A MOBILE-AWARE TRANSMISSION CONTROL PROTOCOL (TCP) FOR WIRELESS COMMUNICATIONS

By

Pamela Chia-pei Lee

Master of Science (Industrial Engineering and Management Sciences)

Northwestern University, 1993

Bachelor of Science (Physics)

National Taiwan University, 1991

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in the School of Engineering Science

SIMON FRASER UNIVERSITY

December, 1996

APPROVAL

Name: Pamela Chia-pei Lee

Degree: Master of Applied Science

Title of thesis: A Mobile-Aware Transmission Control Protocol (TCP) for Wireless Communication

Examining Committee: Dr. Paul Ho
Associate Professor, Engineering Science, Chairman

Dr. R.H.S. Hardy
Professor, Engineering Science
Senior Supervisor

Dr. V. Cuperman
Professor, Engineering Science
Supervisor

Dr. J. Vaisey
Associate Professor, Engineering Science
Internal Examiner

Date Approved: December 6, 1996

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its usrs. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

"A Mobile-Aware Transmission Control Protocol (TCP) for Wireless Communications"

Author:

_____
(signature)

_____
(name)

November 19, 1996
(date)

Abstract

The current Transmission Control Protocol/Internet Protocol (TCP/IP) protocol suite is not designed to support mobile hosts and wireless links, since host migration will result in a loss of connection. All existing network applications must then be restarted. Several solutions have been proposed to provide for host mobility, but they all confine mobility support functions to the IP layer. In theory, mobile hosts can use existing higher-layer protocols such as TCP with any one of these proposed solutions to communicate with the fixed networks. However, this approach can degrade TCP performance significantly. The reason is that TCP cannot differentiate the effects of network mobility from the effects of network congestion. Whenever a packet loss is caused by cell handoffs or by unreliable wireless links, TCP will use the wrong remedy, that is, the congestion control procedures, to alleviate the impact of mobility. In this thesis, we propose a mobile-aware TCP that can distinguish mobility-related packet losses from congestion-related ones and provide mobility alleviation procedures to control the effects of mobility. We also develop analytical and simulation models to evaluate the proposed scheme. Both analytical and simulation results show that our scheme can effectively improve TCP performance in mobile computing environments.

Acknowledgements

First of all, I would like to thank my adviser, Dr. R.H.S. (Steve) Hardy, for his guidance and assistance through the course of this research. Secondly, I would like to thank my family for their encouragement and support. I would also like to thank Motorola for its financial support and MIL 3, Inc. for providing access to the OPNET simulator. Finally, thanks to my groupmates, Paraskevas Polydorou and Panayiotis Toundas, for providing helpful suggestions about writing simulations.

Table of Contents

Approval ii

Abstract iii

Acknowledgements iv

Table of Contents v

List of Figures vii

List of Tables ix

List of Abbreviations x

1. INTRODUCTION 1

    1.1 Impact of Mobility on TCP 2

    1.2 Contributions of the Thesis 3

2. RELATED WORK 5

    2.1 Fast Retransmission Scheme 5

    2.2 I-TCP Scheme 6

    2.3 Snoop Scheme 7

    2.4 Local Handoff Scheme 8

    2.5 AIRMAIL Scheme 9

3. MOBILE-AWARE TCP 10

3.1 Proposed Scheme 11

3.2 Evaluation 13

3.2.1 Analytical Model 13

3.2.2 Simulation Mode 19

4. RESULTS 32

4.1 Analytical Results 32

4.2 Simulation Results 34

4.3 Comparison with the Literature 42

5. CONCLUSIONS 43

5.1 Future Work 44

Appendix 45

References 54

List of Figures

1. Sliding window control model with acknowledgment at end of window 14
2. OPNET modeling structure 20
3. The network model 21
4. The node model of the stationary workstation 22
5. The node model of the mobile workstation 25
6. The gateway node model 26
7. The jammer node model 28
8. The TCP process model 29
9. The process model of the fast retransmission scheme 30
10. The process model of the proposed scheme 30
11. Throughput performances of the no-move TCP, the normal TCP, the fast retransmission scheme, and the proposed scheme 33
12. The behaviors of the TCP sequence number of the normal TCP, the fast retransmission scheme, and the proposed scheme in the no-error environment 36
13. The behaviors of the TCP congestion window of the normal TCP, the fast retransmission scheme, and the proposed scheme in the no-error environment 37

14. The behaviors of the TCP sequence number of the normal TCP, the fast retransmission scheme, and the proposed scheme in the error environment 39

15. The behaviors of the TCP congestion window of the normal TCP, the fast retransmission scheme, and the proposed scheme in the error environment 40

List of Tables

1. Throughput performances of the normal TCP, the fast retransmission scheme, and the proposed scheme in the no-error environment (with 95% confidence interval) 35
2. Throughput performances and bit error rates of the normal TCP, the fast retransmission scheme, and the proposed scheme in the error environment (with 95% confidence interval) 38

List of Abbreviations

AIRMAIL	AsymmetrIc Reliable Mobile Access In Link-layer
ARQ	Automatic Repeat Request
CSMA/CD	Carrier-Sense Multiple Access with Collision Detection
FA	Foreign Agent
FEC	Forward Error Correction
HA	Home Agent
IP	Internet Protocol
I-TCP	Indirect Transmission Control Protocol
MH	Mobile Host
OPNET	OPtimized Network Engineering Tools
RFC	Request For Comment
RTT	Round-Trip Time
SH	Stationary Host
TCP	Transmission Control Protocol

Chapter 1

INTRODUCTION

Computer networks, today, play an important role in providing global communication and information sharing. With the arrival of portable computers, people expect that they can have access to the networks wherever they are and move around the local or wide area networks while retaining their network connections. Unfortunately, the current Transmission Control Protocol/Internet Protocol (TCP/IP) protocol suite is not designed with these mobile hosts (MHs) and wireless links in mind. Host migration will lose its former connection and all existing network applications must be restarted.

Several solutions have been proposed to provide host mobility; however, they all confine mobility support functions to IP layer and attempt to hide mobility from higher layers, so that MHs can continue to run existing higher-layer protocols such as TCP without any modifications. Although this approach is feasible, it will cause serious performance problems in TCP layer [1]. Therefore, TCP should also be modified to support mobility.

This thesis describes a modification scheme to improve TCP performance in mobile computing environments. Before going to that far, we first discuss the performance problems that TCP will experience in such environments.

1.1 Impact of Mobility on TCP

Current TCP has been tuned for wired links and stationary hosts (SHs). It provides end-to-end reliability by requiring each end of a connection to acknowledge the packet it receives from the other end and to retransmit the packet which is not acknowledged in time [2]. TCP maintains a round-trip time (RTT) estimator and a mean deviation to monitor this end-to-end reliable packet transfer. It continually measures the round-trip delay between sending a packet and receiving an acknowledgment for that packet. Every time a new measurement is made, the RTT estimator and the mean deviation are updated. Since TCP uses sliding window mechanism[1] and slow-start algorithm[2] to control the amount of data in transit through the network, if a packet does not receive its acknowledgment within four times the mean deviation from the estimated RTT, TCP will assume that the packet is delayed and lost due to network congestion. It thus retransmits the lost packet and initiates congestion control procedures to allow the network to

[1] The sliding window mechanism allows a sender to transmit several packets before receiving an acknowledgment. The number of unacknowledged packets is known as the window size which is advertised by the receiver. After receiving an acknowledgment for the first packet transmitted, the sender "slides" the window and sends another one [2].

[2] The slow-start algorithm keeps a congestion window and a slow-start threshold in the sender's TCP. When starting or restarting after a timeout, the congestion window is initialized to one packet. Each time an acknowledgment is received, the congestion window will be increased by one packet (exponential increase). When the congestion window reaches the slow-start threshold, it will then be increased by at most one packet for each RTT (linear increase). The sender can transmit up to the minimum of the congestion window and the window size advertised by the receiver [2].

recover: dropping the transmission window size, slowing down the packet-sending rate (slow-start), and doubling the retransmission interval (exponential backoff) [3].

In wireless environments, however, factors which are not related to congestion will also cause packet loss and delay. For example, packets may be lost due to the relatively high bit error rates over wireless links, communication may pause during handoffs, and so on. Besides, these mobility-related packet losses and delays are expected to occur more frequently than congestion-related ones. Since TCP can not distinguish between them, it will initiate congestion control procedures even if a packet loss or delay is caused by mobility, and thus its performance will be significantly degraded.

The effects of mobility on TCP performance have been studied in [4]. Based on experiments performed in a wireless networking testbed, Cáceres and Iftode have shown that in the presence of motion, throughput drops up to 31% and pauses in communication are up to 2.8 seconds. In addition, even though IP-layer connection is reestablished after each handoff, TCP-layer communication still has to wait for the retransmission timeout to resume. The main reasons for the performance degradation are packet loss during handoffs and packet loss due to wireless transmission errors. Lost packets also cause long pauses and trigger congestion control procedures, which further degrade performance.

1.2 Contributions of the Thesis

In this thesis, we propose a mobile-aware TCP to mitigate the effects of mobility on TCP performance. Our scheme consists of two parts: the mobility detection part and the mobility control part. The first part diagnoses network mobility based on the signal strength received by an MH. Then, when a packet loss or delay is mobility-related, the

second part performs mobility alleviation procedures to control the effects of mobility: reducing the packet-sending rate and preserving the transmission window size and the retransmission interval. We also develop analytical and simulation models to compare the performance of the proposed scheme with those of normal TCP and another TCP-layer modification scheme, so that we can evaluate the effectiveness of the proposed scheme. Both the analytical and the simulation results show that our scheme can effectively improve TCP performance in mobile environments.

The remainder of the thesis is organized as follows. Chapter 2 discusses some related work. Chapter 3 describes our modification scheme in detail. Also, our analytical models and simulation models are described in Chapter 3. Chapter 4 presents the analytical results as well as the simulation results, and compares some of these results to those published in the literature. Chapter 5 concludes our research work.

Chapter 2

RELATED WORK

Several schemes have been suggested to improve TCP performance in wireless networks. Five representative ones are fast retransmission [4], I-TCP (Indirect TCP) [5], snoop [6], local handoff [7], and AIRMAIL (AsymmetrIc Reliable Mobile Access In Link-layer) [8]. The first two schemes attempt to make TCP be aware of mobility, while the others intend to hide mobility from TCP. The following sections summarize these proposals and point out their associated advantages and disadvantages.

2.1 Fast Retransmission Scheme

The aim of the fast retransmission scheme is to alleviate the performance degradation caused by handoffs. Because TCP is not aware of mobility, long waits for retransmission timeouts are required for TCP layer to resume communication after handoffs. Thus, the fast retransmission scheme tries to make the sender of a TCP connection immediately communicate with its receiver as soon as each handoff completes. This is done by having

the MH send triplicate TCP acknowledgment packets to inform the sender[3], or by having the IP layer of the MH signal the TCP layer of the MH if the MH is the sender. Once the sender receives such notifications, it will perform existing fast retransmission procedures: retransmitting the earliest unacknowledged packet, dropping the transmission window, and initiating the slow-start algorithm. Note that in networks which guarantee smooth handoffs, that is, in networks which never lose packets during handoffs, it is not necessary to initiate fast retransmissions.

The main advantage of the fast retransmission scheme is that it uses existing TCP fast retransmission procedures to get good performance improvement, with only simple changes to end hosts. This approach also preserves TCP end-to-end reliability. The main disadvantage is that it does not consider the impact of wireless transmission errors on TCP.

2.2 I-TCP Scheme

Based on the fact that the wired links are faster and more reliable than the wireless links, I-TCP scheme splits an end-to-end TCP connection between an MH and an SH into two separate ones:

- In the wireless part, the MH communicates with its current base station using specialized transport protocol whose flow control and congestion control policies are tuned for wireless links.

[3] According to TCP's fast retransmission algorithm, if three or more duplicate acknowledgments are received in a row, it strongly indicates that a packet has been lost. Thus, TCP has to retransmit the lost packet without waiting for a retransmission timeout [2].

- In the wired part, that base station uses normal TCP to communicate with the SH.

The base station is the center point of an I-TCP connection. Thus, during a handoff, the new base station takes over the I-TCP connection from the old base station and becomes the center point.

