TCP over Mobile Ad-Hoc Access Networks

By

Ahmed Moustafa

A thesis submitted to the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Master of Computer Science

Ottawa-Carleton Institute for Computer Science

School of Computer Science

Carleton University

Ottawa, Ontario

December 2002

© Copyright 2002, Ahmed Moustafa

The undersigned recommend to the Faculty of Graduate Studies and Research

acceptance of the thesis

TCP over Mobile Ad-Hoc Access Networks

Submitted by **Ahmed Moustafa**, High Diploma,

in partial fulfillment of the requirements for the

degree of Master of Computer Science

_____

Dr. Thomas Kunz,

Thesis Supervisor

_____

Dr. Frank Dehne, Director of School of Computer Science

Carleton University

December 2002

ABSTRACT

The rapid advancement in portable computing platforms and wireless communication technology has led to significant interest in the design and development of protocols for instantly deployable wireless networks, often referred to as MANET: "Mobile Ad-Hoc Network". This thesis concentrates on the transmission layer which is mostly implemented by TCP (Transmission Control Protocol). TCP suffers a significant drop in its throughput when it runs on a MANET. This significant TCP throughput drop is mainly because of the TCP implicit assumption that any packet loss is due to congestion, which causes TCP to invoke the congestion control algorithms. This assumption is true for the wired network, but not true for the MANET. Several protocols have been proposed for improving the TCP throughput over a Standalone–MANET. To the best of our knowledge this research is the first research that discuss, propose and evaluate transmission control scheme for a MANET connected to the wired network (MANET as Access Network). Our scheme consists of three protocols, the TCP protocol on the wired network, the Snoop protocol on the base station and the ATCP protocol on the MANET. The simulation results show that our scheme was able to improve the TCP end-to-end connection throughput without requiring changes to the TCP stack on the fixed host(s), limiting the changes to the base station and the mobile host(s). This will make deploying such a scheme much easier, as it does not require re-compiling and re-linking existing applications.

ACKNOWLEGMENTS

I have had a great honour to be supervised by Dr. Thomas Kunz. I would like to take this opportunity to acknowledge his scientific guidance and support.

I would like to thank my wife, Mahitab, for her support and encouragement. Her understanding and support made accomplishing my goal easier.

I would like to thank my family, especially my father and my mother, for always being there for me when I needed their prayers.

Above all, I thank Allah (God).

Table of Contents

Chapter 1 - Introduction 1

1.1 Motivation 2

1.2 Background 3

1.3 TCP over Single-Hop Wireless Networks 5

1.4 TCP over Multi-Hop Wireless Networks 6

1.5 Thesis Contribution 8

1.6 Thesis Organization 9

Chapter 2 - TCP Congestion Control Algorithms & IPsec 11

2.1 History 11

2.2 Definitions 12

2.3 TCP Time-out and Retransmission 14

2.4 TCP Congestion Control Algorithms 15

2.4.1 Slow Start and Congestion Avoidance 15

2.4.2 Fast Retransmit and Fast Recovery 17

2.5 IPsec (IP Security) 18

Chapter 3 - TCP over Single-Hop Wireless Networks 20

3.1 Evaluation Criteria 20

3.2 Protocol Classification 22

3.3 TCP-Aware Protocols 23

3.3.1 Protocols that Violate the TCP End-to-End Semantics 23

3.3.2 Protocols that Preserve the TCP End-to-End Semantics 26

3.4 TCP-Unaware Protocols 37

3.4.1 Using Inter-Arrival Times at the Receiver 37

3.4.2 Delayed Duplicate Acknowledgements 39

3.4.3 Fast Retransmissions 42

3.4.4 Freeze-TCP 43

3.4.5 Establishing Two Connections 45

3.5 Summary 48

Chapter 4 - TCP over Multi-Hop Wireless Networks 50

4.1 MANET Architecture Characteristics 50

4.2 Evaluation Criteria 51

4.3 TCP Feedback 52

4.3.1 Protocol Assumptions 53

4.3.2 Protocol Semantics 53

4.3.3 Protocol Evaluation 54

4.4 ATCP: Ad-Hoc TCP 54

4.4.1 Protocol Semantics 55

4.4.2 Protocol Evaluation 58

4.5 TCP-DOOR 58

4.5.1 Protocol Semantics 58

4.5.2 Protocol Evaluation: 61

4.6 TCP-Fixed RTO 61

4.6.1 Protocol Semantics 61

4.6.2 Protocol Evaluation: 62

4.7 Summary 63

Chapter 5 - The Proposed Scheme & Simulation Environment 64

5.1 The Proposed Scheme 64

5.1.1 The Requirements 64

5.1.2 The Scheme 65

5.1.3 The Interaction between TCP, ATCP and Snoop 66

5.2 Simulation 70

5.2.1 Simulation Environment 70

5.2.2 Network Topology 71

5.2.3 Traffic Profile 71

5.2.4 Wired Network Simulation Environment 72

5.2.5 MANET Simulation Environment 72

5.2.6 Simulation Summary 73

5.3 Performance Metric 74

5.4 Factors that Affect the Performance Metric 74

5.4.1 Congestion 75

5.4.2 Route Failures 75

Chapter 6 - Simulation Results 76

6.1 Simulation Sets 76

6.1.1 Simulation Sets Main Configuration 76

6.1.2 Simulation Sets Fixed Configuration 77

6.2 Expected TCP Throughput 78

6.2.1 IEEE 802.11 MAC Data Throughput 79

6.2.2 TCP Data Throughput 81

6.3 Simulation Validation 82

6.4 Simulation Sets Results 85

6.4.1 Simulation Set 1 (TCPReno) 85

6.4.2 Simulation Set 2 (TCPSnoop) 88

6.4.3 Simulation Set 3 (ATCPReno) 93

6.4.4 Simulation Set 4 (ATCPSnoop) 97

6.5 Conclusions 98

Chapter 7 - Conclusions and Future Work 102

7.1 **Conclusions** 103

7.2 Future Work 105

List of Figures

Figure 1.1: MANET in Military Environment 1

Figure 1.2: MANET in Civilian Environment 2

Figure 1.3: Open Systems Interconnection (OSI) Communication Model 3

Figure 1.4: Packet Losses over the Wireless Link 6

Figure 1.5: Mobile Hosts Movements 7

Figure 1.6: Wired – MANET Architecture 8

Figure 3.1: I-TCP Connection Setup 25

Figure 3.2: Data Transfer from a Fixed Host 27

Figure 3.3: Data Transfer from a Mobile Host 28

Figure 3.4: (A) No Timeout (B) Timeout 31

Figure 3.5: WTCP Connection 32

Figure 3.6: M-TCP Connection 35

Figure 3.7: Inter-arrival gap 38

Figure 3.8: An Illustration of Delayed DupAcks Scheme 41

Figure 3.9: Illustration of Increased Throughput Due to Freeze-TCP 44

Figure 3.10: Model of Proposed TCP 46

Figure 4.2: State Transition Diagram for ATCP at the Sender 55

Figure 4.3: A Possible Case of Route Change 58

Figure 5.1: Wired–MANET Architecture Topology 71

Figure 6.1: MAC Interference Among a Chain of Nodes. 80

Figure 6.2: Measured TCP Throughput versus Expected TCP Throughput 84

Figure 6.3: Dropped Packets Dynamics 88

Figure 6.4: TCPReno and TCPSnoop Congestion Window Dynamics 91

Figure 6.5: TCPReno and ATCP Congestion Window Dynamics 95

List of Tables

Table 3.1: Cellular Wireless Network Protocols Evaluation Summary 48

Table 4.1: MANET Protocols Evaluation 63

Table 5.1 Summary of the Network Topology Simulation Parameters 73

Table 5.2 Summary of the Wired Network Simulation Parameters..... 73

Table 5.3 Summary of the MANET Simulation Parameters 74

Table 6.1: Validation Exercise Results..... 83

Table 6.2: TCPReno Throughput without Congestion..... 85

Table 6.3: TCPReno Throughput with Congestion..... 86

Table 6.4: TCPSnoop Throughput without Congestion 88

Table 6.5: TCPSnoop Throughput with Congestion 92

Table 6.6: ATCPReno Throughput without Congestion 93

Table 6.7: ATCPReno Throughput with Congestion 96

Table 6.8: ATCPSnoop Throughput without Congestion 97

Table 6.9: ATCPSnoop Throughput with Congestion 98

Chapter 1

Introduction

A MANET (Mobile Ad-Hoc Network) is a collection of mobile hosts that can dynamically form a network without using any pre-existing network infrastructure and can communicate over relatively bandwidth constrained wireless links. A MANET is required in situations where a fixed communication infrastructure, wired or wireless, does not exist or has been destroyed.

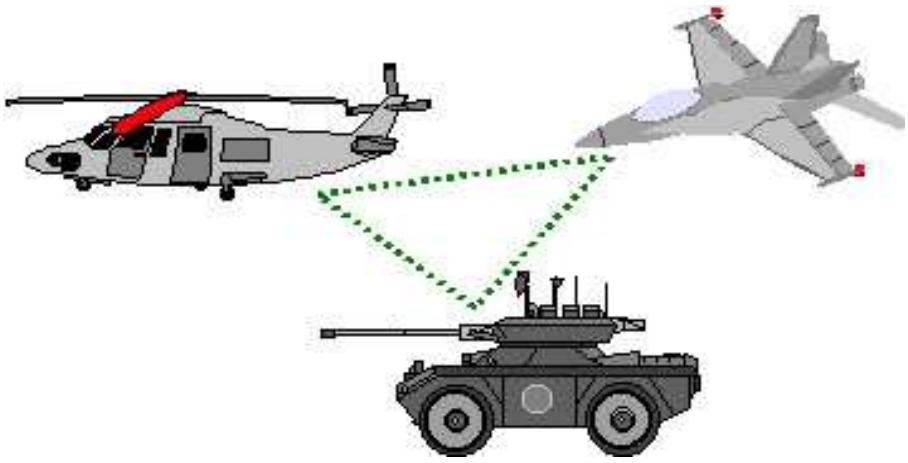

Figure 1.1: MANET in Military Environment

The applications span several different sectors of society. In the military environment they can be used to interconnect military groups working in the desert or in the sea. In

the civilian environment, they can be used to interconnect workgroups moving in an urban or rural area and engaged in collaborative operation such as search and rescue.

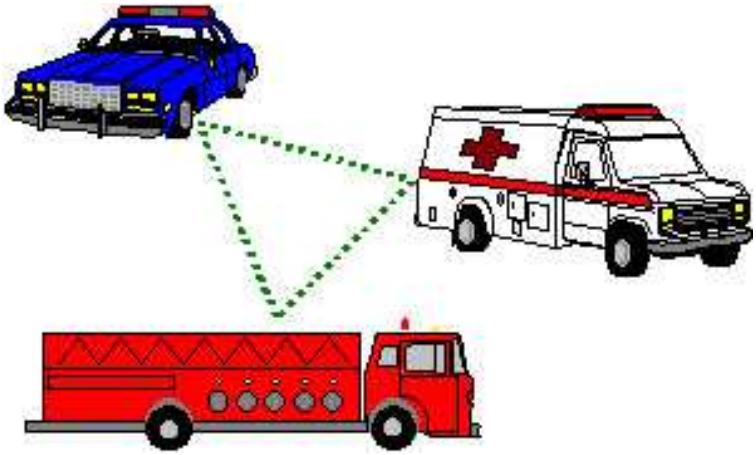

Figure 1.2: MANET in Civilian Environment

In the law enforcement sector, they can be used in applications such as crowd control and border patrol. These network scenarios cannot rely on centralized and organized connectivity. The network is decentralized in a way that all network activity including discovering the topology and delivering messages must be executed by the mobile hosts.

1.1 Motivation

The research in the area of mobile ad-hoc networking is receiving much attention from academia, industry, and government. These networks introduce many complex issues, there are many open problems for research and significant contributions can be made. Research has been done on the different layers of the Open Systems Interconnection (OSI) communication model, especially the first three layers (Physical Layer, Data Link Layer and Network Layer). Relatively little attention was

given to layer 4, the Transport Layer. This thesis concentrates on layer 4, the Transport Layer, which is mostly implemented by TCP (Transmission Control Protocol).

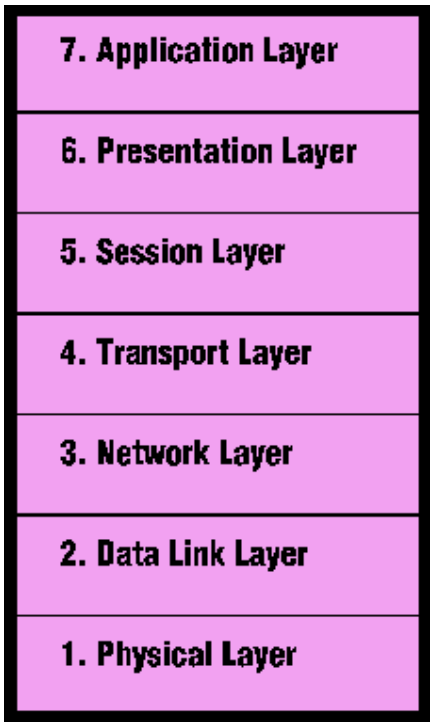

Figure 1.3: Open Systems Interconnection (OSI) Communication Model

1.2 Background

TCP (Transmission Control Protocol) is the most common transport layer protocol used on the Internet. TCP is a set of rules, used along with the Internet Protocol (IP), to send data in the form of message units (Packets), between computers over the Internet. While IP takes care of handling the actual delivery of the data, TCP takes care of keeping track of the individual packets that a message is divided into for efficient routing through the Internet.

For example, when an HTML file is sent from a web server, the transmission control protocol (TCP) in that web server divides the file into one or more packets, numbers the packets, and then forwards them individually to the IP layer. Although each packet has the same destination IP address, it may get routed differently through the network. At the other end (the client program), TCP reassembles the individual packets and waits until they have arrived to forward them to the application as a single file.

TCP is known as a connection-oriented protocol, which means that a connection is established and maintained until such time as the message or messages to be exchanged by the applications at each end have been exchanged.

TCP is a reliable end-to-end transport protocol that was designed and fine-tuned to perform well on networks where the end-to-end connection is wired, and the packet losses are mainly due to congestion. TCP achieves reliability by requiring that the TCP sender retransmits lost packets. For this purpose, the TCP receiver must acknowledge receipt of data packets from the TCP sender. An acknowledgement (ACK) sent by the TCP receiver is cumulative. A new packet received by the receiver is said to be out-of-order (OOO) if it is not the next in-sequence packet expected by the receiver. On receiving this out-of-order (OOO) packet, the receiver sends a duplicate acknowledgement (DupAck), acknowledging all the in-sequence bytes received so far.

The TCP sender determines that a packet is lost by using one of the following:

- Retransmission Timeout: If a timer, set when a packet is transmitted, expires before acknowledgement of the packet is received by the TCP sender, the packet is presumed lost.
- Fast Retransmit: If three DupAcks are received by the TCP sender, that is an indication that a packet has been lost, the TCP sender retransmits the packet without waiting for the retransmission timer to expire.

The TCP sender maintains a congestion window that determines the maximum amount of unacknowledged data sent by the sender. When a packet loss is detected, the TCP congestion control mechanism drastically reduces the congestion window size, effectively reducing the amount of data sent by the sender in one round-trip time (RTT).

1.3 TCP over Single-Hop Wireless Networks

Single-Hop Wireless Network (Cellular Network Architecture) is a network architecture where the last link in the end-to-end connection is a wireless link. In such a network architecture, packet losses are not only because of congestion on the wired part of the connection, but could be because of transmission errors or handoffs between cells on the wireless part of the connection (A cell is a region that is covered by a base station. The mobile host could be within the transmission range of more than one base station. Handoffs occurs when the mobile host moves from a base station transmission range (cell) to another base station transmission range (cell)). TCP assumes that packet losses are mainly due to congestion, on detecting such a packet loss, it triggers slow-start and congestion avoidance algorithms, which could

be unnecessary in case of packet losses due to transmission errors or handoffs between cells on the wireless part of the connection, which results in unnecessarily degrading the TCP end-to-end connection throughput.

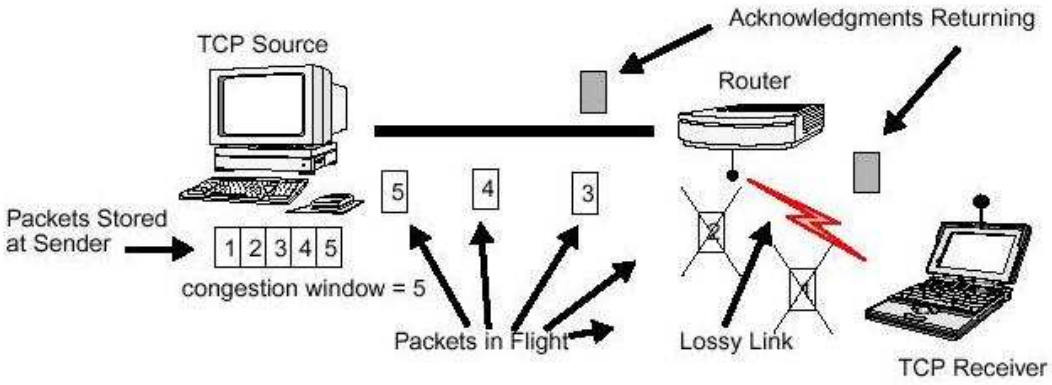

Figure 1.4: Packet Losses over the Wireless Link

Figure 1.4 shows a typical loss situation over the wireless link. Here, the TCP sender is in the middle of a transfer across a two-hop network to a mobile host. The sender's congestion window consists of 5 packets. Of the five packets in the network, the first two packets are lost on the wireless link. TCP assumes that the packet loss was due to congestion, at the wired part of the network, and triggers slow-start and congestion avoidance algorithms.

We discuss and evaluate in Chapter 3 *"TCP over Single-Hop Wireless Networks"*, transmission control protocols that propose solutions to these problems.

1.4 TCP over Multi-Hop Wireless Networks

A Mobile Ad-Hoc Network (MANET) is a collection of mobile hosts that can dynamically form a network without using any pre-existing network infrastructure.

Each mobile host works as a router, for establishing connections between any two mobile hosts. The MANET topology changes rapidly due to the movement of the mobile hosts, which results in route failure between the mobile hosts. In such network architecture, packet losses are not only because of congestion but it could be because of transmission errors or route failure. TCP assumes that packet losses are mainly due to congestion, on detecting such a packet loss, it triggers slow-start and congestion avoidance algorithms, which could be unnecessary in case of packet losses due to transmission errors or route failure, which again results in unnecessarily degrading the TCP end-to-end connection throughput.

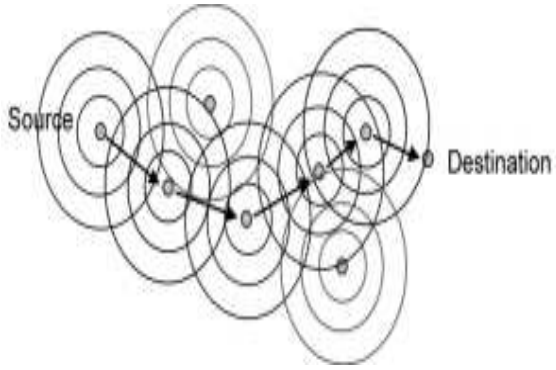

Figure 1.5: Mobile Hosts Movements

Figure 1.5 shows typical mobile hosts movements in a MANET. If any mobile host on the route from mobile host *Source* to mobile host *Destination* moves out of the transmission range of its neighbour mobile host, the route from the mobile host *Source* to the mobile host *Destination* will be invalidated. Invalidating the route will cause packet loss.

