ABSTRACT

KANDULA, DHEERAJ. End-to-end Behavior of Delay Tolerant Networks with Message Ferries. (Under the direction of Associate Professor Rudra Dutta).

Delay Tolerant Networks (DTN) are high delay networks with intermittent connectivity. Transport protocols developed either for high bandwidth networks or low delay networks suffer significantly on these type of networks. We have studied the impact of various transport protocols and application level protocols on a specific type of DTN namely Message Ferry Networks. At present there is no specific transport protocol that adapts well to the characteristics of Message Ferry networks. We developed a protocol that is well suited for Message ferry networks. Our protocol ensures major characteristics of a reliable transport protocol like in order delivery and reliable transfer of data without compromising on the throughput. We simulated our protocol by modifying the TCP process model in Opnet and compared it with standard TCP. The simulation results show a drastic improvement over the standard TCP protocol.

End-to-end Behavior of Delay Tolerant Networks with Message Ferries

by

Dheeraj Kandula

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fullfillment of the
requirements for the Degree of
Master of Science

Computer Science

Raleigh, North Carolina

2008

Approved By:

Dr. David Thuente	Dr. George N. Rouskas

Dr. Rudra Dutta
Chair of Advisory Committee

DEDICATION

To my family

BIOGRAPHY

Dheeraj Kandula graduated from Pondicherry Engineering College with a Bachelor of Technology degree in Computer Science and Engineering, Pondicherry University, Pondicherry, India in May 2001. He started his professional career at HCL Technologies, Chennai, India and spent the next 3 years as a Test Engineer. He then joined Master of Science program in Computer Networking at North Carolina State University (NCSU).

ACKNOWLEDGMENTS

I would like to express my sincere thanks and gratitude to my advisor Dr. Rudra Dutta for his continuous guidance, suggestions and the long hours he spent for me during the entire period of my research. I was able to fulfill my aspiration to develop a network protocol with the motivation and inspiration instilled in me by my advisor. I am grateful to my committee members, Dr. David Thuente and Dr. George Rouskas for their affable nature and constant support during the course of my research.

I would like to thank my parents Dr. K. Satyanarayana Rao and Ms. P.V.S. Sai Kumari, my sister Ms. Radha Kandula and my brother-in-law Mr. Arun Kumar Varala for their love, support and encouragement throughout my life.

I would also like to thank all my friends whose constant support and care has always kept me motivated. I want to specifically thank my close friends Mr. Fernando Barsoba and Ms. Paloma B. Liton, Mr. Manikandan Arumugam, Mr. Mohan Iyer, Mr. Ravikumar Eswaran and Mr. Varadarajan Thulasirajan for their help in making tough choices and support in times of need.

Finally, I would like to thank North Carolina State University for giving me an opportunity to pursue my Masters.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

In today's world accessibility and availability of information irrespective of the field of interest has been made possible by the ubiquitous presence of Internet. This has been made possible even to people on the move due to the advent and advances in the area of wireless networking.

Recently wireless networks have been extended to provide network access in places where wired technology cannot penetrate or the cost of setting up the infrastructure far surpasses its use or the network is required only for a short duration. Typical examples are providing wireless access in the last mile by Wireless Internet Service Providers or setting up a network in a war zone or a network to aid relief work in a disaster zone. Hence wireless networks become the only type of network that can be put in place in these scenarios. The last two scenarios take wireless networks to the extreme called Delay Tolerant Networks (DTNs).

Many researchers have looked into the problem of providing network access or figuring out a feasible path to transfer data across multi-hop wireless networks where connectivity is assumed to be present most of the time. Ad hoc sensor networks is a typical example.

The above scenarios are typical examples where routing more than reliability has been extensively studied by developing ad hoc routing protocols like the common AODV and DSR protocols. However transport protocols have not been researched much in networks where connectivity is highly intermittent and delays are huge like DTNs.

We have looked into a specific type of DTN namely Message ferry networks whose typical application is providing Internet access to places which are disconnected from the wired world. Consider a village which does not have Internet access as it is very remote from a wired infrastructure.

In order to better understand this type of network we analyzed the similarities of this network with currently available and similar technologies on the wired networks namely Ethernet and Token Ring. Chapter 2 provides a comparison of these three type of technologies.

As connectivity is highly intermittent, we studied the feasibility and use of the characteristics of the transport and application protocols in such networks and typical applications that will be used. These are done in Chapter 2 in sections Transport and Applications respectively. We observed that transport protocols are the main area of focus to be addressed. We wanted to develop a highly reliable transport protocol with throughput and complete use of available bandwidth as the major criteria for these type of networks.

A comprehensive study of the various variants of TCP has been done in Chapter 2 and the transport protocols that may be applicable to DTNs have been analyzed in Chapter 3. However, we were unable to fit any of the currently available transport protocols to work well and satisfy the requirements of high throughput and maximum bandwidth usage. Hence we have developed a customized version of TCP specific to Message Ferry DTNs. The characteristics and working of Message Ferry TCP are documented under System Design in Chapter 4 under conceptual design and opnet design.

The results of Message Ferry TCP are present in Chapter 5 under Numerical results. We have observed a drastic improvement in the throughput and bandwidth use of Message Ferry TCP. We conclude the thesis in Chapter 6 with room for future work.

Chapter 2

Context

2.1 DTN/MF

Delay Tolerant Networks are networks with intermittent connectivity which leads to long delays for data to reach from source to destination. This is because the network is partitioned due to either the power drain in the devices or the devices are compromised. Ad hoc and Sensor networks are good examples of DTNs. Consider a battlfield where the various sensor devices are installed on various battle equipment to form a network on the fly. But when the equipment are compromised the main network connectivity devices may be down. Thus the network needs some method to keep the connectivity reestablished either by bringing up new devices or increasing the coverage of existing devices or by making the existing devices to forward messages hoping that the messages will reach the destinations or by providing a device that goes around the area where the devices are present.

Thus the solution can be any of the following

1. Topology Control
    - Bring up new devices that are already installed but not active
    - Increase the power range of the various devices to reestablish a connected network
2. Reactive routing
3. Proactive routing

The solution considered over here is the Proactive routing. The specific solution is Message Ferrying, where the network connectivity is maintained by specific role mobile devices that go around the nodes to transmit and receive data. These special devices are called Message Ferries (or ferries in short). In the literature there have been various types of designs based on the number of ferries, the mobility of the nodes, the number of routes, the buffer level at the nodes and the establishment of inter route contacts.

For a quick recap of the literature, a brief introduction of the terminology is presented here for further reference in the following sections. The design may involve a single route for the whole span of the devices or multiple routes. Single route simplifies the design as this does not involve the problem of assigning nodes to various routes for minimizing estimated weighted delay (EMD) which is the minimization of the net delay in the route. For further details refer to [3]. Multiple routes minimizes the buffer spaces on ferries as the nodes come in contact with the ferry quite often. In single route with multiple ferries, the ferry buffers and delay are less as the traffic load between various nodes is shared by the various ferries. In the two scenarios considered above the ferries do not interact with each other. However the ferries can interact with each other in two ways. If the ferries interact among themselves, it is Ferry relaying and the if the interaction is via nodes which have sufficient buffers to store the messages from the ferries it is node relaying. Node relaying is better than Ferry relaying as the ferries need to synchronize in the ferry relaying in order to meet at regular interaction points called contact points. Thus to maintain the timing in meeting other ferries, the length of each ferry route should be the same. Thus delay in the route is increased and this in turn increases the buffer requirements at various ferries. Of the various designs considered the multiple ferry routes design performs the best and the Ferry Relaying design performs the worst.

Message ferry model has high propagation delay, high transmission delay, high queuing delay when compared to normal connected networks on the one hand and low efficiency and throughput on the other hand. This information can be correlated to the corresponding components in LAN technologies. This report establishes the similarities between the Message ferry model and the LAN technologies like Ethernet and Token Ring. The various characteristics considered are the propagation delay, transmission delay, queuing delay, maximum medium access time/ average medium access time, throughput and latency. The equations for various characteristics for all the three types of networks are provided in the respective sections.

2.2 LAN

The various characteristics are evaluated in detail in this section.

Propagation Delay

The propagation delay ρ of ethernet and of token ring depends on the distance between the source and destination and the speed of the signal in the medium.

The equation is as given below

$$\rho = \frac{Distance_{ij}}{Speed\ of\ the\ signal\ in\ the\ medium} \tag{2.1}$$

The propagation delay in a message ferry network depends on the various scenarios.

For single route, the propagation delay of a message is the time taken by the ferry to reach from the source node i to the destination node j. Let the length between the two nodes be l_{ij}. The constant speed of the ferry is f. The propagation delay is given by the expression

$$\rho = \frac{l_{ij}}{f} \tag{2.2}$$

For multiple routes in which no ferry interacts with any other ferry the propagation delay is again given by the same equation (2.2) for every individual route.

For routes in which the ferries interact the propagation delay depends on the type of relaying.

For node relaying, the common node acts as the contact point for adjacent routes. The message waits in the node - contact point - until the next ferry picks it up. The assumption is that the nodes as well as the ferries buffers are sufficient enough to accomodate the message. Consider for example that the source and destination are separated by one intermediate route as given in Figure (2.1). Thus the number of routes involved are three routes including the routes on which the source node and destination route are present.

Let the distance between the source i and the source route's contact point a be l^s_{ia}. The distance between the two contact points b and c is l^1_{bc}. The distance between the contact point c and the destination j is l^d_{cj}. Let the ferries in each route move with the constant speed f. The waiting time at each contact point is denoted by w Hence the propagation delay is given by

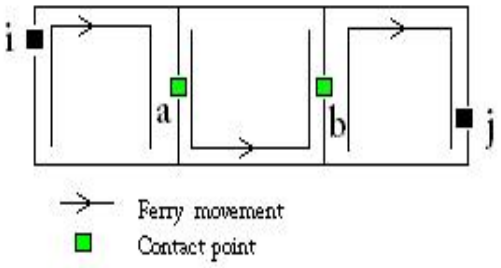

Figure 2.1: Node Relaying

$$\rho = \frac{l_{ia}^{s} + l_{bc}^{1} + l_{cj}^{d}}{f} + w_s + w_1 \tag{2.3}$$

For ferry relaying, the length of every route is assumed to be the same which is the length of the maximum route among all routes. Thus the equation for propagation delay is similar to node relaying but without the waiting time at the nodes as the messages do not wait at the ferries. This is similar to infinitesimal waiting time at the nodes when compared to node relaying. Thus the propagation delay is given by

$$\rho = \frac{l_{ia}^{s} + l_{bc}^{1} + l_{cj}^{d}}{f} \tag{2.4}$$

Thus the propagation delay in a message ferry network with ferry relaying is similar to that of Ethernet or Token ring which depends on the distance between the source and destination. In node relaying, the waiting time at the nodes is similar to congestion in the routers that connect the various Ethernet or Token rings. Higher the congestion higher the delay which is equivalent to larger the difference between the two routes, higher the delay. In single and multi route scenarios, there is no waiting time like a single network connected via a hub or switch. When multiple ferries are present in a single route, it resembles grouping of multiple interfaces of a switch to transfer data which leads to more data throughput with decrease in delay.

Transmission Delay

The transmission delay τ of Ethernet and Token ring depends on the rate of the interface as given by equation (2.5). Let the rate of the interface be r and the frame size be F_{ij}

$$\tau = \frac{F_{ij}}{r} \tag{2.5}$$

The transmission delay of message ferry is similar for the various scenarios. It just depends on the transmission rate of the interface. Let the rate of the wireless interface on the node and the ferry be r. For the sake of simplicity, the message can be transferred during contact between the node and the ferry instead of splitting it for future transmission. The transmission delay is given by

$$\tau = \frac{M_{ij}}{r} \tag{2.6}$$

where M_{ij} is the message size. The transmission delay is independent of the number of the ferries or the number of routes.

Thus like in Ethernet and Token ring the transmission delay is dependent only on the transmission rate of the interface.

Queuing Delay

The queuing delay of Ethernet and Token ring depends on the number of frames in the buffer in front of the frame. For Ethernet it is the sum of the propagation delays and the medium access time of all the packets in front of the frame.

The queuing delay of Token ring is given by the sum of the propagation delays of all frames in front of the frame in the buffer. But here the node transmits until its maximum token holding time (θ) exists. Once it expires it has to release the token for the next higher priority node to obtain the token.

In message ferry networks with mobile nodes, the nodes decide to either meet the ferry or not. This is decided on the amount of timeouts and buffer overflows that may occur if the node does not meet the ferry. If the amount of timeouts and buffer overflows is above a threshold then the node goes to the ferry to deliver the messages. Thus the queuing delay

depends on the threshold value and the maximum queuing delay is given by (2.7) for single ferry and by (2.8) for multiple ferries.

For static nodes there is no timeout or buffer overflow as the nodes meet the ferry during every trip. The equations are the same as for mobile nodes.

$$\text{Queuing delay } = \frac{L}{f} \tag{2.7}$$

$$\text{Queuing delay } = \frac{L}{fm} \tag{2.8}$$

Medium Access Time

The medium access time for LANs is the time taken by the device to start sending its frame on to the physical medium when it is the next frame to be sent on to the medium. Thus for Ethernet and Token ring it depends on how far it has to wait for the medium to be accessed. For Ethernet, it is given by (2.9) and for Token ring by (2.10)

$$\text{Average Medium Access Time } = N(5\rho + \tau) \tag{2.9}$$

$$\text{Maximum Medium Access Time } = \rho + \tau + (N-1)\theta \tag{2.10}$$

In message ferry networks, the medium access time is negligible. The nodes know when the ferry will be within range to transfer/receive data. When multiple nodes are present around when the ferry is nearby, a scheduling policy takes care of scheduling the various nodes to transfer data to the ferry to avoid collision. The ferry collects all the data from all the nodes when it visits the nodes because the route length is optimized to satisfy the bandwidth requirements of the nodes. In message ferrying the traffic rate is known before hand unlike Ethernet or Token ring.

Throughput

The throughput of the network is the average total transmission rate when the network is heavily loaded.

For Ethernet, the throughput is given by (2.11)

$$\eta_{eth} = \frac{1}{1 + 3\frac{\rho}{\tau}} \tag{2.11}$$

For Token ring, the throughput is given by (2.12)

$$\eta_{tr} = \frac{1}{1 + \frac{\rho}{N\theta}} \tag{2.12}$$

where N is the number of nodes in the Token ring.

For message ferrying, the throughput is given by (2.13)

$$\eta_{mf} = \frac{2rN}{L} \tag{2.13}$$

In message ferrying network the throughput is given by the ratio between the time period when the ferry is transmitting or receiving data to or from the nodes to the total time taken to travel the whole route once. Hence converting it to distance traveled, (2.13) is obtained. Thus if the ferry is transmitting or receiving messages without getting into *IDLE* state [2], maximum throughput is achieved. The throughput calculated is for every ferry in the route. Hence it does not differ if it is single ferry or multiple ferries in the route. However if the coverage areas overlap and the ferry goes to IDLE state between contacts with nodes, the throughput will be lesser than (2.13).

Latency

The latency of the network is the time a frame takes from the arrival at a network interface until it reaches the destination. This is the sum of the medium access time, the propagation delay, the transmission delay and the queuing delay.

For message ferrying networks, the latency is the sum of the time spent in the node after the frame is created, the time for the node to come in contact with the ferry and the time taken by the ferry to travel the distance between the source and destination in case of single route. If multiple interacting routes are included the time taken by each ferry to carry the message along each route should also be considered. Thus it is the sum of the relevant equations seen till now.

The similarities and differences between Message Ferrying and LAN technologies are given in Table. 2.1

Table 2.1: Comparison of characteristics among Ethernet, Token Ring and Message Ferry Networks

Characteristics	**Ethernet**	**Token Ring**	**Message Ferry**
Propagation Delay (ρ)	Similar	Similar	Similar
Transmission Delay (τ)	Similar	Similar	Similar
Queuing Delay	Different	Similar	Similar
Medium Access Time	Similar	Similar	Negligible
Throughput	40% - 90%	100%	Depends on nodes
Latency	Similar	Similar	Similar

2.3 Transport

Transport protocols provide end to end services for transfer of data. Certain transport protocols like TCP and SCTP are know for their end to end connectivity, in order delivery, low delay and high reliability in transfer of data. These protocols have an underlining assumption that the network connectivity will be viable for a significant period of time. Message Ferry networks on the other hand are high delay networks and highly disconnected. Thus when transport protocols are ported to message ferry networks, certain unique features specific to transport protocol either degrade the performance or completely fail in this network.

The various characteristics of transport protocols and their impact in message ferry networks are discussed below:

- Reliability
- End to end connection maintenance
- Connection orientedness versus connectionless
- End to end delay estimation
- Connection establishment
- Connection termination
- Effect of congestion
- In order delivery of data
- Throughput

Reliability

Transport protocols like TCP and SCTP provide reliable transfer of data across the network. However on the other hand UDP provides low delay but at the cost of reliability. TCP and SCTP use timers to ensure that the data segments or messages are received by the end host even when the network is prone to losses.