The advantage of such an indirect approach is that it confines the mobility-related problems to the wireless link and uses a mobile-aware transport protocol in the wireless part to improve the end-to-end TCP performance. However, I-TCP acknowledgments are not end-to-end. Since a TCP connection is split into two different ones, the sender may receive the acknowledgments of TCP packets even before these packets are actually received by the receiver. In addition, I-TCP scheme requires the base station to copy data from the wireless part of an I-TCP connection to the wired part and vice versa, and its implementation is more complex.

2.3 Snoop Scheme

Snoop scheme also limits the problems caused by high bit error rates to the wireless link, but it mainly modifies IP to improve TCP performance:

- For packet transfer from an SH to an MH, the base station caches unacknowledged TCP packets and performs local retransmissions across the wireless link by monitoring the acknowledgments received from the MH.
- For packet transfer from an MH to an SH, the base station keeps track of lost packets and generates negative acknowledgments for those packets back to the MH. The MH then retransmits the corresponding lost packets.

As to the problems caused by handoffs, snoop assumes that smooth handoffs are provided in wireless networks so that it can use multicast and buffering methods in those base stations that the MH is likely to migrate to.

Advantages of the snoop scheme are that it uses local retransmissions or notification over the wireless link to improve TCP performance, and it preserves TCP end-to-end reliable packet transfer. One disadvantage is that, if more than one packet is lost per window, the congestion control procedures of TCP will limit the improvement provided by snoop. Also, since networks with little or no overlap between cells can reuse the same frequencies in nearby cells and provide more accurate location information, it is unlikely that all cellular networks will provide enough overlapped regions between cells in the near future to insure handoffs complete before an MH loses contact with the old base station.

2.4 Local Handoff Scheme

The purpose of the local handoff scheme is to decrease handoff latency between adjacent cells in the same subnet. It also utilizes buffering and retransmissions at the base stations (for data transfer from an SH to an MH), or at the MH (for data transfer from an MH to an SH), to reduce packet losses during handoffs. The main advantage of the local handoff scheme is that it can extend well to a large number of MHs which move between small cells. The main disadvantage is that it does not consider the problem caused by wireless transmission errors.

2.5 AIRMAIL Scheme

AIRMAIL scheme is designed for the error-prone wireless links. It uses a combination of forward error correction (FEC) and automatic repeat request (ARQ) techniques for loss recovery over the wireless links. The advantage of the AIRMAIL scheme is that it operates independently of TCP and thus it does not maintain any per-connection state for TCP. The disadvantage of the AIRMAIL scheme is that it does not always result in better performance since retransmissions in the link layer may interfere with the end-to-end retransmissions of TCP [9].

Chapter 3

MOBILE-AWARE TCP

As discussed earlier, since TCP can not distinguish network mobility from network congestion, it will use the wrong remedy, that is, congestion control procedures, to control the effects of mobility. One solution to this problem is to avoid the occurrence of mobility-related packet loss and delay so that TCP does not have to be aware of mobility. It can be done by modifying lower-layer protocols like snoop, local handoff, and AIRMAIL do. But in a real network, it is very likely that an MH will temporarily lose contact with any base station during handoffs. In other words, it is unlikely that the system can avoid packet losses and delays during handoffs[4]. Thus, this solution is not perfect since the congestion control procedures of TCP will ultimately affect TCP performance significantly.

The other solution is to let TCP be aware of mobility so that TCP can use the proper remedy to alleviate the impact of mobility, like fast retransmission and I-TCP do.

[4] In analog cellular systems, handoffs are normally triggered by the network. The network measures the signal quality of the MH and keeps track of the candidate channels and cells available to supply a new path. When the signal quality drops below a given threshold, a handoff order will be sent. The MH then switches to the new channel where the network constructs a new path in advance. In general, the handoff process can take 100-200 milliseconds and thus produces a short interruption of the connection [10].

However, indirect approaches such as I-TCP do not provide end-to-end reliable packet transfer. Applications which depend on the end-to-end TCP-layer acknowledgments, for example, Telnet, can not use the indirect method since the sender may receive the acknowledgments of the sent packets before these packets are truly received by the receiver.

Based on the above facts, fast retransmission scheme may be a good method to improve TCP performance in mobile computing environments. But, as mentioned previously, it only considers the impact of handoffs and does not take the impact of wireless transmission errors into consideration. Therefore, we propose another mobile-aware TCP which intends to alleviate the performance degradation caused by handoffs and by wireless transmission errors without sacrificing TCP end-to-end reliable packet transfer. Also, our scheme tries to achieve good performance improvement with only simple changes to the end hosts.

3.1 Proposed Scheme

There are two parts in the proposed scheme: the mobility detection part and the mobility control part. The main idea is to make TCP be able to differentiate mobility-related packet losses and delays from congestion-related ones and then perform certain procedures to control the effects of mobility. In the mobility detection part, we use the signal strength received by an MH to diagnose mobility. The reason for using this method to distinguish mobility is that if an MH is leaving its current cell or, if multipath fadings of the wireless links exist, the received signal strength of the MH will become weaker. In the mobility control part, we suggest three procedures to mitigate the impact of mobility:

- Preserve the transmission window size
- Reduce the packet-sending rate
- Hold the retransmission interval

The design of these mobility alleviation procedures are based on observing how congestion control procedures, in the presence of mobility, degrade TCP performance. When a timeout occurs because of network mobility, congestion control procedures cause TCP to set its slow-start threshold to one half of the current window size, to initiate the slow-start algorithm, and to double the retransmission interval. The first two procedures reduce the amount of data to be sent over the network, and the third not only causes a long pause in communication but also freezes data transmission. Thus, our mobility alleviation procedures do not change the transmission window size and the retransmission timeout. Our procedures, however, do reduce the packet-sending rate in order to avoid congesting the new cells in the case of handoffs. Also, we use fast retransmission to reduce the waits for retransmission timeouts after handoffs.

As an example of the application of the proposed scheme, suppose an MH receives some packets from an SH and moves between non-overlapped cells. When the signal strength received by the MH is lower than a pre-determined threshold, the MH will send a TCP warning packet to the SH. After receiving this warning packet, the SH will stop sending any packets. Since the SH may have sent several packets to the MH before receiving the warning packet, if any one of these sent packets is lost, the SH will consider that the packet loss is due to network mobility and perform mobility alleviation procedures. In the situation where the MH moves to another cell, the MH will send triplicate TCP acknowledgment packets to the SH as soon as the handoff completes.

Once the SH receives such notifications, it will retransmit the earliest unacknowledged packet and initiate mobility alleviation procedures.

3.2 Evaluation

The performance statistics that we are interested in are the total time delay and the average throughput. To see if the proposed scheme is a better solution to improve TCP performance, we first developed a simplified analytical model to calculate and to compare the performance statistics of the proposed scheme, the normal TCP, and the fast retransmission scheme. Next, we developed simulation models of these schemes for further evaluation. Note that in our analytical and simulation models, we consider bulk data transfer from an SH to an MH. In addition, we consider the situation where an MH moves between non-overlapped cells.

The following subsections describe our analytical and simulation models in detail. The analytical and simulation results are presented in the next chapter.

3.2.1 Analytical Model

The data transfer from an SH to an MH can be modeled as a closed queueing network. This type of network is complicated even though some assumptions are made. In addition, attempting to incorporate TCP's sliding window flow control and congestion control mechanisms in the analysis is not easy because not every packet is necessarily acknowledged and, whenever a packet is delayed or lost, the congestion control procedures are initiated and certain variables are reset depending on their values right

before congestion occurs. Therefore, in order to gain an insight into the performance statistics, it is convenient to make some simplifications and assumptions in the analytical model:

- If all packets in a window are transmitted, the source will stop sending any packet until it receives the acknowledgment.
- Acknowledgment is withheld by the receiver until the last packet in a window is received. This acknowledgment acknowledges all packets in the window.
- Acknowledgment is received by the source right after the last packet in the window reaches the destination.
- All links have the same service rate, which is equal to the service rate of the bottleneck (slowest) link.
- Each node is modeled as an independent M/M/1 queue (Poisson arrivals, exponential service statistics, and one server).

The basic model used in our analysis is shown in Figure 1 [11]. The number of hops M is counted including the source node but excluding the destination node. The rates λ and μ represent the offered load and the service rate respectively. A counter C is assumed

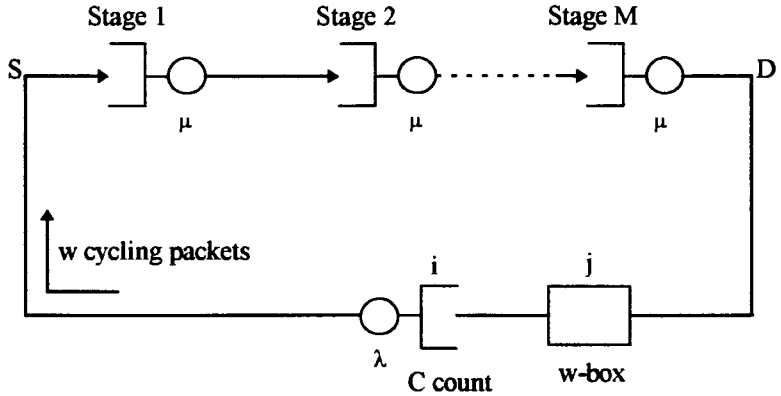

Figure 1. Sliding window control model with acknowledgment at end of window [11]

to be maintained at the source. It is initially set to w (the window size) and is decremented by one whenever a packet is transmitted. If the counter C reaches zero, it means that there are w packets in the network, so the source will stop sending any new packets. The w-box shown is assumed to be kept at the destination. It is initially set to zero and is incremented by one whenever a packet reaches the destination. It stores up to w-1 packets and then on the arrival of the w-th packet, it will reset its value to zero and reinitialize the counter C back to w.

The model is solved approximately in [11] when the offered load λ approaches infinity. It is the case we consider since we model bulk data transfer. The throughput γ, the time delay E(T), and the average number of packets E(n) are given by

$$\gamma = \mu * w / [w + (M - 1) * Tw]$$

$$\text{where } Tw = 1 + 1/2 + 1/3 + \ldots + 1/w \qquad (3.1)$$

$$E(T) = [M - 1 + (1 + w) / 2] / \mu \qquad (3.2)$$

$$E(n) = \gamma * E(T) \qquad (3.3)$$

But, in order to calculate the total time delay and the average throughput of each scheme, some adjustments are still needed. For example, for each scheme pauses in communication have to be assumed, and the window size w has to be increased and decreased dynamically. Based on these adjustments, the total time delay can be calculated by adding up the following two types of delay:

- The E(T)s of different window sizes
- The pauses in communication due to packet losses

and the average throughput can be found by the following two steps:

1. Add up the E(n)s of different window sizes. This will be the total number of packets.

2. Divide the total number of packets calculated in Step 1 by the total time delay. This final value will be the average throughput.

Note that we do not directly use (3.1) to calculate the average throughput because it may not correctly find the different throughput losses caused by different pauses in communication. In other words, using (3.1) may produce the same average throughput for the normal TCP and the fast retransmission scheme – something that is not true.