We discuss and evaluate in Chapter 4 *"TCP over Multi-Hop Wireless Networks",* transmission control protocols that propose solutions to these problems.

1.5 Thesis Contribution

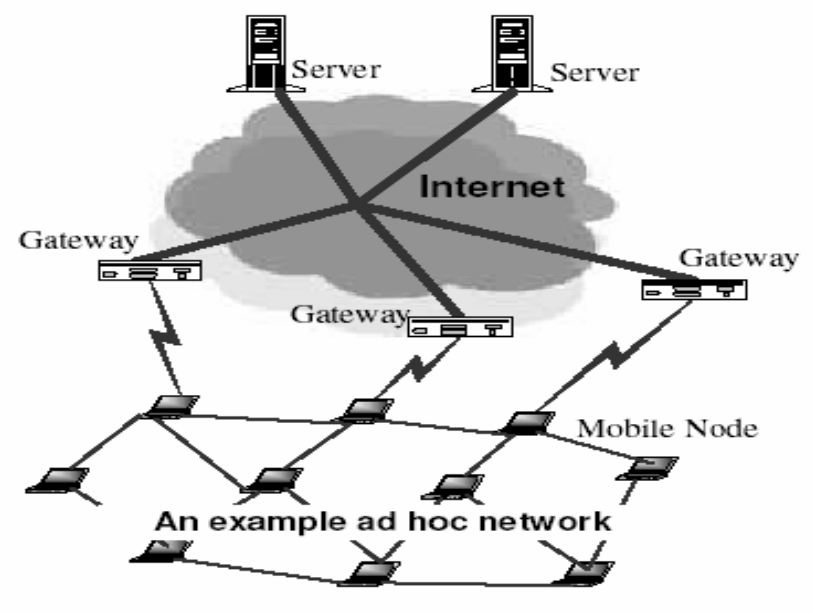

Figure 1.6: Wired – MANET Architecture [22]

Our research concentrates on the protocols we discuss and evaluate in Chapter 3 and in Chapter 4, to choose the most suitable ones, implement them in the NS simulator and modify them to interact together, and then apply it to the Wired-MANET architecture. We implemented the ATCP protocol in the NS simulator and modified the Berkeley Snoop protocol to interact with it.

All of the protocols we discuss in Chapter 4 only propose a solution for TCP over a stand-alone MANET (MANET not connected to the wired network). To the best of our knowledge, this thesis is the first research to discuss, propose and evaluate a transmission control scheme for a MANET that is connected to the wired network

through a base station (Wired – MANET Architecture). Figure 1.6 shows a MANET that is connected to the wired network (Wired – MANET Architecture).

In Chapter 3, we identify evaluation criteria for the Single-Hop Wireless Network and apply these evaluation criteria on the different protocols that are proposed for TCP over Single-Hop Wireless Network to choose the most suitable protocol for this network architecture.

In Chapter 4, we identify evaluation criteria for the Multi-Hop Wireless Network and apply these evaluation criteria on the different protocols that are proposed for TCP over Multi-Hop Wireless Network to choose the most suitable protocol for this network architecture.

The simulation results show that our scheme was able to improve the TCP end-to-end connection throughput without requiring changes to the TCP on the fixed host(s), limiting the changes to the base station and the mobile host(s). This will make deploying such a scheme much easier, as it does not require re-compiling and re-linking the existing applications.

1.6 Thesis Organization

This thesis is divided into seven chapters including this chapter:

Chapter 1: Introduction

Chapter 1 contains the thesis introduction, motivation, background and contribution.

Chapter 2: TCP Congestion Control Algorithms & IPsec

Chapter 2 provides a background on TCP congestion control algorithms and IPsec.

Chapter 3: TCP over Single-Hop Wireless Network

Chapter 3 discusses and evaluates transmission control protocols for a Single-Hop Wireless Network.

Chapter 4: TCP over Multi-Hop Wireless Network

Chapter 4 discusses and evaluates transmission control protocols for a stand-alone Multi-Hop Wireless Network.

Chapter 5: The Proposed Scheme and Simulation

Chapter 5 discusses and proposes our scheme for a MANET connected to the wired network (Wired–MANET Architecture), gives a description of the simulation environment and discusses the performance metric.

Chapter 6: Simulation Results

Chapter 6 gives a description of the different simulation scenarios we are running in our work, lists all the results from our experiments and discusses them.

Chapter 7: Conclusion and Future Work

Chapter 7 presents the conclusion of the thesis and the future work.

Chapter 2

TCP Congestion Control Algorithms & IPsec

In Chapter 1, we explained that TCP assumes that any packet loss is mainly due to congestion in the network, and we explained that this assumption is correct in the case of a wired network, but in the case of a wireless network (either Single-Hop Wireless Network or Multi-Hop Wireless Network) this assumption is not always correct. In case of a Single-Hop Wireless Network, packet losses could be because of wireless transmission errors or handoffs between cells. In case of a Multi-Hop Wireless Network, packet losses could be because of wireless transmission errors or route failure. Due to the importance of the TCP congestion control algorithms in this thesis, we will dedicate Chapter 2 to explain it. We also have a brief description of IPSec, as it will be one of the criteria that we use to evaluate any protocol in Chapter 3 and Chapter 4

2.1 History

TCP congestion control was introduced into the Internet in the late 1980's by Van Jacobson; roughly eight years after the TCP/IP protocol stack had become operational. Immediately preceding this time, the Internet was suffering from

congestion collapse, hosts would send their packets into the Internet as fast as the receiver window would allow, congestion would occur at some router causing packets to be dropped, and the hosts would time-out and retransmit their packets, resulting in even more congestion. Broadly speaking, the idea of TCP congestion control is for each sender to determine how much capacity is available in the network, so it knows how many packets it can safely have in transit. Once a given sender has this many packets in transit, it uses the arrival of an acknowledgment (ACK) as a signal that one of its packets has left the network, and it is therefore safe to insert a new packet into the network without adding to the level of congestion. Using acknowledgments (ACKs) to pace the transmission of packets, TCP is said to be self-clocking. Of course, determining the available capacity in the first place is not an easy task. To make matters worse, because other connections come and go, the available bandwidth changes over time, meaning that any given sender must be able to adjust the number of packets it has in transit.

2.2 Definitions

In this section we provide the definition of several terms that will be used throughout the remainder of this chapter [1].

- SEGMENT: A segment is any TCP/IP data or acknowledgment packet.
- SENDER MAXIMUM SEGMENT SIZE (SMSS):

  The SMSS is the size of the largest segment that the sender can transmit. This value can be based on the maximum transmission unit of the network, the path MTU discovery algorithm, RMSS, or other factors. The size does not include the TCP/IP headers and options.

- RECEIVER MAXIMUM SEGMENT SIZE (RMSS):

  The RMSS is the size of the largest segment the receiver is willing to accept. This is the value specified in the MSS option sent by the receiver during connection start-up. The size does not include the TCP/IP headers and options.

- FULL-SIZED SEGMENT:

  A segment that contains the maximum number of data bytes permitted.

- RECEIVER WINDOW (rwnd):

  The most recently advertised receiver window.

- CONGESTION WINDOW (cwnd):

  A TCP state variable that limits the amount of data a TCP sender can send. At any given time, a TCP sender must not send data with a sequence number higher than the sum of the highest acknowledged sequence number and the minimum of cwnd and rwnd.

- INITIAL WINDOW (IW):

  The initial window is the size of the sender's congestion window after the three-way handshake is completed.

- LOSS WINDOW (LW):

  The loss window is the size of the congestion window after a TCP sender detects loss using its retransmission timer.

- RESTART WINDOW (RW):

  The restart window is the size of the congestion window after a TCP restarts transmission after an idle period.

- FLIGHT SIZE:

  The amount of data that has been sent but not yet acknowledged.

2.3 TCP Time-out and Retransmission

TCP provides reliable data delivery by retransmitting segments that are not acknowledged within some retransmission time-out (RTO) interval. Accurate dynamic determination of an appropriate RTO is essential to TCP performance. RTO is determined by estimating the mean and variance of the measured round-trip time (RTT), i.e., the time interval between sending a segment and receiving an acknowledgment for it. The original TCP specification had TCP update a smoothed RTT estimator (called R) using the low-pass filter

$$R \leftarrow \alpha R + (1 - \alpha) M$$

Where α is a smoothing factor with a recommended value of 0.9. This smoothed RTT is updated every time a new measurement is made. Ninety percent of each new estimate is from the previous estimate and 10% is from the new measurement [19]. Jacobson suggested that calculating the RTO based on both the mean and variance provides much better response to wide fluctuations in the round trip times. As described by Jacobson, the mean deviation is a good approximation to the standard deviation, but easier to compute. This leads to the following equations that are applied to each RTT measurement M.

$$Err = M - A$$

$$A \leftarrow A + gErr$$

$$D \leftarrow D + h (| Err | - D)$$

$$RTO = A + 4D$$

Where A is the smoothed RTT (an estimator of the average) and D is the smoothed mean deviation. Err is the difference between the measured value just obtained and the current RTT estimator. Both A and D are used to calculate the next retransmission time-out (RTO). The gain g is for the average and is set to 1/8 (0.125). The gain for

the deviation is h and is set to 0.25. The larger gain for the deviation makes the RTO go up faster when the RTT changes.

2.4 TCP Congestion Control Algorithms

In a typical congestion situation, there is at least one router whose buffer space is saturated. Such a router is called a bottleneck router. All packets coming into such routers must be discarded. There are four TCP congestion control algorithms that we are going to discuss:

- Slow Start
- Congestion Avoidance
- Fast Retransmit
- Fast Recovery

2.4.1 Slow Start and Congestion Avoidance

A TCP sender must use the slow start and congestion avoidance algorithms to control the amount of outstanding data injected into the network. To implement these algorithms, two state variables are added to the TCP per-connection state. The congestion window (cwnd) is a sender limitation on the amount of data the sender can transmit into the network before receiving an acknowledgment (ACK). The receiver's advertised window (rwnd) is a receiver limitation on the amount of outstanding data. The minimum of cwnd and rwnd controls data transmission.

```
MaxWindow = MIN (cwnd, rwnd)
```

To determine whether the slow start or congestion avoidance algorithm should be used to control data transmission, another state variable is added to the TCP per-

connection state, the slow start threshold (ssthresh). The initial value of the slow start threshold (ssthresh) is equal to the receiver's advertised window (rwnd). The slow start algorithm is used when TCP starts transmission into a network with unknown transmission conditions or after recovering from packet loss detected by the retransmission timer. During slow start, the TCP sender uses the initial window (IW) (the initial value of the congestion window (cwnd)), which must be less than or equal to 2*SMSS bytes and must not be more than 2 segments, to slowly probe the network to determine the available capacity, in order to avoid congesting the network with an inappropriately large amount of data. Then TCP increments the congestion window (cwnd) by at most SMSS bytes for each ACK received that acknowledges new data. Once the congestion window (cwnd) exceeds the slow start threshold (ssthresh), the slow start algorithm ends and the congestion avoidance algorithm starts. During the congestion avoidance, the congestion window (cwnd) is incremented per round-trip time (RTT) using the following formula:

```
cwnd += (MSS * MSS) / cwnd
```

The congestion avoidance algorithm continues until congestion is detected.

When a TCP sender detects segment loss using the retransmission timer, it does the following [1]:

- Sets the value of ssthresh to no more than the value given in the following equation:

  ssthresh = MAX (FlightSize / 2, 2*SMSS)

- Sets the value of cwnd to no more than the loss window, LW.
- Uses the slow start algorithm to increase the window from 1 full-sized segment to the new value of ssthresh, and when it exceeds cwnd, the congestion avoidance algorithm takes over again.

2.4.2 Fast Retransmit and Fast Recovery

When the TCP receiver receives an out-of-order (OOO) segment, it sends an immediate duplicate acknowledgment (DupAck) to the TCP sender. DupAck happens when the TCP receiver receives an out-of-order segment and since it did not receive the segment(s) before this out-of-order segment, it can not acknowledge the reception of this segment. Keeping in mind this fact, the TCP receiver responds with an ACK that has the sequence number of the expected packet, which is the same ACK it used to acknowledge the last in-order segment it received. This DupAck informs the TCP sender that the TCP receiver received an out of order packet and the sequence number of the expected packet. From the TCP sender's perspective DupAcks can be caused by the following [1]:

- Dropped segments.
- The re-ordering of data segments by the network.
- Replication of ACK or data segments by the network.

The TCP sender uses the arrival of three DupAcks (4 identical acknowledgments) as an indication that a segment has been lost and invokes the fast retransmission algorithm, which will retransmit the lost segment without waiting for a retransmission time-out to occur. The rationale behind that is simple, if the TCP sender was able to receive the DupAck that means that the network is not heavily congested, so there is no reason to wait for a time-out and just retransmit the lost segment immediately. After retransmitting the lost segment, the fast recovery algorithm controls the transmission of new data until a non-DupAck arrives. The fast retransmit and fast recovery algorithms are usually implemented together as follows [1]:

1. When the third DupAck is received, set ssthresh to no more than the value given in the following equation.

   ssthresh = MAX(FlightSize / 2, 2*SMSS)

2. Retransmit the lost segment and set cwnd to ssthresh plus 3*SMSS.
3. For each additional duplicate ACK received, increment cwnd by SMSS.
4. Transmit a segment, if allowed by the new value of cwnd and the receiver's advertised window.
5. When the next ACK arrives that acknowledges new data, sets cwnd to ssthresh.

2.5 IPsec (IP Security)

A very important question that we will need to answer when we evaluate the different transmission control protocols *"Is this protocol able to handle encrypted traffic or not?"*

As network security is taken more and more seriously, encryption is likely to be adopted widely. For instance, IPsec is becoming an integral part of IPv6, the next generation IP protocol. In such cases the whole IP payload is encrypted, so that the intermediate nodes may not even know that the traffic being carried in the payload is TCP.

IPsec (Internet Protocol Security) is a developing standard for security at the network or packet-processing layer of network communication. Earlier security approaches have inserted security at the application layer of the communication model. IPsec will be especially useful for implementing virtual private networks (VPNs) and for remote user access through dial-up connection to private networks. A big advantage of IPsec is that security arrangements can be handled without requiring changes to individual

user computers. Cisco has been a leader in proposing IPsec as a standard (or combination of standards and technologies) and has included support for it in its network routers.

IPsec provides two choices of security service: Authentication Header (AH), which essentially allows authentication of the sender of data, and Encapsulating Security Payload (ESP), which supports both authentication of the sender and encryption of data as well. The specific information associated with each of these services is inserted into the packet in a header that follows the IP packet header. Separate key protocols can be selected, such as the ISAKMP/Oakley protocol.

Chapter 3

TCP over Single-Hop Wireless Networks

In this chapter we discuss and evaluate transmission control protocols that have been designed or proposed for TCP over a Single-Hop Wireless Network (Cellular Network Architecture). A Single-Hop Wireless Network is a heterogeneous network where the last link in the end-to-end connection is a wireless link. Communications over wireless links are characterized by limited bandwidth, high latencies, high bit-error rates and temporary disconnections, that network protocols and applications must deal with. In addition, network protocols and applications have to handle user mobility and the handoffs that occur as users move from cell to cell in the cellular network. Throughout this chapter we will assume that congestion occurs on the wired part of the TCP connection and that wireless transmission errors and handoffs between cells occur on the wireless part of the TCP connection.

3.1 Evaluation Criteria

We identified evaluation criteria that we are going to use throughout this chapter to evaluate various transmission control protocols. These evaluation criteria will help us

choose the most suitable transmission control protocol for the cellular network architecture. The following are the evaluation criteria:

- **TCP end-to-end acknowledgments and semantics:**

  The protocol must preserve the TCP end-to-end acknowledgments and semantics, as existing applications may be relying on these semantics.

- **Congestion:**

  The protocol must be able to handle congestion as efficiently as the existing TCP.

- **Wireless transmissions errors:**

  The protocol must be able to distinguish between packet losses due to wireless transmissions errors and packet losses due to congestion.

- **Handoffs between cells:**

  The protocol must be able to distinguish between packet losses due to handoffs between cells caused by the mobility of the mobile host(s) and packet losses due to congestion.

- **Encrypted traffic:**

  The protocol must be able to handle encrypted traffic. As network security is taken more and more seriously, encryption is likely to be adopted very widely. Using IPsec, the whole IP payload is encrypted, so that the intermediate nodes may not even know that the traffic being carried in the payload is TCP packets. Even if the intermediate nodes know that the traffic is TCP packets, it will not be able to look at the TCP headers.

- **TCP on the fixed hosts:**

  Ideally there should not be any change required at the fixed hosts, because they are likely to belong to other organizations, which make them unavailable for

modifications. Another problem with modifying TCP on the fixed hosts is that it will require re-compiling and re-linking all existing applications that use TCP.

- **TCP on the base stations:**

  It will be better if the protocol does not change the TCP on the base station, but if a change to the TCP on the base station has to be brought in for performance enhancement, there is still a need to consider whether the base station will become the bottleneck of the connection or not. It is clear that the base station in some protocols will have to buffer at least some amount of data and do some extra processing for each connection going through them. If hundreds of mobile hosts are in the domain of a base station, it could get overwhelmed with the processing of the traffic associated with each connection. When a mobile host moves from the domain of one base station to another, the entire state of the connection including any data that was buffered for retransmissions may need to be handed over to the new base station. This can cause a significant amount of overhead and might lead to the loss of some packets and the sender dropping the congestion window, which would defeat the original purpose.

3.2 Protocol Classification

To facilitate the comparison between various transmission control protocols, we classified these protocols into different categories as follows:

- **TCP-Aware protocols**

  A TCP-Aware protocol requires a base station to look at the TCP headers and take TCP-specific actions to help improve TCP performance. This category is further divided into two subcategories:

- **Protocols that preserve the TCP end-to-end semantics**

  A TCP protocol that does not violate the TCP end-to-end semantics does not acknowledge to the TCP sender that a packet has been received until the actual TCP receiver receives it.

- **Protocols that violate the TCP end-to-end semantics**

  A TCP protocol that violates the TCP end-to-end semantics acknowledges to the TCP sender that a packet has been received even if the actual TCP receiver did not receive it yet.

- **TCP-Unaware protocols**

  A TCP-unaware protocol does not need a base station to look at the TCP headers and take TCP-specific actions to help improve TCP performance.

3.3 TCP-Aware Protocols

These protocols share the following common evaluation point(s):

- The protocol is unable to handle encrypted traffic.
- The protocol requires some modifications to the base station.

3.3.1 Protocols that Violate the TCP End-to-End Semantics

These protocols share the following common evaluation point(s):

- The protocol violates the TCP end-to-end acknowledgments and semantics.

These protocols violate important evaluation criteria and will not be considered further in our proposed scheme. I-TCP is an example of such protocols.

3.3.1.1 I-TCP

The I-TCP (Indirect TCP) protocol utilizes the resources of the base station(s) to provide transport layer communication between the mobile host(s) and the fixed host(s) [2].

3.3.1.1.1 Protocol Semantics

When a mobile host wants to communicate with a fixed host using I-TCP, a request is sent to the current base station to open a TCP connection with the fixed host on behalf of the mobile host. The mobile host communicates with its base station on a separate connection using version of TCP that is tuned for wireless links and is also aware of mobility. The fixed host only sees an image of its peer mobile host that resides on the base station. This image is handed over to the new base station in case of handoffs between cells. When the base station receives packets sent to the wireless host, it then sends an acknowledgement to the fixed host and then the packets is forwarded to the wireless host [2]. Figure 3.1 shows the setup for an I-TCP connection. The mobile host (MH) establishes a connection with the fixed host (FH) through base station (BS-1) and then moves to another cell under base station (BS-2). When MH requests an I-TCP connection with FH inside the cell of BS-1, BS-1 establishes a socket with the MH address and MH port number to handle the connection with the fixed host. It also opens another socket with its own address and some suitable port number for the wireless side of the I-TCP connection to communicate with the MH. When the MH switches cells, the state associated with the two sockets for the I-TCP connection at BS-1 is handed over to the BS-2. BS-2 then creates the two sockets corresponding to the I-TCP connection with the same endpoint parameters that the sockets at BS-1 had

associated with them. Since the connection endpoints for both wireless and the fixed parts of the I-TCP connection do not change after a move, there is no need to reestablish the connection at the new BS.