Message Ferry networks on the contrary have high delay. Hence reliability cannot be provided similar to TCP or SCTP using timers. Though the timer values can be set to very high values, the estimation of actual time taken will not be accurate as the route taken by the ferry can change in order to accommodate more data collection. Hence reliability should be provided but using different methodology

End to end connection maintenance

UDP does not ensure end to end connection maintenance. It just provides transport layer functionality with characteristics of IP layer delivery. On the other hand TCP and SCTP ensure end to end connection maintenance by maintaining the state of the connection at each end of the connection.

Message ferry network's requirements depend on the type of application requirement. Application requirements are discussed in the next section. Hence if just transfer of data is mandated, a protocol similar to UDP shall suffice. However if end to end connectivity has to be maintained to account for the amount of data transferred and the type of delivery, a protocol similar to TCP should be present.

End to end delay estimation

End to end delay estimation in transport protocols is performed to figure out loss of segments or messages. This ensures reliable transfer of data across the network. Thus the necessasity of this characteristic in message networks depends on the application used.

Connection establishment

In order to maintain end to end connectivity transport protocols establish a connection before transferring data on the connection. This ensures that the connection param-

eters are negotiated before transfer of data so that intermediate routers or other devices do not get overloaded. Even the end points can allocate specific resources for the connection.

In message ferry network, the intermediate device is the ferry. Though the ferry has unlimited buffer space, the ferry has to ensure that the connection be established before transfer of data if the application requires reliable transfer of data.

Connection termination

Connection termination is provided by reliable transport layer protocols to ensure that the resources at the end points are not held up unnecessarily even after the connection has terminated.

Message ferry network's transport layer requires this feature based on the application being supported. For reliable transport, connection termination has to be ensured.

Effect of congestion

TCP and SCTP detect congestion in the network by using lost segments. However in message ferry networks, the ferry has unlimited buffer and the ferry provides only means of communication. Hence congestion effects are not seen in message ferry networks unless buffer constraints are places on the ferry.

In order delivery of data

For applications which require instant delivery of message in order delivery is not strictly enforced like voice and video. However applications like FTP or HTTP require the data to be delivered reliably and in order so that the contents of the files transferred are in tact.

Hence the requirement for message ferry network again depend on the feasibility of the application.

Throughput

UDP provides high throughput as it does not establish connection or ensure in order delivery of data. However on message ferry networks where the delay is very high,

losses have a great impact on the application as the losses can be rectified only on the next visit of the ferry to the node if the ferry has buffered the lost segments.

Hence applications on message ferry networks that require high throughput require reliable delivery of data to ensure that throughput is not impacted.

2.4 Applications

Message ferry delay tolerant networks (MF DTNs) are networks where the the nodes are disconnected i.e. the distance between any two nodes is higher than the transmission radio range. A ferry node, which is mobile with unlimited buffer and energy moves around the deployment area to transfer data among the nodes which can either be mobile or static. Examples include a bus going around to various villages connecting the users for basic data transfer. Thus applications that can be supported are limited.

In the Internet various types of applications exist as MF DTN related constraints are not applicable. The applications can be broadly classified based on round-trip latency, throughput and reliability.

In DTNs, applications cannot work under the same assumptions as on the Internet. In the Internet the applications timeout after a certain period of time based on the interactivity of the application. These applications require very low latency. But in MF DTNs, as the network is disconnected most of the time, low latency applications do not perform well.

We study the performance of various categories of applications based on round trip latency, throughput and reliability. Round-trip latency(RTL) defines the amount of time the data takes to reach from the source to the destination and back to the source with the response if required. Otherwise it is just the one way latency. Throughput is the amount of data that can be transferred in unit time. This is generally in kB/sec or MB/sec. Reliability ensures that the data reaches the intended recipient and not lost. TCP provides in order reliability.

Based on these three properties, applications can be categorized under eight distinct classes. They are listed in Table. (2.2) along with examples where applicable.

The various classes mentioned above are studied to figure out if these applications can perform equally well in MF DTNs.

Table 2.2: Classification of Applications

Category			Examples
RTL	Throughput	Reliability	
Low	Low	Low	Temperature Readings
Low	Low	High	ATM Trans., Remote Login, Routing Updates,IM, Browsing
Low	High	Low	Streaming - Voice, Video
Low	High	High	Fast switching - Among core routers
High	Low	Low	e-mail, Image transfer, Offline Money transfer
High	Low	High	
High	High	Low	File transfer, Buffered - Voice, video
High	High	High	Remote Data Backup/Storage, Remote Data Processing

2.4.1 Low RTL, Low Throughput, Low Reliability.

This class of applications needs low latency such that the data reaches the users within a short period. However the amount of data to be transferred is low. Thus a typical example is temperature reading of a region. Even if certain readings are lost, this does not impact the user much. But the data should reach the user regularly. Considering from a MF DTN point of view this application does not suit a DTN as the delay in DTNs is not predictable. Thus regular periodic readings cant be delivered to the user.

2.4.2 Low RTL, Low Throughput, High Reliability.

Applications in this class require all data to reach the destination but the amount of data to be transferred is less. This include applications which are critical to users or errors in the transfer cannot be tolerated and demand low latency. Applications like ATM transaction, routing updates, instant messaging, remote login and browsing come under this category. These applications also do not fit in MF DTN as short periodic delivery of data is not feasible.

2.4.3 Low RTL, High Throughput, Low Reliability.

In this class of applications, huge amounts of data have to be transferred. Even if certain packets are lost, this does not impact the user at the other end. But the data

has to be sent regularly. Streaming applications like live video and audio come under this category. Even if some voice or video packets are lost, it is not noticeable at the other end. Again due to the periodicity of packet arrival, these applications do not fit well on DTNs.

2.4.4 Low RTL, High Throughput, High Reliability.

In certain applications, losses cannot be tolerated as processing time is very crucial and retransmissions cannot be accepted. Also the amount of data to be transferred is huge. Data transfer between switches on high speed networks over long distances fall under this category. The data changes so fast that old data is no longer usable. Hence retransmission is of no use. These are time critical applications and MF DTNs cannot handle this type of applications.

2.4.5 High RTL, Low Throughput, Low Reliability.

Here the user does not wait for the output but instead assumes that the application displays the output or stores the output so that it can be viewed later when it is available. Thus the end nodes involved can retransmit the data when data is lost. Hence applications like e-mail transfer, image transfer and offline money transfer which do not involve huge amounts of data fall under this class. As the output is not demanded instantly, these applications fit in a MF DTN. In this class, it does not matter where the other end is either in the same ferry network or part of a connected network.

2.4.6 High RTL, Low Throughput, High Reliability.

Similar to the above class of applications, these donot demand immediate response to the user, but here the data has to be transferred reliably else the processing power at the nodes is wasted. Thus this class of applications also fit well in a MF DTN.

2.4.7 High RTL, High Throughput, Low Reliability.

In this class of applications, huge amounts of data have to be transferred but losses can be tolerated as this does not impact the user. Examples include huge file transfers that include transfer of audio and video files for later viewing. These applications do not fit in

DTNs as storage is limited at the mobile nodes. However if the other end has unlimited buffer space, these applications work well in MF DTNs.

2.4.8 High RTL, High Throughput, High Reliability.

The last category of applications requires that the data has to be transferred reliably as the processing at the other end infers or collects statistics based on the data and the processing power is limited at the other end. Hence retransmission wastes valuable processing power which is not acceptable. It may also increase the queue size at the other end which in turn leads to packet drops. Examples include remote printing and backing up data. Hence this class does not fit well in MF DTNs as the queue keeps building up at the other end. However if processing power and buffers are unlimited, this class fits a MF DTN.

Table 2.3: Applicability to MF DTN

Category			DTN
RTL	Throughput	Reliability	
Low	Low	Low	No
Low	Low	High	No
Low	High	Low	No
Low	High	High	No
High	Low	Low	Yes
High	Low	High	Yes
High	High	Low	No(With limited buffers), Yes(With huge buffers)
High	High	High	No(With limited buffers), Yes(With huge buffers)

2.5 Prior Work

TCP has been designed for wired networks where the losses are considered due to network congestion than due to other causes as the wired networks have very low bit error rates and this option is ruled out. Wireless networks on the other hand have losses due to lossy channels, partition in the network and disconnection due to hand off.

The lossy nature of the channels leads to bit errors in the packets and these are dropped at the lower layer before they reach the transport layer. Thus the transport layer

detects this as a packet loss but not a corrupted packet. Hence TCP invokes congestion control in order to rectify the loss. This should be avoided because congestion control lowers the packet transfer rate which does not work out well in this scenario. TCP congestion control must be invoked only when there is real congestion in the network.

The wireless nodes get separated from the rest of the network due to obstacles in the path between the nodes or due to node mobility. This leads to packet drops at either the mobile station or the base station. During transfer from one cell to another cell, the mobile station does not transfer any data until the hand off is done. This also leads to packet losses at the transport layer. For both the above scenarios, TCP invokes congestion control which is not required or suggested.

Many protocols have been developed in order to rectify these issues. The various protocols that are discussed here are classified into four categories - End-to-end mechanisms, feedback based, split connection and separate layer implementation. In end to end mechanisms, the changes are implemented at the end nodes rather than either at the intermediate devices or any new layers are introduced. WTCP and transaction TCP come under this category. In feedback based protocols, the network layer or the data link layer provides feedback about the network status. Freeze-TCP and Adapted TCP fall under this category. In split connection based protocols, the connection between the end stations are split in between into two connections. The split mostly happens at the base station as that is the entry point into the wireless world. I-TCP and M-TCP are examples. In the last category, a separate layer is introduced either above the transport layer or below it. ATCP and MTCP come under this category. In order to be comprehensive, BIC and CUBIC have been discussed as these protocols enhance the throughput in high bandwidth delay networks.

2.5.1 Various TCPs

The four categories of the various protocols are discussed in this section.

Stream Control Transmission Protocol (SCTP)

SCTP [21] is a message oriented protocol with framing across message boundaries unlike TCP. SCTP creates multiple streams to transmit data to the destination. Losses in

one stream do not affect the other streams. In order to track loss of messages and locate the specific stream, two sets of sequence numbers are used - Transmission Sequence Number that tracks the messages and their losses and the Stream ID/Stream Sequence Number pair.

Loss in a particular stream leads to reception of messages with discontinuous Stream Sequence Number. Thus the stream can be identified and messages from the specific stream are buffered while the other streams have normal operation.

However the congestion and flow control mechanism are common across all the streams while multi-homing feature in SCTP allows the streams to be routed across different networks. This allows retransmission messages to be sent along a different path for higher probability of delivery.

Thus because of a common congestion and flow control mechanism, the impact due to wireless loss is similar to TCP.

ATCP

ATCP [10] places TCP in persist state when the loss is due to BER or partition of network. When ECNs are received from the network, ATCP does not perform any operation on TCP allowing it to go through normal TCP congestion control.

When packets are lost due to BER, ATCP - a layer between TCP and IP - puts TCP into persist mode and keeps retransmitting the lost segments and forwards the ACKs to TCP when they are received and removes the TCP from persist state. Even if the RTO is above the expire, ATCP puts TCP into persist mode and TCP comes back to normal mode when ATCP receives the ACKs.

When the network is partitioned placing the source and destination in separate networks due to movement of the nodes, Destination unreachable ICMP messages are generated by the network. ATCP on receiving these signal, puts TCP to persist state. ATCP maintains its own timers and retransmits the segments. When the network gets connected again the ACKs received by ATCP are forwarded to TCP placing it back in normal operation.

The experiments were conducted using 31 kbps links across the various nodes. The results show that ATCP does not bring down the cwnd to minimum when there is loss due to bit error rate or partition or route change. Thus it enhances the performance of TCP in wireless links but does not alter the window size based on the bandwidth of the network.

Hence it does scale well for higher Bandwidth delay product(BDP).

MTCP

In MTCP [7], a separate session layer protocol (MHP - Mobile Host Protocol) is introduced on top of TCP both on the mobile hosts and the base stations. There are no changes on the fixed hosts. The connection from the mobile host to the fixed host is split into two connections - one over the wireless link from the mobile host to the base station and the other from the base station to the host on the wired network. If the destination is also a mobile host, the connection is split into three connections.

There are two types of implementations. In MTCP (Multiple TCP), the connection is split into two and TCP is used for both the connections. In SRP (Selective Repeat Protocol), a protocol more optimized for the wireless characteristics is used to establish the transport connection on the wireless link. SACK is used to recover missing segments in an out of sequence transmission.

When data is transferred by the mobile host, the MHP layer on the base station, buffers the segments as the MTU on the wireless less is less than the wired network. This is done to efficiently use the bandwidth on the wired network.

MHP also takes care of hand offs from one base station to another. During the hand off process the remote process may loose some segments and are retransmitted. The paper does not explain about the impact on the throughput due these losses.

The experiments conducted show a performance improvement on MTCP and more improvement on SRP. For values of the improvement refer to the paper.

Adapted TCP

In Adapted TCP[9], the authors consider the scenarios when the source is fixed and mobile unlike other papers which consider only fixed hosts. In order to improve the performance of TCP, the network layer is assumed to provide feedback when the mobile host disconnects and reconnects to the network using *disconnect* and *connect* signals.

The RTO timers are stopped when there is an open window and the disconnection occurs. If it closed window, the RTO is allowed to expire. During connection, if the window is open, the packets are reconnected and new RTO is created, else RTO is made to expire.

When an RTO event occurs and if the network was disconnected at that time instance, the *sshthresh* is set to the current cwnd and *cwnd* is set to one. If not, retransmission occurs without changes in the two parameters. This is the scenario for MH to FH where MH is the source.

For FH to MH transmission, the ACKs of the last two bytes are delayed for at most 500 ms. Hence when a disconnection occurs, it is signaled to the transport layer. Upon connection, the first byte is acked with zero window size and the second with full window size. This makes the FH to retransmit all the unacked segments. Thus congestion control is avoided when packets are lost due to MH disconnection.

The performance were tested for WLAN and WWAN. The results indicate better performance for both the scenarios. The performance of ATCP is better at higher RTT values as normal TCP has higher RTO values. This occurs when the MH is the source. For FH to MH, the performance is similar to Freeze-TCP for lower RTT but less than Freeze-Tcp when the RTT values are high. Again this protocol does not address the behavior during higher bandwidth delay product.

Freeze TCP

In Freeze TCP [4], the mobile nodes can sense that they are about to be handed off to another base station. Also the mobile node can detect a disconnection based on the fading signal strength. Freeze-Tcp uses this information to send zero window size to the sender so that the sender can get into persist mode. This prevents the sender from going into slow start phase which under utilizes the bandwidth.

The experiments were conducted for 10 Mbps (local and remote connection), 100 Mbps (local), 38.4 Kbps(PPP) (local and remote). The results show that for 100 Mbps, the performance was very good. For 10 Mbps, the performance was better than normal TCP but at times it was the other way round but the loss was very marginal. For 38.4 Kbps, the performance was similar to normal TCP.

Thus Freeze-Tcp does not perform well for routes with short thin pipes as the packet drops at intermediate routers cannot be controlled by Freeze-Tcp unlike M-TCP which configures intermediaries like the base station. The bandwidth delay product for thin pipes is less. Hence the impact due to Freeze Tcp does not improve the performance against normal TCP. Freeze-tcp performs well when the bandwidth delay product is large.

I-TCP

I-TCP [5] considers a network with one end of the connection being mobile. The mobile host (MH) on the wireless link communicates with a fixed host (FH) on the wired network. I-TCP divides the connection into two parts, one part from the MH to the mobile support router (MSR) - also called the center point - and another part between the MSR and the FH. When the MH moves from one MSR to another MSR, the FH is not aware of the shift. The MH initiates a communication to the FH via the MSR which uses the MH's address and port to initiate a connection with the FH. On moving to the new MSR, the same information is used by the new MSR. This maintains the end to end connection instead of being teared down by the FH. When the MH moves to the new MSR, the MSR initiates slow start and congestion control with the mobile but with the retransmission timers being reset. This improves throughput because the distance between the MH and MSR is less and hence the RTT is less which in turns makes the window size to reach its previous state much faster than the window size in normal TCP connection after disconnection. The MSR also puts the FH in persist state if the buffers at the MSR get full during the short disconnection between the MH and the MSR.

The performance of I-TCP in LAN and WAN environment has been studied. The performance improvement is greater in the WAN environment than the LAN improvement because the RTT of the whole connection is larger in the WAN setup while the RTT for the wireless link is relatively small in WAN than in LAN setups. This protocol does not cater to the high delay and bandwidth scenario.

M-TCP

In M-TCP [6], the network infrastructure assumed is a three-level hierarchy. The mobile hosts get connected to the base stations which are in turn connected to the supervisory hosts (SH). The base stations are relieved of hand offs and the SH take care of hand offs when the mobile hosts move from one SH to another. A bandwidth management module is present at the SH which allocates bandwidth for each mobile connection and adjusts it periodically based on current bandwidth usage and other mobile connections. It is assumed that the BER is low as the link layer uses FEC to reduce BERs.

The connection between a host and the mobile host is split into two at the SH. The SH has two modules implemented, the SH-TCP and the M-TCP. The SH-TCP communi-

cates with the sender and the M-TCP with the mobile host. The SH acks data only when it receives ACKs from the MH but retains the ACK of the last byte. When a disconnection occurs at the MH, the SH-TCP sends an ACK for the last byte with a window size of zero forcing the sender into persist mode. When the MH reconnects, it sends a greeting packet to the SH and this removes the source from the persist state.