As an example, suppose:

- The packet size is 1000 bytes.
- The maximum window size is 4000 bytes (4 packets).
- The number of hops M is 2.
- The service rate μ is 1600000 bits per second (200 packets per second).
- Handoff happens every 8 seconds.
- Beacon period is 1 second[5].
- The minimum retransmission timeout is 1 second.
- An SH sends 2000000 bytes (2000 packets) of data to an MH.
- All packet losses are caused by handoffs.

and we want to find the total time delay and the average throughput of the normal TCP. According to TCP's slow-start algorithm, the window size w will start at one packet, then two, then four, and stay at four if no packet is lost. Thus, using (3.1), (3.2), and (3.3), the throughput γ, the time delay E(T), and the average number of packets E(n) at w = 1 are

$$\gamma = 200 * 1 / [1 + (2 - 1) * 1] = 100 \quad \text{packets/second}$$
$$E(T) = [2 - 1 + (1 + 1) / 2] / 200 = 0.01 \quad \text{seconds}$$

[5] A beacon is an advertisement message that broadcasts the information of a cell. If the beacon of a cell is broadcasted every one second, then the beacon period of that cell is one second.

$$E(n) = 100 * 0.01 = 1 \quad \text{packet}$$

At this stage, the total data sent, the total time delay, and the total number of packets are

$$\text{total data sent} = 1000 \quad \text{bytes}$$
$$\text{total time delay} = 0.01 \quad \text{seconds}$$
$$\text{total number of packets} = 1 \quad \text{packet}$$

When w becomes two:

$$\gamma = 200 * 2 / [2 + (2 - 1) * (1 + 1 / 2)]$$
$$= 114.285714 \quad \text{packets/second}$$
$$E(T) = [2 - 1 + (1 + 2) / 2] / 200 = 0.0125 \quad \text{seconds}$$
$$E(n) = 114.285714 * 0.0125 = 1.428571 \quad \text{packets}$$
$$\text{total data sent} = 1000 + 2000 = 3000 \quad \text{bytes}$$
$$\text{total time delay} = 0.01 + 0.0125 = 0.0225 \quad \text{seconds}$$
$$\text{total number of packets} = 1 + 1.428571 = 2.428571 \quad \text{packets}$$

When w becomes four and stays at four until the first handoff occurs:

Window Size (packet)	γ (packets/second)	E(T) (seconds)	E(n) (packets)	
4	131.506849	0.0175	2.301370	┐
4	131.506849	0.0175	2.301370	├ 455 rows
:	:	:	:	┘
4	131.506849	0.0175	2.301370	

$$\text{total data sent} = 3000 + 4000 * 455 = 1823000 \quad \text{bytes}$$
$$\text{total time delay} = 0.0225 + 0.0175 * 455 = 7.985 \quad \text{seconds}$$
$$\text{total number of packets} = 2.428571 + 2.301370 * 455 = 1049.551921 \quad \text{packets}$$

During the transmission of the next four packets, since handoff occurs and the MH moves between non-overlapped cells, some of the packets will be lost. If we assume that all four packets are lost (the worst case scenario), the total data sent and the total number of packets will be the same as those calculated before the handoff occurs:

$$\text{total data sent} = 1823000 + 0 = 1823000 \quad \text{bytes}$$

total number of packets = 1049.551921 + 0 = 1049.551921 packets

If we also assume that the MH receives a beacon from the new cell one second after leaving the old cell (the worst case scenario), the retransmitted packet sent after the first retransmission timeout will be lost as well. In other words, there will be two retransmission timeouts: the first one will be one second but the second one will be two seconds (TCP's exponential backoff). Thus, the total time delay after the handoff will be

total time delay = 7.985 + 1 + 2 = 10.985 seconds

When the first handoff completes, based on TCP's congestion control procedures, the slow-start threshold will be set to two packets[6], and the window size w will start at one packet again, then two, then three, then four, and stay at four until next handoff occurs:

Window Size (packet)	γ (packets/second)	E(T) (seconds)	E(n) (packets)	
1	100	0.01	1.000000	
2	114.285714	0.0125	1.428571	
3	124.137931	0.015	1.862069	
4	131.506849	0.0175	2.301370	┐
:	:	:	:	├ 42 rows
4	131.506849	0.0175	2.301370	┘
3	124.137931	0.015	1.862069	

total data sent = 1823000 + 1000 + 2000 + 3000 * 2 + 4000 * 42

= 2000000 bytes

total time delay = 10.985 + 0.01 + 0.0125 + 0.015 * 2 + 0.0175 * 42

= 11.7725 seconds

total number of packets = 1049.551921 + 1 + 1.428571 + 1.862069 * 2 + 2.301370 * 42

= 1152.362170 packets

[6] Although there are two retransmission timeouts, which means that the slow-start threshold should be set to one packet, the minimum value of the slow-start threshold is two packets [2].

Note that after sending out two packets, the window size w becomes three instead of four. The reason is that because the slow-start threshold is set to two, the window size w, when reaching that value, can only be increased by at most one packet [2]. As can be seen, since all data have been sent to the MH before the second handoff occurs, the average throughput can thus be found:

$$\begin{aligned} \text{average throughput} &= 1152.362170 / 11.7725 \\ &= 97.885935 \quad \text{packets/second} \\ &= 783087.480 \quad \text{bits/second} \end{aligned}$$

The code for the analytical calculation is shown in the Appendix. Note that since the usefulness of the analytical model is to gain an insight into the evaluation result, we only consider handoff-related packet losses and delays in our analytical model. Error-related packet losses are considered in the simulation models.

3.2.2 Simulation Models

We use OPtimized Network Engineering Tools (OPNET) [12] to construct our simulation models. OPNET is a window-based simulation package with a graphical user interface. It's modeling structure is depicted in Figure 2. As can be seen, OPNET modeling is hierarchical. Thus, in order to simulate wireless communication networks with detailed protocol modeling, we need to build a few models: the topology of a communication network is modeled at the network level, and the layered protocols such as TCP and IP are implemented at the node level and the process level.

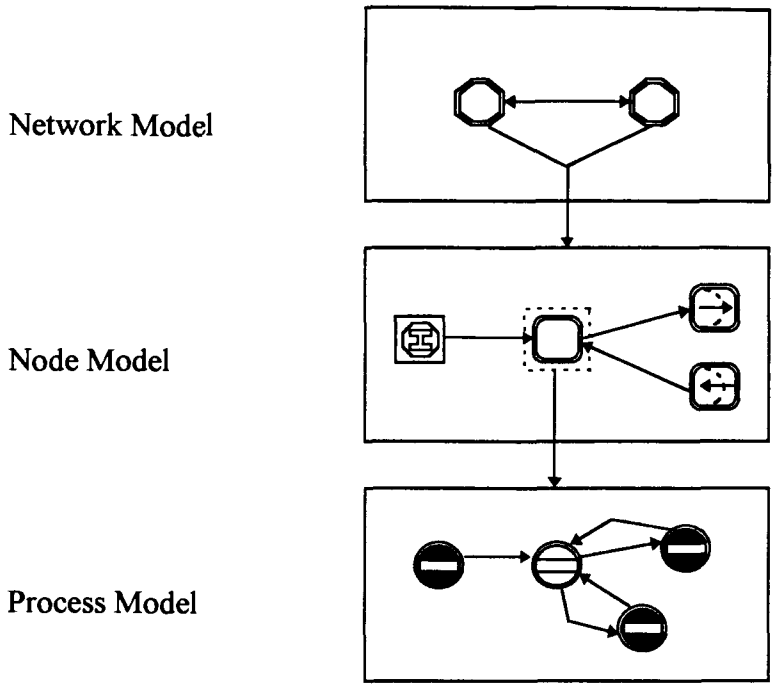

Figure 2. OPNET modeling structure

A. Network Configuration

The network model developed for our study is shown in Figure 3. This model considers the situation where the *MH*, a member of the *HA* (Home Agent), is away from home. The *SH*, the *HA*, and the *FA*s (Foreign Agents[7]) are connected to a 10-Megabits/second (Mbps) Ethernet. The *MH*s, the *HA*, and the *FA*s are connected to a 2-Mbps wireless local area network. Cells defined in the wireless network cover an indoor environment, and the error modality is multipath interference. We make the *MH* move periodically between the cells defined by *FA_1* and *FA_2* so that we can explore the effects of handoffs. We also use two jammer nodes (*MH1* and *MH2*) to introduce radio channel interference in both cells in order to investigate the impact of wireless transmission errors.

[7] A router that assists a locally reachable MH that is away from its home network [13].

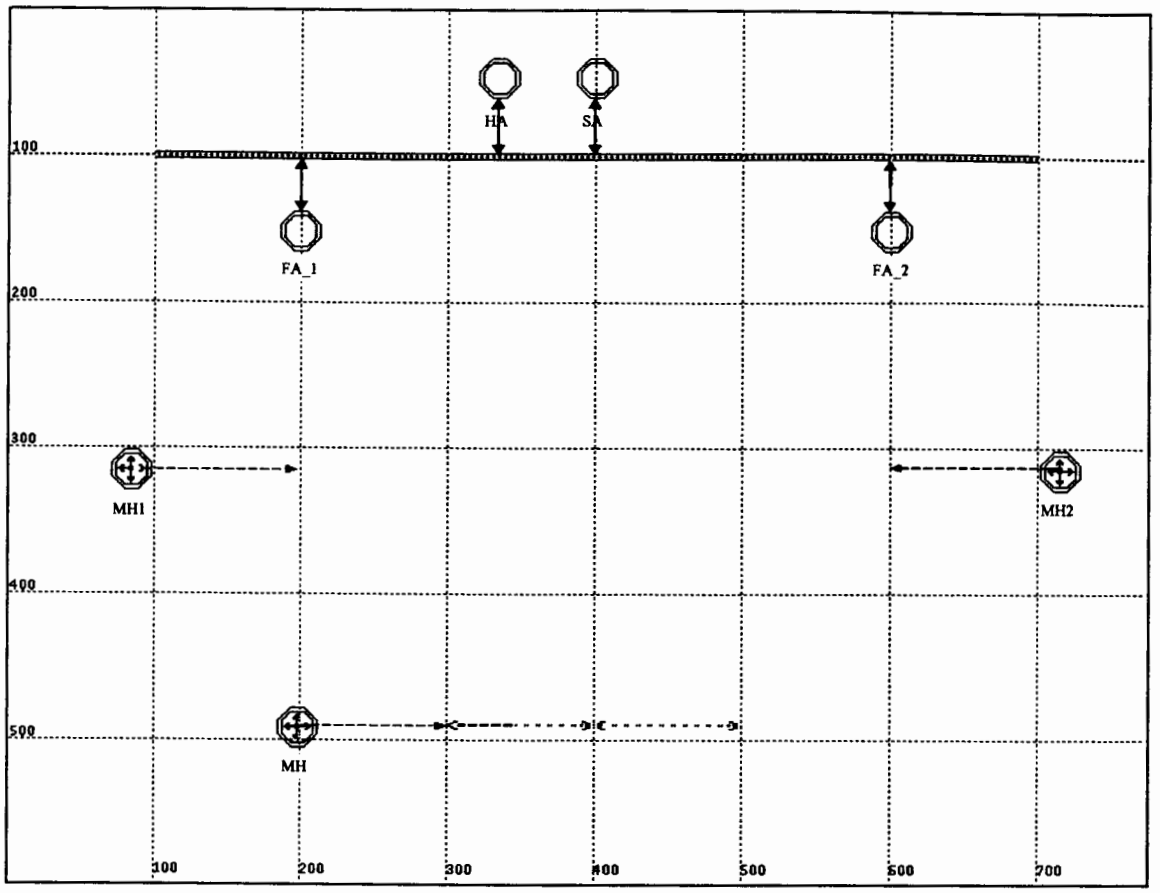

Figure 3. The network model

Mobility support in our model is provided by Mobile IP [13]. Each agent, either an *HA* or an *FA*, broadcasts a beacon periodically and is responsible for its home *MH*s and the visiting *MH*s. In the model shown in Figure 3, packets originated from the *MH* are sent via the *FA* (either *FA_1* or *FA_2* depending on the cell that the *MH* is in). However, packets destined to the *MH* are first routed to the *HA*, redirected to the *FA* (either *FA_1* or *FA_2* depending on the cell that the *MH* is in), and finally delivered to the *MH*.

B. Layered Architecture

Four different types of node model were developed for our network. They are described as follows.

1. Stationary Workstation (*SH*)

This workstation model and the associated process models are provided by OPNET. As can be seen in Figure 4, this node model consists of several modules that implement various protocols of the TCP/IP protocol stack. Two network connections, *app1* and *app2*, can be established concurrently. Outgoing packets generated by *app1* and *app2* are passed downward through the TCP layer (*tcp*), the IP layer (*ip_encap* and *ip*), and the *arp* module to the Ethernet (*mac*, *defer*, *bus_tx*, and *bus_rx*). Incoming packets are passed upward through the protocol stack in a similar manner.

