.4$.

Figure 5.8 gives a similar picture to Figure 5.7. Again, performance of I-TCP and EI-TCP is unchanged comparing to Figure 5.6 and 5.7. TCP shows a sharper throughput decrease than the indirect protocols as PLR increases from 0, but the throughput difference between TCP and the indirect protocols maximizes at some PLR, and then the difference narrows down with further PLR increase until it finally vanishes as PLR grows above certain value. Comparing to Figure 5.7, Figure 5.8 shows a larger performance difference between TCP and the indirect protocols. A maximum throughput drop of 0.9 Kbps, or almost 15%, is seen for TCP compared to the indirect protocols. Also, TCP cannot achieve the performance of the indirect protocols until PLR increases to 60% or above.

With $P_{BG}$ decreased further, the mean error burst size becomes even larger. According to the above explanation, this translates into a more severe problem of redundant retransmission for TCP. A longer error burst means an even higher probability of having a
retransmission timeout and a subsequent redundant retransmission at the TCP sender. Moreover, for the same PLR, a lower $P_{BG}$ means that the mean time between two error bursts is longer. This lowers the probability of having another error burst during the timer backoff period and results in slower RTO adaptation to the LLR delay increase, as shown in Figure 5.8 by the wider range of PLR values that cause degraded TCP performance.

Figure 5.8 Transport layer throughput comparison for TCP, I-TCP and EI-TCP, with link layer retransmissions over the wireless link of $P_{BG} = 0.4$.
Chapter 6 Conclusions

The tremendous growth in the use of portable computing devices and the Internet in recent years is driving the demand for mobile computing support through wireless internetworking. This also drives a lot of research activities focusing on various aspects of mobile computing and wireless internetworking. One area of research interest is to address efficiency issues of existing internetworking protocols when applied to the new wireless internetworked environment. This thesis contributes to this area of the existing work base of wireless internetworking research. It addresses the performance impact on TCP due to the presence of a lossy wireless link along the end-to-end data path between the communicating end hosts.

The primary contribution of this thesis is the proposal of EI-TCP as an alternative transport protocol for the wireless internetworked environment. The beauty of EI-TCP is that it achieves performance improvement without requiring changes to current Internet hosts. The architecture is also flexible enough to allow adaptation to wireless links of different link layer characteristics. In addition, EI-TCP improves on the earlier I-TCP proposal by maintaining the end-to-end semantics of TCP acknowledgments. Simulation studies show that no loss of performance occurs as a result of providing end-to-end acknowledgments to the indirect transport architecture. On the contrary, with end-to-end acknowledgments on EI-TCP, the performance degradation seen with I-TCP as a result of the vast bandwidth difference between the wired and wireless I-TCP connection segments is eliminated.

To further utilize the benefits of the indirect transport protocol architecture, modifications to standard TCP Reno are proposed in this thesis for the wireless transport
Chapter 6 Conclusions

connection of I-TCP and EI-TCP. These modifications adapt the wireless transport protocol to the specific characteristics of the wireless link depending on whether a reliable link layer is used. Without a reliable link layer, the fast retransmit algorithm of TCP Reno is modified to take one duplicate acknowledgment instead of three to indicate a lost TCP segment. This boost to the loss recovery process of the wireless TCP connection is shown to give drastically improved I-TCP and EI-TCP performance. With a reliable link layer, the TCP Reno retransmission mechanism is turned off to avoid redundant retransmissions by the transport layer. Simulation results indicate up to 15% better throughput performance over TCP, which is subject to the redundant retransmission problem.

In contrast with previous work in the area, whose results are mostly based on studies in a wireless LAN environment, the results in this thesis are obtained from a wide-area based wireless internetworking model. Since the majority of mobile computing activities are perceived to involve wide area mobility, the results from this thesis should have greater practical interest. In modelling the error characteristics of the wireless link, a more sophisticated burst error model is used instead of the independent bit error model used in some earlier work. The burst error model is shown to be more accurate in modelling a fading channel. Simulation results are obtained with variations in not only the PLR, but also the error distribution parameter, to provide a more complete picture of protocol performance characteristics.