The M-TCP at the mobile host freezes the timers when the MH disconnects. On reconnection the M-TCP sends the sequence number of the last byte received thus far and unfreezes the timers. The M-TCP at the SH determines that the MH has disconnected because it does not receive ACKs from the MH for a long duration of time. This moves the M-TCP at the SH into persist state. After the MH reconnects, it is forced to respond to the persist probe packets from the M-TCP on the SH. The greeting from the MH puts the sender back to normal mode by the SH.

The experiments have been performed over short RTT and long RTT ranges. The performance is better on shorter RTTs than on longer RTTs.

Binary Increase Congestion Control

BI-TCP [11] scales well when the BDP of the network is high i.e. high speed networks with long delays. BI-TCP performs like normal TCP when the BDP is less. When the BDP is above a threshold, the Binary increase congestion control kicks in. Thus this protocol performs well for all scenarios. But this protocol assumes that losses are due to congestion as it assumes the work on wired networks. When a loss is detected, the new target -window size - is calculated which is based on the current window size and the window size after it has been reduced by a scaling factor β.

CUBIC [12] is an enhanced version of BIC which increases the fairness of BIC while maintaining the scalability and stability of BIC. The main change is in the calculation of the congestion window size. Here also the loss is considered only due to congestion like in BIC.

Transaction - TCP

Transaction TCP [13] is very useful in cases where the number of data transfers is less like in DNS address resolution of host name. The SYN, FIN and DATA are sent in

the same segment. There is no separate connection establishment and termination phase. Once the other end receives data it sends back the ack for the SYN, the required data and its FIN. The data is ACKed by the sender. Thus the whole data transfer takes place using three transfers. Thus this reduces the number of segment exchange and the percentage decrease in number of packets reduces as the number of packets exchanged increase.

WTCP

Wireless TCP (WTCP) [8] is developed for Wireless WANs which are characterised by non-congestion related packet loss, very low bandwidth, large round trip times, asymmetric channels and occasional disconnections. WTCP works on congestion control and reliability in WWAN networks unlike other works which try to reduce the lossy nature of the network but retain the congestion mechanism of TCP.

WTCP implements a rate based mechanism for congestion control unlike normal TCP which uses ACKs for self-clocking. WTCP uses the ratio between the receiver's inter-packet delay and the sender's inter-packet delay to estimate the transmission rate in the network. When congestion is incipient, the rate is adjusted according to the calculations performed at the receiver. When the network is in congestion the transmission rate is decreased aggressively to reduce packet loss and move out of congestion quickly.

For reliability, WTCP uses Selective ACKs instead of RTOs as in normal TCP. By looking at the ACKs from the receiver, the sender can determine the lost packets in the stream. As the sender has to receive sender's transmission rate, the ACKs have to be received periodically (5 secs) else it is considered as a blackout.

The experiment results show that TCP does not distinguish between congestion and losses and hence WTCP is able to get back to its sending rate after blackout while TCP goes to slow start. Also the rate of transmission is not affected by the random losses as they are within the threshold limits of WTCP.

Sensor Transport Control Protocol (STCP)

Sensor Transport Control Protocol (STCP) [17] is a generic, scalable and reliable transport layer protocol. As the sensor nodes are energy constraint, much of the functionality is implemented in the base station which is mostly the data sink node for the network

where the energy, memory and processing power are unlimited compared to the nodes. The sensor nodes have either a continuous flow or event based flow.

For continuous flow, the base station responds with NACKs and for event based with ACKs. In the continuous flow the sensors initiate an association with the base station. Thus the base station has knowledge about the time interval between successive packets from the sensor nodes as they are periodic transmissions. In event based, the timing of packets cannot be determined.

In order to provide congestion detection and avoidance, the intermediate nodes set the congestion notification bit in the data packet which is notified to the sensor node via the acknowledgment packet to the sensor node.

In order to provide the required reliability, the base station determines if the NACK has to be sent for a certain packet based on the requirements of the reliability. If the reliability is above the desired reliability level, no NACKs are sent. Similarly for event based flows, the sensor nodes removes the packets if the reliability is above the threshold else they will be retransmitted after a timer fires.

The paper presents the result for number of NACKs being sent, average packet latency and energy spent for different reliability. The unnecessary NACKs were higher for lower values of Estimated trip time (ETT) - the difference between the clock time at the node and the base station. The results are considered with loss in the intermediate nodes. The latency is proportional to the ETT. For nodes farther from the base station, the ETT is higher.

The energy spent was higher for higher reliability requirements with 30% loss at each node.

Snoop Protocol

The Snoop protocol [19] makes changes to the network layer at the base station. The routing code is modified which basically caches the data packets sent by the FH and retransmitting data over the wireless link based on the acknowledgments received from the MH.

The protocol as described above involves two main methods snoop_data and snoop _ack. The snoop_data method works as follows. If the packet received is a new packet, the packet is cached at the base station and forwarded to the MH. IF the packet is not, the

packet is forwarded to the MH and the retransmission timer at the base station is reset. However, if the packet is not in sequence and not cached at the base station, the packet is forwarded and marked as due to congestion loss in the wired network.

The snoop_ack method handles the acks as described below. If a new ack is received, it is forwarded to the FH and the cache is cleared along with updating RTT. If the ack is an ack already seen, it is silently discarded. However a duplicate ack is handled in two ways. If it is first dup ack, the packet is retranmissted with a high priority else it is discarded.

For transmission from the MH to the FH, the MH uses SACKs to handle multiple dropped packet on the wireless links in a single window. The performance of Snoop protocol 20 times better than normal TCP with errors in the wireless medium. Under very low or no errors, the protocol performs similar to TCP and thus there is no overhead due to snoop protocol.

TCP Westwood

TCP Westwood [22] makes changes to the sender side congestion window based on the rate of the connection by monitoring the rate of returning ACKs. When congestion occurs, regular TCP drops down the congestion window to 1 whereas TCPW goes back to the state it was present in before the congestion. The paper proves that the throughput improvement is more pronounced on lossy links like the wireless networks.

2.5.2 Performance comparison

In order to compare the performance of various TCPs, let us describe the various scenarios under which they are compared. For effecient working of TCP, the window size should be twice the bandwidth delay product in order to effectively use the whole bandwidth of the connection. Based on the bandwidth and delay of the network connections, the networks can be classified into four categories: *Short thin pipes(STN), short fat pipes(SFN), long thin pipes(LTN) and long fat pipes(LFN).* The thick and thin depend on the bandwidth of the network. The higher the bandwidth, the fatter the pipe is. The short and long represent the delay in the network which is half the round trip time of a packet from the source to the destination and back to the source provided the path is symmetric. The higher

the delay, the longer is the distance between the source and destination (considered as a pipe). The performance of various TCPs without loss under these four categories is given in Table. 2.4. Table. 2.5 refers to the same parameters under lossy conditions.

The following is the legend for certain parameters being used in Table.2.4 and Table. 2.5.

MTT - Mean Time to Transfer.

UU - Underutilization of network bandwidth.

TDT - Throughput when Disconnection Time increases.

SI - Stability index is the standard deviation normalized by the average throughput.

T - Throughput without loss.

The signaling mechanism in the various TCPs is given in Table.2.6. If the changes are made only on the end hosts, the signaling system is considered to be end-to-end. For some protocols, changes are made in the base station and these are considered intermediaries.

Table 2.4: Evaluation of various characteristics - Without Loss

TCP variant	**STN**	**SFN**	**LTN**	**LFN**
A-TCP MTT (1MB)	 ↓ 1/3	 ↓	 ↓	UU
Adapted-TCP Throughput TDT	 ↑ 40%	 ↑	 ↑↑ 150%-50%	UU
FAST TCP				
FREEZE-TCP Throughput	 ↑↑	 ↑↑↑	 ↑	UU
I-TCP Throughput	 ↑	 ↑	 ↑↑	UU
M-TCP MTT (0.5MB, 1MB)	 ↓	 ↓	 ↓↓	UU
MTCP MTT (100KB) Hand off Time(↑)	 ↓(20%) Loss	 in between Loss	 ↓3 times Loss	UU Loss
Snoop Protocol Throughput	 No change	 No change	 No change	UU
STCP				
TCP Westwood RTT Fairness Friendliness	 ↑ No change	 ↑ No change	 ↑ No change	UU
Transaction-TCP Total required Packets	 ↓↓	 ↓	 ↓	UU ↓
WTCP MTT (100KB) Number of packets transferred Bandwidth Fairness	 ↓ ↑ ↑	 ↓ ↑ ↑	 ↓ ↑↑ ↑	UU
BI-TCP Throughput SI	 TCP	 TCP	 TCP	 Better $0.1 < SI < 0.3$
S-TCP Throughput SI	 TCP	 TCP	 TCP	 Best $SI < 0.1$
Fast-TCP Throughput SI	 TCP	 TCP	 TCP	 Better $0.1 < SI < 0.3$
HS-TCP Throughput SI	 TCP	 TCP	 TCP	 Better $0.1 < SI < 0.3$
HSTCP-LP Throughput SI	 TCP	 TCP	 TCP	 Good $SI > 0.3$
H-TCP Throughput SI	 TCP	 TCP	 TCP	 Better $0.1 < SI < 0.3$

Table 2.5: Evaluation of various characteristics - With Loss

TCP variant	**STN**	**SFN**	**LTN**	**LFN**
A-TCP				UU
MTT (1MB)	$\downarrow$ 1/3	$\downarrow$	$\downarrow$	
Adapted-TCP				UU
Throughput	$\uparrow$	$\uparrow$	$\uparrow\uparrow$	
TDT	40%		150%-50%	
FAST TCP				
FREEZE-TCP				UU
Throughput	$\uparrow\uparrow$	$\uparrow\uparrow\uparrow$	$\uparrow$	
I-TCP				UU
Throughput	$\uparrow$	$\uparrow$	$\uparrow\uparrow$	
M-TCP				UU
MTT (0.5MB, 1MB)	$\downarrow$	$\downarrow$	$\downarrow\downarrow$	
MTCP				UU
MTT (100KB)	$\downarrow$(20%)	in between	$\downarrow$3 times	
Handoff Time($\uparrow$)	Loss	Loss	Loss	Loss
Snoop Protocol				UU
Throughput	$\uparrow$ ($1to20$)			
STCP				
TCP Westwood				UU
Throughput(Loss)	$\uparrow$ (394%)	$\uparrow$	$\uparrow$	
Friendliness	$\uparrow$	$\uparrow$	$\uparrow$	
LER(Burst Errors)	T($\uparrow$)	T($\uparrow$)	T($\uparrow$)	
HER(Burst Errors)	Poor	Poor	Poor	
Blackouts(0.1sec)	T($\uparrow$167%)	T($\uparrow$)	T($\uparrow$)	
Blackouts(Larger)	Poor	Poor	Poor	
Transaction-TCP				UU
Total required Packets	$\downarrow\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
WTCP				UU
MTT (100KB)	$\downarrow$	$\downarrow$	$\downarrow$	
Number of Packets trans.	$\uparrow$	$\uparrow$	$\uparrow\uparrow$	
Bandwidth Fairness	$\uparrow$	$\uparrow$	$\uparrow$	
BI-TCP				
Throughput	TCP	TCP	TCP	
S-TCP				
Throughput	TCP	TCP	TCP	
Fast-TCP				
Throughput	TCP	TCP	TCP	
HS-TCP				
Throughput	TCP	TCP	TCP	
HSTCP-LP				
Throughput	TCP	TCP	TCP	
H-TCP				
Throughput	TCP	TCP	TCP	

Table 2.6: End to End connection for various TCPs

TCP variant	**Signaling**
A-TCP	End-to-end
Adapted-TCP	End-to-end
Bic-TCP	End-to-end
FREEZE-TCP	End-to-end
I-TCP	Intermediaries
M-TCP	Intermediaries
MTCP	Intermediaries
TCP	End-to-end
Snoop Protocol	Intermediaries
STCP	Intermediaries
TCP Westwood	End-to-end
Transaction-TCP	End-to-end
WTCP	End-to-end

Chapter 3

Problem Definition

3.1 Basic Problem

From the analysis of transport and application protocols in the previous chapter we conclude that applications which are highly delay tolerant and which do not require immediate response are well suited for Message Ferry networks. Typical applications include file transfers, video downloads and download of mails from the server for a specific region.

From the study of transport protocols in Chapter 2, currently available transport protocols like [11], [12] are designed for maximum bandwidth utilization but do not address the issue of high disconnectivity. Protocols like [7], [5], [6], [4] , [19] segregate the impact of losses and network disconnection due to wireless network from affecting the wired network at the transport layer. Their approach is very pertinent to Message Ferry networks.

Message ferry networks are networks where the ferry - a mobile device - keeps collecting information from various nodes in the area by visiting each node along its path and transferring data destined to them. In order to achieve this, the ferry is equipped with unlimited buffer space. The ferry tries to adjust its route so that it transfers and receives all the data available at the nodes. This is feasible either by pre-defined amount of transfer from the nodes or by slowing down at the nodes to gain more time when the ferry is near the node.

Hence in order to provide high throughput and maximum bandwidth usage, a

typical transport protocol with both the characteristics of protocols like BI-TCP, CUBIC and protocols like MTCP, M-TCP, Freeze TCP and Snoop is required for Message Ferry networks. In the next Chapter we analyze the applicability of the second set of protocols on message ferry networks.

3.2 TCP and MF

3.2.1 Freeze-TCP

Freeze-TCP expects the receiver to send a Zero Window Advertisement to the sender so that the sender can stop transmitting any more data and avoid changing its window to one segment when the receiver senses a disconnection or handoff. The base station and the sender require no change like other protocols. However only one end of the connection is on the wireless network. The effect of Freeze-TCP with both ends in the wireless networks is not addressed.

The receiver is expected to send a Zero Window Advertisement (ZWA) at least one RTT prior to disconnection to ensure that at least one ZWA reaches the sender. Freeze-TCP also assumes that the disconnection occurs when data is being sent by the receiver. The impact when the receiver is idle is not addressed. Also the behavior when the link is down when the connection is established is not addressed. However we assume that the receiver doesn't send any data unless there is a viable data transfer so that the timers do not timeout at the receiver.

Data transfer - Freeze-TCP - Mobile Host to Mobile Host

The topology is shown in Figure (3.2).

- Open a TCP connection between MH A and MH B
- Send data from MH A to MH B
- Data reaches MH B and acknowledgments are sent to MH A
- MH A receives acknowledgments

- MH A senses a disconnection and sends ZWA to MH B
- MH B goes into persist state
- MH A reconnects and sends 3 duplicate ACKs of the last segment to MH B
- MH B comes out of persist state and resumes standard TCP operation

When both MH A and MH B get disconnected, they send ZWA to the other. We assume that the ZWA reach each other successfully. When they are reconnected, they send the 3 duplicate ACKs. However if both do not get reconnected, segments may be lost and a host may get into slow start phase.

Data transfer - Freeze-TCP - Mobile Host to Mobile Host via Ferry

The topology is shown in Figure (3.3).

- Open a TCP connection from Node 2 to Node 4
  - Node 2 senses the presence of ferry nearby
  - Node 2 sends SYN to ferry
  - Ferry does not respond with any acknowledgments and hence the connection times out
- Ferry out of communication range
  - Node 2 senses no presence of ferry and does not transfer any data

However when the timers at the nodes are changed to reflect the ferry route time, the data throughput rate will be less as the acknowledgments do not arrive until the ferry comes back. But even with this modification, the connection is established only when the ferry is within communication. When the ferry is not within communication range, the connection aborts as all the SYN segments get lost.

3.2.2 Snoop Protocol

Snoop protocol introduces a snoop module at the base station in the network layer which monitors every packet that crosses the base station in both directions. This maintains

end to end semantics of TCP unlike MTCP and I-TCP. Snoop protocol is efficient when the lost segments are within a single window of segments being transferred.

A typical topology for snoop protocol is shown in Figure (3.1).

The snoop protocol mostly handles packets from the FH to MH. However, in order to handle packet losses from the MH to FH, the snoop implementation modifies the TCP protocol at the MH so that Selective Acknowledgment (SACK) is enabled. Hence when packets transmitted from MH to FH are lost, the BS sends SACK information so that the MH can retransmit only the lost packets.

When a packet arrives from the fixed host (FH) on the wired network to be sent to the mobile host (MH) on the wireless network, the snoop layer caches the packet in its buffer and forwards the packet to the MH. If an out of sequence packet that has been cached is received again at the base station, it is assumed as being timed out at the FH and the retransmission count is reset. However if the segment had a lower sequence number than the last ACK received, the latest ACK is sent to the FH. If an out of sequence packet that has not been cached has been received, it is marked as having experienced congestion loss as it has been lost on its previous transmission from the FH.

When an ACK is received from the MH and it is less than the latest ACK it is silently discarded as TCP ACKs are cumulative. When a new ACK is received, either a retransmission is done or the buffer cache is cleared for all the ACKed packets and the ACK is forwarded to the FH. When a duplicate ACK is received, it is either forwarded to the FH or discarded or the packet is retransmitted to the MH. For a more detailed description refer to ([19])

The simulation results show improvement in the throughput which indicates that snoop protocol can handle higher bit error rates than standard TCP.