The modules *app1* and *app2* only act as simple data generators and receivers. They utilize the *tcp* module below to communicate reliably with other application

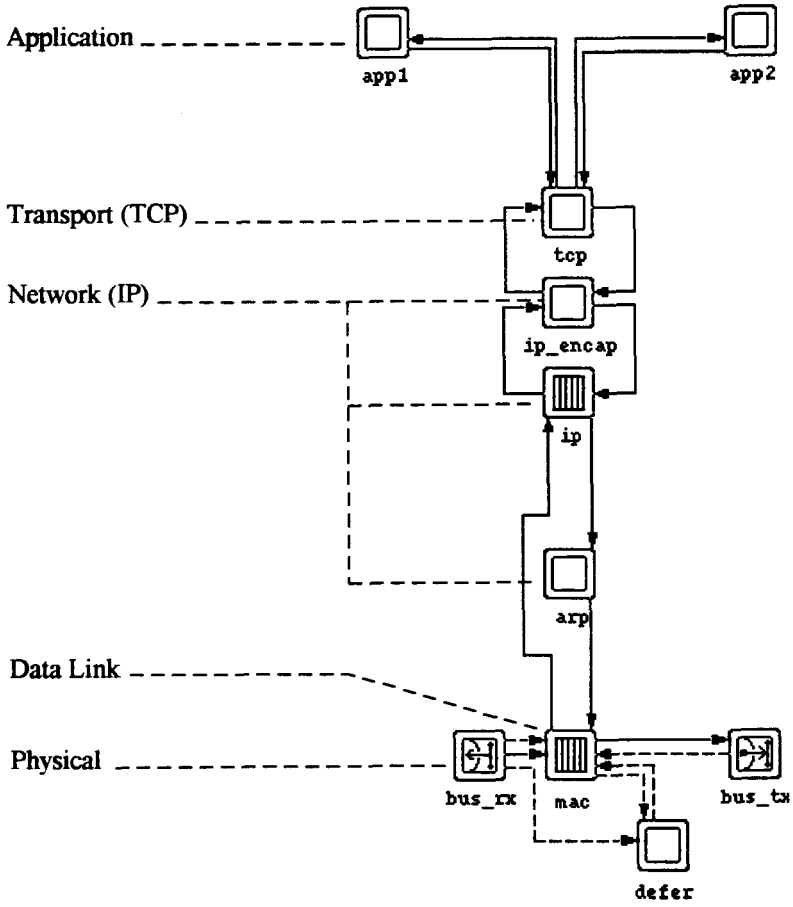

Figure 4. The node model of the stationary workstation

modules in the network.

The *tcp* module is based on the specifications of TCP in Request for Comment (RFC) 793 and RFC 1122. Features incorporated in this module include:

- Three-way handshake connection establishment and termination
- End-to-end reliable packet transfer based on acknowledgments and retransmissions
- Reordering out-of-sequence data
- Sliding-window flow control and slow-start algorithm
- Congestion avoidance and control
- RTT measurement
- Persistence timeout
- TCP's urgent mode

Features not implemented in this module are:

- TCP quiet time concept which avoids a TCP connection with the same local and foreign IP addresses and port numbers to be established too soon after rebooting.
- TCP checksum which discards any packet being modified in transit

Since rebooting is irrelevant to our simulation and packet losses due to invalid TCP checksum is rare (<< 1%) [2], omitting the two TCP features will not significantly affect our simulation results. Note that proposed TCP modifications such as fast retransmit/fast recovery and timestamp option are not implemented in this module.

The role of the *ip_encap* module is to encapsulate higher layer data into IP datagrams and then to send these datagrams to the *ip* queue module for routing. When

the *ip* queue module accepts datagrams from the *ip_encap* module, it will hold them for a processing delay and then send them into the network. For datagrams arriving from the network, the *ip* queue module performs reassembly of the fragmented datagrams and forwards complete datagrams to the *ip_encap* module, which then decapsulates these datagrams and forwards them to TCP.

The *arp* module provides a mapping between the IP addresses and the Ethernet addresses. This module is necessary since the IP layer and the Ethernet use different forms of address: the address of an IP datagram is 32-bit while that of an Ethernet frame is 48-bit.

The remaining modules, *mac*, *defer*, *bus_tx*, and *bus_rx*, compose the Ethernet interface of the workstation: the *mac* module performs the media access protocol, the *defer* module performs the carrier sensing function for the *mac* module, and the *bus_rx* and the *bus_tx* modules provide physical access to the bus link.

2. Mobile Workstation (*MH*)

Figure 5 displays the node model of our mobile workstation. It is identical to that of the *SH* node model except:

- A mobility support module, *micp*, is added to the IP layer.
- A wireless Ethernet interface is used in the lower layers.
- A *pointer* module is added.

The *micp* module performs the registration procedures described in Mobile IP. When the *MH* moves to a new cell, the *micp* module detects a beacon in the new cell, sends a *registration request* message to the new serving *FA*, and updates the routing table of the *MH* to make the new serving *FA* be the default router of the *MH*. It then

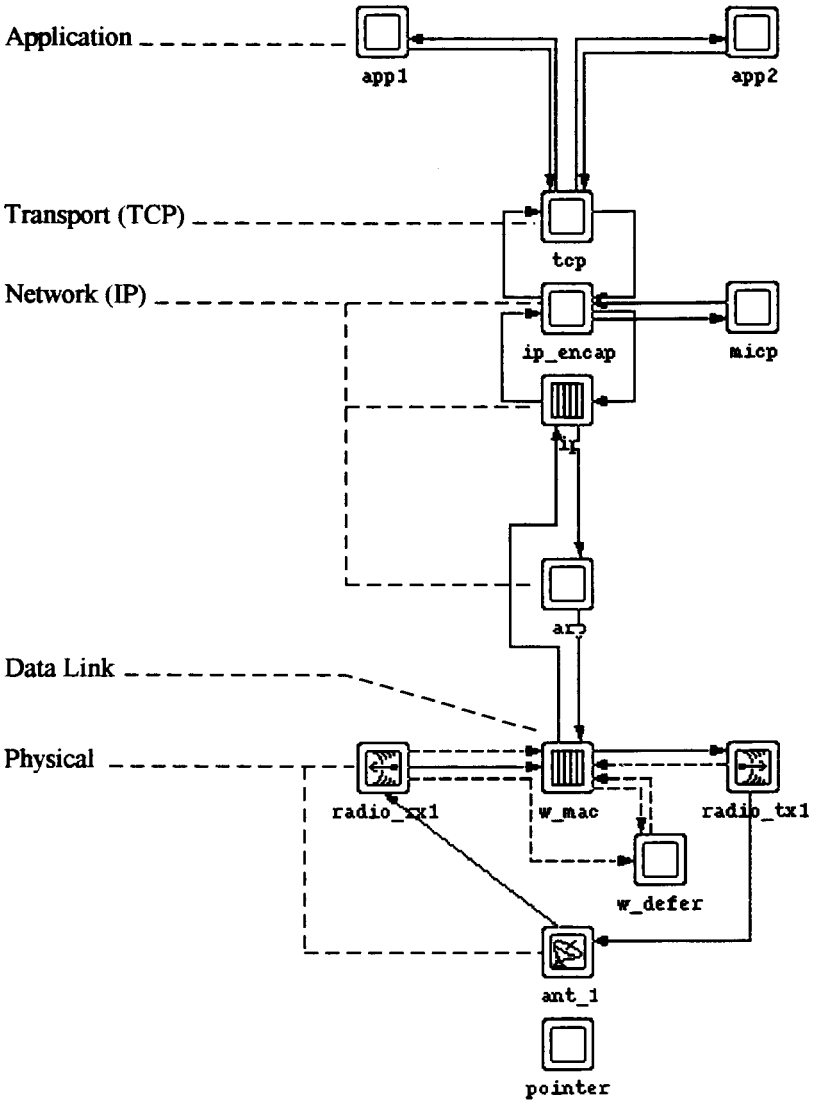

Figure 5. The node model of the mobile workstation

waits for a *registration reply* message from the new serving *FA* to confirm the completion of the handoff. If any of the registration messages is lost, the *micp* module will keep greeting whenever it detects a beacon until a *registration reply* message is received.

The *w_mac*, *w_defer*, *radio_tx1*, *radio_rx1*, and *ant_1* modules constitutes the wireless Ethernet interface of the mobile workstation. This wireless Ethernet interface, like the wired one, also uses the carrier-sense multiple access with collision detection

(CSMA/CD) protocol. But, instead of using the *bus_tx* and the *bus_rx* modules, it uses the *radio_tx1*, *radio_rx1*, and *ant_1* modules to provide the physical access to the radio link. The *radio_tx1*, *radio_rx1*, and *ant_1* modules are built-in OPNET node objects. Radio-link characteristics such as modulation method and link bandwidth can be specified via object attributes menus.

The purpose of the *pointer* module is to let the mobile workstation listen to the different broadcast channels in different cells.

3. Gateway (*HA* or *FA*)

The structure of a gateway node model is shown in Figure 6. As can be seen, this model does not have the application and the TCP layers because its responsibility is to route packets, not to generate or to receive packets. In addition, since the gateway nodes in our network are connected to both the wired and the wireless networks, this

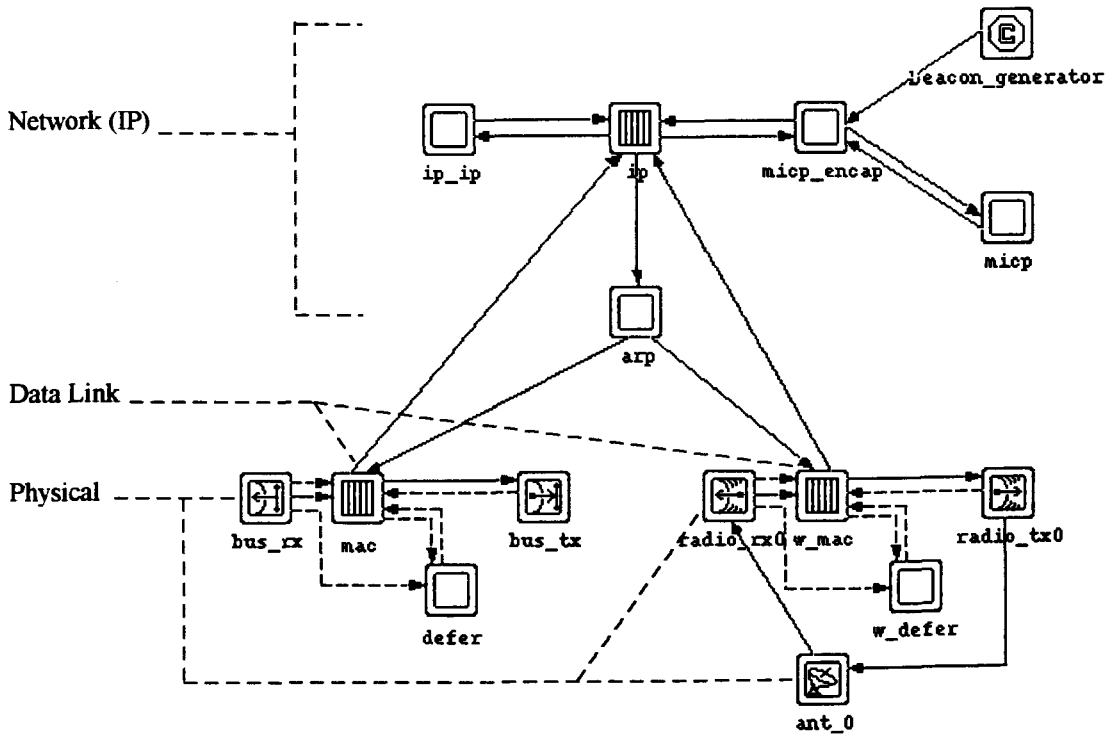

Figure 6. The gateway node model

model has two interfaces: a wired and a wireless Ethernet interfaces.

The *beacon_generator*, *micp*, *micp_encap*, and *ip_ip* modules are used for IP mobility support. They perform different functions specified in Mobile IP:

- The *beacon_generator* module periodically broadcasts a beacon.
- The *micp* module performs different registration procedures depending on the role of the gateway. For a serving *FA*, if a *registration request* message is received from the *MH*, the *micp* module will relay the message to the *HA* and add the *MH* to its routing table; once a *registration reply* message is received from the *HA*, the *micp* module will relay the message to the *MH*. For an *HA*, if a *registration request* message is received from the serving *FA*, the *micp* module will send a *registration reply* message to the serving *FA* and update its routing table.
- The *micp_encap* module encapsulates the messages received from the *beacon_generator* and the *micp* modules into IP datagrams and then sends these datagrams to the *ip* queue module for routing. For datagrams received from the *ip* queue module, the *micp_encap* module will decapsulate them and forward the messages to the *micp* module.
- The *ip_ip* module of an *HA* encapsulates IP datagrams received for the *MH* into other IP datagrams and forwards them to the serving *FA*. The *ip_ip* module of the serving *FA* decapsulates IP datagrams received for the *MH* and delivers the inner IP datagrams to the *MH*.