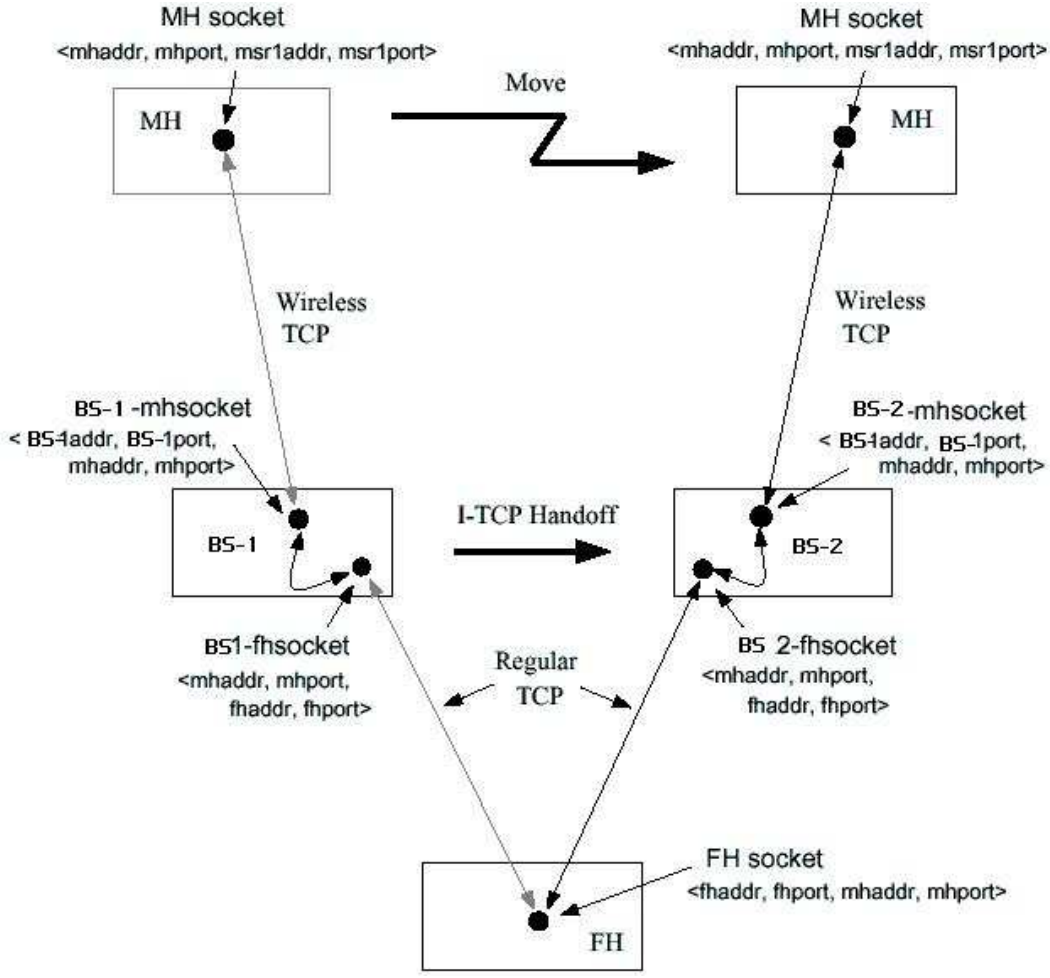

Figure 3.1: I-TCP Connection Setup [2]

3.3.1.1.2 Protocol Evaluation:

- The protocol handles congestion.
- The protocol handles wireless transmissions errors.
- The protocol handles handoffs between cells.
- The protocol does not change TCP on the fixed host.

- The protocol changes TCP on the base station.

3.3.2 Protocols that Preserve the TCP End-to-End Semantics

These protocols share the following common evaluation point(s):

- The protocol preserves the TCP end-to-end acknowledgments and semantics.

3.3.2.1 Berkeley Snoop

The Berkeley Snoop protocol modifies the network layer software on the base station(s) to cache the TCP packets and perform local retransmissions across the wireless links [3].

3.3.2.1.1 Protocol Semantics

- **Data Transfer from a Fixed Host**

  The base station routing code is modified by adding a module called snoop, which monitors every packet that passes through the connection in either direction. The Snoop module maintains a cache of TCP packets sent from the fixed host that have not yet been acknowledged by the mobile host. When a new packet arrives from the fixed host, the snoop module adds it to its cache and passes the packet on to the routing code, which performs the normal routing functions. The snoop module also keeps track of all the acknowledgments sent from the mobile host. When a packet loss is detected (either by the arrival of a DupAck or by local timeout), the snoop module retransmits the lost packet to the mobile host if it has the packet cached. Thus, the base station hides the packet loss from the fixed host by not propagating duplicate acknowledgments, thereby preventing unnecessary congestion control mechanism invocations [3].

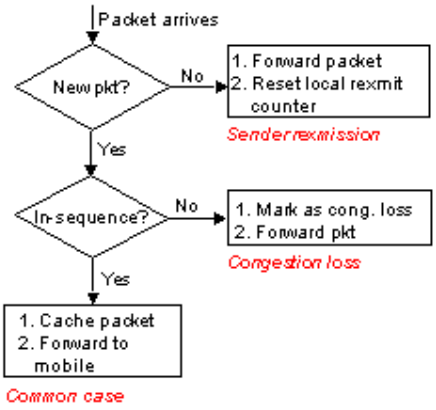

Figure 3.2: Data Transfer from a Fixed Host [3].

- **Data Transfer from a Mobile Host**

  There is no way for the mobile host to know if the loss of a packet happened on the wireless link or elsewhere on the network. Since TCP performs retransmissions on the basis of round-trip estimates for the connection, mobile host timeouts for a packet lost on the wireless link will happen much later than they should. Snoop modifies the TCP code on the mobile host to improve this situation. The base station keeps track of the packets that were lost in any transmitted window, and generates negative acknowledgments (NACKs) for those packets back to the mobile host. These NACKs are sent when either a threshold number of packets have reached the base station or when a certain amount of time has expired without any new packets from the mobile host. The mobile host uses these NACKs to selectively retransmit lost packets [3].

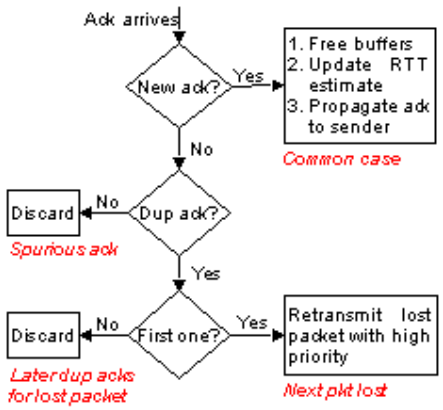

Figure 3.3: Data Transfer from a Mobile Host [3].

- **Mobility Handling**

  Handoffs in the "Berkeley Snoop Module" protocol are based on multicast. Most often, they are mobile-initiated and occur when the mobile host discovers a base station with a stronger signal than the current one. When a handoff is requested by the mobile host or anticipated by the base station, the nearby base stations join a multicast group and begin receiving packets destined for the mobile host. This allows them to begin building up their snoop caches for this mobile host. However, during this period packets and acknowledgments from the mobile host continue to pass only through the primary base station. Therefore, the nearby buffering base stations cannot snoop on any acknowledgments for the caches they are mirroring. This prevents snoop from freeing up packets that have safely reached the mobile host. Once the handoff occurs information to synchronize the snoop cache is sent to the base station that the mobile host has transferred to [3].

3.3.2.1.2 Protocol Evaluation

- The protocol handles congestion.
- The protocol handles wireless transmissions errors.

- The protocol handles handoffs between cells.
- The protocol does not change TCP on the fixed host.
- The protocol does not change TCP on the base station.

3.3.2.2 Multiple Acknowledgements

The Multiple Acknowledgments protocol modifies TCP on the base station and partially acknowledges a packet that is experiencing difficulty on the wireless link and retransmits it [5].

3.3.2.2.1 Protocol Definitions

- Ack_p: Partial acknowledgment with sequence number N_a informs the sender that the base station has received the packet(s) with sequence numbers up to Na – 1.
- Ack_c: Complete acknowledgement that has the same semantics as the normal TCP acknowledgement.
- *RTT*: The round trip time between the TCP Sender and the TCP Receiver.
- RTT_B: The round trip time between the base station and the mobile host.
- *RTO*: The maximum time the TCP Sender waits for an end-to-end acknowledgement.
- RTO_B: The maximum time the base station waits for a mobile host to acknowledge a packet.

3.3.2.2.2 Protocol Semantics

- **The Sender:**

  Ack_p with sequence number N_a, informs the sender that the base station received all the packets with a sequence number up to N_a – 1, and that it is experiencing

problems to forward these packets through the wireless link (these packets spent more than RTO_B at the base station without being successfully transmitted to the mobile host). The sender can identify these packets (these packets have sequence numbers between the last Ack_c and $N_a - 1$) and retransmit them. The sender must update *RTO* to give more time to the base station to accomplish the retransmissions and *RTT* will be updated when either Ack_p or Ack_c is received for a packet [5]. Receiving Ack_c means that the receiver got the packet [5].

- **The base station:**
  - **Data received:**

    On receiving a new packet in the normal TCP sequence, the base station buffers the packet, put a timestamp on it, set the timer RTO_B and tries to forward it to the mobile host. If RTO_B expires then the base station sends back to the sender a cumulative acknowledgement Ack_p with the sequence number of this packet plus 1. On receiving an out of sequence packet that has been buffered earlier, the base station will check the packet sequence number as follows. Let S_a be the sequence number for the last acknowledged packet. If $S > S_a$, and the packet is still in the cache, then Ack_p will be sent back to the sender. Otherwise, it will be processed as a new packet. If $S \leq S_a$, then a fake Ack_c will be generated holding the identity of the mobile host and sent back to the sender. On receiving an out of sequence packet that has not been buffered earlier, the base station will forward the packet to the mobile host [5].
  - **Retransmissions:**

    When RTO_B expires, this means that either the packet has not been sent yet or that it has been lost. The packet must then be retransmitted and an Ack_p is sent back to the sender if the base station has received all packets before it [5].

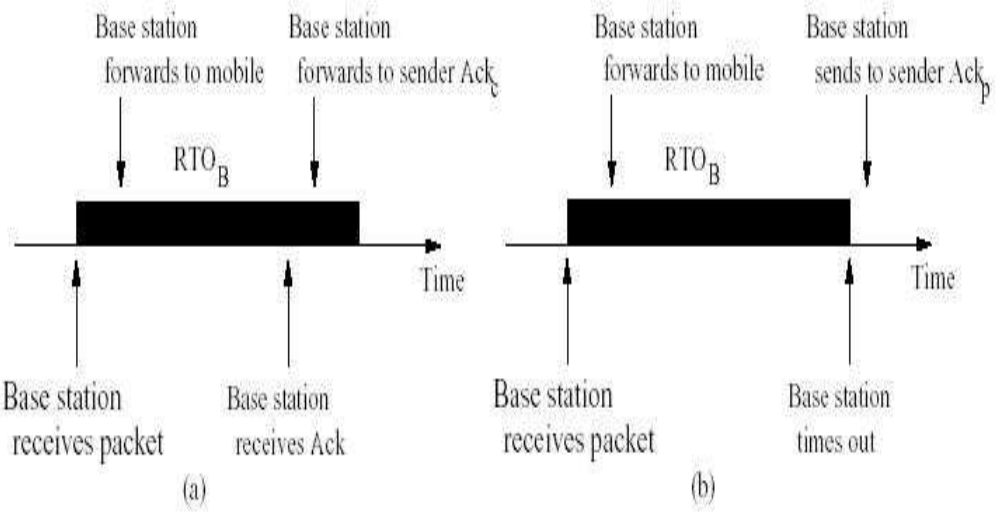

Figure 3.4: (A) No Timeout (B) Timeout [5]

- **New acknowledgement received:**

  Since the protocol must hold even if the TCP sender does not recognize Ack_p, acknowledgments for the receiver must be processed at the base station. These acknowledgments will be processed exactly as in Snoop. If it is a new ACK, the base station will purge its buffer, forward the acknowledgement Ack_c to the sender and update RTT_B. If it is a spurious ACK, It is discarded. If it is a duplicate acknowledgement: The idea here is to forward the necessary acknowledgements to the TCP protocol and to discard those who trigger unnecessary retransmissions from the sender [5].

3.3.2.2.3 Protocol Evaluation:

- The protocol handles congestion.
- The protocol handles wireless transmissions errors.
- The protocol does not handle handoffs between cells.

- The protocol changes TCP on the fixed host.
- The protocol changes TCP on the base station.

3.3.2.3 WTCP

The WTCP protocol modifies TCP on the base station to cache the TCP packets destined to a mobile host and retransmits these packets in case of loss [18].

3.3.2.3.1 Protocol Semantics

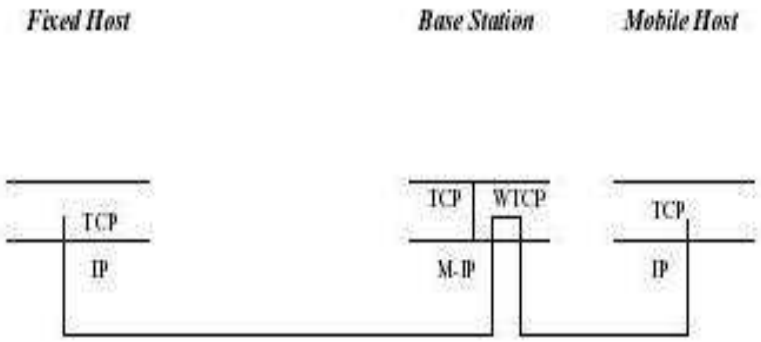

Figure 3.5: WTCP Connection [18]

- **Receiving data segments from the fixed host:**

  The network layer protocol (M-IP) running on the base station detects any TCP segment that arrives for a mobile host and sends it to the WTCP input buffer [18]. If this segment is the next segment expected from the fixed host, it is stored in the WTCP buffer along with its arrival time, and the receive array is updated. The receive array maintains the sequence numbers of the segments received by WTCP. The sequence number for the next segment expected from the fixed host will increase by the number of bytes received. If the newly arriving segment has a larger sequence number than what is expected, the

segment is buffered, the arrival time is recorded and the receive array is updated, but the sequence number of the next packet expected is not changed. If the sequence number of the segment is smaller than what is expected, the segment is ignored [18].

- **WTCP sends data segments to mobile host:**

  On the wireless link, WTCP tries to send the segments that are stored in its buffer. WTCP maintains state information for the wireless connection, such as sequence number of last acknowledgment received from the mobile host, and the sequence number of the last segment that was sent to the mobile host [18]. From its buffer, WTCP transmits all segments that fall within the wireless link transmission window. Each time a segment is sent to the mobile host the timestamp of the segment is incremented by the amount of time that segment spent in the WTCP buffer. When a segment is sent to the mobile host the base station schedules a new timeout if there is no other timeout pending [18].

- **WTCP receives acknowledgement from mobile host:**

  The base station acknowledges a segment to the fixed host only after the mobile host actually receives and acknowledges that segment. Also, the round trip time seen by the fixed host is the actual round trip time taken by a segment to reach and the acknowledgment to return from the mobile host. Based on duplicate acknowledgment or timeout, the base station locally retransmits lost segments. In case of timeout, the transmission window for the wireless connection is reduced to just one segment assuming a typical loss on the wireless link is going to follow [18]. If only one segment is lost and the following segment(s) pass(es) through, the loss would likely be indicated by a duplicate acknowledgment. In case of timeout, by quickly reducing the

transmission window, potentially wasteful wireless transmission is avoided and the interference with other channels is reduced.

3.3.2.3.2 Protocol Evaluation:

- The protocol handles congestion.
- The protocol handles wireless transmissions errors.
- The protocol does not handle handoffs between cells.
- The protocol does not change TCP on the fixed host.
- The protocol changes TCP on the base station.

3.3.2.4 M-TCP

The M-TCP protocol changes TCP on the base station and splits the TCP connection between the fixed host and the mobile host at the base station. The TCP protocol between the base station and the mobile host is a special version of TCP [8].

3.3.2.4.1 Protocol Semantics:

When BS-TCP (Base Station – TCP is a special version of TCP that runs on the base station and communicates with the TCP sender) on the base station receives a segment from the TCP sender, it passes the segment on to the M-TCP (Mobile – TCP is a special version of TCP that runs on the base station and communicates with the mobile host) client on the same base station. However, it does not ACK this data until the mobile host does. The M-TCP client will notify BS-TCP of the mobile host ACKs. To ensure that the sender does not go into congestion control, when ACKs do not arrive, the "M-TCP" protocol chokes the TCP sender by sending an ACK with zero window size advertised when the mobile host is disconnected, and allows the

sender to transmit at full speed when the mobile host reconnects by manipulating the TCP sender's window [8].

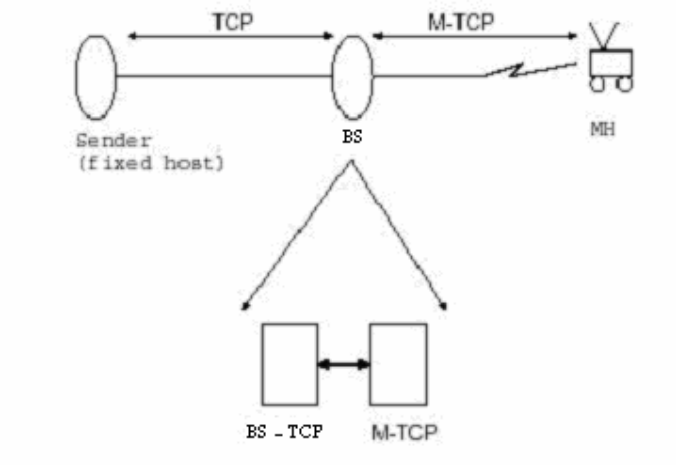

Figure 3.6: M-TCP Connection [8]

Let W denote the currently advertised receive window at BS-TCP. Say the window contains $w <= W$ bytes. Assume that the mobile host has Ack'ed bytes up to $w' <= w$. BS-TCP sends an ACK or ACKs for bytes up to $w' - 1$ in the normal way. As and when the mobile host ACKs more data, more ACKs are generated but one last byte is always left unacknowledged. Say the mobile host disconnects after having Ack'ed bytes up to w'. The M-TCP client assumes that the mobile host has been temporarily disconnected because it stops receiving ACKs for bytes transmitted after w'. M-TCP sends an indication of this fact to BS-TCP who then sends an ACK for the w'th byte to the sender. This ACK will also contain a TCP window size update that sets the sender's window size to zero. When the TCP sender receives the window update, it is forced into persist mode. While in this state, it will not suffer from retransmit timeouts and will not exponentially back off its retransmission timer, nor will it close its congestion window. If the mobile

host has not disconnected but is in a cell with very little available bandwidth, BS-TCP will still send an ACK for byte w' with a window size set to 0. BS-TCP estimates the round trip time (RTT) to the TCP sender and estimates the RTO interval. It uses this information to preemptively shrink the sender's window before the sender goes into exponential back off (to implement this scheme, a timer is maintained at the base station that is initialized to this estimated RTO value). When a mobile host regains its connection, it sends a greeting packet to the base station. M-TCP is notified of this event and it passes on this information to BS-TCP, which in turn, sends an ACK to the sender and reopens its receive window (and hence the sender's transmit window). The window update allows the sender to leave persist mode and begin sending data again. Since the sender never timed out, it never performed congestion control or slow start. Thus the sender can resume transmission at full speed. The sender will begin sending from byte w + 1 here though, possibly duplicating previous sends, so it does not shrink the window more often than necessary.