Data transfer - Snoop protocol - Fixed Host to Mobile Host

The topology is shown in Figure (3.1).

- Open a TCP connection from FH to MH
- Send Data from FH to MH.
- Base station receives segments in order. Caches the segments received and forwards the segments to the MH.

Figure 3.1: Fixed and Mobile Host Network

- Segments arrive at MH.
  - MH sends acknowledgment
  - Base station receives the acknowledgment
    - If a new acknowledgment, clear the cache of the segments and estimate the Round Trip Time(RTT) for the last hop. Forward the Acknowledgment(s) to the FH
    - If an old acknowlegement, discard it
    - If a Duplicate acknowledgment (DUPACK), resend the lost segment if in the cache immediately if this is the first DUPACK else ignore the DUPACK if this DUPACK is expected. If the DUPACK has no segment in the cache or the segment experienced congestion, forward the DUPACK to the FH as the segment related to this DUPACK has not reached the BS. This may be due to congestion in the wired network.
- Segment is lost
  - If no previous segments arrived at the MH, no ACKs will be sent to BS. The BS timeouts because its persist timer timeouts and resends the segments
  - If previous segments arrived at the MH, DUPACKs will be sent to the BS

- Segment arrives out of order.
  - If the segment is not already in cache, the segment has been lost earlier due to congestion or segment has been reordered. The segment is marked as congestion experienced.

- If the segment is already in the cache, the segment may be retransmitted by the FH. The retransmission counter is reset and the cached packets are sent to the MH.

Data transfer - Snoop protocol - Mobile Host to Fixed Host

- Open a TCP connection from MH to FH
- Send data from MH to FH
- Segments arrive at BS. Send them to the FH.
- Certain segments lost
  - Send Acknowledgment for last received in sequence segment
  - Send SACK information which indicates out of segments received at the BS
  - MH retransmits the lost segments indicated in the SACK

Data transfer - Snoop protocol - Mobile Host to Mobile Host

The topology is shown in Figure (3.2).

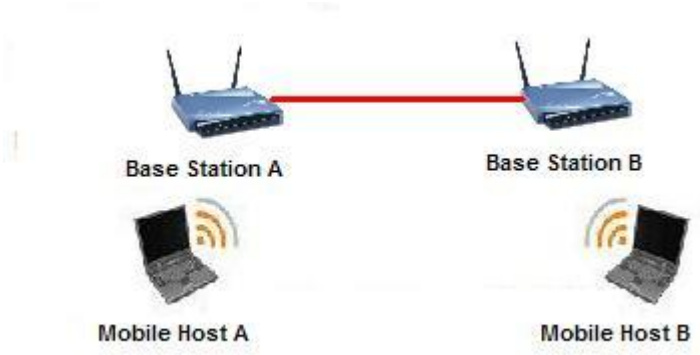

Figure 3.2: End-End Mobile Hosts Network

- Open a TCP from MH A to MH B
- Send data from MH A to MH B

- Segments arrive at BS associated with MH A - referred as BS A
  - BS A sends the segments to BS B which caches and forwards the segments to MH B
  - MH B sends acknowledgments. BS B clears the cache at BS B and forwards the acks to MH A.
- Segments do not arrive at BS A
  - If no segments were sent previously, the MH A times out and retransmits the segments
  - If segments already arrived at BS A, SACK information is sent to MH A
- Segments arrive at BS A. do not reach MH B
  - BS A sends the segments to BS B which caches and forwards the segments to MH B
  - Segments do not arrive at MH B. BS B times out and resends the segments

Data transfer - Snoop protocol - Mobile Host to Mobile Host via Ferry

The topology is shown in Figure (3.3).

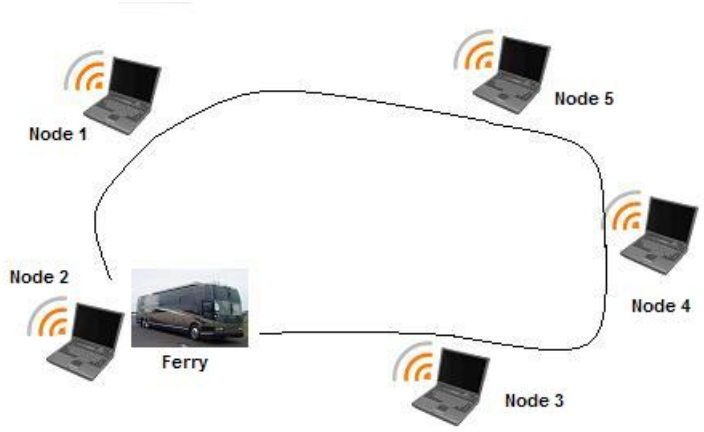

Figure 3.3: Message Ferry Networks

- Open a TCP connection from Node 2 to Node 4

- Ferry within communication range
  - The SYN segments time out as the ferry does not ACK the segments
- Ferry out of communication range
  - Node 2 times out and keeps retransmitting data until the ferry reaches the node or the connection is aborted

However even if the timeout values are high i.e. if the timeout is equivalent to the total time taken by the ferry to complete one trip, the timeout values have to be changed if there is a difference in the time taken by the route when new nodes comes up or go down. Thus timeouts may occur if there is a delay in the ferry arrival at the node. However even if timeouts are estimated correctly, the throughput will be very low as the ferry does not acknowledge any segments. The acknowledgments are sent only by the nodes. But even in this scenario, the connection is aborted when the ferry is out of communication range as all the SYN segments are lost due to out of communication range loss.

3.2.3 MTCP

MTCP addresses the drop in throughput due to smaller MTU in wireless links, wireless loss and loss of packets due to handoff by introducing a new session layer above TCP namely Mobile Host Protocol (MHP).

The TCP connection spans a short wireless network and then on to a wired network to a fixed host. MHP is introduced in the base station and the mobile host so that the impact of erratic behavior in the wireless links does not affect the TCP connection over the wired network. The fixed host is not altered.

When an application on the mobile host wants to establish a connection to a fixed host, the MHP on the mobile host establishes a connection to the MHP peer on the current base station. The base station's MHP establishes a separate TCP connection to the fixed host. If the two ends are mobile hosts, the TCP connection is split into three separate connections. The transport connection between the mobile host and the base station has smaller MTU while the TCP connection on the wired network is higher. The base station buffers the received packets to be split across desired MTU boundaries. When an application

on the fixed host wants to establish a connection to the mobile host, the connection goes through the MHP on the base station.

The transport protocol between the mobile host and the base station is either TCP or a customized wireless transport protocol. The first option is MTCP (Multiple TCP) and the other is Selective Repeat Protocol(SRP). SRP is customized for wireless networks and the throughput is better than TCP.

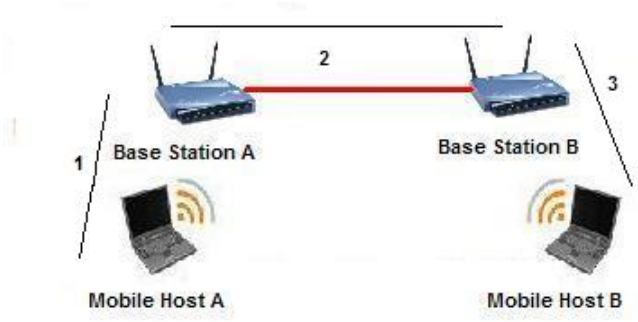

Figure 3.4: Split TCP Connections

Data transfer - MTCP - Mobile Host to Mobile Host

The topology is shown in Figure (3.4).

- Open a TCP connection from Node 2 to Node 4
- Send data from Node 2 to Node 4
- MH A opens a TCP connection to BS A. BS A opens a TCP connection to BS B. BS B in turn opens a TCP connection to MH B
- Segments are sent to BS A
- BS A acks the segments from MH A
- BS A sends the segments by buffering the segments and segmenting across the wired MTU to BS B which acks them
- BS B buffers the segments and segments across the wireless MTU and sends the segments to MH B

- If losses occur standard TCP or SRP takes care of them

Data transfer - MTCP - Mobile Host to Mobile Host via Ferry

The topology is shown in Figure (3.3).

- Open a TCP connection from Node 2 to Node 4
- Send data from Node 2 to Node 4
- Ferry within communication range
  - The MHP on node 2 opens a TCP connection to MHP on ferry
  - The ferry ACKs the segments from node 2 and stores the segments in its buffers
  - The connection on node 2 times out when the ferry moves out of communication range
  - The ferry moves to node 4. The MHP on the ferry sets a TCP connection with MHP on node 4 and transfers the data.
  - The connection on node 4 also times out and aborts after some retries when the ferry goes out of communication range.
  - When the ferry returns to node 2, the connection is aborted if the route trip time is large. Similar behavior is observed at node 4
- Ferry out of communication range
  - The TCP connection times out and aborts after certain number of retries

However if the timers are equivalent to the route trip time of the ferry, the throughput will be sufficiently high for connections that get established with the ferry. But for those connections that are initiated when the ferry is out of range, the connections will be aborted due to out of communication SYN segment drops.

3.2.4 M-TCP

M-TCP uses a split TCP approach but the ACKs are not generated by the device where the split occurs, instead the original ACKs are forwarded to the required destination.

The architecture is a three level architecture. The MHs are connected to the base stations which are in turn connected to the Supervisor Hosts (SH). The TCP split occurs at the SH. The TCP modifications are made at the base station and the MH. The FH is unaltered.

When a TCP segment is received, the segment is received by the TCP client at the SH - SH-TCP. This module forwards the segment to the M-TCP module on the SH. The M-TCP sends the segments to MH. The ACKs received by M-TCP are forwarded to SH-TCP which forwards all the ACKs except the ACK for the last byte. SH-TCP doesn't generate its own ACKs as in MTCP and I-TCP. Hence M-TCP ensures end to end TCP semantics. However when the MH gets disconnected, the M-TCP client does not get ACKs for certain bytes of data that have been sent to the MH. The M-TCP notifies the SH about this disconnection which in turn sends zero window update to the sender in the ACK for the last byte. This puts the sender in persist mode. On reconnection, the MH sends a greeting packet to the SH. This is notified by M-TCP to SH-TCP which opens up the window on the sender.

Hence the disconnection at the wireless network does not decrease the congestion window size at the sender.

Data transfer - M-TCP - Mobile Host to Mobile Host

The topology is shown in Figure (3.2).

- Open a TCP connection from MH A to MH B
- Send data from MH A to MH B
- The segments are intercepted by M-TCP at the SH and forwarded to SH-TCP. SH-TCP sends the segments to SH-TCP at the SH near MH B
- The SH-TCP at the SH near MH B sends the segments to M-TCP. M-TCP opens a TCP connection with MH B
- MH B acknowledges the segments
- SH-TCP at SH near MH B forwards the acks of all the segments except the last byte to SH-TCP at SH near MH A

- The SH-TCP at SH near MH A forwards all the ACKs to MH A
- When MH A or MH B disconnect, the SH-TCP at the SH send the ACK of the last byte with a zero window. This puts the other end in persist state
- When the disconnected MH reconnects, it sends a greeting packet to its SH which is sent to the destination

Data transfer - M-TCP - Mobile Host to Mobile Host via Ferry

The topology is shown in Figure (3.3).

- Open a TCP connection from Node 2 to Node 4
  - Node 2 sends a SYN to the ferry
  - The connection establishment times out as the ferry does not ACK in this scenario

However if the timers are equal to the route trip time of the ferry, the throughput will be low. When the ferry is out of communication range, the connections are aborted as the SYN segments are not received by the ferry.

3.2.5 I-TCP

Indirect TCP is based on Indirect protocol which splits the TCP connection between a MH and a FH at the base station into two separate TCP connections so that the special requirements of the wireless medium can be handled by the TCP connection on the wireless medium.

The mobile host when it wants to establish a connection to a fixed host, the I-TCP on the mobile host sends a request to the base station. The base station in turn establishes a separate TCP connection to the fixed host with the base station as the relay agent between the two TCP connections. Hence the base station ACKs data before the actual destination sends the ACKs.

The throughput improvement in a WAN environment is significant than in a LAN environment with I-TCP while TCP suffers because of packet loss due to handoff between the cells when the MH moves from one cell to another and due to general wireless loss.

Data transfer - I-TCP - Mobile Host to Mobile Host

The topology is shown in Figure (3.2).

- Open a TCP connection from Node 2 to Node 4
- Send data from Node 2 to Node 4
- MH A opens a TCP connection to BS A. BS A opens a TCP connection to BS B. BS B in turn opens a TCP connection to MH B
- Segments are sent to BS A
- BS A acks the segments from MH A
- BS A sends the segments to BS B which acks them
- BS B buffers the segments and sends the segments to MH B
- If losses occur standard TCP handles them

Data transfer - I-TCP - Mobile Host to Mobile Host via Ferry

The topology is shown in Figure (3.3).

- Open a TCP connection from Node 2 to Node 4
- Send data from Node 2 to Node 4
- Ferry within communication range
  - The TCP at node 2 opens a TCP connection with ferry
  - The ferry ACKs the segments from node 2 and stores the segments in its buffers
  - The connection on node 2 times out when the ferry moves out of communication range
  - The ferry moves to node 4. The ferry opens a TCP connection with node 4 and transfers the data.
  - The connection on node 4 also times out and aborts after some retries when the ferry goes out of communication range.

  - When the ferry returns to node 2, the connection is aborted if the route trip time is large. Similar behavior is observed at node 4

- Ferry out of communication range

  - The TCP connection times out and aborts after certain number of retries

However if the timers are equivalent to the route trip time of the ferry, the throughput will be sufficiently high for connections that get established with the ferry. But for those connections that are initiated when the ferry is out of range, the connections will be aborted due to out of communication SYN segment drops.

3.2.6 TCP Spoofing

In TCP spoofing [18],an intermediate node acknowledges the data destined for a destination before the data actually reaches the actual destination. This speeds up the exponential rate of increase in slow-start as the ACKs are received within a short period rather than after traversing the actual path of the connection.

The ACKs from the actual receiver when received by the intermediate host are suppressed than being forwarded to the source. This avoids duplicate ACKs being received by the source of the connection.

In Satellite networks, the spoofing is done at the uplink device to the satellite. This is usually the satellite gateway. This reduces the Round Trip Time (RTT) of a TCP connection as the terrestrial networks have less propagation delay than satellite links. The satellite gateway forwards the packets up the satellite link and toward the destination. Actual ACKS that are received from the destination are suppressed. When packets are lost across the path to the destination, the satellite gateway retransmits them upon expiration of timers maintained on the gateway. Thus satellite gateways have abundant buffer space to store the packets from various connections.

TCP spoofing is transparent to the sender and receiver. The sender and receiver see an improvement in throughput as RTT is reduced.

Data transfer - TCP Spoofing - Mobile Host to Mobile Host via Satellite link

The topology is shown in Figure (3.5)

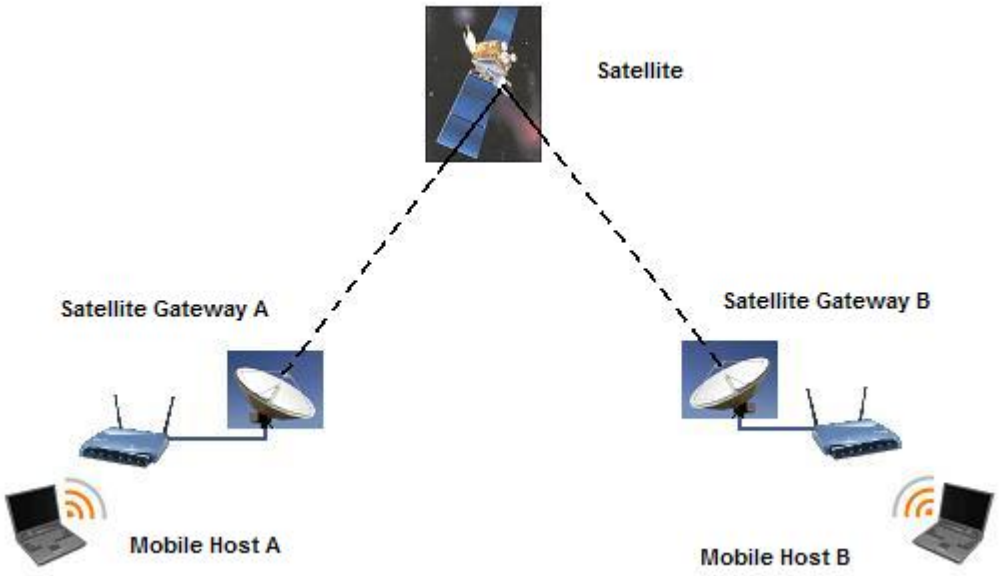

Figure 3.5: Satellite Network

- Open a TCP connection from MH A to MH B
- Send data from MH A to MH B. The satellite gateway acts as a splitting point for the connection.
- Segments arrive at satellite gateway.
- The satellite gateway ACKs the data and caches the segments
- The satellite gateway sends the data to MH B
- MH B sends ACKs to MH A
- The ACKs from MH B are suppressed by the gateway
- If packets are lost, the gateway retransmits them instead of the original source retransmitting them

Data transfer - TCP Spoofing - Mobile Host to Mobile Host via Ferry

The topology is shown in Figure (3.3).