As to the remaining modules, they are the same as those used in the stationary and the mobile workstations.

4. Jammer (*MH1* or *MH2*)

Our jammer node model is depicted in Figure 7. Jammer packets generated by the modules *source1* and *source2* are passed to the radio link (*tx1*, *tx2*, and *ant*) in order to affect the radio receiver of the *MH* and that of the *FA* (either *FA_1* or *FA_2* depending on the cell that the jammer node is in). All modules in this node model are built-in OPNET node objects. Jammer-packet and radio-link characteristics can be specified via objects menus. Note that this node model does not implement layered protocols such as TCP and IP because it is simply used to create radio interference.

Figure 8 shows the process model that is used in the *tcp* module of the above workstation nodes. It uses a state transition diagram with embedded C code to handle the activities of the TCP connections. It also includes a child process model to implement the functionality of the normal TCP. The process starts from the *init* state where all necessary initializations are performed. It then goes to the *active* state to wait for the commands and indications. When a command or an indication arrives, the process will make a transition from the *active* state to the appropriate state, perform the task specified in that state, and then return to the *active* state to wait for the next command or indication. The states *OPEN*, *SEND*, *RECEIVE*, *CLOSE*, and *ABORT* perform "connection open", "data pass",

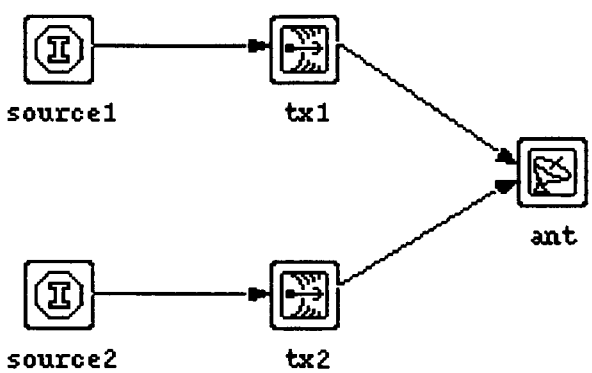

Figure 7. The jammer node model

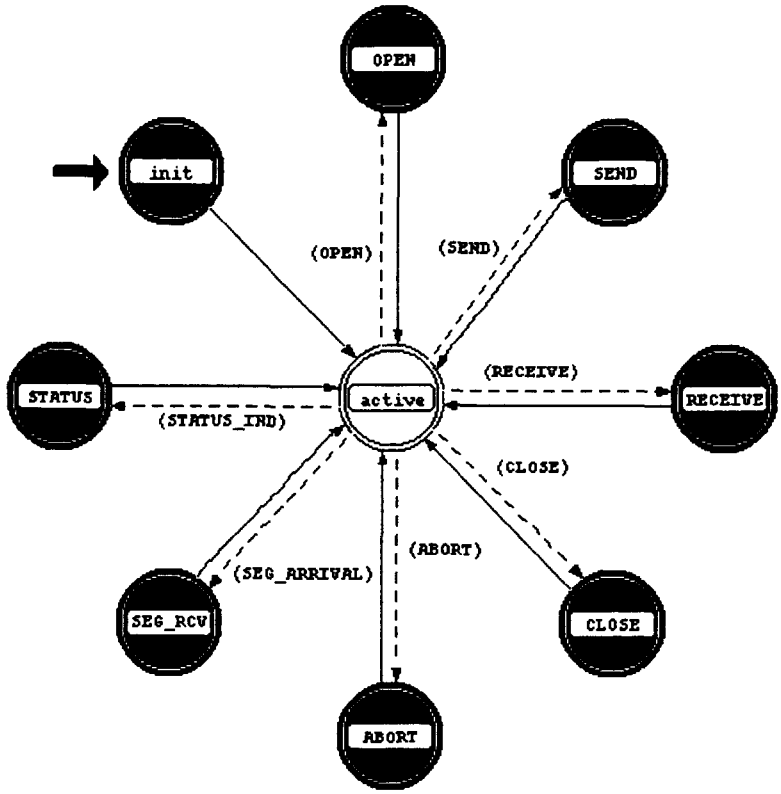

Figure 8. The TCP process model

"data receive", "connection close", and "connection abort" for the application commands of the same name. The state *SEG_RCV* performs appropriate TCP mechanisms to process packets received from the network. The state *STATUS* processes the indications received from connection processes.

In order to implement the fast retransmission and the proposed schemes, the above TCP process model was modified. Figure 9 and Figure 10 display the process models of these two schemes respectively. For the fast retransmission scheme, we added one state called *FAST_RE* and modified the included child process model so that when handoff completes, if the *MH* is the receiver:

- The process in the *MH* will move to the FAST_RE state to generate and to send triplicate acknowledgment packets to the *SH*.

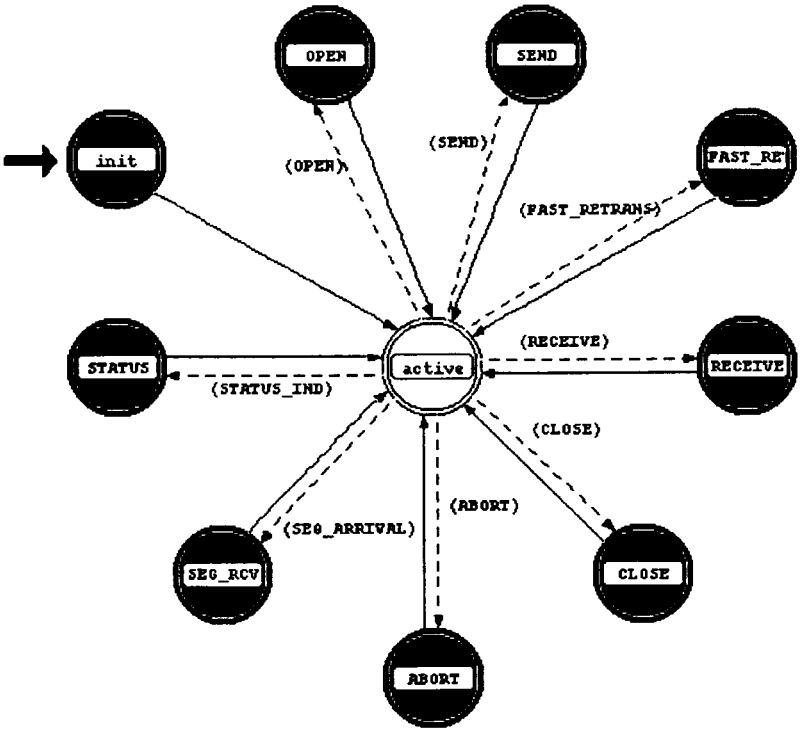

Figure 9. The process model of the fast retransmission scheme

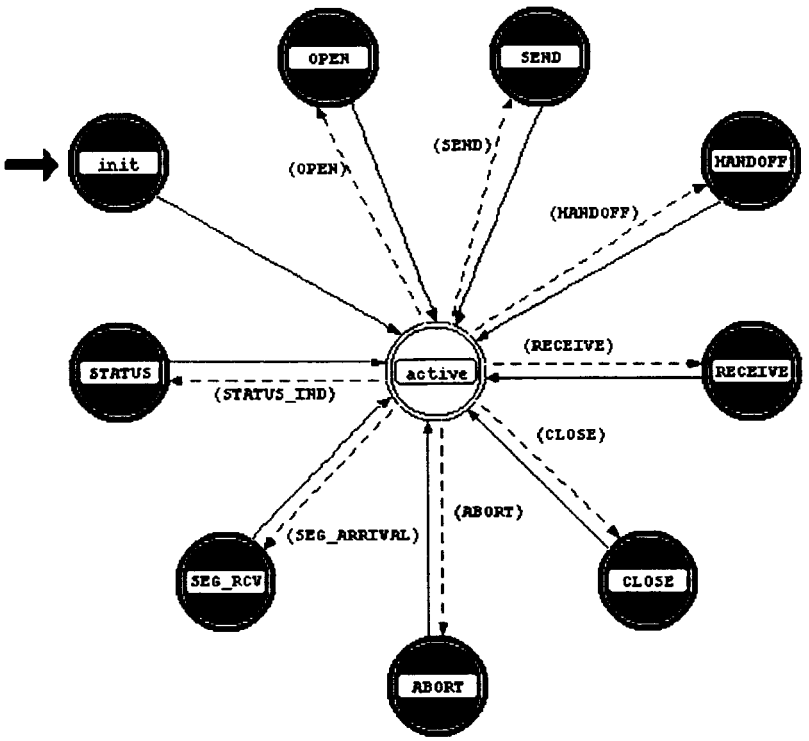

Figure 10. The process model of the proposed scheme

- Once the process in the *SH* receives these acknowledgment packets, it will perform fast retransmission procedures: retransmitting the earliest unacknowledged packet, dropping the transmission window, and initiating the slow-start algorithm.

If the *MH* is the sender, the process in the *MH* will move to the *FAST_RE* state to perform fast retransmission procedures. We also modified the *micp* module in the *MH* so that when the *micp* module receives a *registration reply* message indicating the completion of a handoff, it will signal the *tcp* module.

For the proposed scheme, we added a state called *HANDOFF* in the TCP process model and modified the included child process model so that when the signal strength received by the *MH* is lower than a pre-determined threshold, the process in the *MH* will move to the *HANDOFF* state to generate a warning message, to send this warning message to the *SH*, and to turn on its mobility indicator. When the process in the *SH* receives the warning message, it will turn on its mobility indicator. Once a retransmission timeout occurs, the process in either the *MH* or the *SH* will perform mobility alleviation procedures: preserving the transmission window size, reducing the packet-sending rate, and holding the retransmission interval. We also modified the *w_mac* module in the *MH* so that if the signal strength received is lower than a pre-determined value, the *w_mac* module will signal the *tcp* module. Since the proposed scheme uses fast retransmission as well, modifications made for the fast retransmission scheme are included in the *HANDOFF* state and the relevant process models.

Chapter 4

RESULTS

In this chapter, we present the analytical and the simulation results of the normal TCP, the fast retransmission scheme, and the proposed scheme. We also compare some of the results with those published in the literature.

4.1 Analytical Results

The values of the parameters we considered are as follows:

- The packet size is 536 bytes (default value in most TCP implementations).
- The maximum window size is 4096 bytes (default value in most TCP implementations).
- The number of hops M is 3 (the size of our simulated network).
- The service rate μ is 2 Mbps (used in WaveLAN radio network).
- Handoff happens every 8 seconds (according to the literature [4]).
- Beacon period is 1 second (according to the literature [4]).
- The minimum retransmission timeout is 1 second (according to the literature [4]).

- The SH sends 4 Mbytes of data to the MH (according to the literature [4]).
- The time between receiving a beacon and performing fast retransmission is 0.2 second (according to the literature [4] for the calculations of the fast retransmission and the proposed schemes).

These values are the ones we used for or obtained from our simulations. In addition, we assume that all packets in a window during handoffs are lost, and that the MH receives a beacon from the new cell one second after leaving the old cell.