Handoff between cells is handled by the handoff module. When the handoff starts, M-TCP is notified and the send window is shrunk on the TCP side. Then the connection state is serialized at the old base station and passed to the new base station. At the new base station, the connection state is unserialized and the TCP send window is reopened. The new base station reopens the send window at the TCP sender after handoff is completed.

3.3.2.4.2 Protocol Evaluation

- The protocol handles congestion.
- The protocol handles wireless transmissions errors.

- The protocol handles handoffs between cells.
- The protocol does not change TCP on the fixed host.
- The protocol changes TCP on the base station.

3.4 TCP-Unaware Protocols

These protocols share the following common evaluation point(s):

- The protocol preserves the TCP end-to-end acknowledgments and semantics.
- The protocol is able to handle encrypted traffic.
- The protocol does not change TCP on the base station.
- The protocol does not require any modifications to the base station.

3.4.1 Using Inter-Arrival Times at the Receiver

The Using Inter-Arrival Times at the Receiver protocol uses the packet inter-arrival time at the receiver as an indication to distinguish between packets lost on the wired network and packets lost on the wireless network [6].

3.4.1.1 Protocol Semantics

Figure 3.7 shows three scenarios that may occur when the sender sends packets 1, 2 and 3 to the receiver [6].

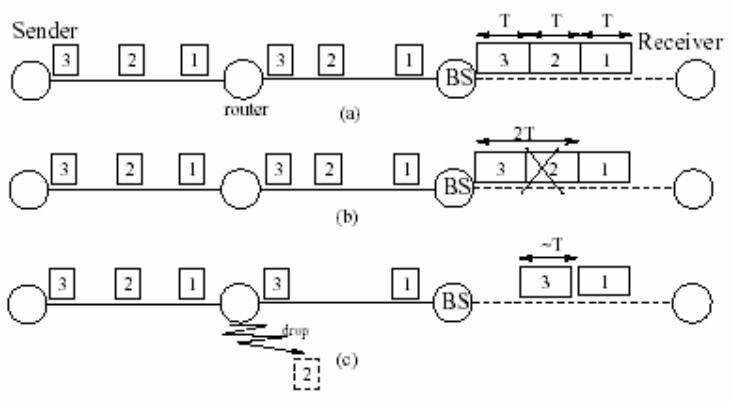

Figure 3.7: Inter-arrival gap [6]

- Figure 3.7 (a) shows that none of the packets is lost. Therefore, the receiver receives all of the packets. In this case, the "packet inter-arrival gap", or the time between arrivals of consecutive packets, is approximately equal to the time *T* required to transmit one packet on the wireless link.
- Figure 3.7 (b), shows that packet 2 is lost when transmitted over the wireless link. In this case, the time between the arrivals of the two packets received by the receiver (i.e., packets 1 and 3) is *2T*, since packet 2(lost due to transmission) uses the wireless link for *T* time units.
- Figure 3.7 (c), shows that packet 2 is lost due to queue overflow (congestion) at the intermediate router. In this case, the inter-arrival gap between packets 1 and 3 will be comparable to *T* (this holds true if packet 3 arrives at the base station either before, or just after the base station has transmitted packet 1). When packet 2 is lost and packet 3 arrives at the receiver, packet 3 is called an out-of-order packet.

Based on the above observations, the following heuristic has been developed:

- Let *Tmin* denote the minimum inter-arrival time observed so far by the receiver during the connection.
- Let *P0* denote an out-of-order packet received by the receiver.
- Let *Pi* denote the last in-sequence packet received before *P0*.
- Let *Tg* denote the time between arrivals of packets *P0* and *Pi*.
- Finally, let the number of packets missing between *Pi* and *P0* be *n* (assuming that all packets are of the same size).

If $(n + 1)$ *Tmin* <= *Tg* < $(n + 2)$ *Tmin*, then the *n* missing packets are assumed to be lost due to wireless transmission errors. Otherwise, the *n* missing packets are assumed to be lost due to congestion.

3.4.1.2 Protocol Evaluation

- The protocol handles congestion.
- The protocol handles wireless transmissions errors.
- The protocol does not handle handoffs between cells.
- The protocol changes TCP on the fixed host.

3.4.2 Delayed Duplicate Acknowledgements

The Delayed Duplicate Acknowledgments protocol implements a link level retransmission scheme at the base station for packets that are lost on the wireless link [20].

3.4.2.1 Protocol Semantics

The base station implements a link level retransmission scheme for packets that are lost on the wireless link due to transmission errors. This link level retransmission scheme will be triggered by using link level acknowledgements [20]. The TCP receiver attempts to reduce interference between TCP and link level retransmissions by delaying third and subsequent DupAcks for some interval *D*. Specifically, when out-of-order (OOO) packets are received, the TCP receiver responds to the first consecutive OOO packets by sending DupAcks immediately. However, DupAcks for further consecutive OOO packets are delayed for duration *D*. If the next in-sequence packet is received within the *D* interval, then the delayed DupAcks are not sent. Otherwise, after the *D* interval, all the delayed DupAcks are released [20].

- **Packet is lost due to transmission errors:**

  If *D* is large enough to allow time for link level retransmission (from the base station) of the lost packet, then the retransmitted packet reaches the TCP receiver before third and subsequent DupAcks can be sent. In this case, the delayed DupAcks will not be sent, and on receiving the retransmitted packet, the receiver will acknowledge receipt of all the in-sequence data (including the retransmitted packet). Since the TCP sender does not receive more than two DupAcks, it will not fast retransmit. If *D* is not large enough to allow time for link level retransmission (from the base station) of the lost packet, the TCP receiver sends all the delayed DupAcks to the TCP sender, but after waiting for *D* time, which can lead to degradation in performance, compared to standard TCP. Standard TCP will send the third DupAck without any delay, thus initiating fast retransmission sooner than the proposed protocol.

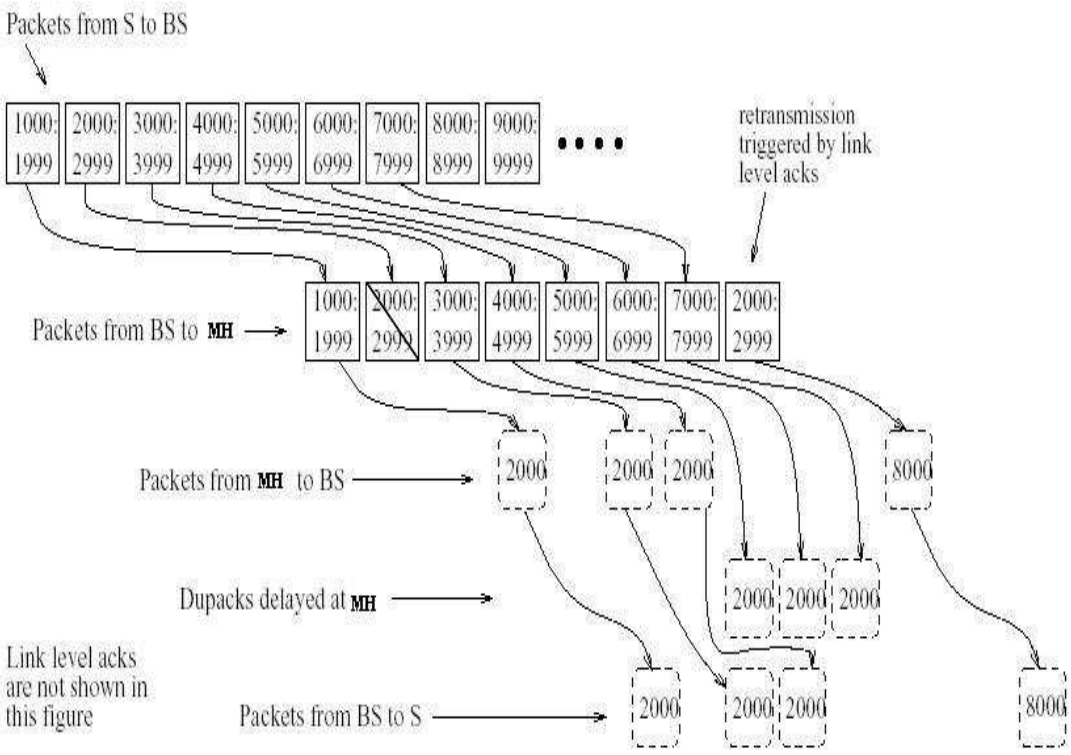

Figure 3.8: An Illustration of Delayed DupAcks Scheme [20]

In Figure 3.8 boxes containing two sequence numbers denote TCP data packets, whereas the boxes containing a single sequence number are TCP ACKs. Recall that a TCP acknowledgement that contains sequence number 2000 denotes that the receiver has received all bytes through 1999, but not byte 2000. A diagonal line through the data packet 2000:2999 sent by the base station (BS) denotes that the packet is lost due to transmission errors. Retransmission of packet 2000:2999 by the base station (BS) is triggered by link level ACKs. The first two DupAcks with sequence number 2000 are sent by the TCP receiver to the base station (BS), and propagated by the base station (BS) to the TCP sender. The third and subsequent DupAcks are delayed at the receiver. In this case, the retransmitted packet is assumed to be delivered to the

receiver within the *D* interval. Hence, receiver MH will discard the delayed DupAcks when the retransmitted packet is received.

- **Packet is lost due to congestion:**

  The TCP receiver will delay the third and subsequent DupAcks for *D* time, which can lead to degradation in performance, compared to standard TCP. Standard TCP will send the third DupAck without any delay, thus initiating fast retransmission sooner than the proposed protocol.

3.4.2.2 Protocol Evaluation

- The protocol does not handle congestion efficiently.
- The protocol handles wireless transmissions errors.
- The protocol does not handle handoffs between cells.
- The protocol does not change TCP on the fixed host.

3.4.3 Fast Retransmissions

The Fast Retransmissions protocol resumes communication immediately after handoffs between cells complete, without waiting for a retransmission timeout [9].

3.4.3.1 Protocol Semantics

The idea behind the Fast Retransmissions protocol is for the transport protocol to resume communication immediately after handoffs between cells complete, without waiting for a retransmission timeout. It achieves that by causing the TCP receiver to send triplicate acknowledgments to the TCP sender, which will activate the fast retransmission procedure. When activated, the fast retransmission procedure

immediately retransmits the earliest unacknowledged packet, drops the transmission window, and initiates the fast-recovery algorithm. The rationale behind fast retransmissions is that triplicate acknowledgments clearly indicate that packet loss has occurred, and thus there is no need to wait for a timeout before retransmission. As soon as routes become consistent following a handoff between cells, the fast retransmission procedure gets invoked as follows: First, the Mobile IP software on the mobile host signals the TCP software on the mobile host when a greeting acknowledgment arrives from the new base station. Second, the TCP software on the mobile host forwards the signal (three identical but ordinary TCP acknowledgment packets) over the network to the fixed host. Third, the TCP software on the fixed host invokes the fast retransmission procedure when it receives such signal [9].

3.4.3.2 Protocol Evaluation

- The protocol handles congestion.
- The protocol does not handle wireless transmissions errors.
- The protocol handles handoffs between cells.
- The protocol does not change TCP on the fixed host.

3.4.4 Freeze-TCP

The Freeze-TCP protocol puts the TCP sender into a persist state where it freezes all of its re-transmit timers and congestion window.

3.4.4.1 Protocol Semantics

A mobile node can monitor signal strengths in the wireless antennas and detect an impending handoff. In such a case, it can advertise a zero window size, and send out a few ZWA (Zero Window Advertisement) acknowledgements. The sender, upon seeing a ZWA, freezes all of its re-transmit timers, congestion window and enters a persist mode, in which it will send ZWPs (Zero Window Probes) until the receiver's window opens up. The interval between successive ZWPs grows exponentially (exponential back-off) until it reaches 1 minute, where it remains constant. Eventually the receiver responds to a ZWP with a non-zero window size, and the sender will continue transmission using a window size consistent with the advertised value [14].

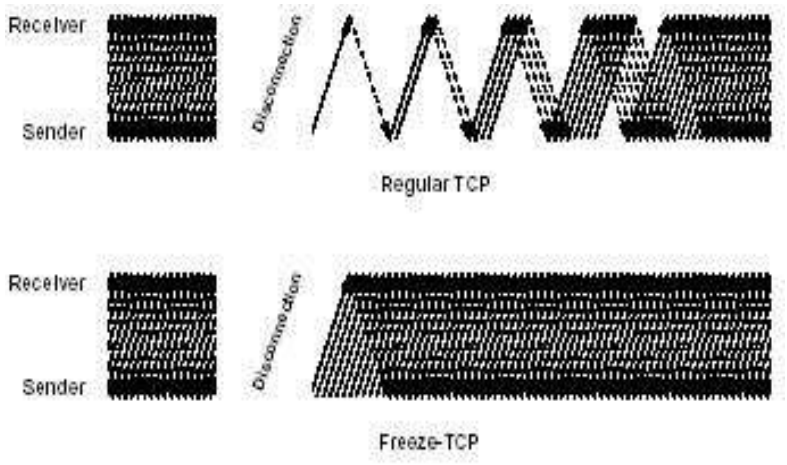

Figure 3.9: Illustration of Increased Throughput Due to Freeze-TCP [14]

Figure 3.9 illustrates the increase of the TCP throughput due to the use of Freeze-TCP. Regular TCP, when the sender does not receive an ACK for a previously sent packet, well eventually timeout, assuming that there is congestion and will drop its transmission window size before retransmitting packets, initiate congestion control or

avoidance mechanisms (e.g., slow start) and reset its retransmission timer. This will lead to degradation in the TCP throughput, which does not happen in the case of Freeze-TCP.

The question is *"How much in advance of the disconnection should the receiver start advertising a window size of zero?"* This period is the *"Warning Period"* prior to disconnection. Ideally, the warning period should be long enough to ensure that exactly one ZWA gets across to the sender. If the warning period is any longer, the sender will be forced into ZWP mode prematurely, thereby leading to idle time prior to the disconnection. If the warning period is too small, there might not be enough time for the receiver to send out a ZWA, which will cause the sender's congestion window to drop due to packets lost during the disconnection. So this warning period should be equal to RTT (The round trip time between the TCP Sender and the TCP Receiver).

3.4.4.2 Protocol Evaluation

- The protocol handles congestion.
- The protocol does not handle wireless transmissions errors.
- The protocol handles handoffs between cells.
- The protocol does not change TCP on the fixed host.

3.4.5 Establishing Two Connections

The Establishing Two Connections protocol establishes a TCP connection from the fixed host to the base station that serves as control connection, to estimate the network congestion between the fixed host and the mobile host.

3.4.5.1 Protocol Semantics

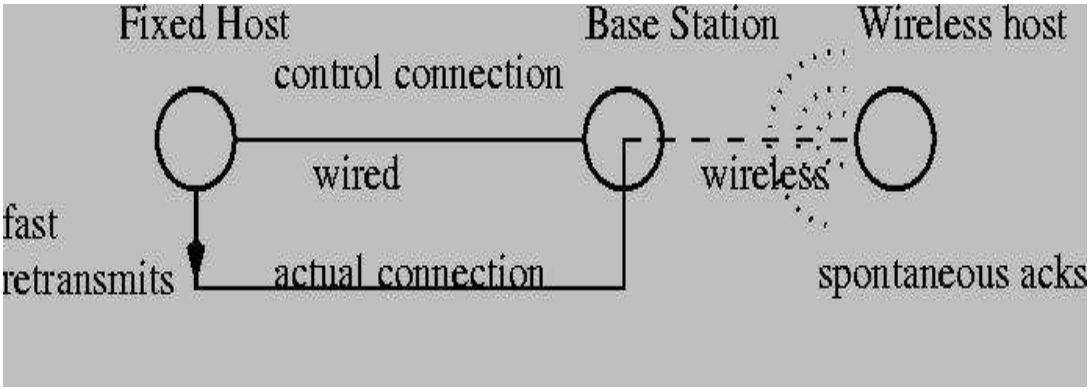

Figure 3.10: Model of Proposed TCP [4]

- **The Sender:**

  When a TCP connection is to be opened between a fixed host and a mobile host, the sender opens two independent connections, one between the fixed host and the base station and the other between the fixed host and the mobile host [4]. The connection between the fixed host and the base station serves as a control connection, to estimate the network congestion that occurs between the two ends. With the assumption that the mobile host receives all packets through the base station, the protocol expects that packets on both connections should be routed by almost the same route, and would see the same network congestion. Packets on the control connection are sent at some regular interval. The TCP sender will also send packets on the connection with the mobile host. Comparing the fraction of the packets that are acknowledged within the timeout on the two connections, the TCP sender will determine if there is congestion on the wired part of the connection, which is causing packets to be lost, or there are transmission errors on the wireless part of the

connection. If the fraction is significantly different in the two cases, then the conclusion is that transmission errors are causing the packet loss, in this case, the TCP sender will be prevented from reducing the congestion window size, and it will continue its linear increment at the same rate. If the two fractions are the same, then the normal congestion control mechanisms are allowed to proceed as it would in ordinary TCP.

- **The Mobile Host:**

  At the mobile host side, to prompt the sender to retransmits packets that are lost due to transmissions errors or otherwise, the fast retransmissions technique is employed. The sender, on seeing a DupAck, decides that its previous packet sent has been lost or corrupted. Hence, it retransmits the subsequesnt packet even before the timer expires and restarts the timer.

3.4.5.3 Protocol Evaluation

- The protocol handles congestion.
- The protocol handles wireless transmissions errors.
- The protocol does not handle handoffs between cells.
- The protocol changes TCP on the fixed host.

3.5 Summary

✓ Means that the protocol satisfies this evaluation criteria.

X Means that the protocol does not satisfy this evaluation criteria.

	TCP end-to-end ACKs and semantics	Congestion	Wireless transmission errors	Handoffs between cells	Encrypted traffic	No Change for TCP on the fixed hosts	No Change for TCP on the base stations
Berkeley Snoop	✓	✓	✓	✓	X	✓	✓
Multiple ACKs	✓	✓	✓	X	X	X	X
WTCP	✓	✓	✓	X	X	✓	X
M-TCP	✓	✓	✓	✓	X	✓	X
I-TCP	X	✓	✓	✓	X	✓	X
Using Inter-Arrival Times at the Receiver	✓	✓	✓	X	✓	✓	✓
Delayed DupAcks	✓	X	✓	X	✓	✓	✓
Fast Retransmissions	✓	✓	X	✓	✓	✓	✓
Freeze-TCP	✓	✓	X	✓	✓	✓	✓
Making Two Connections	✓	✓	✓	X	✓	X	✓

Table 3.1: Cellular Wireless Network Protocols Evaluation Summary

Table 3.1 shows that the Berkeley Snoop, the Using Inter-Arrival Times at the Receiver, the Fast Retransmissions and the Freeze-TCP protocols are the only protocols that meets most of the evaluation criteria. The Fast Retransmissions and the Freeze-TCP protocols are unable to handle wireless transmission errors. This is a very

important criteria, as it affects the TCP throughput. At the same time, there is no easy work around for it. The Inter-Arrival Times at the Receiver is unable to handle handoffs between cells. This is another important criteria, as it affects the TCP throughput. At the same time, there is no easy work around for it. The Berkeley Snoop protocol handles wireless transmission errors and handoffs between cells but unable to handle encrypted traffic. This is a less important criteria than wireless transmission errors and handoffs between cells, and there are possible ways to work around this problem. It could be solved by making the SNOOPing base station a party to the security association between the client and the server or terminate the IPsec tunnelling mode at the SNOOPing base station.