- Open a TCP connection from Node 2 to Node 4
- Send data from Node 2 to Node 4
- Ferry within communication range
  - The ferry ACKs the data and caches the data
  - When the ferry leaves node 2, the connection times out and the connection is aborted
  - The connection of the ferry also times out as node 2 goes out of communication range
  - When the ferry reaches node 4, the segments are transmitted to node 4 if the connection is still live. If the connection is aborted, no data will be forwarded to node 4. Hence node 2 would have been acknowledged for certain segments and then the connection will be aborted
  - Again when the ferry moves out of range of node 4, the connection at node 4 times out
- Ferry out of communication range
  - The connection times out as no ACKs are received

Hence certain data will be acknowledged but the connections will be aborted later. In the scenario where the ferry is out of communication range, the connections are aborted because the ferry cannot receive those segments.

Chapter 4

System Design

4.1 Conceptual Design

In order to ensure that TCP works for message ferry networks, the following changes are made to TCP. Before getting into the details of the changes, a summary of the changes is given below:

- TCP at the nodes store the application data received when the ferry is not within communication range
- TCP starts the timers only when the ferry is within communication both on the ferry and the node
- When the ferry is within communication range, it sends a window update for every connection on the ferry. This opens up the window on the nodes which is closed when the ferry goes out of communication range
- The ferry estimates the amount of data that will be sent for each connection as it internally maintains the list of connections to a particular node
- The nodes also estimate the amount of data that will be sent to the ferry as it stores all the data that need to be sent to the ferry

- At the node if there is a new request from an application, the node satisfies the request only if there is sufficient bandwidth to fulfill the request else the node stores this request and fulfills the request during the next trip of the ferry
- When the ferry leaves the node, it sends a zero window update for every connection on the node so that the TCP on the node enters persist state

Additional details that are required to understand the establishment of a connection, data transfer and termination of a connection are explained below

4.1.1 Window Size Estimation - Node

The node stores the data received from the application in its internal buffer specific for every TCP connection until the ferry is within communication range. The duration when the ferry is in communication range with the node represents the communication window for the node. When the ferry is within communication range, the TCP variant on the node, estimates the amount of data that can be sent for each connection. This estimation is based on the number of connections on the node and the window size advertised by the ferry. This is shown is Figure (4.1). The window size at the node for every connection $Connection_{window\ size}$ is given by Eqn. (4.1).

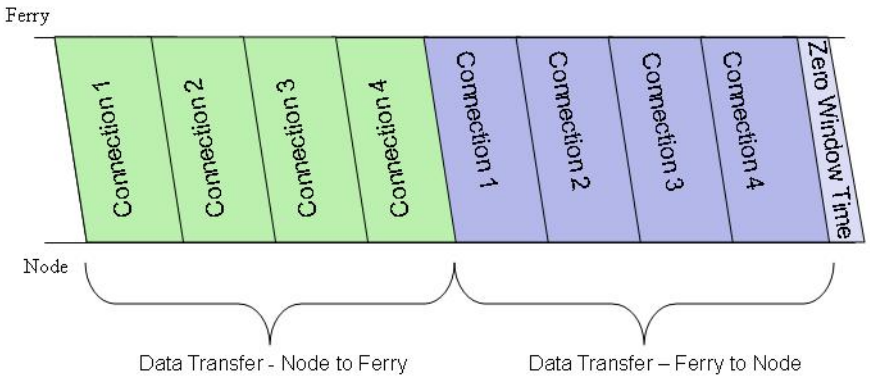

Figure 4.1: Timing Diagram - Whole Bandwidth occupied by connections

$$Connection_{windowsize} = \frac{Time\ of\ contact\ with\ the\ ferry * Wireless\ Bandwidth}{Number\ of\ connections\ +\ 1} \tag{4.1}$$

From the window size per connection, the amount of data that can be sent to the ferry $Data_{ferry}$ is calculated as given below

$$Data_{ferry} = MIN(Connection_{window\ size}, Ferry_{window\ Size}) \tag{4.2}$$

The number of connections is incremented by 1 to allow any new connection that is requested after the estimation. Thus the maximum amount of data that can be transferred between a node and the ferry is limited by the number of connections from the node and the ferry's advertised window size. However after the estimation if a new connection request arrives, it is satisfied only if there is sufficient bandwidth. When the amount of data transferred by every connection is less, most of the window is wasted. This excess window bandwidth is allocated to new requests. This is shown is Figure (4.2). However if the data transferred from the node is significantly large, only one extra connection will be allowed.

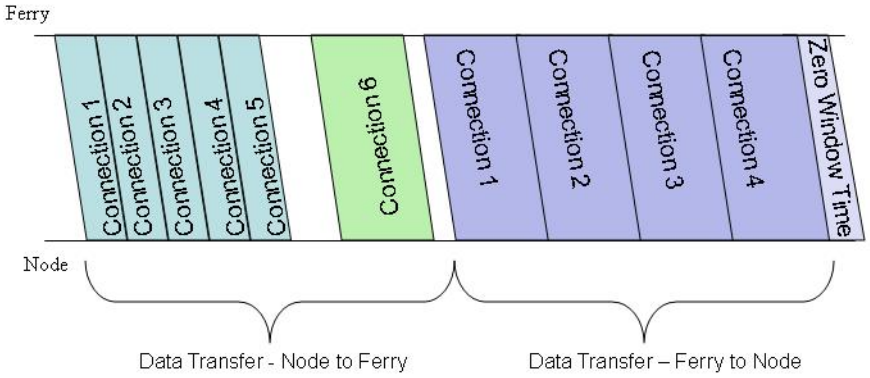

Figure 4.2: Timing Diagram - Free Bandwidth occupied by a new connection

Once the window size has been estimated, the bandwidth for acknowledgments has to be allocated. This is done to ensure that the acknowledgments are received by the node. Hence the window size is decremented by the maximum expected number of acknowledgments. As the ferry acknowledges every segment with an ACK, the number of ACKs is equal to the number of segments sent by the node. The equations below evaluates the final value for the window size

$$ACK_{count} = \frac{Data_{ferry}}{MTU\ Size} \tag{4.3}$$

$$Data_{ferry} = Data_{ferry} - ACK_{count} * ACK\ Size \tag{4.4}$$

4.1.2 Window Size Estimation - Ferry

Ferry on arrival at a node determines the number of connections destined for that node. The ferry also calculates the window size allowed for every connection. A similar set of equations as in section "Sender Side" are used to estimate the window size for every connection before sending any data to the node, the ferry sends a window update based on this calculation to the node for every connection. This opens up the window on the node for every connection

The equivalent equations are given below

$$Connection_{windowsize} = \frac{Time\ of\ contact\ with\ the\ node * Wireless\ Bandwidth}{Number\ of\ connections\ +\ 1} \tag{4.5}$$

From the window size per connection, the amount of data that can be sent to the ferry $Data_{node}$ is calculated as given below

$$Data_{node} = MIN(Connection_{window\ size}, Node_{window\ Size}) \tag{4.6}$$

As data may already exist in the ferry's buffer that has to received from the node on the previous trip of the ferry, this amount is deduced from the calculation of available window size on the ferry

$$Data_{node} = Data_{node} - Amount\ of\ data\ in\ ferry's\ buffer \tag{4.7}$$

$$ACK_{count} = \frac{Data_{node}}{MTU\ Size} \tag{4.8}$$

$$Data_{node} = Data_{node} - ACK_{count} * ACK\ Size \tag{4.9}$$

4.1.3 TCP State Transition Diagram for Nodes

The state transition diagram at the node is not changed. It is the same as the standard TCP mentioned in RFC 793 ([20]).

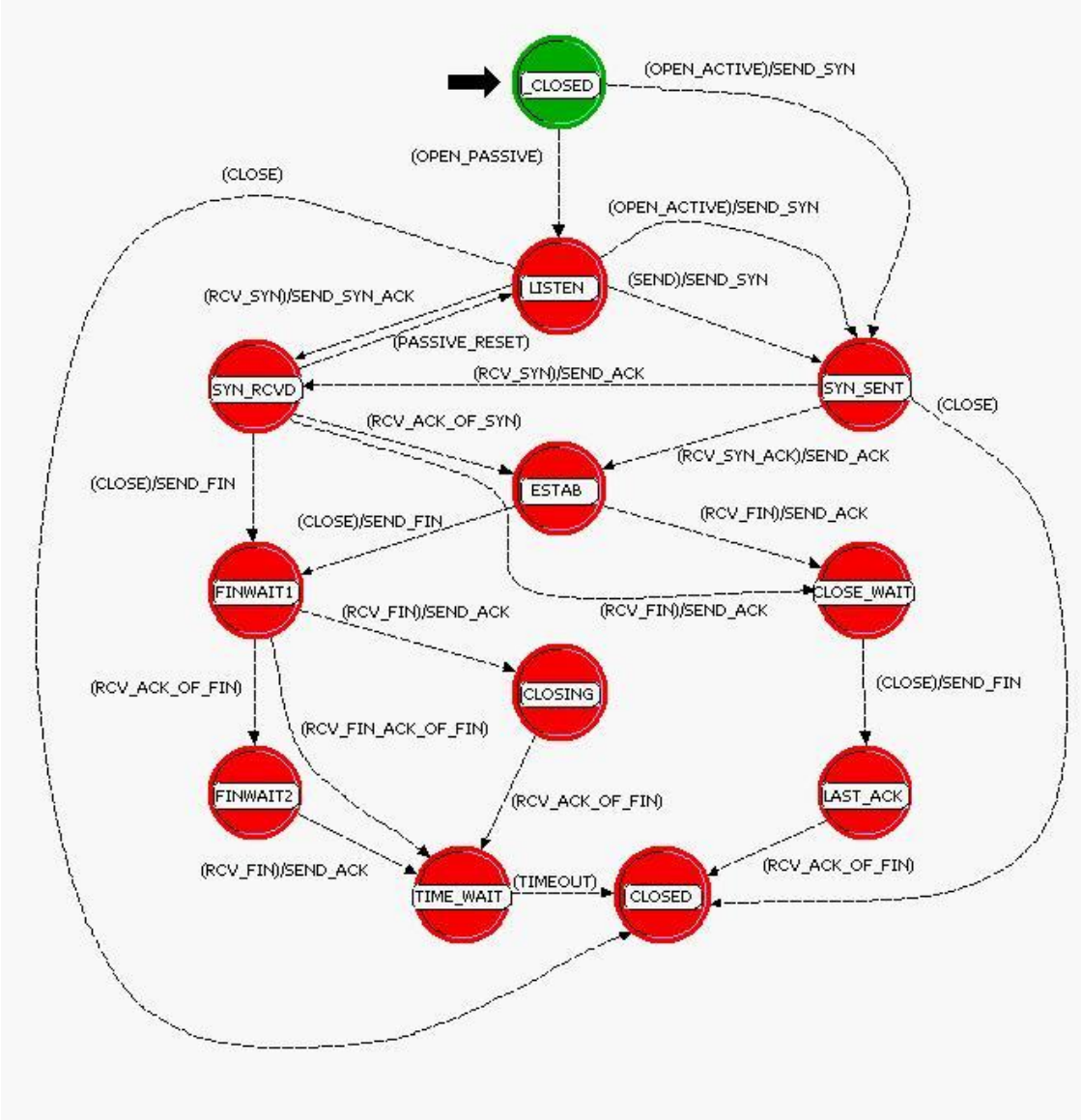

Figure 4.3: Node - TCP State Diagram

4.1.4 TCP State Transition Diagram for Ferry

The state transition diagram at the ferry is shown in Figure (4.4). When a node initiates a new connection with the ferry, the connection on the ferry progresses through a set of states. The various states are : INIT, ESTABLISHED-1, ESTABLISHED-2, ESTABLISHED, FIN-1, FIN-ACK, FIN-2, CLOSING and CLOSED. CLOSED state is imaginary as the connection is either not established or terminated.

INIT : When a SYN is received, the connection enters this state. The connection in this state initiates the internal buffers and sends a SYN ACK for the SYN being received. Once the SYN ACK has been sent, the connection moves to ESTABLISHED-1 state.

ESTABLISHED-1 : In this state, the connection sends ACKs for segments received from the node and stores the segments in the internal buffer that has been created for every connection in INIT state. If the connection receives a FIN segment from the node, the connection moves to ESTABLISHED-2 state. The ESTABLISHED state is not entered as the SYN from the other end of the connection is not yet received. The other end SYN will be received only when the ferry reaches the destination.

ESTABLISHED-2 : The connection stays in this state until it receives the SYN from the other end of the connection.

ESTABLISHED : The connection in this state performs normal data transfer as in standard TCP. The connection ACKs segments received from either end of the connection and stores only the data segments in it's internal buffer.

FIN-1 : The connection enters this state when a FIN segment has been received from one end of the connection. The connection ACKs this segment and stores the FIN segment in the internal buffer.

FIN-ACK : The connection enters this state when the actual ACK for the FIN received from the other end is received. This helps the connection to keep track of the actual ACKs received.

FIN-2 : The connection enters this state when the second FIN is received. This FIN has to be from the other end of the connection.

CLOSING : The connection enters this state when the ACK for the second FIN is received.

CLOSED: The connection enters this state when the four way connection termination is complete. This state represents the removal of the connection or the absence of a

connection.

4.1.5 Session Initiation

The session is considered between node 2 and node 4 in Figure (3.3) where node 2 initiates the connection.

Session initiation at a node is similar to standard TCP except that the ACKs in the three-way initiation are pseudo ACKs. When the ferry is within communication range, node 2 initiates a connection by sending a SYN segment to node 4 and initiates a retransmission timer. The connection at node 2 moves to SYN-SENT state. The ferry intercepts the SYN segment and responds with a SYN ACK segment with MTU and window size. The ferry stores the SYN in its internal buffer and starts a retransmission timer for the SYN ACK and progresses the connection to ESTABLISHED-1 state. When node 2 receives the SYN ACK segment from the ferry, an ACK is sent and the retransmission timer is stopped. The node's connection moves to ESTAB state. The session termination is diagrammatically represented in Figure (4.5) and Figure (4.6).

If any of the segments are lost, the retransmission timer expires and the segments are retransmitted utmost 3 times. The connection is aborted when the limit exceeds by transmitting a RESET segment. However the wireless medium is considered to be loss free as no two nodes are within communication range with each other. Thus loss occurs mainly due to buffer overflow in the lower layers at either the node or the ferry.

4.1.6 Data Transfer

Data transfer between the node and the ferry is significantly different from standard TCP. When the node's connection is in ESTAB state the node sends segments to the ferry according to the window size estimated by equation(4.4) instead of initiating slow start phase as in standard TCP. If the amount of data available at the node is less than the window size, the window remains open for any new data transfer. But if the node's data exceeds the window size, the node transfers data only on the next trip of the ferry.