The throughput performances of the no-move TCP, the normal TCP, the fast retransmission scheme, and the proposed scheme are shown in Figure 11. As can be seen, if we compare the normal TCP with the proposed scheme, throughput can significantly improve from 0.808098 Mbps to 1.077282 Mbps. The main reason for this significant improvement is that the proposed scheme continues its data transfer as soon as a handoff completes, but the normal TCP has to wait for a longer retransmission timeout before retransmitting. In addition, the proposed scheme only has one handoff occurs (total time

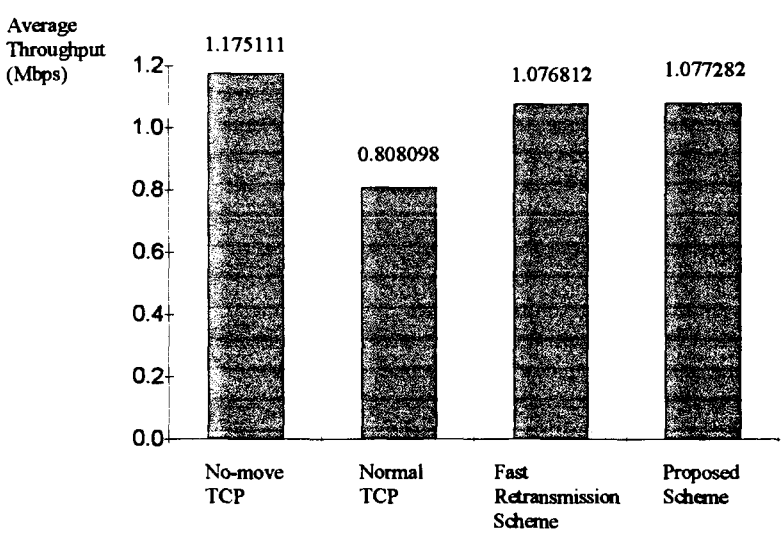

Figure 11. Throughput performances of the no-move TCP, the normal TCP, the fast retransmission scheme, and the proposed scheme

delay = 14.452800 seconds), while the normal TCP has two handoffs occur (total time delay = 19.284960 seconds). If we compare the fast retransmission scheme with the proposed scheme, throughput can only improve slightly: from 1.076812 Mbps to 1.077282 Mbps. The reason for the similar performances is that both schemes use fast retransmission to reduce the long wait for a retransmission timeout. However, since the proposed scheme uses mobility alleviation procedures which do not drop the transmission window size nor initiate the slow-start algorithm, it still performs slightly better than the fast retransmission scheme. As a result, the above performance comparisons tell us that the proposed scheme can perform better than the other two schemes.

4.2 Simulation Results

In our simulations, we performed data transfer over one TCP connection from the *SH* to the *MH*. The values of the parameters we considered, in addition to those described in the previous section, are as follows:

- The maximum acknowledgment delay is 0.01 second.
- The initial retransmission timeout is 3.0 seconds.
- The maximum retransmission timeout is 60 seconds (default value in most TCP implementations).
- The RTT gain is 0.125 (default value in most TCP implementations).
- The deviation gain is 0.25 (default value in most TCP implementations).
- The RTT deviation coefficient is 4.0 (default value in most TCP implementations).
- The persistence timeout is 1.0 second.

Most of the values of the TCP parameters are the default values used in the actual TCP implementations [2]; however, since the wireless link used in our simulations is quite fast (2 Mbps), parameters such as the maximum acknowledgment delay, the initial retransmission timeout, and the persistence timeout are shorter than the default values in order to avoid unnecessary waits for sending data over the TCP connection. Also, parameters such as the handoff interval, the beacon period, and the minimum retransmission timeout are chosen according to those published in the literature [4].

We first ran our simulations under the no-error environment. This environment was generated by disabling the jammer nodes. For each of the schemes, we performed ten simulation runs, and the results are summarized in Table 1. From this table, we can see

Scheme Name	Throughput (Mbps)	Standard Deviation (Mbps)
Normal TCP	0.742783 ± 0.000213	0.000297
Fast Retransmission Scheme	1.024693 ± 0.000631	0.000883
Proposed Scheme	1.029711 ± 0.001232	0.001723

Table 1. Throughput performances of the normal TCP, the fast retransmission scheme, and the proposed scheme in the no-error environment (with 95% confidence interval)

that the proposed scheme performs significantly better than the normal TCP, and that it only performs slightly better than the fast retransmission scheme. These comparisons can also be observed from Figure 12 and Figure 13, which depict the behaviors of the TCP sequence number and congestion window of each scheme respectively for one of our simulation runs. In both figures, the empty regions with or without a dot inside reflect the

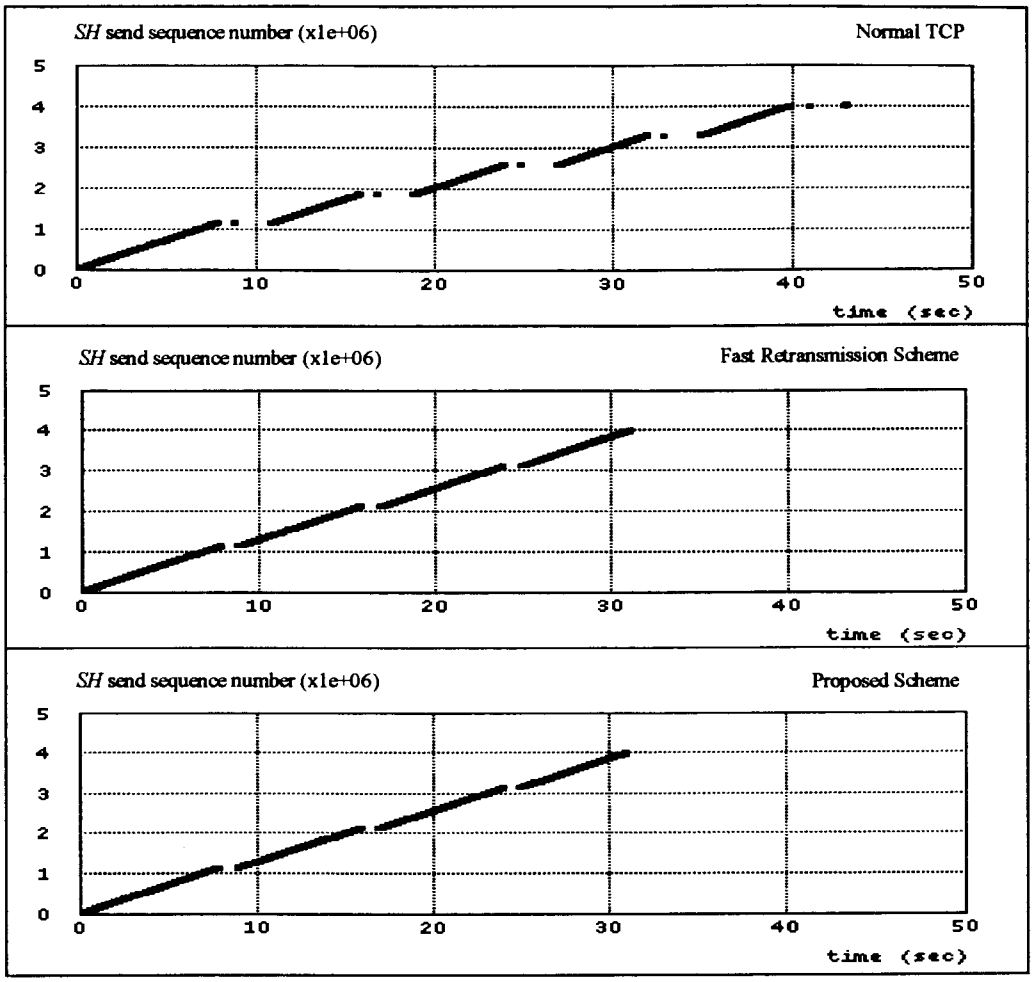

Figure 12. The behaviors of the TCP sequence number of the normal TCP, the fast retransmission scheme, and the proposed scheme in the no-error environment

effects of handoffs. Because packets are lost within these regions, the sender's TCP stops sending any packets, waits for the retransmission timeouts, and then retransmits the lost packets and new packets. For the normal TCP scheme, since the retransmitted packets (the dots within the empty regions) are also lost, the sender's TCP has to wait for longer retransmission timeouts in order to continue its packet transfer. Thus, as shown in Figure 12 and Figure 13, since the normal TCP spends about 10 more seconds to transmit 4 Mbytes of data than the fast retransmission and the proposed schemes, its throughput will

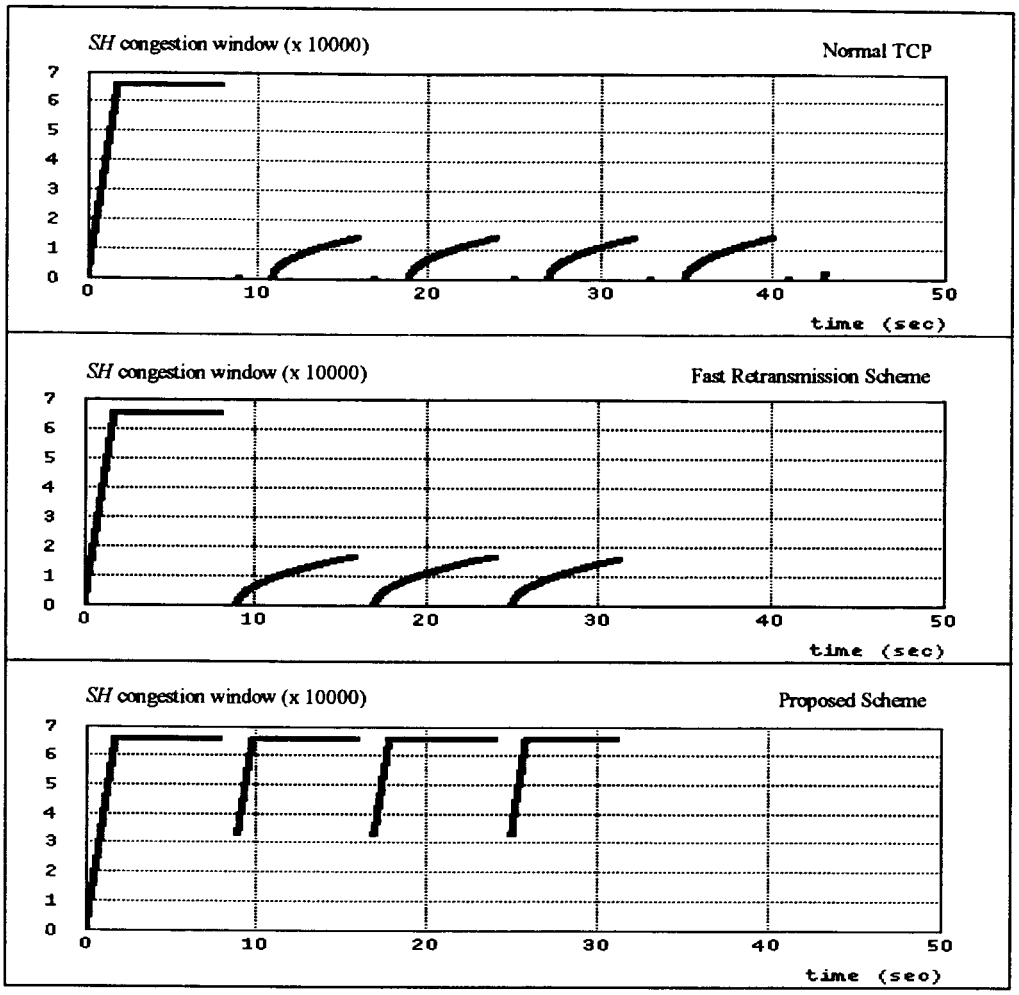

Figure 13. The behaviors of the TCP congestion window of the normal TCP, the fast retransmission scheme, and the proposed scheme in the no-error environment

be significantly less than those of the other two schemes. The performance of the fast retransmission and the proposed schemes seem to be the same (they both spend about 31.5 seconds to transmit 4 Mbytes of data). But, from Figure 13, we can see that the congestion window of the proposed scheme, unlike that of the fast retransmission scheme, does not start from the origin after handoffs (because of mobility alleviation procedures). It means that more data can be sent through the network after handoffs complete. Thus, the proposed scheme requires less time to complete 4-Mbyte data transfer than the fast

retransmission scheme, and its throughput is slightly higher than that of the fast retransmission scheme.