Chapter 4

TCP over Multi-Hop Wireless Networks

In Chapter 3, we discussed and evaluated transmission control protocols that have been designed or proposed for TCP over a Single-Hop Wireless Networks (Cellular Network Architecture). In this chapter we discuss and evaluate transmission control protocols that have been designed or proposed for TCP over a Multi-Hop Wireless Networks (Mobile Ad-Hoc Network Architecture). A Multi-Hop Wireless Network is a collection of mobile hosts that can dynamically form a network without using any pre-existing network infrastructure. Communications over wireless links are characterized by limited bandwidth, high latencies, high bit-error rates and temporary disconnections, that network protocols and applications must deal with. In addition, network protocols and applications have to handle route failure, which is caused by the rapid changes in the topology of the MANET due to the mobility of the mobile hosts.

4.1 MANET Architecture Characteristics

A MANET has a wireless, multi-hop network architecture that has the following characteristics:

- Each mobile host functions as both a host and a router.
- The control of the network is distributed among the mobile hosts, and each mobile host acts as a relay as needed.
- The topology of the MANET potentially changes every time a mobile host moves, which requires efficient routing protocols to maintain the multi-hop routes.
- The rate of topology change is dependent on the extent of mobility and the transmission range of the mobile hosts.
- Routes are heavily dependent on the relative location of the mobile hosts.
- Routes may be repeatedly invalidated in a sporadic and arbitrary fashion due to the mobility of the mobile hosts.
- The mobility of a single mobile host may affect several routes that pass through it.

4.2 Evaluation Criteria

The MANET architecture characteristics are different from the Cellular Network architecture characteristics, so the evaluation criteria for the transmission control protocols designed or proposed for the MANET Architecture are different from the evaluation criteria for the transmission control protocols designed or proposed for the Cellular Network Architecture. We identified evaluation criteria that we are going to use throughout this chapter to evaluate these transmission control protocols. These evaluation criteria will help us choose the most suitable transmission control protocol for the MANET architecture. The evaluation criteria are divided into two sets, the first set contains evaluation criteria which are common between the Cellular Network

architecture and the MANET architecture, and the second set are specific to the MANET architecture.

The following is the set of common evaluation criteria:

- **TCP end-to-end acknowledgments and semantics**
- **Congestion**
- **Wireless transmissions errors**
- **Encrypted traffic**

The following are the MANET architecture specific evaluation criteria:

- **Route Failure:** The protocol must be able to distinguish between packet losses due to route failure caused by the mobility of the mobile hosts, and packet losses due to congestion.
- **Route Congestion Window:** The protocol must not use the old route congestion window (cwnd) size for the new route, since it is unlikely that the conditions that existed on the old route that determined the congestion window for this route will be the same conditions on the new route and will therefore result in the same congestion window.

4.3 TCP Feedback

The TCP Feedback protocol utilizes the network layer feedback. In case of a route failure, the TCP sender is sent a Route Failure Notification packet, allowing it to enter persist state. When the route is reestablished the TCP sender is sent a Route Reestablishment Notification packet, allowing it to resume transmission [10].

4.3.1 Protocol Assumptions

- The wireless links are bi-directional.
- A reliable data link layer protocol is implemented.
- A suitable routing protocol is implemented to establish and maintain routes between a source and a destination.
- All packets carry the source and destination id's so that the network layer can identify the source and destination address of each packet.

4.3.2 Protocol Semantics

- **Route Failure Notification (RFN):**

  When a mobile host detects a route failure due to the mobility of the next mobile host along that route, it sends a RFN packet to the source. Each intermediate node along the path that receives the RFN packet discards the route, dropping any data packets, and forwards the RFN packet backwards, unless it has an alternate route to the destination, in which case, the RFN packet is discarded. When the source (i.e. TCP sender) receives the packet, it goes into persist state, in which it stops sending packets, invalidates its existing timers and freezes its TCP window, and state. It then sets a timer for route reestablishment timeout [10].

- **Route Reestablishment Notification (RRN):**

  If an intermediate mobile host that has previously forwarded a RFN to the source discovers a new route to the destination, it will forward a RRN to the source. Any other RRNs that are received by a mobile host on the path are discarded [10]. When the source (i.e. TCP sender) receives the RRN, it goes

into an active state in which it flushes out all the unacknowledged packets in its current window, thaws it's frozen TCP timers, window, and state, and resumes transmissions as if the failure had never occurred.

- **Route Failure Timer:**

  If a RRN does not arrive, the Route Failure Timer ensures that the source (i.e. TCP sender) does not persist indefinitely. When the timer expires, the source behaves as if it had received a RRN, hence, relying on the normal TCP mechanisms to handle the failure [10].

4.3.3 Protocol Evaluation

- The protocol preserves the TCP end-to-end acknowledgments and semantics.
- The protocol handles congestion.
- The protocol does not handle wireless transmissions errors.
- The protocol handles route failures.
- The protocol uses the old route congestion window size for the new route.
- The protocol is unable to handle encrypted traffic.

4.4 ATCP: Ad-Hoc TCP

The ATCP protocol utilizes the network layer feedback. In case of route failure, an ICMP message, "Destination Unreachable", will allow the TCP sender to enter persist state. The ECN flag differentiates between packets losses due to congestion and packet losses due to wireless transmission errors. In case of loss the TCP sender enters persist state and in case of congestion ATCP will not interfere with TCP congestion control algorithms [17].

4.4.1 Protocol Semantics

ATCP is a layer between IP and TCP that listens to the network state information provided by ECN (Explicit Congestion Notification) messages and by ICMP "Destination Unreachable" messages and then puts TCP into the appropriate state. Thus, on receipt of a "Destination Unreachable" message, the TCP sender enters the persist state until a new route is found, ensuring that the sender does not invoke the congestion control algorithms [17]. ECN is used as a mechanism by which TCP is notified of impending network congestion along the route. On receipt of an ECN, the sender invokes congestion control without waiting for a timeout event [17]. Let us consider Figure 4.2, which illustrates ATCP's four possible states (normal, congestion, loss and disconnected).

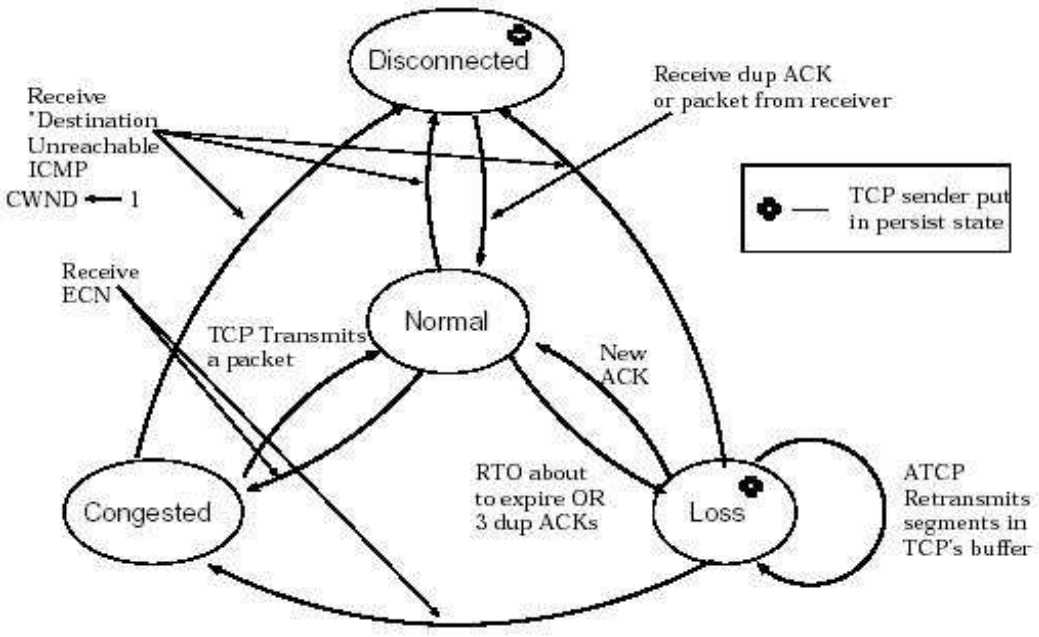

Figure 4.2: State Transition Diagram for ATCP at the Sender [17]

When the TCP connection is initially established, ATCP at the sender is in the normal state. In this state, ATCP does nothing and is invisible. Let us now examine ATCP's behaviour under five circumstances [17]:

- Loss: ATCP in its normal state counts the number of duplicate ACKs received for any segment. When it sees that three DupAcks have been received, it does not forward the third DupAck but puts TCP in persist mode. Similarly, when ATCP sees that TCP's RTO is about to expire, it again puts TCP in persist mode. After ATCP puts TCP in persist mode, ATCP enters the loss state. In the loss state, ATCP transmits the unacknowledged segments from TCP's send buffer. It maintains its own separate timers to retransmit these segments in the event that ACKs are not forthcoming. Eventually, when a new ACK arrives (i.e., an ACK for a previously unacknowledged segment), ATCP forwards that ACK to TCP, which also removes TCP from persist mode. ATCP then returns to its normal state [17].
- Congested: ATCP assumes that when the network detects congestion, the ECN flag is set in ACK and data packets. Let us assume that ATCP receives this message when in its normal state. ATCP moves into its congested state and does nothing. It ignores any DupAcks that arrive and it also ignores imminent RTO expiration events. In other words, ATCP does not interfere with TCP's normal congestion behaviour. After TCP transmits a new segment, ATCP returns to its normal state [17].
- Disconnected: Node mobility in ad hoc networks causes route re-computation or even temporary network partition. When this happens, ATCP assumes that the network generates an ICMP “Destination Unreachable” message in response to a packet transmission. When ATCP receives this message, it puts

the TCP sender into persist mode and enters the disconnected state. TCP periodically generates probe packets while in persist mode. When, eventually, the receiver is connected to the sender, it responds to these probe packets with an ACK. This removes TCP from persist mode and moves ATCP back into normal state. In order to ensure that TCP does not continue using the old CWND value, ATCP sets TCP's CWND to one segment at the time it puts TCP in persist state. The reason for doing this is to force TCP to probe the correct value of CWND to use for the new route [17].

- Other Transitions: Finally, when ATCP is in the loss state, reception of an ECN or an ICMP “Source Quench” message will move ATCP into congested state and ATCP removes TCP from its persist state. Similarly, reception of an ICMP “Destination Unreachable” message moves ATCP from either the loss state or the congested state into the disconnected state and ATCP moves TCP into persist mode [17].
- Effect of Lost Messages: If an ECN message is lost, the TCP sender will continue transmitting packets. However, every subsequent ACK will contain the ECN thus ensuring that the sender will eventually receive the ECN causing it to enter the congestion control state as it is supposed to. Likewise, if there is no route to the destination, the sender will eventually receive a retransmission of the ICMP “Destination Unreachable" message causing TCP to be put into the persist state by ATCP. Thus, in all cases of lost messages, ATCP performs correctly [17].

4.4.2 Protocol Evaluation

- The protocol preserves the TCP end-to-end acknowledgments and semantics.
- The protocol handles congestion.
- The protocol handles wireless transmissions errors.
- The protocol handles route failure.
- The protocol does not use the old route congestion window size for the new route.
- The protocol is able to handle encrypted traffic.

4.5 TCP-DOOR

The TCP-DOOR protocol improves the TCP performance by detecting and responding to out-of-order packet delivery events, which are results of frequent route changes.

4.5.1 Protocol Semantics

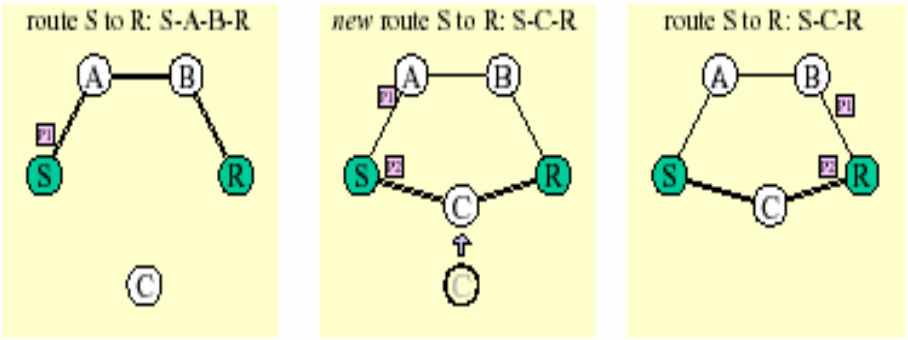

Figure 4.3: A Possible Case of Route Change [21]

- **Out-of-Order Delivery:**

  In an ideal TCP session, the packets sent by one end should arrive at the other end in sequence and in order. However, there are two cases in which this ideal order can be violated. The first case is the Out-of-Sequence event where retransmissions occur due to packet losses. The second case is the Out-of-Order (OOO) event where a packet sent earlier arrives later than a subsequent packet.

  OOO often implies route changes in the network. Consider two hosts communicating over a fixed route in the network. Assuming the FIFO queues are used at every node, all packets should be received in the same order as they are sent. However, if the route changes during the communication session so that a later packet P2 takes a different path than an earlier packet P1 (Figure 4.3), and if the new route taken by P2 is faster or shorter than the one taken by P1, P2 can arrive before P1, therefore OOO delivery can happen.

- **Detecting Out-of-Order Events:**
  - Sender Detecting Out-of-Order ACK Packets (D1)

    When the sender receives an ACK, it compares the sequence number it carries with the one in the previous ACK, and if the current sequence number is smaller, the sender can surely declare OOO. Detecting OOO delivery of duplicate ACK packets requires additional ordering information, because otherwise two duplicate ACK packets would have identical content. To achieve this, a one-byte TCP option was added to the TCP ACK header, called ACK duplication sequence number (ADSN). When the TCP receiver sends the first ACK packet for a data segment, the ADSN option is set to zero. It increments the ADSN number whenever it sends out a duplicate ACK for the same sequence number. This way, each duplicate ACK will carry a different

ADSN field and the TCP sender can compare this field to detect an OOO delivery.

- Receiver Detecting Out-of-Order Data Packets

  To reliably detect OOO data packets, strict ordering information must be included in the TCP data packet. A two-byte TCP option was added to the TCP header, called TCP packet sequence number (TPSN). Starting from zero, and incremented with every data packet sent (including retransmissions), this TPSN records the exact order of the data packet stream. With TPSN option, the TCP receiver can detect OOO reliably.

- **Responding to Out-of-Order Events**
  - Temporarily Disabling Congestion Control

    Since OOO is likely to be caused by route changes and not by congestion, the TCP sender should temporarily disable the congestion control algorithms whenever an OOO condition is detected. That is, for a time period T1 after detecting an OOO, the TCP sender will keep its state variables constant, such as the retransmission timer (RTO) and the congestion window size.

  - Instant Recovery during Congestion Avoidance

    The detection of the OOO condition implies that a route change event has just happened. Therefore, if during the past time period T2 the TCP sender has suffered from congestion and entered the congestion avoidance state, it should recover immediately to the state before such congestion avoidance action was invoked. The fact that the TCP sender has begun receiving ACK means that the disruption is over and the TCP should quickly resume the pre-route-change state, without going through slow start or linear window recovery. Similarly, if congestion avoidance was triggered by a three-duplicate-ACK event, these

duplicate ACKs were likely caused by OOO deliveries or temporary route disruption and not by congestion losses. And likewise, this condition would have passed and TCP should resume its previous state.

4.5.2 Protocol Evaluation:

- The protocol preserves the TCP end-to-end acknowledgments and semantics.
- The protocol handles congestion.
- The protocol does not handle wireless transmissions errors.
- The protocol handles route failure.
- The protocol uses the old route congestion window size for the new route.
- The protocol is able to handle encrypted traffic.

4.6 TCP-Fixed RTO

The TCP-Fixed RTO protocol assumes that consecutive timeouts are an indication of route failure rather than congestion [12].

4.6.1 Protocol Semantics

The fixed RTO technique does not rely on the feed-back from lower layers. A heuristic is employed to distinguish route failures and congestion. When timeouts occur consecutively, the sender assumes a route failure happened rather than network congestion. The unacknowledged packet is retransmitted again but the RTO is not doubled a second time. The RTO remains fixed until the route is re-established and the retransmitted packet is acknowledged [12].

4.6.2 Protocol Evaluation:

- The protocol preserves the TCP end-to-end acknowledgments and semantics.
- The protocol does not handle congestion.
- The protocol does not handle wireless transmissions errors.
- The protocol handles route failure.
- The protocol uses the old route congestion window size for the new route.
- The protocol is able to handle encrypted traffic.

4.7 Summary

✓ Means that the protocol satisfies this evaluation criteria.

X Means that the protocol does not satisfy this evaluation criteria.

	TCP end-to-end ACKs and semantics	Congestion	Wireless transmission errors	Route Failure	Route Congestion Window	Encrypted traffic
TCP-Feedback	✓	✓	X	✓	X	X
ATCP	✓	✓	✓	✓	✓	✓
TCP-DOOR	✓	✓	X	✓	X	✓
TCP-Fixed RTO	✓	✓	X	✓	X	✓

Table 4.1: MANET Protocols Evaluation

Table 4.1 shows that the ATCP protocol is the only protocol that meets all the evaluation criteria.

Chapter 5

The Proposed Scheme & Simulation Environment

In Chapter 3, we discussed and evaluated transmission control protocols that have been designed for TCP over Single-Hop Wireless Networks (Cellular Architecture), and in Chapter 4 we discussed and evaluated transmission control protocols that have been designed for TCP over Multi-Hop Wireless Networks (Standalone–MANET Architecture). In this chapter we discuss, propose and evaluate a transmission control scheme for TCP over Multi-Hop Wireless Access Networks (Wired–MANET Architecture).

5.1 The Proposed Scheme

5.1.1 The Requirements

We identified requirements that must or should exist in the protocol(s) we are going to choose to be part of our proposed scheme. Below is a list of the requirements:

- The protocol must preserve the TCP end-to-end acknowledgments and semantics.

- The protocol must handle congestion as efficiently as TCP.
- The protocol must handle wireless transmissions errors.
- The protocol must handle route failure.
- The protocol must handle handoffs between base stations.
- The protocol must use a new congestion window size for the new route.
- The protocol must not change TCP on the fixed hosts.
- The protocol must not change TCP on the base stations.
- The protocol should handle encrypted traffic.

5.1.2 The Scheme

We compared the requirements from the previous section with the evaluation that was done in Chapter 4 and concluded that the ATCP protocol is the best protocol that matches these requirements. ATCP was designed for a Standalone-MANET Architecture, because of that the following requirement is not met "The protocol must not change TCP on the fixed hosts". Applying the ATCP protocol to a Wired–MANET Architecture would definitely change TCP on the fixed hosts. To avoid changing TCP on the fixed hosts and violating one of our requirements, we needed another protocol that could be applied on the base station and at the same time will not interfere with the ATCP protocol on the MANET. To be able to choose this protocol we had to do the same comparison we did before again, but this time we compared the requirements with the evaluation that was done in Chapter 3. We concluded that the Snoop protocol is the best protocol that matches these requirements. The only requirement that is not met by the Snoop protocol is its inability to handle encrypted traffic, which could be solved as we mentioned earlier in the Summary section of Chapter 3.

As a result from the previous discussion we are proposing to use the TCP protocol on the wired network, the Snoop protocol on the base station and the ATCP protocol on the MANET. The Snoop protocol enables us to avoid changing TCP on the fixed host (FH) and to limit the change to the base station and the mobile host(s). A very important question to answer is *"How would the TCP, ATCP and Snoop protocols interact together?"*

5.1.3 The Interaction between TCP, ATCP and Snoop

It is very important to define the interaction between TCP, ATCP and Snoop to make sure that there are no conflicts between these three protocols. To understand how these three protocols will interact together to improve the TCP throughput, let us consider the following four scenarios:

- Data sent from a fixed host to another fixed host

  In this case the data transmission is located on the wired network (The base station and the MANET are not involved). The TCP protocol will control the data transmission.