However the ferry stays in communication with the node such that all data gets transferred. Hence the amount of data transferred from the node empties the send buffer at the node. This assumption is based on the fact that the ferry estimates the route path

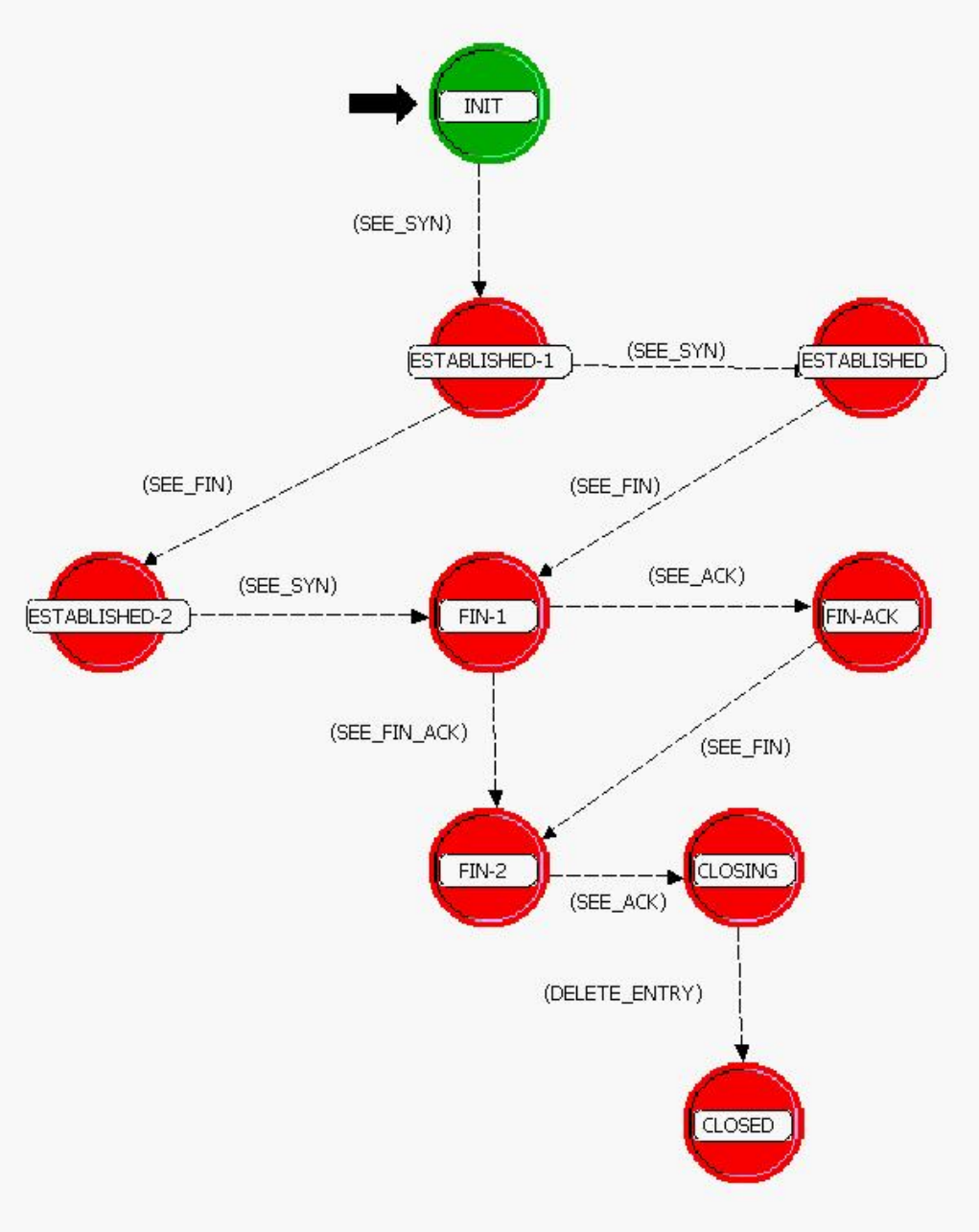

Figure 4.4: Ferry - TCP State Diagram

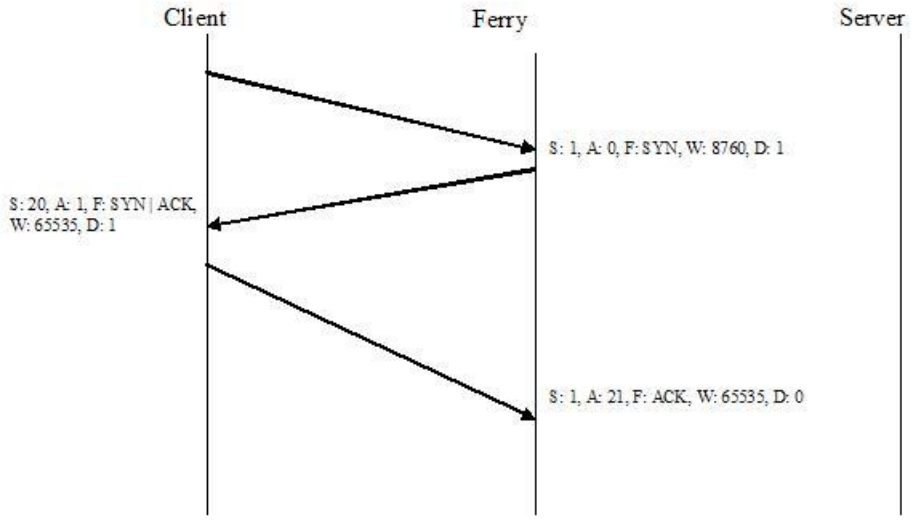

Figure 4.5: Session Initiation - Between Client and Ferry

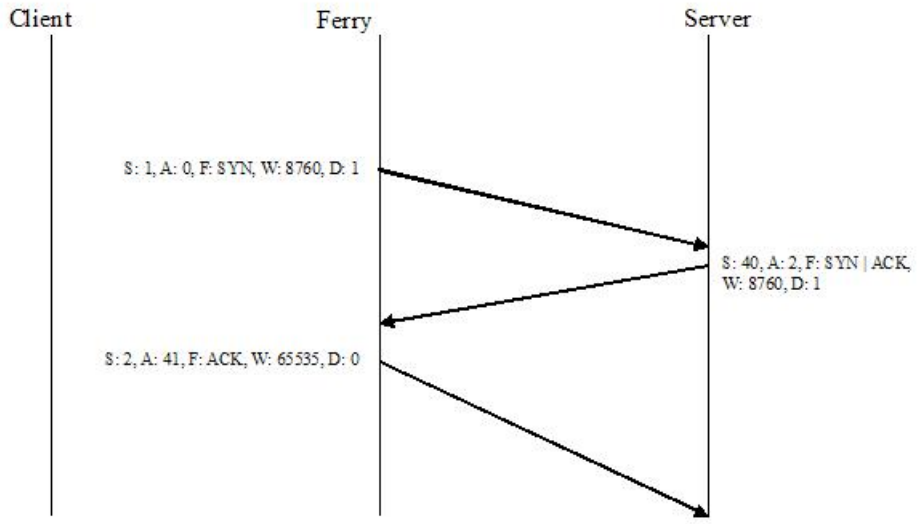

Figure 4.6: Session Initiation - Between Ferry and Server

based on the amount of data available at each and every node.

The data transfer between the node and the ferry is diagrammatically represented in Figure (4.7) and Figure (4.8)

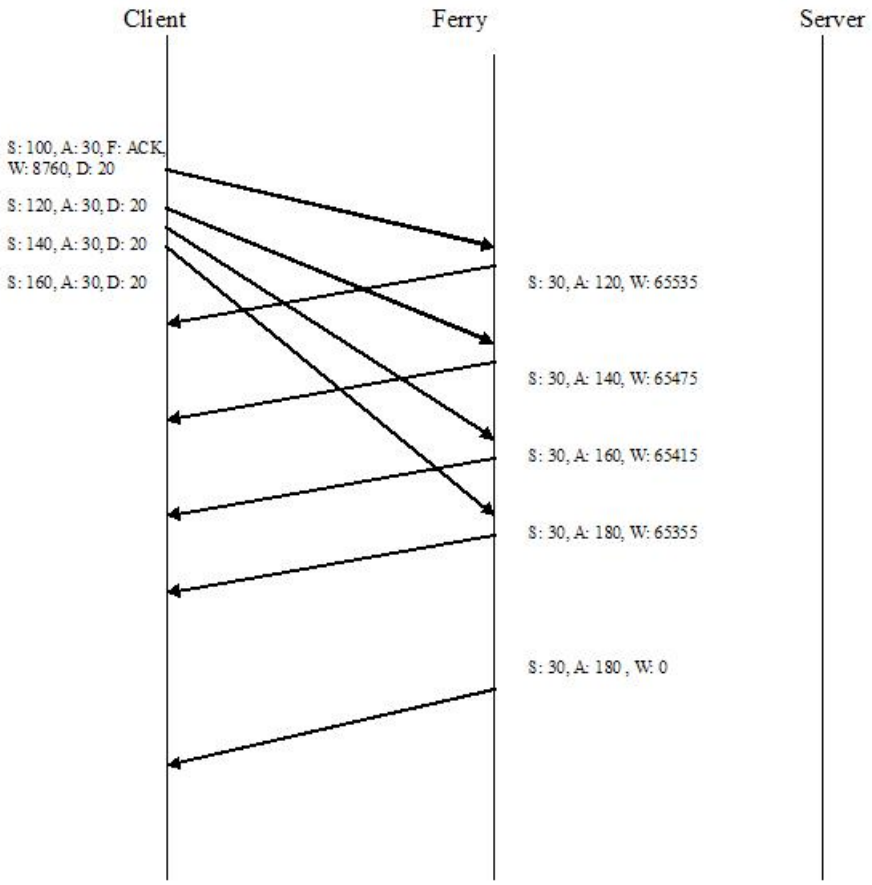

Figure 4.7: Data Transfer - Between Client and Ferry

Node 2 transfers data according to its permitted window size to the ferry and starts a retransmission timer. The ferry ACKs each and every segment. Thus the number of ACKs received is equal to the number of segments transmitted. The ferry stores the segments in it's internal buffer to be later transferred to node 4. However if there is any segment loss, the retransmission timer times out and the segment is retransmitted only if the window is open. If the window closes, the node enters persist state and no retransmissions

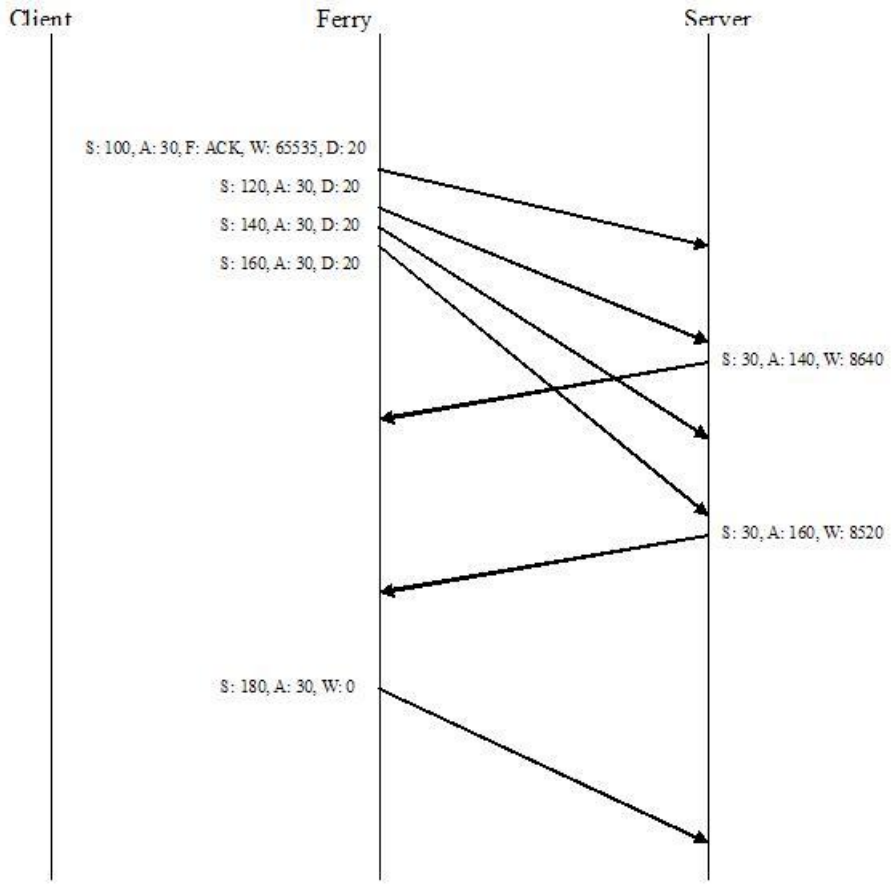

Figure 4.8: Data Transfer - Between Ferry and Server

occur. However the node does not send zero window probes as the bandwidth allocated for this node has been consumed by this node. Before the ferry goes out of communication range of node 2, the ferry transmits a zero window update for the connection. The zero window update is transmitted for each and every connection on the node. The zero window is sent significantly ahead to avoid out of communication range losses. However even if the zero window updates are lost, the node does not send any more data beyond the permitted limit as the node will close down its window size when the transmission duration elapses.

When the ferry reaches node 4, the ferry transmits segments according to the window size of node 4 and starts a retransmission timer. Hence when node 4 updates it's window size, the ferry transmits further segments until the transmission duration elapses. The ACKs received by the node are not stored in the ferry as the ferry locally ACKs the segment rather than end to end.

If there are any segment losses, the retransmission timer times out and the segments are resent based on the window size and the remaining bandwidth allocated to the connection. Thus the bandwidth available between the ferry and the node are utilized to the maximum extent possible. Segment loss occur due to buffer overflow at the various layers rather than loss due to wireless medium or collision due to simultaneous transmission by the nodes.

4.1.7 Session Termination

Session termination at a node is similar to standard TCP except that the ACKs in the four-way termination are pseudo ACKs. When node 2 sends a FIN segment for the connection, the Ferry intercepts the FIN segment and responds with a pseudo ACK on behalf of the destination and stores the FIN segment. The ferry's connection moves to FIN-1. The reception of the ACK at the node moves the connection to FIN-WAIT2.

When the ferry reaches node 4, the ferry transmits the FIN and starts a retransmission timer for the FIN segment. On reception of the FIN, the node's connection changes to CLOSE-WAIT state. When the ferry receives the actual ACK from the node, the connection's state changes to FIN-ACK. When data transfer from node 4 is done, node 4 transmits a FIN segment and changes to LAST-ACK state and starts a retransmission timer for the FIN. When the FIN reaches the Ferry, the ferry's connection responds with an ACK and changes state to CLOSING. Node 4 moves to CLOSED state on reception of the ACK and

the connection is cleared.

The session termination is diagrammatically represented in Figure (4.9) and Figure (4.10).

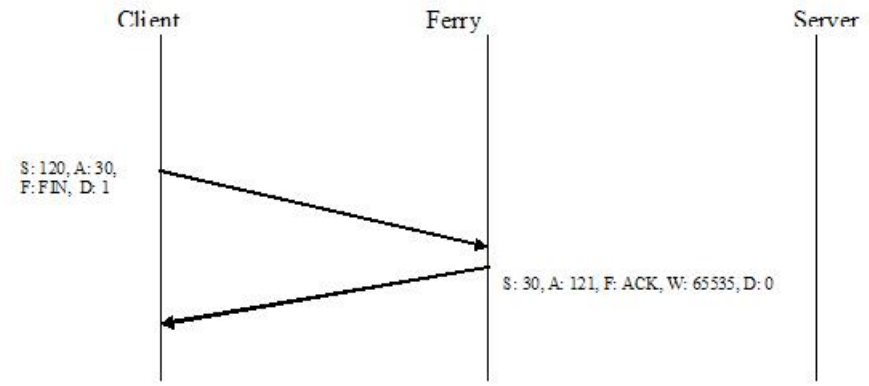

Figure 4.9: Session Termination - Between Client and Ferry

If any of the segments are lost, the segments are retransmitted thrice and if the limit is exceeded, the connection is aborted by sending a RESET segment. However if the transmission duration ends before the ACK is received, the node's connection enters persist state on the reception of a zero window update from the ferry. When the ferry revisits node 4, the window opens up and the connection resumes.

4.2 Opnet Design

Opnet is a Discrete Event Simulator (DES) used to model communication networks. In order to understand the simulation of TCP variant for Message Ferry networks, a brief introduction about Opnet is provided here.

The various layers of the TCP/IP protocol stack are implemented as individual modules. Each module has a set of processes associated with it. The modules communicate among themselves using streams. Streams provide a communication link between every layer. Thus TCP layer has two different streams - one for the IP layer below and the other for the Application layer above.

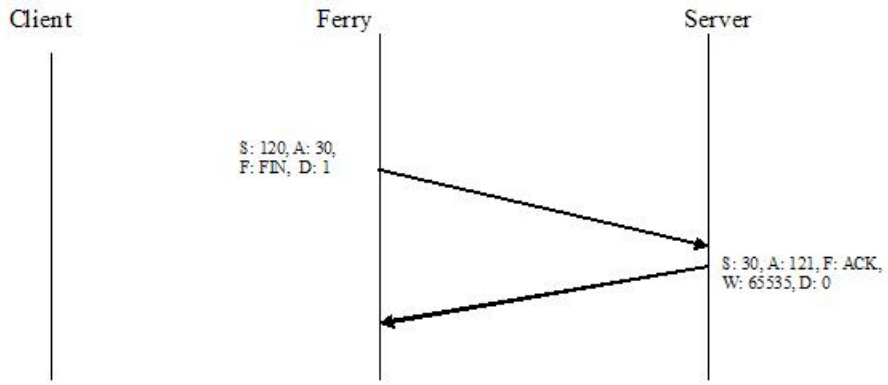

Figure 4.10: Session Termination - Between Ferry and Server

The arrival of a segment from the IP layer is notified by a stream interrupt to TCP layer. The interrupt has associated with it an Interface Control Information (ICI) which is used to transfer information across processes. The IP module creates the ICI with information like the source address of the segment along with the segment. The ICI is destroyed by the receiving process - TCP - once it has accessed the required information. The TCP layer pushes the segment down to IP layer using stream interrupts also associated with an ICI.

The application layer communicates with TCP layer using commands and indications. The commands used by applications are mentioned in the following section. The indications sent by TCP to the application layer are in the form of stream interrupts with an associated ICI.

The main processes associated with the TCP layer - TCP module - are the TCP Connection Manager process and the TCP Connection process.

4.2.1 TCP Connection Manager Process

The TCP manager is responsible for the following activities

- Process data from various TCP related applications
- Forward data received from IP layer to corresponding applications

- Initiate and maintain various TCP connections

Process data from various TCP related applications

The application sends data to the TCP manager to be forwarded to the required destination. The various commands that the application uses to either initiate or transfer or close a connection are the following

- OPEN - To open a new TCP connection to corresponding application on another host.
- SEND - To send data or messages to corresponding application on another host.
- RECEIVE - To request for data or messages to be forwarded to the application.
- CLOSE - To close a connection when all data has been transferred to TCP layer.
- ABORT - To prematurely close a connection under exceptional conditions.

Forward data received from IP layer to corresponding applications

The segments from IP layer are received at the TCP layer by the TCP connection manager. The specific command that TCP connection manager uses is SEG_RCV. Once a segment is received from IP layer the segment is sent to the corresponding TCP connection by using the source port, destination port, source IP address and destination IP address. If no connection exists and the segment is not a SYN segment the segment is discarded and a Reset segment is sent to the source of the segment.