To further extend the work of this thesis, the following possible directions of future research are suggested:

1. This thesis presents performance data by computer simulations only. Real life
Internet traffic characteristics are much more complex than those modelled in the simulations. Implementation and experimentation of the protocols in a live wireless internetworked environment is highly desirable for validating the results presented here.

2. Simulation results show significant performance improvement by tuning the wireless TCP Reno connection to the wireless link characteristics. This topic is worth further investigation. Some earlier proposals on improving TCP performance over wireless networks that involve TCP modifications, mentioned in Chapter 1, can be implemented on the wireless TCP connection. The indirect architecture takes care of the interoperability problems. A newer TCP implementation known as TCP Vegas [41], with its revised retransmission strategy, is also a good potential candidate. Some custom transport protocols [29, 30] designed for the wireless transport connection in other I-TCP-like environments may also be studied for EI-TCP.

3. This thesis assumes a system model in which MH mobility is confined within cells controlled by a single MSC. Support for MH mobility between different MSCs, such as moving between networks run by different operators, is a desirable feature. The current IETF Mobile IP standard [42] provides support for forwarding IP datagrams from the home MSR of the MH to the MSR in the visiting network. With the route optimization scheme proposed for Mobile IP [43], IP datagrams destined for the MH in the visiting network do not need to be routed through the home MSR, in which case the indirect transport connection state information has to be migrated.
to the new MSR. Handoff schemes for I-TCP connections have been proposed [44, 45]. Further work is required to evaluate these schemes with EI-TCP working in conjunction with either basic or route-optimized Mobile IP.
### Glossary

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>Acknowledgment</td>
</tr>
<tr>
<td>AMPS</td>
<td>Advanced Mobile Phone System</td>
</tr>
<tr>
<td>ARQ</td>
<td>Automatic Repeat Request</td>
</tr>
<tr>
<td>BER</td>
<td>Bit Error Rate</td>
</tr>
<tr>
<td>BS</td>
<td>Base Station</td>
</tr>
<tr>
<td>CDMA</td>
<td>Code Division Multiple Access</td>
</tr>
<tr>
<td>CDPD</td>
<td>Cellular Digital Packet Data</td>
</tr>
<tr>
<td>EI-TCP</td>
<td>End-to-End-Acknowledged Indirect TCP</td>
</tr>
<tr>
<td>FEC</td>
<td>Forward Error Correction</td>
</tr>
<tr>
<td>FH</td>
<td>Fixed Host</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>GBN</td>
<td>Go-Back-N</td>
</tr>
<tr>
<td>GSM</td>
<td>Global System for Mobile Communications</td>
</tr>
<tr>
<td>ICMP</td>
<td>Internet Control Message Protocol</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IGMP</td>
<td>Internet Group Management Protocol</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>I-TCP</td>
<td>Indirect TCP</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>LLR</td>
<td>Link Layer Retransmission</td>
</tr>
<tr>
<td>MH</td>
<td>Mobile Host</td>
</tr>
<tr>
<td>MSC</td>
<td>Mobile Switching Centre</td>
</tr>
<tr>
<td>MSR</td>
<td>Mobility Support Router</td>
</tr>
<tr>
<td>MSS</td>
<td>Maximum Segment Size</td>
</tr>
<tr>
<td>NAK</td>
<td>Negative Acknowledgment</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistant</td>
</tr>
<tr>
<td>PDU</td>
<td>Protocol Data Unit</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>PLR</td>
<td>Packet Loss Rate</td>
</tr>
<tr>
<td>RTO</td>
<td>Retransmission Timeout</td>
</tr>
<tr>
<td>RTT</td>
<td>Round Trip Time</td>
</tr>
<tr>
<td>SACK</td>
<td>Selective Acknowledgment</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TDMA</td>
<td>Time Division Multiple Access</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>WDN</td>
<td>Wireless Data Network</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
</tbody>
</table>
References