- Data sent from a mobile host to another mobile host

  In this case the data transmission is located on the MANET (The base station and the wired network are not involved). The ATCP protocol will control the data transmission.

- Data sent from a fixed host to a mobile host

  Let us assume that we have a data stream that is being transferred from a TCP sender running on the fixed host which is located on the wired network to an ATCP receiver running on the mobile host which is located on the MANET

through the Snoop protocol which is located on the base station. The following five states are possible:

- Normal State (The Entire Network)
    - The TCP sender will send the data packets.
    - The Snoop protocol will cache the data packets.
    - The ATCP receiver will receive and acknowledge the data packets.
    - The Snoop protocol will remove the data packet from its cache.
    - The TCP sender will receive the acknowledgments.
- Congested State (The Wired Network)
    - The TCP sender will send the data packets.
    - The data packets will be dropped before being cached by the Snoop protocol.
    - The TCP sender will know about the congestion either by receiving three duplicate acknowledgments (DupAcks) or by its RTO timer expiration events and will invoke the congestion control algorithms.
- Congested State (The MANET)
    - The TCP sender will send the data packets.
    - The data packets will be dropped after being cached by the Snoop protocol.
    - The ECN flag will be set.
    - The Snoop protocol will ignore duplicate acknowledgments (DupAcks) and its RTO timer expiration events (i.e. it will not try to send the data packets from its own cache so it does not add to the level of congestion already experienced on the MANET).

- The TCP sender will know about the congestion either by receiving three duplicate acknowledgments (DupAcks) or by its RTO timer expiration events and will invoke the congestion control algorithms.

- Loss State (The MANET)
  - The TCP sender will send the data packets.
  - The data packets will be lost after being cached by the Snoop protocol.
  - The Snoop protocol will know about the loss either by receiving duplicate acknowledgment (DupAck) or by its RTO timer expiration events and will send the lost data packets from its own cache and put the TCP sender into persist state by sending an ACK with a zero receiver window size (The Snoop protocol differentiates between the loss and congestion by checking the ECN flag).
- Disconnected State (The MANET)
  - The TCP sender will send the data packets.
  - The data packets will be lost after being cached by the Snoop protocol.
  - The Snoop protocol will know about the loss either by NACKs (Acknowledgments with a negative sequence number generated by the routing protocol to inform the sender about the route failure), receiving three duplicate acknowledgments (DupAcks) or its RTO timer expiration events and will send the lost data packets from its own cache and put the TCP sender into persist state by sending an ACK with a zero receiver window size.

- Data sent from a mobile host to a fixed host

  Let us assume that we have a data stream that is being transferred from an ATCP sender running on the mobile host which is located on the MANET to a TCP

receiver running on the fixed host which is located on the wired network through the Snoop protocol which is located on the base station. The following five states are possible:

- Normal State (The Entire Network)
    - The ATCP sender will send the data packets.
    - The TCP receiver will receive and acknowledge the data packets.
    - The ATCP sender will receive the acknowledgments.
- Congested State (The Wired Network)
    - The ATCP sender will send the data packets.
    - The data packets will be dropped.
    - The ATCP sender will know about the congestion either by receiving three duplicate acknowledgments (DupAcks) or by its RTO timer expiration events and will invoke the congestion control algorithms.
- Congested State (The MANET)
    - The ATCP sender will send the data packets.
    - The data packets will be dropped.
    - The ECN flag will be set.
    - The ATCP sender will know about the congestion either by receiving three duplicate acknowledgments (DupAcks) or by its RTO timer expiration events and will invoke the congestion control algorithms.
- Loss State (The MANET)
    - The ATCP sender will send the data packets.
    - The data packets will be lost.
    - The ATCP sender will know about the loss either by receiving three duplicate acknowledgments (DupAcks) or by its RTO timer expiration

events and will put itself into persistent state (The ATCP sender differentiates between the loss and congestion by checking the ECN flag).

- Disconnected State (The MANET)
    - The ATCP sender will send the data packets.
    - The data packets will be lost before being cached by the Snoop protocol.
    - The ATCP sender will know about the loss by receiving NACKs from the routing protocol and will put itself into persistent state.

5.2 Simulation

5.2.1 Simulation Environment

Our simulation environment is based on the *NS* network simulator [13] with extensions from the CMU Monarch project [11] [7]. The extensions include a set of MANET routing protocols, an implementation of the BSD ARP protocol, the modeling of an IEEE 802.11 wireless LAN, a radio propagation model and mechanisms to model node mobility using pre-computed mobility patterns. We chose to keep most of the simulations parameters identical to those in [7] and [15] with a few exceptions to be able to compare and validate the results.

5.2.2 Network Topology

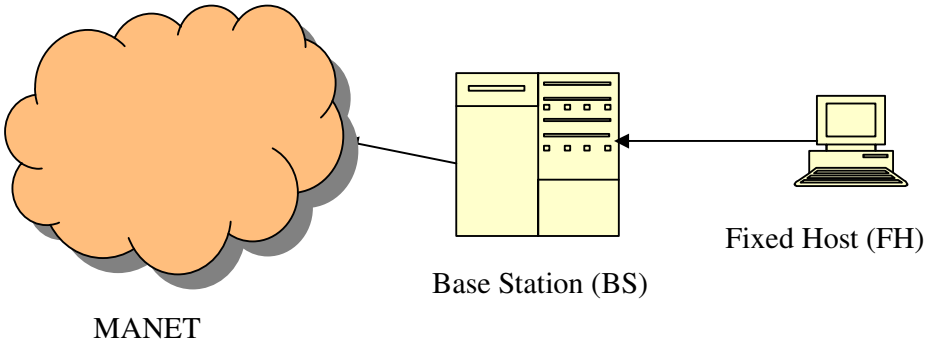

Figure 5.1: Wired–MANET Architecture Topology

Our network topology consists of two networks (Wired Network and Wireless Network). The wired network consists of one fixed host (FH) and a base station (BS), the fixed host (FH) is connected to the base station (BS) and both of them are located at the short edge of the simulation field. The wireless network consists of 50 mobile hosts that are randomly placed on a 1500 meter * 300 meter field at the beginning of the simulation.

5.2.3 Traffic Profile

One TCP connection is established from a fixed host sender (FH) to a mobile host receiver (MH). Over it a FTP file transfer is conducted for 900 seconds. The TCP packet size is 536 bytes, and the maximum congestion and receiver window is eight packets.

5.2.4 Wired Network Simulation Environment

5.2.4.1 MAC Layer

The *NS* uses the IEEE 802.3 MAC layer as the default MAC layer for the wired network.

5.2.4.2 Links

Full duplex links with a 10 Mbps bandwidth and 10 ms delay which should simulate wired links.

5.2.5 MANET Simulation Environment

5.2.5.1 MAC Layer

We use the IEEE 802.11 MAC layer for the MANET. This choice was based on the availability in the *NS* simulator and because most of the research that we refer to in the thesis uses it. The extensions from the CMU Monarch project include the modeling of an IEEE 802.11 MAC layer, with a bandwidth of 2 Mbps and radio transmission radius of 250 m.

5.2.5.2 Routing Protocol

We use the DSDV routing protocol. This choice was based on the availability in the *NS* simulator and because DSDV is the only routing protocol in the *NS* simulator that works with the Wired-MANET architecture. The extensions from the CMU Monarch project include an implementation of the DSDV routing protocol.

5.2.5.3 Mobile Hosts Movement

In the *NS*, mobile host(s) moves according to the random waypoint mobility model. In the random waypoint model, each mobile node chooses a random destination in the field and a random speed that is uniformly distributed between 0 m/s and 20 m/s and

then travels to the destination in a straight line. Once the mobile node arrives at its destination, it pauses, chooses another destination, and continues. We used a pause time of 0 so that each node is in constant movement throughout the simulation.

5.2.6 Simulation Summary

The following tables give a summary of the simulation parameters we discussed earlier:

Number of Fixed Hosts (FH)	1
Number of Base stations	1
Number of Mobile Hosts	50
Field Size	1500 m * 300 m
Simulation Time	900 s
TCP Packet Size	536 bytes
Maximum Congestion Window	8
Maximum Receiver Window	8

Table 5.1 Summary of the Network Topology Simulation Parameters

MAC Layer	IEEE 802.3
Link Bandwidth	10 Mbps
Link Delay	10 ms

Table 5.2 Summary of the Wired Network Simulation Parameters

MAC Layer	IEEE 802.11
Link Bandwidth	2 Mbps
Radio Transmission Radius	250 m
Routing Protocol	DSDV
Mobile Host Speeds	0, 5, 10, 15 and 20 m/s
Pause Time	0

Table 5.3 Summary of the MANET Simulation Parameters

5.3 Performance Metric

The main motivation behind our research is to improve the TCP throughput on the Wired–MANET architecture, so it was naturally for us to choose the TCP throughput as our performance metric.

5.4 Factors that Affect the Performance Metric

In the Wired–MANET architecture there are many different factors that could affect the TCP throughput, our research concentrated on two of these factors to measure and analyze their effect on the TCP throughput. These factors are:

- Congestion at an intermediate host(s) on the wired part of the network, which is caused by competing traffic.
- Route failures between the mobile hosts on the MANET, which is caused by the mobility of the mobile hosts and is dependent on the mobility rate.

5.4.1 Congestion

Congestion is the effect of having traffic routed from a fast link to a slower link, which causes the traffic packets to be discarded. We introduced congestion at the wired network by routing competing traffic through the same link as the normal traffic in a periodical manner, 20 seconds in every 50 seconds simulation time. In Figure 5.1, both the normal traffic and the competing traffic are going from FH to BS. From FH we are sending 1000 packets every second and the packet size is 10000 bytes. This results in a bandwidth of 10.0 Mbps for the link between FH and BS for the competing traffic, when we add our normal traffic bandwidth this will exceed the link capacity (10 Mbps) and will result in congesting the link between FH and BS.

5.4.2 Route Failures

Route failures are the effect of mobile host(s) mobility. The route failure rate depends on the mobility rate and the transmission range of the mobile host(s). The mobility rate of a mobile host is determined by the mobility speed and the pause time. In our simulation we used a fixed pause time 0 so that the mobile hosts is always moving with the specified mobility speed. We ran our simulations using five different mobility speeds ranging from 0 m/s to 20 m/s in steps of 5 m/s. The mobility speed is actually the mobile host maximum speed and not the exact mobile host speed. The mobile host exact speed is uniformly distributed between 0 and the mobility speed. We will be using the term mobility speed to mean maximum speed. The IEEE 802.11 transmission radius is 250 m.

Chapter 6

Simulation Results

In this chapter, we present and discuss the results from the simulations that were conducted to evaluate the proposed scheme. In order for us to be able to evaluate the proposed scheme, we had to run four different sets of simulation. The details of each simulation set are explained in the next section.

6.1 Simulation Sets

Our simulation sets consist of two configurations. The first configuration is the main configuration of each simulation set, which is different from one simulation set to another. The second configuration is the fixed configuration of the simulation which is the same for all the four simulation sets.

6.1.1 Simulation Sets Main Configuration

The following are the main configuration of each simulation set, which describes the following:

- The transmission control protocol running on the wired part of the network.
- If the Berkeley Snoop protocol is running on the base station or not.

- The transmission control protocol running on the MANET part of the network.
    - Simulation Set (1) Main Configuration (TCPReno)
        - The TCPReno protocol on the wired part of the network.
        - No Berkeley Snoop protocol on the base station.
        - The TCPReno protocol on the MANET part of the network.
    - Simulation Set (2) Main Configuration (TCPSnoop)
        - The TCPReno protocol on the wired part of the network.
        - The Berkeley Snoop protocol on the base station.
        - The TCPReno protocol on the MANET of the network.
    - Simulation Set (3) Main Configuration (ATCPReno)
        - The ATCP protocol on the wired part of the network.
        - No Berkeley Snoop protocol on the base station.
        - The ATCP protocol on the MANET part of the network.
    - Simulation Set (4) Main Configuration (ATCPSnoop)
        - The TCPReno protocol on the wired network of the network.
        - The Berkeley Snoop protocol on the base station.
        - The ATCP protocol on the MANET part of the network.

6.1.2 Simulation Sets Fixed Configuration

The following are the fixed configuration for the simulation, which describe the following:

- If there was congestion generated on the wired part of the network or not.
    - No Congestion
    - Congestion
- The different mobility rate of the mobile host(s).

- Mobile Host(s) move at speed 0 m/s (i.e. Stationary Mobile Hosts)
- Mobile Host(s) move at speed 05 m/s
- Mobile Host(s) move at speed 10 m/s
- Mobile Host(s) move at speed 15 m/s
- Mobile Host(s) move at speed 20 m/s

For each simulation in a simulation set we conducted the simulation using ten different movement scenario files. The movement scenario file defines the initial position of the mobile host(s), the destinations to which the mobile host(s) move and the mobility rate (i.e. speed) of the mobile host(s).

6.2 Expected TCP Throughput

When we started the simulation, we asked ourselves two important questions, and we had to answer them before we start our simulation. Here are the two questions:

- *What should be the expected TCP throughput?*
- *How would we verify that the TCP throughput is valid?*

We can answer these questions for the first set of simulation "Simulation Set (1)" with no congestion and no mobile host(s) movement, which is enough to validate that the TCP throughput we are getting is valid, because the remaining simulation sets depend on this set. These two questions are closely related to each other, because if we are able to calculate the expected TCP throughput using all the factors that affect it, we can compare between this expected TCP throughput and the TCP throughput we are getting and we will know if the results are valid or not.

6.2.1 IEEE 802.11 MAC Data Throughput

[16] discusses the IEEE 802.11 MAC data throughput of a MANET that is not connected to the wired network. [16] calculates that the IEEE 802.11 MAC data throughput will be at most the result from the following equation:

$$\frac{\text{Data Packet}}{\text{Data Packet} + \text{Data Packet Overhead}} * 2 \text{ Mbps}$$

Data Packet Overhead = RTS + CTS + MAC Header + MAC ACK

- RTS (Request to Send) Size = 40 Bytes.
- CTS (Clear to Send) Size = 39 Bytes.
- MAC Header Size = 47 Bytes.
- MAC ACK Size = 39 Bytes.

If we apply this equation to the data packet size we are using, which is 536 bytes, the IEEE 802.11 MAC data throughput should be at most:

$$\frac{536}{536 + 40 + 39 + 47 + 39} * 2 \approx 1.53 \text{ Mbps}$$

From applying this equation, we find that the IEEE 802.11 MAC data throughput could be at most 1.53 Mbps for 536 bytes data packets. When various inter-frame timings are also accounted for, this limit is reduced to 1.43 Mbps [16]. This result here represents the IEEE 802.11 MAC data throughput for a receiver that is one hop away from the sender, but when it comes to a chain of nodes, the IEEE 802.11 MAC

data throughput is even lower. The following analysis is discussing this issue. Let us consider Figure 6.1 where we have a chain of 6 nodes, node 1 is the sender and node 6 is the receiver. Assume for the moment that the radios of nodes that are not neighbours do not interfere with each other. Nodes 1 and 2 can not transmit at the same time because node 2 can not receive and transmit simultaneously. Nodes 1 and 3 can not transmit at the same time because node 2 can not correctly hear 1 if 3 is sending. Nodes 1 and 4 can transmit at the same time. This leads to a channel utilization of 1/3 [16]. If one assumes that radios can interfere with each other beyond the range at which they can communicate successfully, the IEEE 802.11 MAC data throughput is worse. IEEE 802.11 MAC nodes in the *NS* simulator can correctly receive packets from 250 meters away, but can interfere at 550 meters.

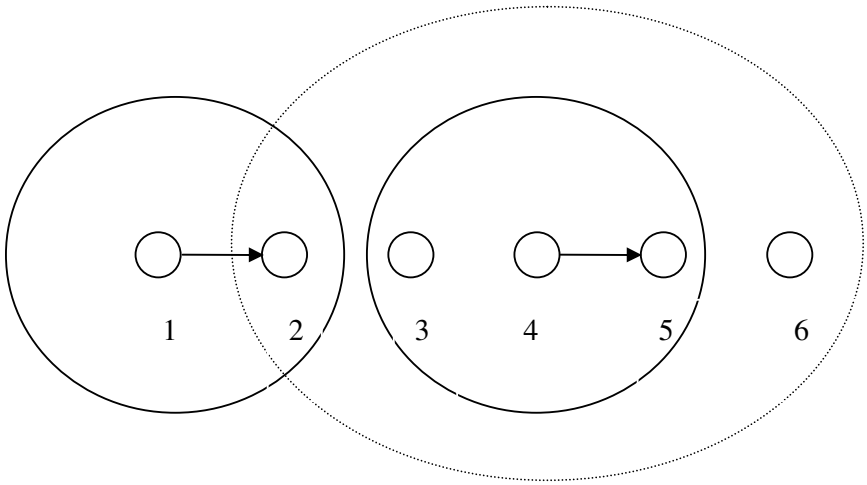

Figure 6.1: MAC Interference Among a Chain of Nodes.

In Figure 6.1, the solid-line circle denotes a node's transmission range. The dotted-line circle denotes a node's interference range. Node 4's transmission will corrupt node 1's

transmission at node 2 [16]. Therefore [16] expects the maximum utilization of a chain of ad hoc nodes in the *NS* simulator to be 1/4.

6.2.2 TCP Data Throughput

To calculate the TCP data throughput we can use the same equation but after modifying it to account for the various overheads the data packet experiences during the transmission from the sender to the receiver. We modified the data packet overhead from the previous section to add the IP header overhead and the TCP header overhead. In IEEE 802.11, each unicast data packet received by the receiver will be acknowledged by an ACK, which should be counted as an overhead to the data packet. The TCP data throughput will be at most the result of the following equation:

$$\frac{\text{Data Packet Size}}{\text{Data Packet} + \text{Data Packet Overhead} + \text{TCP ACK Overhead}} * 2 \text{ Mbps}$$

Data Packet Overhead= TCP H. + IP H. + RTS + CTS + MAC H. + MAC ACK

TCP ACK Overhead = TCP H. + IP H. + RTS + CTS + MAC H. + MAC ACK

- TCP Header Size = 20 Bytes.
- IP Header Size = 20 Bytes.

If we apply this equation to the data packet size we are using, which is 536 bytes, the TCP data throughput should be at most:

Data Packet Overhead = 20 + 20 + 40 + 39 + 47 + 39 = 205

TCP ACK Overhead = 20 + 20 + 40 + 39 + 47 + 39 = 205

$$\frac{536}{536+205+205} * 2 \approx 1.13 \text{ Mbps}$$

From this equation, we expect the TCP data throughput to be at most 1.13 Mbps for 536 bytes data packets. When various inter-frame timings are also accounted for, this limit is reduced to 0.93 Mbps (0.1 Mbps for the Data Packet and 0.1 Mbps for the TCP ACK) [16]. This result here represents the TCP data throughput for a receiver that is one hop away from the sender, but when it comes to a chain of nodes, we will need to apply the channel utilization equation from the previous section, which will reduce the TCP data throughput to at most 0.23 Mbps for a chain of 6 nodes.