Next, we ran our simulations under the error environment. This error environment was generated by making the jammer nodes periodically send fixed-sized jammer packets to the wireless link. In order to simulate the burst-error situation, the fixed-sized jammer packets were chosen to be 1000 bit. Also, since wireless transmission errors may occur more frequently than handoffs in a real environment, the period of sending jammer packets was chosen to be 2 seconds (which is shorter than the 8-second handoff interval). Again, we performed ten simulation runs for each of the schemes. These results are summarized in Table 2. In this more realistic environment, the throughput of the proposed scheme is

Scheme Name	Throughput (Mbps)	Standard Deviation (Mbps)	Bit Error Rate
Normal TCP	0.556677 ± 0.055184	0.077148	7.152947×10^{-5}
Fast Retransmission Scheme	0.676479 ± 0.025146	0.035154	1.048137×10^{-4}
Proposed Scheme	0.756661 ± 0.038687	0.054084	7.176836×10^{-5}

Table 2. Throughput performances and bit error rates of the normal TCP, the fast retransmission scheme, and the proposed scheme in the error environment (with 95% confidence interval)

still significantly higher than that of the normal TCP (about 36% higher), and it is about 12% higher than that of the fast retransmission scheme. In addition, the bit error rate[8] of the proposed scheme is about the same as that of the normal TCP, but it is lower than that of the fast retransmission scheme.

[8] The bit error rate represents the average bit error rate of the channel measured at the *MH*'s radio receiver. It is computed by dividing the total number of bit errors by the total number of bits that arrived at the channel (both with and without errors) during the transmission of the 4-Mbyte data.

If we look at Figure 14 and Figure 15, which display the behaviors of the TCP sequence number and congestion window of each scheme respectively for one of our simulation runs, we can see why the proposed scheme has better performance than the normal TCP and the fast retransmission scheme. The pauses in these two figures reflect either the effects of handoffs (at the 8th, 16th, 24th, 32th, 40th, and 48th second) or the effects of wireless transmission errors. As shown, because the normal TCP has several long pauses (the empty regions with a dot inside) while the other two schemes only have short pauses, it needs more time to complete 4-Mbyte data transfer and thus it has the

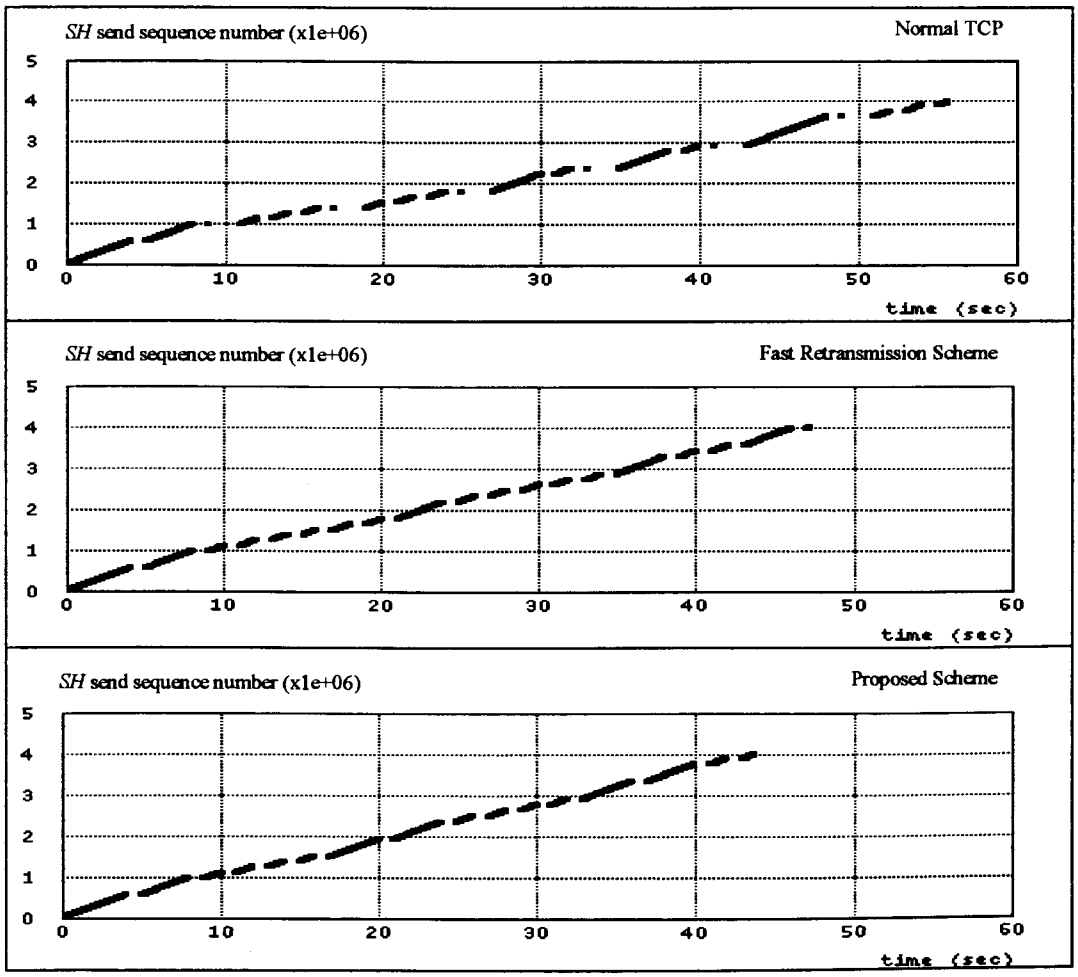

Figure 14. The behaviors of the TCP sequence number of the normal TCP, the fast retransmission scheme, and the proposed scheme in the error environment

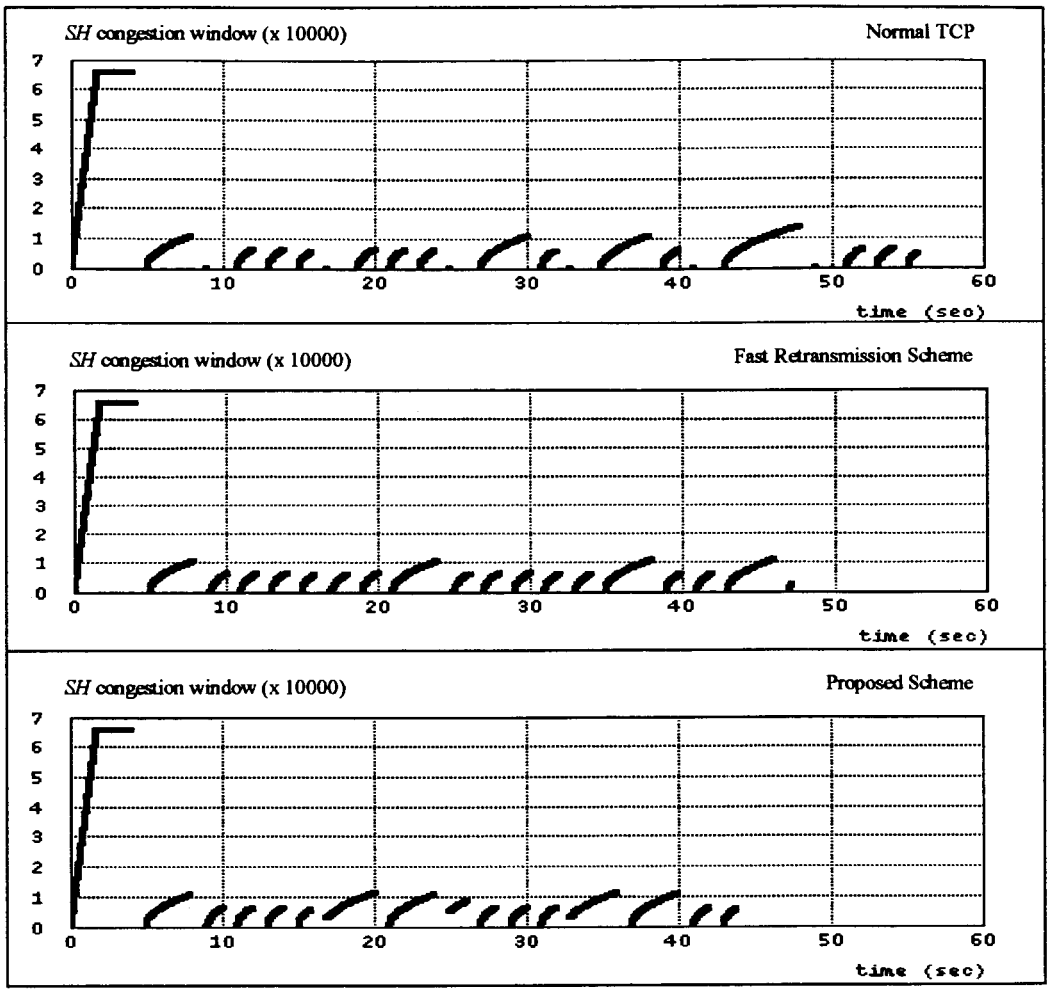

Figure 15. The behaviors of the TCP congestion window of the normal TCP, the fast retransmission scheme, and the proposed scheme in the error environment

lowest throughput. In addition, since the fast retransmission scheme has more short pauses (the empty regions) than the proposed scheme, its throughput is less than that of the proposed scheme. We can also see from these figures that the fast retransmission scheme has the most number of pauses (excluding those caused by handoffs) during the lifetime of the TCP connection. It implies that there are more error-related packet losses in the fast retransmission scheme. This explains why the bit error rate of the fast retransmission scheme is higher than the other two schemes. Since the normal TCP and

the proposed scheme have about the same number of pauses, their bit error rates are similar.

In summary, the above performance comparisons show that the proposed scheme has better performance than the other two schemes, whether in the no-error or in the error environment. Note that if the values of the parameters such as the maximum window size (either in the no-error or in the error environment), the handoff interval (either in the no-error or in the error environment), the size of the jammer packets (in the error environment), and the period of sending jammer packets (in the error environment) are changed, the absolute performances of these schemes will be different. However, the relative performance comparisons will be the same, that is, the proposed scheme will still perform better than the normal TCP and the fast retransmission scheme. For example, if we decrease the period of sending jammer packets, which causes more packet losses, all three schemes will have more pauses in communication and thus require more time to transmit 4 Mbytes of data. Consequently, their throughputs and bit error rates will decrease as well. But, as all three schemes require more time to transmit 4 Mbytes of data, there will be more handoff-related and more error-related packet losses during the lifetime of the data transfer. Since more handoff-related packet losses cause more long pauses for the normal TCP, its throughput will still be lower than the other two schemes. Also, since the fast retransmission scheme does not consider the impact of wireless transmission errors, more error-related packet losses will make the throughput of the fast retransmission error lower than that of the proposed scheme. Thus, the proposed scheme will still have better performance than the other two schemes.

4.3 Comparisons with the Literature

The throughput performances of the normal TCP and the fast retransmission scheme, under no-error situation, have been studied in [4]. Based on experiments performed in a wireless LAN testbed (with a bandwith of 2 Mbps for the wireless link), Cáceres and Iftode have shown that in the no-error environment, the throughput of the normal TCP is about 1.100 Mbps and the throughput of the fast retransmission scheme is about 1.380 Mbps. Although the absolute results of these schemes are not quite the same as our analytical and simulation results, the relative performance comparisons are quite similar. In the experimental results, the throughput of the fast retransmission scheme is about 0.28 Mbps higher than that of the normal TCP. In our analytical and simulation results, the throughput improvement is about 0.27 Mbps and 0.28 Mbps respectively. Thus, we can be sure that our analytical and simulation models are correct.

Chapter 5

CONCLUSIONS

We have described a mobile-aware TCP to alleviate the impact of mobility on TCP performance. The key point in this proposed scheme is to differentiate between mobility-related packet losses and delays from congestion-related ones based on the signal strength received by the MH. In addition, based on observing how congestion control procedures degrade TCP performance, the proposed scheme provides three mobility alleviation procedures in order to control the effects of mobility.

We have also developed analytical and simulation models to compare the performance of the proposed scheme with those of the normal TCP and the fast retransmission scheme in order to evaluate the effectiveness of the proposed scheme. Both the analytical and the simulation results show that the proposed scheme performs significantly better than the normal TCP (over 30%), and that it also performs better than the fast retransmission scheme (up to 12%). These comparisons indicate that our mobile-aware TCP can effectively improve TCP performance in wireless computing environments.