Initiate and maintain various TCP connections

The various TCP connections at the node are maintained as separate TCP connection processes. Each TCP connection has a separate Transmission Control Block (TCB). This structure has information regarding the connection identifier, application identifier, state of the connection, type of service, local port, destination port, local address , remote address, associated TCP connection process, and certain statistical information.

Whenever either the application requests a new connection or a SYN segment is received, a new TCB entry is created and added to the list of entries maintained by the

TCP manager. When the connection is either aborted or terminated, the entry is removed from the list.

4.2.2 TCP Connection process

The TCP connection process is specific to every TCP connection created on the node either due to a request from the local application or the remote application. The TCP connection process is represented as the TCP state diagram. Every TCP connection has connection specific variables called state variables. These are similar to thread specific variables. The commands received from either the application or IP layer are processed in this process by specific events. The various events are described in the next section.

4.2.3 Events

The events that handle the various commands from the application or the interrupts from the IP layer can be classified into three groups and are mentioned below :

- TCPC_EV_OPEN_ACTIVE
- TCPC_EV_OPEN_PASSIVE
- TCPC_EV_SEND
- TCPC_EV_RECEIVE
- TCPC_EV_CLOSE
- TCPC_EV_ABORT
- TCPC_EV_SEG_ARRIVAL
- TCPC_EV_RCV_SYN
- TCPC_EV_RCV_SYN_ACK
- TCPC_EV_RCV_ACK_OF_SYN
- TCPC_EV_RCV_FIN
- TCPC_EV_RCV_ACK_OF_FIN

- TCPC_EV_RCV_FIN_ACK_OF_FIN
- TCPC_EV_PASSIVE_RESET
- TCPC_EV_ABORT_NO_RST
- TCPC_EV_SEND_ACK
- TCPC_EV_SEND_ZERO_WINDOW
- TCPC_EV_SEND_DATA

Application Specific Events

The following are used to handle application specific commands

- TCPC_EV_OPEN_ACTIVE
- TCPC_EV_OPEN_PASSIVE
- TCPC_EV_SEND
- TCPC_EV_RECEIVE
- TCPC_EV_CLOSE
- TCPC_EV_ABORT

TCP Connection Process Events

The following events are used to handle events specific to arrival of various segments.

- TCPC_EV_RCV_SYN
- TCPC_EV_RCV_SYN_ACK
- TCPC_EV_RCV_ACK_OF_SYN
- TCPC_EV_RCV_FIN
- TCPC_EV_RCV_ACK_OF_FIN

- TCPC_EV_RCV_FIN_ACK_OF_FIN
- TCPC_EV_PASSIVE_RESET
- TCPC_EV_ABORT_NO_RST

Transmission and Reception of specific segments

The following events are used to either send segments to the network or receive segments from the network.

- TCPC_EV_SEG_ARRIVAL
- TCPC_EV_SEND_ACK
- TCPC_EV_SEND_ZERO_WINDOW
- TCPC_EV_SEND_DATA

4.2.4 Design of Message Ferry TCP

In order to explain the design of Message Ferry TCP in Opnet the following topology is used.

Simulation Topology

The topology consists of 7 nodes - 4 of which take up the role of clients, 2 of them act as servers and a special node called the ferry. All the nodes are wireless enabled. The client and server nodes are static while the ferry is a mobile node with a wireless interface. The path taken by the ferry is considered a straight line and is represented by the solid line connecting each and every node. The ferry starts at node 1 and comes back to node 1 after visiting node 2, 3, 4, 5 and 6 in the sequence mentioned.

The various applications configured on the nodes are defined using the "Application Definition" icon named Application and the set of applications on each node is defined using the "Profile Definition" icon named Profile. The topology shown above has been configured for FTP.

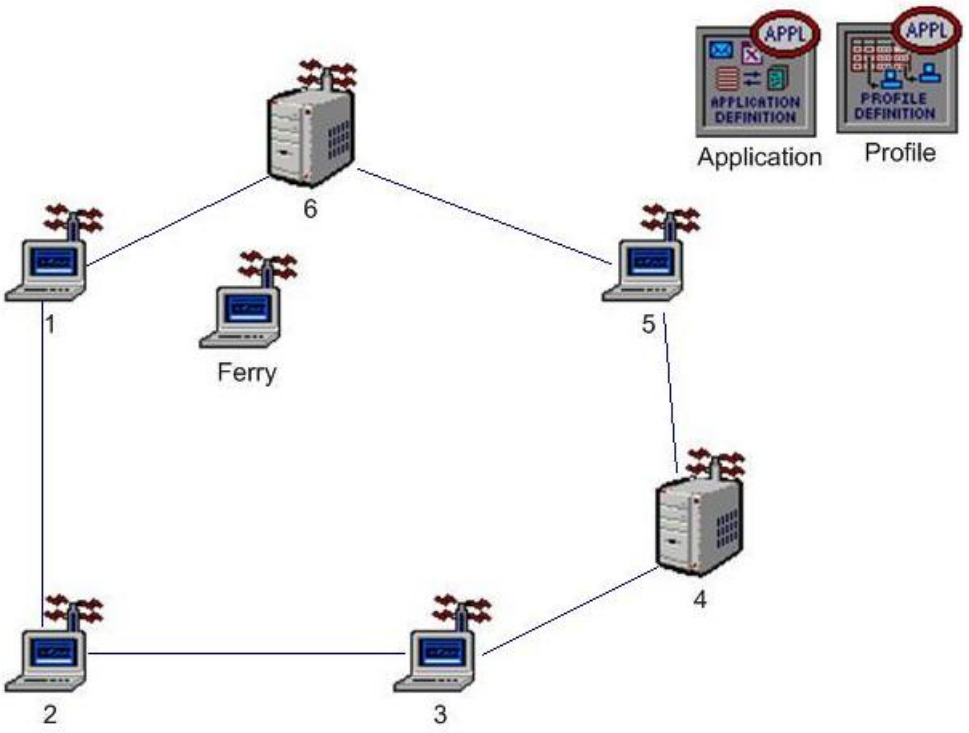

Figure 4.11: Message Ferry Network Topology

Timers

In order to enable the node or ferry to send data to the ferry or node, respectively, only when the ferry is communication range with the nodes, timers are used to implement both at the nodes and the ferry for the arrival and departure of the ferry. Thus the transit time - represented by the transit timer - of the ferry between any two nodes and the duration of time the ferry maintains wireless communication with the node - difference between the ferry departure and ferry arrival at a node - can be modified using these timers. The duration of the difference between ferry arrival and ferry departure timer estimates the communication window size. In the simulation, the communication window is equally shared by the ferry and the node. Thus when the ferry arrives at a node, the ferry transfers the data destined to the node followed by the node. These timers are configured in the TCP connection manager component as this component has control of the various TCP connections both at the nodes and the ferry.

Ferry

The ferry on arrival near a node estimates the amount of data for every connection and updates this information to the node as window updates. Thus the ferry splits its part of the communication window across all connections irrespective of the amount of data available for every connection. The ferry transfers data to the node either until the node window size becomes zero or its communication window timer expires. However if the node's window closes, it continues to transfer data once the window opens up until the window again closes or the allocated bandwidth for the connection expires or the communication window for the ferry expires.

The ferry receives the segments from the node and inserts them into a list ordered by the sequence number of the segments received which is maintained specific to each connection for every node.

Node

To establish an end to end connectivity with a remote host, an application sends the request to the TCP layer. The TCP connection manager receives the request and buffers the various commands of the request as a list of commands specific to each connection.

When the ferry is within communication range and the duration of data transfer from node to ferry has commenced, which is the second half of the communication window, the TCP connection manager processes each and every command present in the list and creates the necessary events to handle the various commands. The OPEN command from the application creates the TCPC_EV_OPEN_ACTIVE event which generates a SYN segment to be sent to the other end. The sequence number of the segment is calculated based on the amount of time elapsed in the simulation both at the ferry and the node. SEND and RECEIVE commands generate TCPC_EV_SEND and TCPC_EV_RECEIVE events. These events forward the data to the TCP connection process and requests for arrived data from the connection process respectively. The CLOSE command creates the TCPC_EV_CLOSE event which generates a FIN segment to close the TCP connection in the TCP connection process. The ABORT command creates the TCPC_EV_ABORT to send a RST segment to the other end.

The connection manager primarily processes all the queued requests and then satisfies any new requests based on the remaining bandwidth available for this node.

The data sent by the manager to the TCP process is buffered by the TCP connection process. The TCP connection process divides the data across segment boundaries and forwards it to the IP layer based on the amount of data that can be forwarded to the ferry. This is calculated from the amount of time allocated for this connection and the window size received from the ferry. Thus the window size with respect to the ferry denotes the amount of data that can be transferred to the ferry as the ferry is not constrained by buffer size because it has unlimited buffer. However the window size with respect to the node denotes the size of receive buffer at the node. The TCP connection process transmits any available data until it runs out of ferry window size or the departure of the ferry which ever occurs first.

Chapter 5

Numerical Results

This chapter presents the various results obtained from the Opnet simulation.

5.1 Time line diagrams of MF TCP connections

5.1.1 Normal Data Transfer

The time line diagram of the simulation output of a file transfer using Message Ferry TCP is shown below.

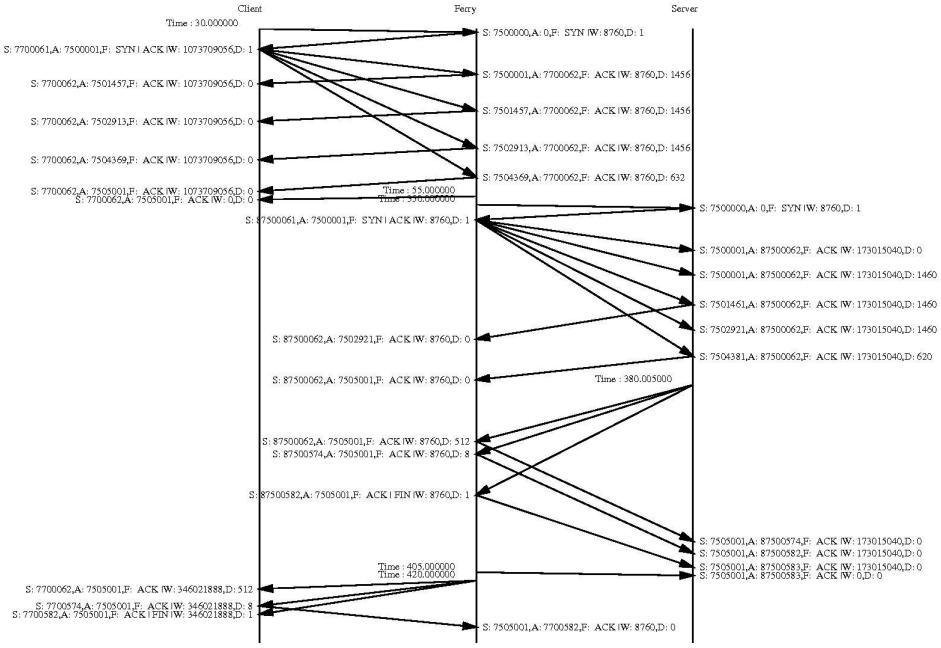

Figure 5.1: Normal Data Transfer - Connection initiation and data transfer

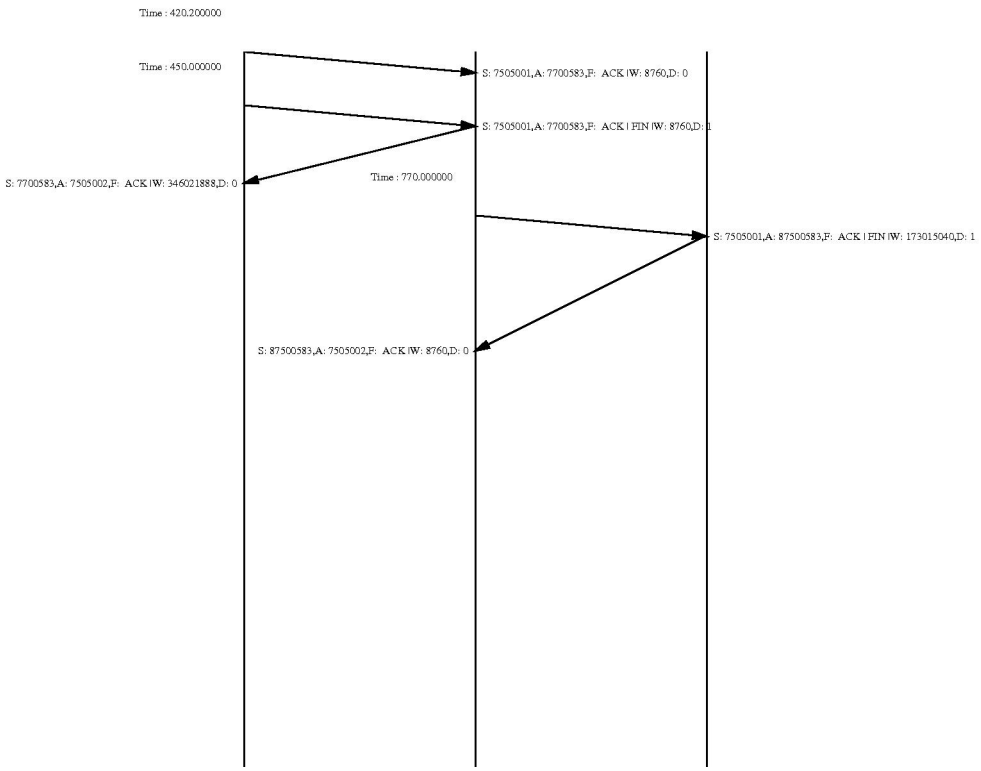

Figure 5.2: Normal Data Transfer - Connection termination

5.1.2 SYN segment loss

The time line diagram of the simulation output when a SYN segment is lost is shown below.

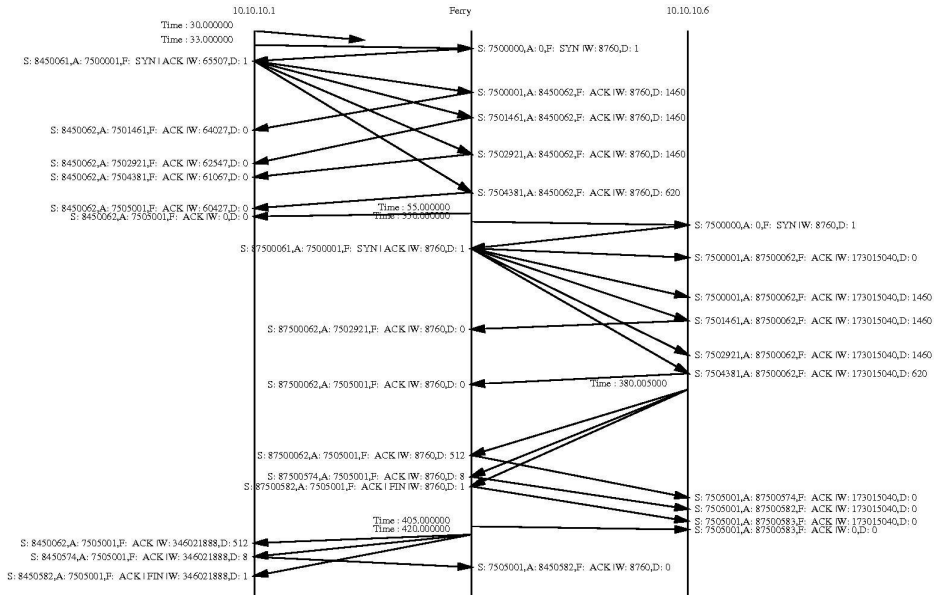

Figure 5.3: SYN Loss - Connection initiation and data transfer

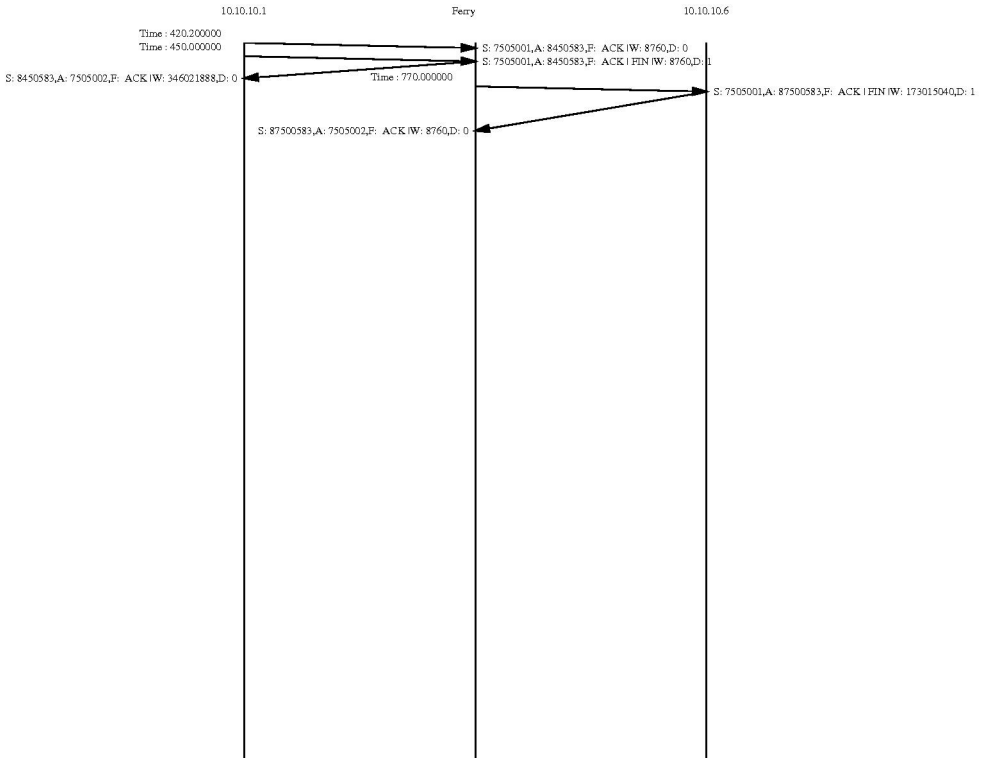

Figure 5.4: SYN Loss - Connection termination

5.1.3 Data segment loss

The time line diagram of the simulation output when a DATA segment is lost is shown below.