6.3 Simulation Validation

In order for us to validate our simulation setup, we conducted a validation exercise that has the following configuration:

- Main Configuration
    - The TCPReno protocol on the wired part of the network.
    - No Berkeley Snoop protocol on the base station.
    - The TCPReno protocol on the MANET part of the network.
- Fixed Configuration
    - No Congestion
    - Mobile Host(s) move at speed 0 m/s (i.e. Stationary)

The validation exercise was conducted using different movement files that are only different in the location of the receiver. In our simulation setup the receiver distance

from the sender could be in a range from one hop to eight hops. We conducted our validation exercise by running five simulations for each case and measured the TCP throughput. The following are the results from the validation exercise:

Number of Hops	Measured Throughput Mbps	Standard Deviation (STDEV)	Expected Throughput Mbps
1	0.92	0.01	0.93
2	0.43	0.01	0.47
3	0.25	0.03	0.31
4	0.18	0.07	0.23
5	0.15	0.04	0.19
6	0.14	0.03	0.16
7	0.12	0.03	0.13
8	0.12	0.03	0.12

Table 6.1: Validation Exercise Results

Table 6.1 shows the results of the validation exercise. We notice that as the chain gets longer, the average TCP throughput approaches 0.12 Mbps for 536 bytes data packets, or almost 1/7 utilization of the maximum 0.93 Mbps. This utilization is substantially less than the 1/4 utilization expected by [16] in the previous section. To shed light on the discrepancy between 1/4 and 1/7, [16] conducted a set of simulations to verify their channel utilization equation and they found that as the chain gets longer, they approach a utilization of 0.13 Mbps for 536 bytes packets, or 1/7 of the maximum of 0.93 Mbps for 536 bytes packets [16]. They investigated this discrepancy and found that the IEEE 802.11 MAC is capable of sending at the optimal rate, but does not discover the optimum schedule of transmission. IEEE 802.11 MAC fails to achieve the optimum chain schedule because an 802.11 MAC node's ability to send is affected by the amount of contention it experiences [16].

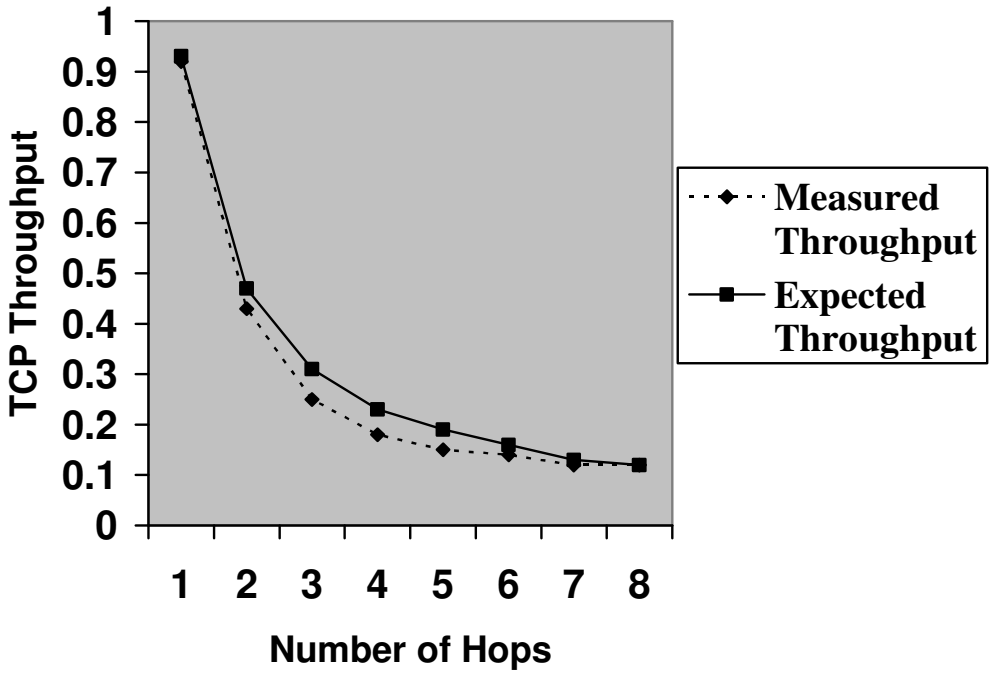

Figure 6.2: Measured TCP Throughput versus Expected TCP Throughput

Figure 6.2 shows the measured TCP throughput versus the expected TCP throughput. The maximum TCP throughput we can achieve is at one hop, almost 0.93 Mbps for both of them, with the increase in the number of hops the TCP throughput approaches almost 0.12 Mbps for both of them**.** We notice that in the cases from two hops to seven hops, the measured TCP throughput is less than the expected TCP throughput, because the interference that the mobile host experiences from other mobile hosts depends on the location of the mobile host in the Ad-Hoc chain. For example if we have a chain of seven mobile hosts, mobile host one experiences interference from three other mobile hosts, but mobile host three experiences interference from five other mobile hosts, which means that mobile host one can inject more data than mobile host three.

Conclusion

Conducting this validation exercise and comparing the TCP throughput obtained from it with the expected TCP throughput demonstrates that our simulation setup is valid and that it can be used for the four simulation sets.

6.4 Simulation Sets Results

After validating our simulation setup in the previous section, we conducted our four simulation sets using the configurations described in Section 6.1. In the following subsections we will present the simulation results. Each simulation result is an average of ten different runs. Each run uses a different movement scenario file. The scenario files are different in the initial positions, the movement directions and the mobility rate (i.e. speed) of the mobile hosts.

6.4.1 Simulation Set 1 (TCPReno)

6.4.1.1 No Congestion

Table 6.2 presents the results for simulation set 1 (TCPReno) without congestion artificially introduced on the wired part of the TCP connection.

Mobile Hosts Speed m/s	TCPReno Throughput Mbps	Standard Deviation (STDEV)
0	0.26	0.26
5	0.12	0.11
10	0.12	0.09
15	0.03	0.04
20	0.03	0.03

Table 6.2: TCPReno Throughput without Congestion

Analysis

Table 6.2 shows the following:

- The maximum TCP throughput was achieved in the 0 m/s scenario.
- Increasing the mobile host(s) mobility rate over the 0 m/s caused the TCP throughput to drop significantly.
- The TCP throughput drop rate depends on the mobile host(s) mobility rate.
- The higher the mobility rate, the lower the TCP throughput.

Conclusion

We conclude that the mobile host(s) mobility rate is an important factor that affects the end-to-end TCP throughput. We also conclude that the TCP throughput achieved for all scenarios other than the stationary scenario is very low and can be improved substantially.

6.4.1.2 Congestion

Table 6.3 presents the results for simulation set 1 (TCPReno) with congestion artificially introduced at an intermediate host on the wired part of the TCP connection.

Mobile Hosts Speed m/s	TCPReno Throughput Mbps	Standard Deviation (STDEV)
0	0.09	0.07
5	0.05	0.04
10	0.05	0.03
15	0.01	0.01
20	0.01	0.01

Table 6.3: TCPReno Throughput with Congestion

Analysis

Table 6.3 shows the following:

- The TCP throughput achieved is substantially less than the TCP throughput achieved in Table 6.2 (TCPReno Throughput without Congestion) for all scenarios.
- The TCP throughput drop rate is not the same for all scenarios, because of the mobile host(s) mobility factor.

Conclusion

We conclude that congestion on the wired part of the TCP connection is an important factor that affects the end-to-end TCP throughput.

Congestion Validation

To make sure that congestion is the actual factor for the TCP throughput drop, we randomly selected the scenario where the mobile hosts move at 10 m/s and monitored the TCP packets being dropped on the link between fixed host (0) and fixed host (1). We found that the TCP packets are only being dropped on that link when we artificially introduce congestion.

Figure 6.3 shows the dynamics of the number of TCP packets being dropped as a function of the simulation time. We notice that during the first 250 seconds, there are no TCP packets being dropped on that link, because the TCP connection was not established yet. Starting from second 250 up to the end of the simulation, the TCP packets are only being dropped in the 20 second intervals during which we artificially introduce congestion, which shows that the drop in the TCP throughput was because of our artificial congestion.

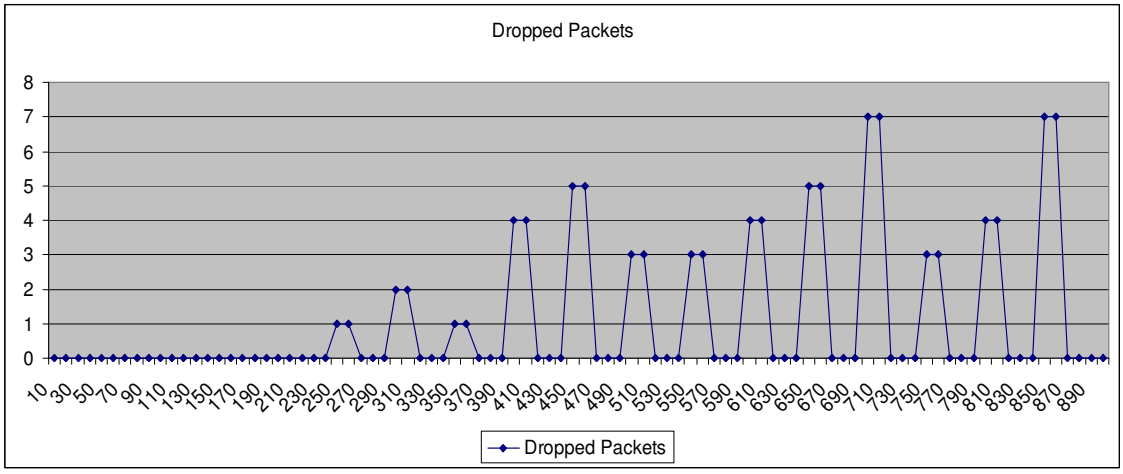

Figure 6.3: Dropped Packets Dynamics

6.4.2 Simulation Set 2 (TCPSnoop)

6.4.2.1 No Congestion

Table 6.4 presents the results for simulation set 2 (TCPSnoop) without congestion artificially introduced at the wired part of the TCP connection.

Mobile Hosts Speed m/s	TCPSnoop Throughput Mbps	Standard Deviation (STDEV)
0	0.26	0.25
5	0.12	0.11
10	0.14	0.07
15	0.03	0.02
20	0.03	0.03

Table 6.4: TCPSnoop Throughput without Congestion

Analysis

Table 6.4 shows the following:

- In the 0 m/s scenario the TCP throughput achieved is the same as the TCP

throughput achieved in Table 6.2 (TCPReno Throughput without Congestion). This is because there are no route failures in this scenario. This means that the Snoop protocol will not improve the TCP throughput.

- In the 5, 10, 15 and 20 m/s scenarios the TCP throughput achieved is the same as the TCP throughput achieved in Table 6.2 (TCPReno Throughput without Congestion) (We will explain later why the TCPSnoop throughput of the 10 m/s scenario is not different from the TCPReno throughput). We expected the TCP throughput to improve, because there are route failures in these scenarios. This means that the Snoop protocol should improve the TCP throughput.
- To explain why the TCP throughput in the 10 m/s scenario is not different from the TCPReno throughput, we performed statistical analysis to make sure that the results of the ten simulation runs for each simulation set (TCPReno and TCPSnoop) are really different. The statistical analysis showed that the 95% confidence interval for the TCPReno throughput is [0.07086, 0.1765], and the 95% confidence interval for the TCPSnoop throughput is [0.09172, 0.1789]. Not only do these two confidence intervals overlap, but they also contain the respective other average throughput, so there is a strong statistical evidence that the results are not different. This means that the Snoop protocol did not improve the TCP throughput for this scenario as well.
- To make sure that these results are actually valid, we did the following:
    1. We started by making sure that the Snoop protocol in the *NS* simulator works as it should. Using *NS* we simulated a single-hop wireless network (Cellular Network Architecture) that has artificially introduced wireless transmission errors in two scenarios. In the first scenario we did not have the Snoop protocol running on the base station, and we measured the TCP

throughput. In the second scenario we did have the Snoop protocol running on the base station, and we measured the TCP throughput. Comparing the results from both scenarios shows that the TCP throughput had actually increased in the second scenario. Doing this validation exercise assured us that the Snoop protocol works correctly in the *NS* simulator.

2. Our next step was to check the semantics of the Snoop protocol, to explain why there was no improvement in the TCP throughput. There is a very important condition that must be met in order for the Snoop protocol to improve the TCP throughput. The condition is that the disconnection time (the time in which there is no route available to the receiving mobile host) should be less than the TCP sender time-out (RTO) (TCP sender time-out (RTO) is based on the round trip time (RTT) estimates we discussed in Chapter 2). In other words, if a packet was lost on the wireless link due to route failure, the Snoop protocol will be able to retransmit this packet successfully (hence avoid invoking congestion control algorithms) only if the route was established in a time that is less than the TCP sender time-out (RTO). This means that the Snoop protocol will only improve the TCP throughput in the case of short disconnections. We monitored some packets in the Snoop cache and found that some of these packets are actually spending more time than the TCP Sender time-out (RTO) without being successfully transmitted to the receiver, which means that the TCP sender will time-out and invoke congestion control algorithms. Having this situation occurring frequently causes the congestion control algorithms to be invoked frequently, which will cause the TCP throughput to drop substantially.

Conclusion

We conclude that the Snoop protocol will only improve the TCP throughput in case the disconnection times were less than the TCP sender time-out (RTO).

Congestion Window Validation

Figure 6.4 shows the dynamics of the TCP congestion window as a function of the simulation time for the 10 m/s scenario. Each result represents the congestion window average for 10 seconds. The congestion window in the *NS* simulator is measured by packets not by bytes as in real TCP.

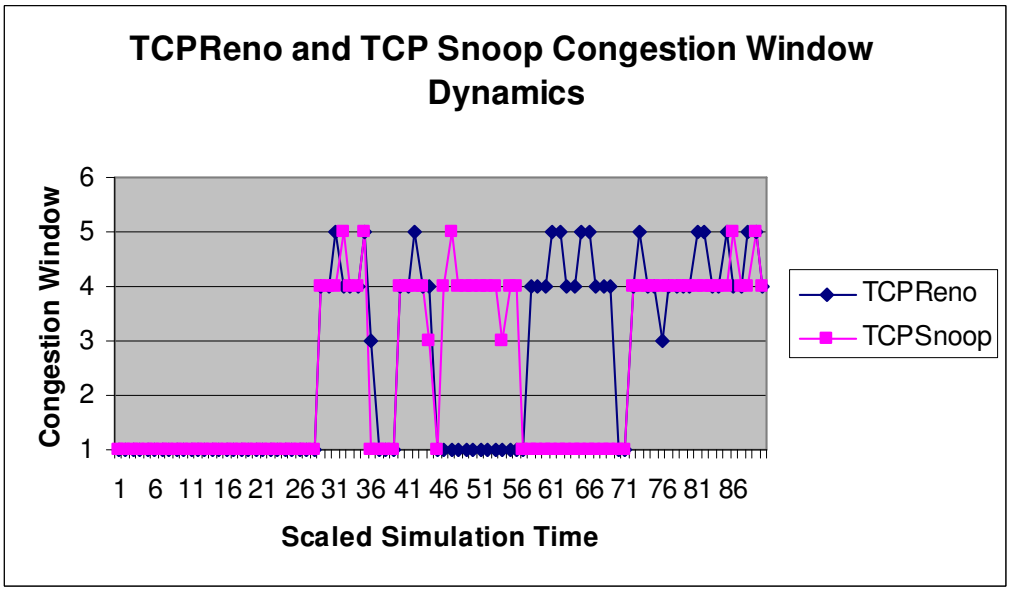

Figure 6.4: TCPReno and TCPSnoop Congestion Window Dynamics

Figure 6.4 shows that for the first 290 seconds the TCP connection was not established so the congestion window is the same for TCPReno and TCPSnoop. Then the TCPSnoop congestion window gets higher than the TCPReno congestion window.

Then the TCPReno congestion window gets higher than the TCPSnoop congestion window. Then they both become the same again. This shows that the average congestion window for both throughout the simulation is almost the same, which results in having almost the same TCP throughput.

6.4.2.2 Congestion

Table 6.5 presents the results for simulation set 2 (TCPSnoop) with congestion artificially introduced at an intermediate host on the wired part of the TCP connection.

Mobile Hosts Speed m/s	TCPSnoop Throughput Mbps	Standard Deviation (STDEV)
0	0.09	0.07
5	0.07	0.06
10	0.06	0.06
15	0.01	0.01
20	0.01	0.02

Table 6.5: TCPSnoop Throughput with Congestion

Analysis

Table 6.5 shows the following:

- In the 5 and 10 m/s scenarios the TCP throughput achieved is higher than the TCP throughput achieved in Table 6.3 (TCPReno Throughput with Congestion). This is because congestion increases the TCP packets round trip time (RTT), which in turn increases the TCP sender time-out (RTO). This means that there will be enough time for the Snoop protocol to retransmit the TCP packets, the TCP receiver to acknowledge the receipt of these packets and for the TCP sender to receive these acknowledgments before the TCP sender times out.

- In the 15 and 20 m/s scenarios the TCP throughput achieved is the same as the TCP throughput achieved in Table 6.3 (TCPReno Throughput with Congestion). This is because the increase in the TCP sender time-out (RTO) was not enough for the Snoop protocol to retransmit the TCP packets from its own cache, the TCP receiver to acknowledge the receipt of these packets and for the TCP sender to receive these acknowledgments before the TCP sender times out.

Conclusion

We conclude that using the Snoop protocol on the base station improved the TCP throughput in case there was congestion on an intermediate node on the wired part of the TCP connection.

6.4.3 Simulation Set 3 (ATCPReno)

6.4.3.1 No Congestion

Table 6.6 presents the results for simulation set 3 (ATCPReno) without congestion artificially introduced on the wired part of the TCP connection.

Mobile Hosts Speed m/s	ATCPReno Throughput Mbps	Standard Deviation (STDEV)
0	0.26	0.26
5	0.17	0.16
10	0.16	0.14
15	0.04	0.04
20	0.04	0.04

Table 6.6: ATCPReno Throughput without Congestion

Analysis

Table 6.6 shows the following:

- In the 0 m/s scenario the ATCP throughput achieved is the same as the TCP throughput achieved in Table 6.2 (TCPReno Throughput without Congestion), as expected. This is because there are no route failures in this scenario. This means that ATCP will not improve the TCP throughput.
- In the 5, 10, 15 and 20 m/s scenarios the ATCP throughput achieved is higher than the TCP throughput achieved in Table 6.2 (TCPReno Throughput without Congestion). We expected the TCP throughput to improve, because there are route failures in these scenarios. This means that ATCP will improve the TCP throughput.
- The TCP throughput improvement rate depends on the mobility rate.
- The higher the mobility rate, the lower the TCP throughput improvement.
- ATCP improved the TCP throughput in a range of 33% to 42%.

Conclusion

We conclude that using ATCP improves the TCP throughput, because it does not allow that the TCP sender times out and invokes the congestion control algorithms, by putting it into a persist state.

Congestion Window Validation

Figure 6.5 shows the dynamics of the TCP congestion window as a function of the simulation time for the 10 m/s scenario. Each result represents the congestion window

average for 10 seconds. The congestion window in the *NS* simulator is measured by packets not by bytes as in real TCP.