5.1 Future Work

We would like to further investigate the performance of our mobile-aware TCP under the following situations:

- Different types of error pattern, especially those with higher bit error rate ($\sim 10^{-3}$)
- Different combinations of the parameters, such as packet size, window size, and handoff interval
- A lot of concurrent TCP connections
- Different types of network topology, especially those with heterogeneous networks interconnected

Based on these studies, we may find a better set of mobility alleviation procedures to further improve TCP performance. In addition to this, we would also like to implement our scheme in a real network to verify the TCP performance improvement.

Appendix

Code for the Analytical Calculation

```
#include <stdio.h>
#include "qmodel.h"

main( )
{
   char  line[MAXSIZE];
   int  j, k, l, m, z, wp0;
   double  *table;
   double  psize, wsize, nhop, srate, hint, tdata, beacon, minrto, num, wp1, wp, a[300],
           b[300], cwnd, cwnd1, maxcwnd, ssthresh, ssthresh1, tsdata, tsdata1, tstime,
           tstime1, tspac, tspac1, swin, swin1, t, r, x, y, Th, Th1, rto;

   /* parameters setting */

   /* fixed parameters: */
   /*    1. service rate */
   /*    2. beacon period */
   /*    3. minimum retransmission timeout */

   printf("The default values of the parameters are as follows:\n\n");
   printf("    1. Packet size = 536 bytes\n");
   printf("    2. Receiver's advertised window = 4096 bytes\n");
   printf("    3. Number of hops = 3\n");
   printf("    4. Handoff interval = 8 seconds\n");
   printf("    5. Total data sent = 4000000 bytes\n");
   printf("    6. Service rate = 2000000 bps\n");
   printf("    7. Beacon period = 1 second\n");
   printf("    8. Minimum retransmission timeout = 1 second\n\n");

   psize = PACSIZE;
```

```
wsize = WINSIZE;
nhop = NUMHOP;
srate = SERVRATE;
hint = HANDINT;
tdata = DATASENT;
beacon = BEACON_P;
minrto = MIN_RTO;

j = 1;
while (j != 0)
      {
      printf("The values in 1-5 can be changed.  Choose the number you want\n");
      printf(" (otherwise, press ENTER):");
      num = rp(line, MAXSIZE);
      if (num != 0.0)
         {
         if (num = = 1.0)
            {
            printf("Packet size = ");
            num = rp(line, MAXSIZE);
            if (num != 0.0)
               psize = num;
            }
         else if (num = = 2.0)
            {
            printf("Receiver's advertised window = ");
            num = rp(line, MAXSIZE);
            if (num != 0.0)
               wsize = num;
            }
         else if (num = = 3.0)
            {
            printf("Number of hops = ");
            num = rp(line, MAXSIZE);
            if (num != 0.0)
               nhop = num;
            }
         else if (num = = 4.0)
            {
            printf("Handoff interval = ");
            num = rp(line, MAXSIZE);
            if (num != 0.0)
               hint = num;
            }
         else if (num = = 5.0)
            {
            printf("Total data sent = ");
```

```
            num = rp(line, MAXSIZE);
            if (num != 0.0)
               tdata = num;
            }
         }
      else
         j = 0;
      }

/* throughput-delay table */

wp1 = wsize / psize;
wp0 = (int) wp1;
wp = (double) wp0;
srate /= 8.0;        /* service rate (in bytes per second) */

for (j = 1; j <= (wp0+1); ++j)
      {
   a[j] = gamma(nhop, (double)j, srate);
   b[j] = delay(nhop, (double)j, srate, psize);
   }

/* normal TCP (no move) */

cwnd = 1.0;              /* initial congestion window (in packet) */
maxcwnd = MAXWND / psize;    /* maximum congestion window (in packets) */
ssthresh = MAXWND / psize;   /* initial slow-start threshold (in packets) */
tsdata = 0.0;            /* initial total data sent (in byte) */
tstime = 0.0;            /* initial total time delay (in second) */
tspac = 0.0;             /* initial total number of packets (in byte) */

while (tsdata < tdata)
    {
    if (cwnd <= wp1)
       {
       swin = cwnd;
       tsdata += swin * psize;
       if (tsdata > tdata)
          break;
       table = &b[(int)swin];
       t = *table;
       tstime += t;
       table = &a[(int)swin];
       r = *table;
       tspac += r * t;
       }
    else
```

```
        {
        swin = wp1;
        tsdata += wsize;
        if (tsdata > tdata)
           break;
        t = delay(nhop, swin, srate, psize);
        tstime += t;
        table = &a[wp0];
        x = *table;
        y = *(table+1);
        r = inter(wp, swin, x, y);
        tspac += r * t;
        }

     if (cwnd < ssthresh)
        cwnd *= 2.0;
     else
        cwnd += 1.0;

     if (cwnd > maxcwnd)
        cwnd = maxcwnd;
     }

if (tsdata != tdata)
   {
    if (swin < wp1)
       tsdata -= swin * psize;
    else
       tsdata -= wsize;
    }

swin = (tdata - tsdata) / psize;

if (swin != 0.0)
   {
   t = delay(nhop, swin, srate, psize);
   tstime += t;
   z = (int) swin;

   if (z = = 0)
      {
       x = 0.0;
       table = &a[1];
       y = *table;
       }
   else
      {
```

```
        table = &a[z];
        x = *table;
        y = *(table+1);
        }

    tspac += inter((double)z, swin, x, y) * t;
    }

Th = tspac * 8 / tstime;

printf("\n----- no-move TCP -----\n");
printf(" Total number of packets = %f bytes\n", tspac);
printf(" Total time delay = %f seconds\n", tstime);
printf(" Throughput = %f bits per second\n", Th);
printf("            ( 100%% )\n\n");

/* normal TCP, fast retransmission scheme, proposed scheme     */

/* assumptions: */
/*    1. all packets in a window are lost during handoffs */
/*    2. mobile host receives a beacon from the new cell */
/*       one second after leaving the old cell */

rto = minrto + minrto * 2;
j = 0;
l = 0;
for (j = 0; j < 3; ++j)
    {
    cwnd1 = 1.0;
    ssthresh1 = MAXWND / psize;
    tsdata1 = 0.0;
    tstime1 = 0.0;
    tspac1 = 0.0;
    k = 1;
    while (tsdata1 < tdata)
        {
        while (tstime1 >= (hint*(k-1)) && tstime1 < (hint*k))
            {
            if (cwnd1 <= wp1)
                {
                swin1 = cwnd1;
                table = &b[(int)swin1];
                t = *table;
                tstime1 += t;
                if (tstime1 > (hint*k))
                    break;
                tsdata1 += swin1 * psize;
```

```
            if (tsdata1 > tdata)
                break;
            table = &a[(int)swin1];
            r = *table;
            tspac1 += r * t;
            }
        else
            {
            swin1 = wp1;
            t = delay(nhop, swin1, srate, psize);
            tstime1 += t;
            if (tstime1 > (hint*k))
                break;
            tsdata1 += wsize;
            if (tsdata1 > tdata)
                break;
            table = &a[wp0];
            x = *table;
            y = *(table+1);
            r = inter(wp, swin1, x, y);
            tspac1 += r * t;
            }

        if (cwnd1 < ssthresh1)
            cwnd1 *= 2.0;
        else
            cwnd1 += 1.0;

        if (cwnd1 > maxcwnd)
            cwnd1 = maxcwnd;
        }

if (tstime1 > (hint*k) && tsdata1 != tdata && l == 0)
    {
    tstime1 = tstime1 - t + rto;

    for (m = 0; m < 2; ++m)
        {
        if (cwnd1 <= wp1)
            ssthresh1 = cwnd1;
        else
            ssthresh1 = wp1;

        ssthresh1 /= 2.0;

        if (ssthresh1 < 2.0)
            ssthresh1 = 2.0;
```

```
            cwnd1 = 1.0;
            }
        }
    else if (tstime1 > (hint*k) && tsdata1 != tdata && l = = 1)
        {
        tstime1 = tstime1 - t + rto;

        for (m = 0; m < 2; ++m)
            {
            cwnd1 /= 2.0;

            if (cwnd1 < 1.0)
                cwnd1 = 1.0;
            }

        table = &b[1];
        t = *table;
        tstime1 += t;
        tsdata1 += 1 * psize;
        table = &a[1];
        r = * table;
        tspac1 += r * t;
        }

    k += 1.0;
    }

if (tsdata1 != tdata)
   {
    if (swin1 < wp1)
       tsdata1 -= swin1 * psize;
    else
       tsdata1 -= wsize;
    }

tstime1 -= t;
swin1 = (tdata - tsdata1) / psize;
z = (int)swin1;

if (swin1 != 0.0)
   {
    t = delay(nhop, swin1, srate, psize);
    tstime1 += t;

    if (z == 0)
       {
```

```
        x = 0.0;
        table = &a[1];
        y = *table;
        }
    else
       {
        table = &a[z];
        x = *table;
        y = *(table+1);
        }

    tspac1 += inter((double)z, swin1, x, y) * t;
    }

Th1 = tspac1 * 8 / tstime1;

if (rto = = (minrto + minrto * 2) && l = = 0)
   {
    printf("----- Normal TCP -----\n");
    printf(" Total number of packets = %f bytes\n", tspac1);
    printf(" Total time delay = %f seconds\n", tstime1);
    printf(" Throughput = %f bits per second\n", Th1);

    Th1 /= Th * 0.01;

    printf("            ( %f%% )\n\n", Th1);

    rto = minrto + 0.2;
    }
else if (rto = = minrto + 0.2 && l = = 0)
    {
    printf("----- Fast Retransmission Scheme -----\n");
    printf(" Total number of packets = %f bytes\n", tspac1);
    printf(" Total time delay = %f seconds\n", tstime1);
    printf(" Throughput = %f bits per second\n", Th1);

    Th1 /= Th * 0.01;

    printf("            ( %f%% )\n\n", Th1);

    l = 1;
    }
else if (rto = = minrto + 0.2 && l = = 1)
    {
    printf("----- Proposed Scheme -----\n");
    printf(" Total number of packets = %f bytes\n", tspac1);
    printf(" Total time delay = %f seconds\n", tstime1);
```

```
            printf(" Throughput = %f bits per second\n", Th1);

            Th1 /= Th * 0.01;

            printf("            ( %f%% )\n\n", Th1);

            }
        }
    return 0;
}
```

References

[1] A. Myles and D. Skellern, "Comparison of Mobile Host Protocols for IP," *Journal of Internetworking : Research and Experience*, vol. 4, no. 4, pp. 175-194, December 1993.

[2] W. R. Stevens, TCP/IP Illustrated, Volume 1: The Protocols, Addison-Wesley, Reading, MA, May 1994.

[3] V. Jacobson, "Congestion Avoidance and Control," *Proceedings ACM SIGCOMM '88*, pp. 314-329, August 1988.

[4] R. Cáceres and L. Iftode, "The Effects of Mobility on Reliable Transport Protocols," *Proceedings of the 14th International Conference on Distributed Computing Systems*, pp. 12-20, June 1994.

[5] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," *Proceedings of the 15th International Conference on Distributed Computing Systems*, pp. 136-143, June 1995.

[6] H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks," accepted for publication in *ACM Wireless Networks*.

[7] R. Cáceres and V. N. Padmanabhan, "Fast and Scaleable Handoffs for Wireless Internetworks," to appear in *Proceedings of ACM MobiCom '96*, November 1996.

[8] E. Ayanoglu, S. Paul, T. F. LaPorta, K.K. Sabnani, and R.D.Gitlin, "AIRMAIL: A link-layer protocol for wireless networks," *ACM Wireless Networks*, no. 1, pp. 47-60, February 1995.

[9] A. DeSimone, M. C. Chuah, and O. C. Yue, "Throughput Performance of Transport-Layer Protocols over Wireless LANs," *Proceedings of Globecom '93*, pp. 542-549, December 1993.

[10] V. K. Garg and J. E. Wilkes, Wireless and Personal Communications Systems, Prentice-Hall, Reading, NJ, 1996.

[11] M. Schwartz, Telecommunication Networks: Protocols, Modeling and Analysis, Addison-Wesley, Reading, MA, November 1988.

[12] MIL 3, Inc., Optimized Network Engineering Tools (OPNET), Release 2.5.A, 1994.

[13] C. Perkins, "IP Mobility Support," IETF Internet draft, January 1995.