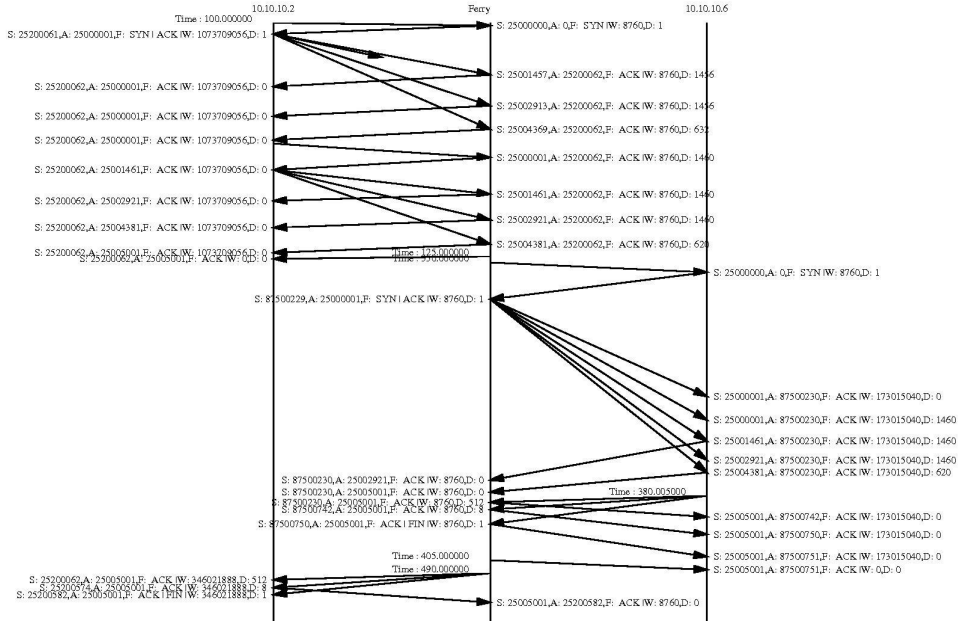

Figure 5.5: Data Loss - Connection initiation and data transfer

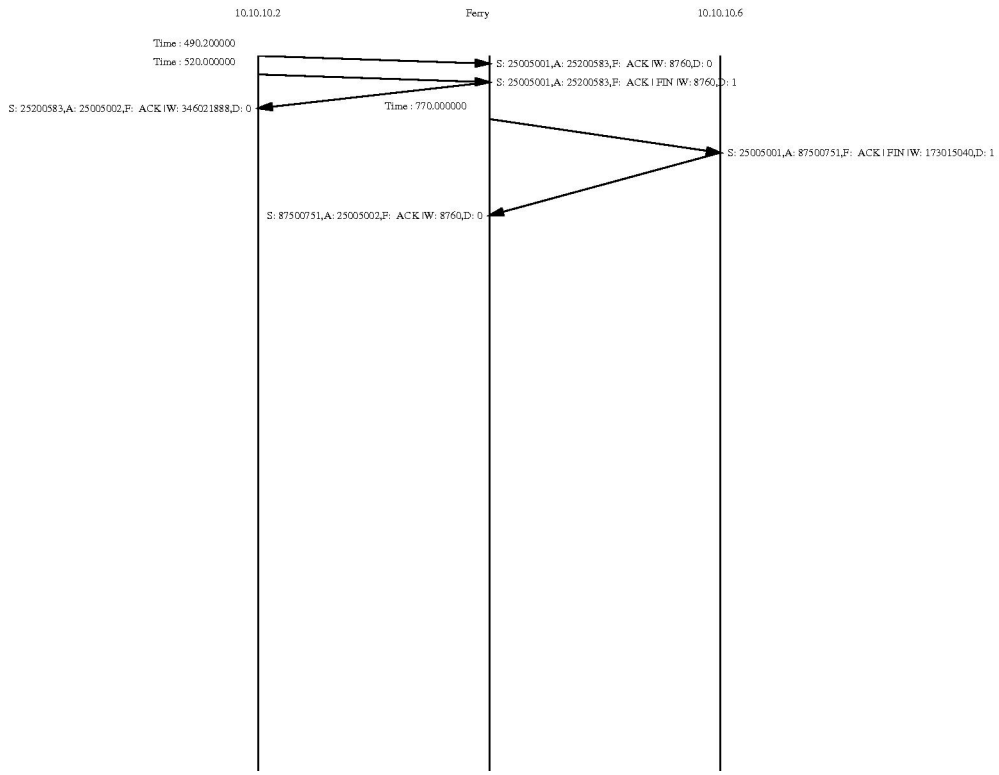

Figure 5.6: Data Loss - Connection termination

5.1.4 FIN segment loss

The time line diagram of the simulation output when a FIN segment is lost is shown below.

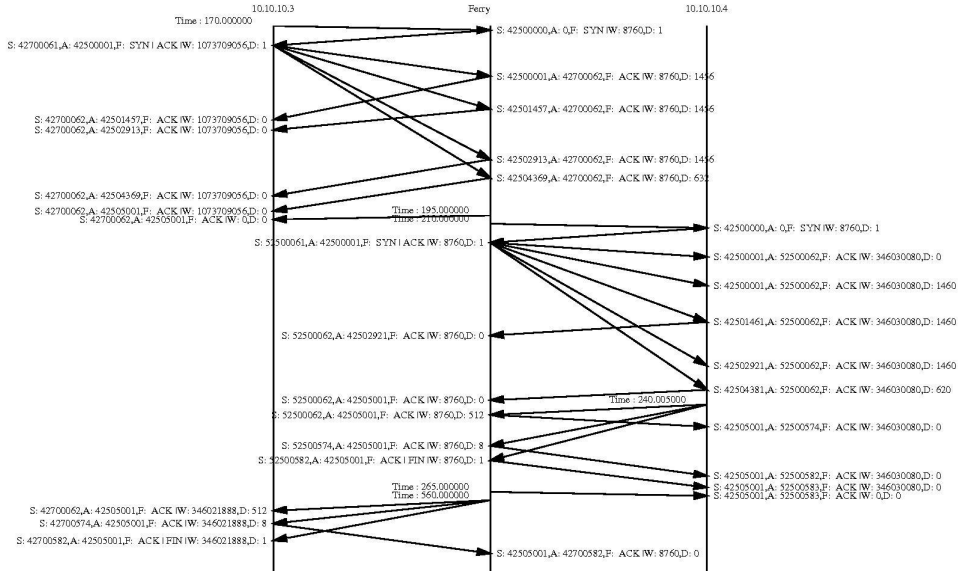

Figure 5.7: FIN Loss - Connection initiation and data transfer

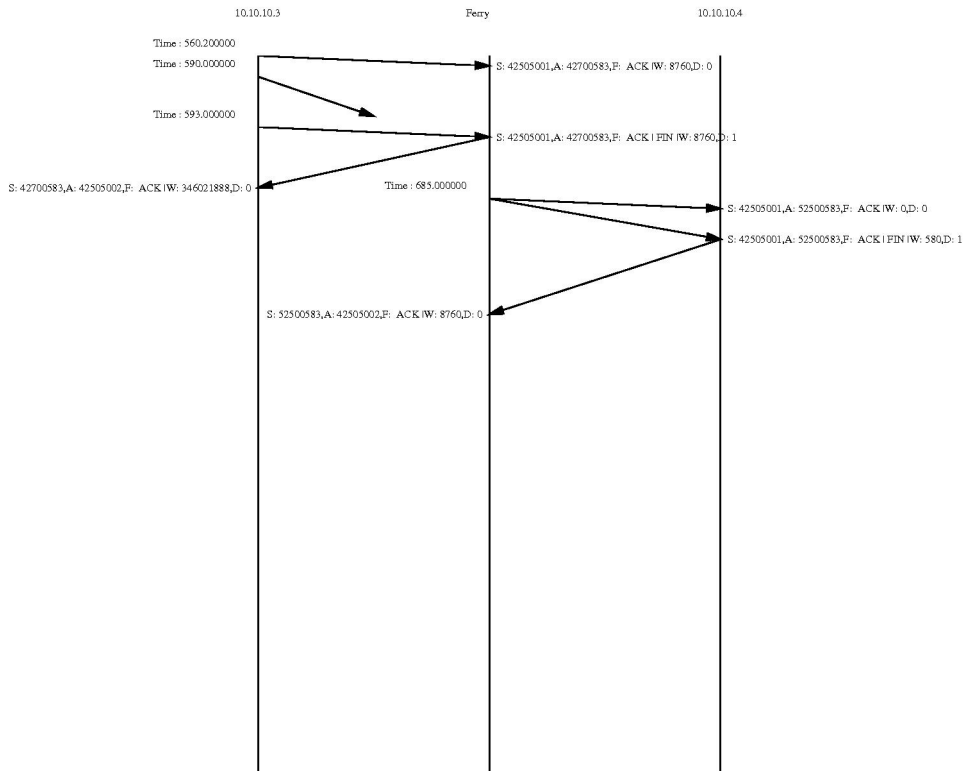

Figure 5.8: FIN Loss - Connection termination

5.2 Comparison of Standard TCP and Message Ferry TCP

We ran simulation for transferring a 10K file from a server to a client. The total time taken for the whole file transfer from the connection initiation using SYN segments till the ACKing of FIN segments increased exponentially as the total route trip of the ferry was increased from 9 sec to 9000 sec for standard TCP. The Message Ferry TCP utilized the available window size advertised by the ferry at the server to transfer all the segments within just one round trip. The inclusion of the acknowledgment of the final FIN actually made up for two total trips. The Figure (5.9) illustrates the exponential increase by standard TCP.

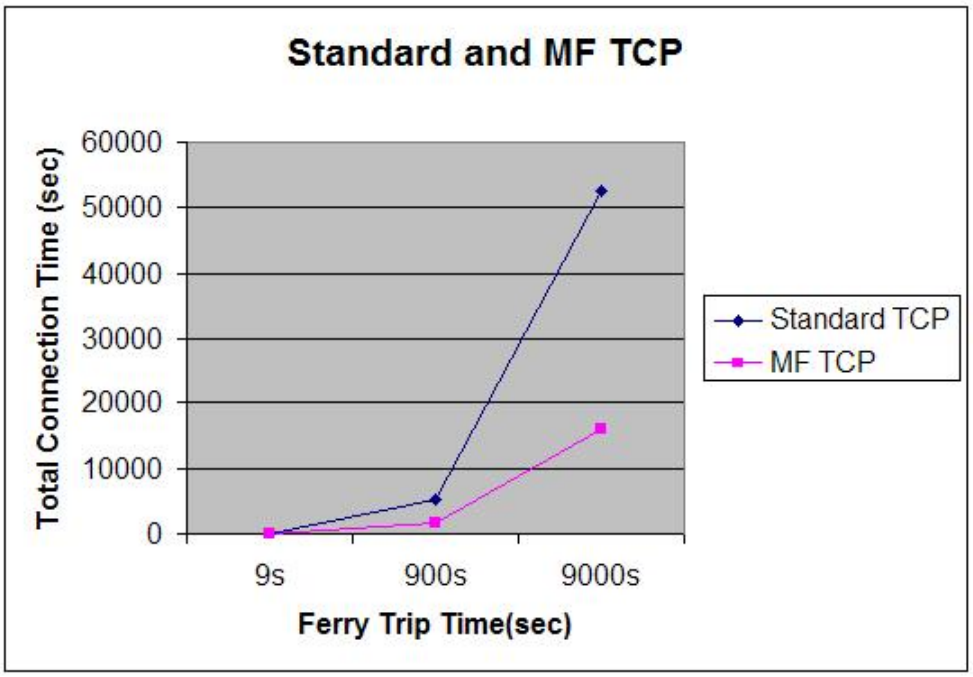

Figure 5.9: Comparison of Message Ferry TCP and Standard TCP

Chapter 6

Conclusion and Future Work

We analyzed the feasibility of various currently available transport protocols on Message Ferry networks. However we could not exactly fit any one protocol that satisfies this requirement. We have developed our own protocol in order to meet the requirements of high throughput and maximum bandwidth utilization in a highly disconnected network. Our simulation results show a huge improvement in the throughput and bandwidth utilization.

We further want to analyze out protocol with various constraints like huge losses in the wireless medium, multiple ferries in the same route, presence of multiple ferry routes and buffer constraints at the ferry. Buffer management at the ferries is one major area that we are interested in as this leads to selecting the right buffer management technique suitable in order to reduce the impact on Message Ferry TCP.

Bibliography

[1] W. Zhao, M. Ammar, and E. Zegura. "Proactive Routing in Highly partitioned Wireless and Ad hoc Networks", in proc. 9th IEEE Workshop on Future Trends in Distributed Computing Systems (FTDCS). May, 2003.

[2] W. Zhao, M. Ammar, and E. Zegura. "A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks", in proc. The 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc2004). May, 2004.

[3] W. Zhao, M. Ammar, and E. Zegura. "Controlling the Mobility of Multiple Data Transport Ferries in a Delay-Tolerant Network", in proc. IEEE Infocom 2005. Mar, 2005.

[4] 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

[5] A. Bakre and B. R. Badrinath. "I-TCP: Indirect TCP for Mobile Hosts", 15th International Conference on Distributed Computing Systems, 1994.

[6] K. Brown and S. Singh. "M-TCP: TCP for Mobile Cellular Networks", ACM SIGCOMM Computer Communication Review, 27(5):1943, 1997.

[7] R. Yavatkar and N. Bhagawat. "Improving End-to-End Performance of TCP over Mobile Internetworks", IEEE 1994 Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, (1994).

[8] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. "WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks", in Proceedings of the Fifth An-

nual ACM/IEEE International Conference on Mobile Computing and Networking, pages 231241, ACM Press, 1999.

[9] A. K. Singh and S. Iyer. "ATCP: Improving TCP Performance over Mobile Wireless Environments", in Fourth IEEE Conference on Mobile and Wireless Communications Networks, Stockholm, Sweden, September 2002.

[10] J. Liu and S. Singh. "ATCP: TCP for mobile ad hoc networks", IEEE JSAC, vol. 19, no. 7, pp. 1300 1315, Jul. 2001.

[11] L. Xu, K. Harfoush, and I. Rhee. "Binary Increase Congestion Control for Fast Long-Distance Networks", INFOCOM 2004

[12] L. Xu, K. Harfoush, and I. Rhee. "CUBIC: A New TCP-Friendly High-Speed TCP Variant", Third International Workshop on Protocols for Fast Long-Distance Networks, 2005

[13] R.Braden. "T/TCP – TCP Extensions for Transactions Functional Specification, RFC 1644", http://rfc.sunsite.dk/rfc/rfc1644.html

[14] V. Jacobson. "Congestion Avoidance and Control", Proceedings of ACM SIGCOMM 88, Stanford, CA, USA, August 1988.

[15] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, R.H. Katz. "A Comparison of Mechanisms for Improving TCP Performance Over Wireless Links", IEEE/ACM Transactions on Networking, Volume 5, Issue 6, December 1997, pp. 756769.

[16] Balakrishnan, H., S. Seshan, E. Amir and R. H. Katz. "Improving TCP/IP Performance Over Wireless Networks", MOBICOM95, Berkeley, CA, USA, November 1995

[17] Y. G. Iyer, S. Gandham, and S. Venkatesan. "STCP: A Generic Transport Layer Protocol for Wireless Sensor Networks", Proc. IEEE ICCCN 2005, San Diego, CA, Oct. 1719, 2005.

[18] J. Ishac and M. Allman. "On the performance of TCP spoofing in satellite networks", in Proc. IEEE MILCOM'01 Conference, Oct. 2001.

[19] E. Amir, H. Balakrishnan, S. Seshan, and R. H. Katz. “Efficient TCP over networks with wireless links”, in Proc. Fifth IEEE Workshop of Hot Topics in Operating Systems (May 1995).

[20] J. Postel. “Transmission control protocol. Request for Comments 793”, September 1981.

[21] L. Ong. “An Introduction to the Stream Control Transmission Protocol (SCTP). Request for Comments 3286”, May 2002.

[22] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang. “TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links”, in ACM MOBICOM, pages 287297, 2001.