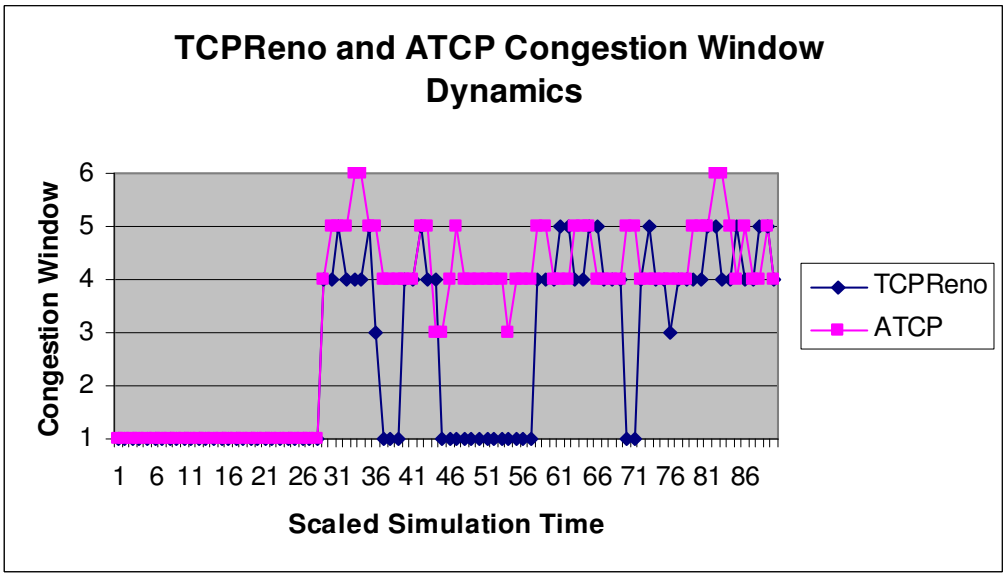

Figure 6.5: TCPReno and ATCP Congestion Window Dynamics

Figure 6.5 shows that for the first 290 seconds the TCP connection was not established so the congestion window is the same for TCPReno and ATCP. Then the ATCP congestion window is consistently higher or the same as the TCPReno congestion window.

6.4.3.2 Congestion

Table 6.7 presents the results for simulation set 3 (ATCPReno) with congestion artificially introduced at an intermediate host on the wired part of the TCP connection.

Mobile Hosts Speed m/s	ATCPReno Throughput Mbps	Standard Deviation (STDEV)
0	0.09	0.07
5	0.07	0.08
10	0.07	0.09
15	0.02	0.03
20	0.02	0.03

Table 6.7: ATCPReno Throughput with Congestion

Analysis

Table 6.7 shows the following:

- In the 0 m/s scenario the ATCP throughput achieved is the same as the TCP throughput achieved in Table 6.3 (TCPReno Throughput with Congestion), as expected.
- In the 5, 10, 15 and 20 m/s scenarios the ATCP throughput achieved is higher than the TCP throughput achieved in Table 6.3 (TCPReno Throughput with Congestion).
- ATCP improved the TCP throughput in a range of 40% to 100%.

Conclusion

We conclude that using ATCP improved the TCP throughput in case there was congestion on an intermediate node on the wired part of the TCP connection and route failures on the MANET.

6.4.4 Simulation Set 4 (ATCPSnoop)

6.4.4.1 No Congestion

Table 6.8 presents the results for simulation set 4 (ATCPSnoop) without congestion artificially introduced on the wired part of the TCP connection.

Mobile Hosts Speed m/s	ATCPSnoop Throughput Mbps	Standard Deviation (STDEV)
0	0.26	0.26
5	0.17	0.15
10	0.16	0.13
15	0.04	0.03
20	0.04	0.04

Table 6.8: ATCPSnoop Throughput without Congestion

Analysis

Table 6.8 shows the following:

- The TCP throughput achieved is the same as the TCP throughput achieved in Table 6.6 (ATCPReno Throughput without Congestion) for all scenarios.
- These results were expected after we knew the results from simulation set 2 (TCPSnoop) and that the Snoop protocol was not able to improve the TCP throughput because of the long disconnection times.

Conclusion

We conclude that using ATCP and the Snoop protocol together did not have negative impact on the TCP throughput achieved by the ATCP.

6.4.4.2 Congestion

Table 6.9 presents the results for simulation set 4 (ATCPSnoop) with congestion artificially introduced at an intermediate host on the wired part of the TCP connection.

Mobile Hosts Speed m/s	ATCPSnoop Throughput Mbps	Standard Deviation (STDEV)
0	0.09	0.07
5	0.07	0.08
10	0.07	0.06
15	0.02	0.02
20	0.02	0.03

Table 6.9: ATCPSnoop Throughput with Congestion

Analysis

Table 6.9 shows the following:

- The TCP throughput achieved is the same as the TCP throughput achieved in table 6.7 (ATCPReno Throughput with Congestion) for all scenarios.

Conclusion

We conclude that using ATCP and the Snoop protocol together did not have negative impact on the TCP throughput in case there was congestion on an intermediate node on the wired part of the TCP connection.

6.5 Conclusions

The TCP throughput of a MANET connected to the wired network through a base station depends on different factors, such as congestion (Wired Network or MANET), wireless transmission errors (MANET) and route failures (MANET). In our

simulation we did not discuss or measure the impact of wireless transmission errors or MANET congestion on the TCP throughput. We discussed and measured the impact of having wired network congestion and the impact of having MANET route failures on the TCP throughput.

From Simulation Set 1 (TCPReno) we conclude that the TCP throughput depends on how frequent the routes between the mobile host(s) get invalidated. As we can see from the results in Table 6.2 (TCPReno Throughput without Congestion), increasing the mobile host(s) mobility rate decreases the TCP throughput. We also conclude that the TCP throughput depends on the level of congestion the wired network is experiencing. As we can see from the results in Table 6.3 (TCPReno Throughput with Congestion), in the 0 m/s scenario (Stationary Scenario) where there is no route failures and only the wired network congestion exist, the TCP throughput decreased from 0.26 Mbps to 0.09 Mbps. In the rest of the scenarios in Table 6.3 (TCPReno Throughput with Congestion), the TCP throughput was impacted by both wired network congestion and MANET routes failures.

From simulation set 2 (TCPSnoop) we conclude that the Snoop protocol was not able to increase the TCP throughput, because the disconnection times on the MANET caused by route failures which is caused by the mobility of the mobile host(s) is greater than the TCP Sender time-out (RTO). This means that there was not enough time for the Snoop protocol to retransmit the TCP packets from its own cache, the TCP receiver to acknowledge the receipt of these packets and the TCP sender to receive the acknowledgment before the TCP sender times out. At the same time, the Snoop protocol did not have a negative impact on the TCP throughput. The results in

Table 6.4 ((TCPSnoop Throughput without Congestion) are similar to the results in Table 6.2 (TCPReno Throughput without Congestion). We also conclude that in the case of wired network congestion, having the Snoop protocol running on the base station did have positive impact on the TCP throughput in case of wired network congestion, because having congestion on the network increases the TCP packets round trip time (RTT), which in turn increases the TCP sender time-out (RTO), giving more time for the Snoop protocol to retransmit the TCP packets from its own cache, the TCP receiver to acknowledge the receipt of these packets and the TCP sender to receive the acknowledgment before the TCP sender times out.

From simulation set 3 (ATCPReno) we conclude that the ATCP protocol did not have a negative impact on the TCP throughput in the 0 m/s scenario (Stationary Scenario) and was able to improve the TCP throughput in all other scenarios. Comparing Table 6.6 (ATCPReno Throughput without Congestion) and Table 6.2 (TCPReno Throughput without Congestion) shows that the ATCP protocol was able to improve the TCP throughput in a range from 33% to 42%. The ATCP sender was able to differentiate between packet losses due to congestion and packet losses due to route failures, so in the case of route failures, it put itself into a persist mode where it does not send any data packets and keeps on probing the ATCP receiver until it gets an Acknowledgment back, then it moves itself to a normal state, where it sends the data packets. We also conclude that in the case of wired network congestion, having the ATCP protocol controlling the data transmission did not have a negative impact on the TCP throughput in the 0 m/s scenario (Stationary Scenario) and was able to improve the TCP throughput in all other scenarios. Comparing Table 6.7 (ATCPReno Throughput with Congestion) and Table 6.3 (TCPReno Throughput with Congestion)

shows that the ATCP protocol was able to improve the TCP throughput in a range from 40% to 100%.

From Simulation set 4 (ATCPSnoop) we conclude that our scheme did not have a negative impact on the TCP throughput. The results in Table 6.8 (ATCPSnoop Throughput without Congestion) are similar to the results in Table 6.6 (ATCPReno Throughput without Congestion). This means that our scheme was able to achieve the same TCP throughput without having to change TCP on the fixed hosts and only limit the changes to the base station and the mobile hosts. We also conclude that in the case of wired network congestion our proposed scheme did not have a negative impact on the TCP throughput. The results in Table 6.9 (ATCPSnoop Throughput with Congestion) are similar to the results in Table 6.7 (ATCPReno Throughput with Congestion).

Overall, our results show that the TCP throughput can be improved on the Wired-MANET architecture. We also conclude that achieving this goal does not require changing TCP on the wired network and we can limit the changes to the base station(s) and the mobile host(s). Our scheme was able to improve the TCP throughput and at the same time meets all the requirements.

Chapter 7

Conclusions and Future Work

TCP assumes that any packet loss is mainly due to congestion in the network, which is true in the case of a wired network, but in the case of a wireless network, either single-hop or multi-hop, this assumption is not always true. In the wireless network packet losses could be because of wireless transmission errors or handoffs between cells in the case of single-hop wireless network or because of wireless transmission errors or route failures in the case of multi-hop wireless network. This assumption causes the TCP throughput to drop significantly when it runs on a wireless network. Several protocols were proposed to address this problem either in the single-hop wireless network or in the multi-hop wireless network. For the single-hop wireless network we identified specific evaluation criteria that we used to evaluate some of these protocols to try to find the best protocol for this architecture. Our evaluation shows that the Snoop protocol is the best protocol for a single-hop wireless network. For the multi-hop wireless network we identified specific evaluation criteria that we used to evaluate some of these protocols to try to find the best protocol for this architecture. Our evaluation shows that the ATCP protocol is the best protocol for a multi-hop wireless network. Our research concentrates on the Wired-MANET architecture which combines both wired networks and multi-hop wireless networks. To the best of our knowledge this research is the first research to address the TCP

throughput in Wired–MANET architecture. The Wired–MANET architecture has some special characteristics as it combines both reasons for packet loss in the wired network (i.e. congestion) and in the multi-hop wireless network (i.e. wireless transmission errors and route failures). This means that any protocol proposal for the Wired–MANET architecture must account for these characteristics. We identified a list of requirements that must exist in any protocol or scheme being proposed for the Wired–MANET architecture. The ATCP protocol is the best protocol matching these requirements but it violates an important requirement, which is "The protocol must not change TCP on the fixed host(s)". To avoid changing TCP on the fixed host(s) we decided to use the Snoop protocol that runs on the base station and interact with TCP on the fixed host(s) and ATCP on the MANET. Our research concentrated on only two factors that affect the TCP throughput in the Wired–MANET architecture and measured the impact of these two factors on the TCP throughput through simulations using the *NS* network simulator. These two factors are congestion on the wired network and route failures on the MANET. We measured the impact of these two factors by running four different sets of simulation (TCP, TCPSnoop, ATCP and ATCPSnoop) and artificially introduced congestion on the wired network and route failures on the MANET. The following section summarizes the conclusions from our research.

7.1 Conclusions

In order to achieve TCP throughput improvement in the Wired-MANET architecture, the TCP sender needs to know the exact cause of the packet loss and not just assume that the packet loss was because of congestion. In our research we compared between two different methods that achieve this goal, the following are the two methods:

- The TCP sender: The first method is to change the TCP sender, so that it would listen to the network layer feedback from intermediate hops and put itself into the appropriate state.
- The base station: The second method is to use a protocol that runs on the base station and listen to the network layer feedback from intermediate hops and put the TCP sender into the appropriate state.

The ATCP protocol implements the first method, which means that if the TCP sender is running on the fixed host, the ATCP protocol would change TCP on the fixed host. The ATCPSnoop scheme implements the second method, which means that it does not require changing TCP on the fixed host even if the TCP sender is running on it.

We conducted our simulation and we concluded the following from the results:

- The maximum TCP throughput achieved is in the stationary scenarios.
- The TCP throughput depends on the mobility rate of the mobile host(s), the higher the mobility rate, the lower the TCP throughput.
- The TCP throughput depends on the level of congestion in the wired network.
- The Snoop protocol did not improve the TCP throughput in the non-congestion scenarios because of the long disconnection times the MANET encounters.
- The Snoop protocol did improve the TCP throughput in the congestion scenarios because congestion increases the packet RTT (round trip time) which increases the TCP sender RTO (retransmission time-out).

- The ATCP and ATCPSnoop protocols did improve the TCP throughput in a range of 33% to 42% in the non-congestion scenarios.
- The ATCP and ATCPSnoop protocol did improve the TCP throughput in a range of 40% to 100% in the congestion scenarios.

We conclude that allowing the TCP sender to know the exact cause of the packet loss will help improve the TCP throughput significantly. We also conclude that achieving this goal does not require changing the TCP on the fixed host(s) and could be achieved by other methods, as demonstrated by ATCPSnoop. Changes can be limited to the base station and the mobile host(s), which will make deploying such a scheme much easier.

7.2 Future Work

As we have mentioned earlier, we consider this research to be the first research to discuss, propose and evaluate a scheme for improving the TCP throughput over the Wired-MANET architecture, which leaves the door open for more research to be done in this area and more problems to be solved. This section tries to shed the light on some of these research topics.

We identified a list of requirements, and based on these requirements we proposed our scheme, which actually consist of three protocols interacting together to achieve the improvement in the TCP throughput. These requirements were based on our own expectations and it could be changed to include more requirements or remove some of the requirements, which of course could change the protocols being selected. This could have negative or positive impact on the TCP throughput achieved. Even

applying the same requirements to a different set of protocols could result in negative or positive impact on the TCP throughput achieved.

We did not study the impact of having the TCP sender running on the MANET and the TCP receiver running on the wired network. In our simulation the TCP sender was always running on the wired network and the TCP receiver was always running on the MANET. We expect that the TCP throughput would still improve. Additional simulations need to be conducted to support this expectation.

We did not study the impact of having both the TCP sender and the TCP receiver running on the MANET and the routing protocol decides to route the traffic through the wired network. We expect that the TCP throughput would still improve. Additional simulations need to be conducted to support this expectation.

We did not study the impact of congestion in the MANET or wireless transmission errors in the MANET on the TCP throughput. In our simulation we studied the impact of congestion in the wired network and route failures in the MANET on the TCP throughput. In both scenarios we expect the Snoop protocol to enhance the TCP throughput. Additional simulations need to be conducted to support this expectation.

We did not study the impact of handoffs between base stations (mobile host moves from a base station to another base station) on the TCP throughput. In our simulation we only had one base station so there were no handoffs. We expect that the TCP throughput would still improve as this scenario is a special case of route failure.

We did not study the effect of multiple TCP connections on the TCP throughput. [22] discusses this issue in details. It would be interesting to see how much improvement we can get with our proposed scheme in the case of multiple TCP connections.

We did not study the bandwidth fairness of our scheme. It would be interesting to compare the bandwidth fairness between the ATCPSnoop scheme and the TCPReno protocol.

We did not study the scalability issue of the Snoop protocol. When handoffs occurs the entire connection state will be transferred between the old base station and the new base station. Additional simulations need to be conducted to measure the effect of frequent handoffs on the ATCPSnoop scheme.

Our scheme was able to improve the TCP throughput, but that raises some very important questions, "Is that the maximum improvement we can achieve?" If not, "What is the maximum improvement?"

Bibliography

[1] M. Allman, V. Paxson and W. Stevens, *"RFC 2581 TCP Congestion Control"*, http://www.ietf.org/rfc/rfc2581.txt?number=2581, April 1999

[2] A. Bakre and B. R. Badrinath, *"I-TCP: Indirect TCP for Mobile Hosts"*, Int. Conf. on Distributed Computing Systems, pages 136-143, June 1995, Vancouver, Canada

[3] H. Balakrishnan, S. Seshan, and R. H. Katz, *"Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks"*, ACM Wireless Networks, Vol. 1, No. 4, pages 469-481, December 1995

[4] S. Banerjee, J. R. Goteti and G. Krishnamoorthy, *"Extending TCP for Wireless Networks"*, http://www.cs.umd.edu/users/suman/docs/711s97/711s97.html

[5] S. Biaz, M. Mehta, S. West and N. H. Vaidya, *"TCP over Wireless Networks Using Multiple Acknowledgements"*, Technical Report 97-001, Computer Science, Texas A&M University, January 1997, Texas, USA

[6] S. Biaz and N. H. Vaidya, *"Discriminating Congestion Losses from Wireless Losses using Inter-Arrival Times at the Receive"*, Technical Report 98-014, Texas A&M University, June 1998, Texas, USA

[7] J. Broch, D. A. Maltz, D.B. Johnson, Y. Hu and J. Jetcheva, *"A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols"*, Int. Conf. on Mobile Computing and Networking, pages 85-97, October 1998, Dallas, Texas, USA

[8] K. Brown and S. Singh, *"M-TCP: TCP for Mobile Cellular Networks",* ACM Computer Communications Review, Vol. 27, No. 5, pages 19-43, 1997

[9] R. Caceres and L. Iftode, *"Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments"*, IEEE Journal on Selected Areas of Communications, Vol. 13, No 5, pages 850-857, June 1995

[10] K. Chandran, S. Raghunathan, S. Venkatesan and R. Prakash, *"A Feedback-Based Scheme for Improving TCP Performance in Ad Hoc Wireless Networks"*, IEEE Personal Communications Magazine, Vol. 8, No. 1, pages 34- 39, February 2001

[11] CMU Monarch Group, *"CMU Monarch Extensions to the NS Simulator"*, http://monarch.cs.cmu.edu/cmu-ns.html, August 2002

[12] T. D. Dyer and R. V. Boppana, *"A Comparison of TCP Performance over Three Routing Protocols for Mobile Ad Hoc Networks"*, ACM Int. Symposium on Mobile Ad Hoc Networking and Computing, pages 56-66, October 2001, Long Beach, California

[13] K. Fall and K. Varadhan, *"The NS Manual (Formerly NS Notes and Documentation)"*, http://www.isi.edu/nsnam/ns/doc/index.html, August 2002

[14] T. Goff, J. Moronski, D. S. Phatak and V. Gupta, *"Freeze-TCP: A True End-To-End TCP Enhancement Mechanism for Mobile Environments"*, IEEE Conference on Computer Communications, pages 1537-1545, 2000

[15] G. Holland and N. Vaidya, *"Analysis of TCP Performance over Mobile Ad Hoc Networks"*, Int. Conf. on Mobile Computing and Networking, pages 219-230, August 1999, Seattle, Washington, USA

[16] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee and R. Morris, *"Capacity of Ad Hoc Wireless Network"*, ACM SIGMOBILE, pages 61-69, July 2001, Rome, Italy

[17] J. Liu and S. Singh, *"ATCP: TCP for Mobile Ad-Hoc Networks"*, IEEE Journal on Selected Areas in Communication, Vol. 19, No. 7, pages 1300-1315, July 2001

[18] K. Ratnam and I. Matta, *"WTCP: An Efficient Mechanism for Improving TCP Performance over Wireless Links"*, IEEE Symposium on Computers and Communications, June 1998. Athens, Greece

[19] University of Southern California, *"RFC 793 Transmission Control Protocol",* http://www.ietf.org/rfc/rfc0793.txt?number=793, September 1981

[20] N. H. Vaidya, M. Mehta, C. Perkins, G. Montenegro, *"Delayed Duplicate Acknowledgements: A TCP-Unaware Approach to Improve Performance of TCP over Wireless"*, Technical Report 99-003, Computer Science Department, Texas A&M University, February 1999, Texas, USA

[21] F. Wang and Y. Zhang, *"Improving TCP Performance over Mobile Ad Hoc Networks with Out-of-Order Detection and Response"*, ACM Int. Symposium on Mobile Ad Hoc Networking and Computing, pages 217-225, June 2002, Lausanne, Switzerland

[22] K. Xu, S. Bae, S. Lee and M. Gerla, *"TCP Behavior across Multi-Hop Wireless Networks and the Wired Internet"*, Int. Workshop on Wireless Mobile Multimedia, pages 41-48, September 2002, Atlanta, Georgia, USA