UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

DESIGN AND EVALUATION OF PROTOCOLS FOR WIRELESS NETWORKS TAKING INTO ACCOUNT THE INTERACTION BETWEEN TRANSPORT AND NETWORK LAYERS

A Dissertation

SUBMITTED TO THE GRADUATE FACULATY

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

By

Waleed Saleh Al-Numay

Norman, Oklahoma

2003

DESIGN AND EVALUATION OF PROTOCOLS FOR WIRELESS NETWORKS TAKING INTO ACCOUNT THE INTERACTION BETWEEN TRANSPORT AND NETWORK LAYERS

A DISSERTATION APPROVED FOR THE

SCHOOL OF COMPUTER SCIENCE

BY

[illegible]

Sridhar Radhakrishnan, Chair

[illegible]

S. Lakshmivarahan

[illegible]

Sudarshan K. Dhall

John K. Antonio

John K. Antonio

Marilyn Breen

Marilyn Breen

Acknowledgments

All my thanks and prayers are due to the one and only one ALLAH. He created me and gave me the ability to succeed.

Next, I wish to express my sincere gratitude to my dissertation advisor Dr. Sridhar Radhakrishnan for his continuous guidance and inspiration without which this work would have been impossible. Sharing his experience, knowledge, and creativity was the foundation of this dissertation. I am very thankful to his encouragement, tolerance, and patients.

A special thanks to my dissertation committee Dr. John Antonio, Dr. Sudarshan Dhall, Dr. S. Lakshmivarahan, and Dr. Marilyn Breen for their support and guidance.

I am very grateful to the support and guidance of the Royal Highness Prince Bander bin Sultan for his emotional support and great representation of Saudi Arabia in the US. His words and encouragements were there for me when I needed them the most.

A million thanks to my lovely wife, Nada Fahad Al-Othman. She is my motivator to continuo the PhD. Without her support, I would quit long time ago. She was there for me either in my good day or bad ones. This dissertation is not possible without you, Nada.

Thanks to my parents Saleh and Hilah. Many thanks to Dr. Ibrahim Al-Numay for his care and advices. Thanks to my brothers, Abdulrahman, Abdullah, Dr. Norah, Dr. Mohammed, Asma, and Sattam.

Finally, I would like to thank my colleagues Bharath Ramanujam, Tao Zheng, and Shankar Banik for sharing their experience and knowledge.

Contents

1. Introduction **1**

1.1. Mobile vs. Fixed networks 1

1.2. Connectionless vs. connection-oriented networks 3

1.3. Mobile networks: Applications 4

1.4. Issues for Protocol Layers in Mobile Networks 5

1.4.1. Application Layer 6

1.4.2. Transport Layer 7

1.4.3. Routing Layer 9

1.5. TCP in Mobile Networks 12

1.5.1. Organization of Mobile TCP schemes 13

1.5.2. Mobile TCP proposals 15

1.6. Dissertation overview and Organization 33

2. Mobile TCP and Rerouting in Connection-less Environment **36**

2.1. Introduction 36

2.2. Interaction between mobile TCPs and rerouting schemes 39

2.2.1. Cell forwarding Interaction protocol 41

2.2.2. Full rerouting Interaction protocol 44

2.2.3. Tree rerouting Interaction protocol 45

2.3. Evaluation metrics 47

2.4. Analytical model and simulation 49

2.5. Case study: Mobile IP and mobile TCP schemes 57

2.6.1 Analytical model 58

2.6. Summary 62

3. Mobile TCP and Rerouting in Connection-oriented Environment **63**

3.1. Introduction 63
3.2. Interaction between mobile TCP's and rerouting schemes 65
3.2.1. Mobile TCP and Full rerouting 65
3.2.2. Mobile TCP and Cell forwarding 69
3.2.3. Mobile TCP and Partial rerouting 71
3.3. Protocol Verification and proof of correctness 78
3.4. Evaluation metrics 82
3.5. Analytical model 83
3.4.1 With-hint 85
3.4.1.1. Buffer requirement 85
3.4.1.2. Disruption time at MH 86
3.4.2 Without-hint 91
3.4.2.1. Buffer requirement 91
3.4.2.2. Disruption time at MH 92
3.6. Results 95
3.7. Summary 104

4. JTCP: Join Protocols for Improving Performance of TCP **106**
4.1. Introduction 106
4.2. Related work. 109
4.3. Flow Competition and Throughput 111
4.4. The Join TCP Protocol 114
4.4.1. When to Join 119
4.5. Improvements to the algorithm 121
4.6. Simulation and results 122
4.7. Summary and conclusions 127

5. Tracking Updates in Distributed Wi-Fi Architectures: Flooding Protocols and Application Fine Tuning **129**
5.1. Introduction 129
5.2. The Distributed Wi-Fi Architecture and Routing 131

5.3. Flooding Protocol and its variants 133

5.3.1. Pure flooding 136

5.3.2. Tree flooding 137

5.3.3. Space partitioned tree flooding (SPTF) 139

5.4. Intelligent Tree Flooding Algorithm 145

5.5. Simulations and results 148

5.6. Summary 158

6. Conclusion and future work **160**

6.1. The problem 160

6.2. Proposals organization 161

6.3. Proposals of independent improvement 161

6.4. Proposals of dependent improvement 163

6.5. Future work 164

List of Figures

1.1 IEEE 802.11 and Cellular Networks 2
1.2 Region of influence 2
1.3 Illustrating some of the issues that need to be addressed by each layer of the protocol stack 7
1.4 Rerouting after handoff 10
1.5 Various wireless TCP schemes under consideration 14
1.6 Kazunori and Nakase proposal 19
1.7 Indirect TCP algorithm 23
1.8 Indirect TCP hand-off 23
1.9 Estimating Buffer Occupancy at MH 27
1.10 METP protocol 32
1.11 Header of METP packet 33

2.1 Cell forwarding and mobile TCP interaction protocol 42
2.2 Full rerouting and mobile TCP interaction protocol 44
2.3 Tree rerouting and mobile TCP interaction protocol 46
2.4 Disruption time evaluation 53
2.5 Buffer requirement at old BS 54
2.6 Simulation results for Buffer requirement at old BS for cell forwarding with different values of $N_{oBS\text{-}nBS}$ 55
2.7 Sender's throughput 56
2.8 Different TCP schemes on top of mobile IP, without simultaneous bindings 59
2.9 Different TCP schemes on top of mobile IP, with simultaneous bindings 60

3.1 Interaction protocol between Mobile TCP and

Full Rerouting – without hint 65
3.2 Interaction protocol between Mobile TCP and
Full Rerouting, with hint 66
3.3 Interaction protocol between Mobile TCP and
Cell Forwarding – without hint 69
3.4 Interaction protocol between Mobile TCP and
Cell Forwarding with hint 71
3.5 Interaction protocol between Mobile TCP and
Partial Rerouting – without hint 72
3.6 Interaction protocol between Mobile TCP and
Partial Rerouting – with hint 73
3.7 Interaction between mobile TCP schemes and
Tree rerouting-without hint 74
3.8 Interaction between mobile TCP schemes and
Tree rerouting-with hint 76
3.9 Protocol Correctness Proof 79
3.10 A finite state diagram for the interaction protocol of cell forwarding
and mobile TCP in connection-oriented environment 81
3.11 Analytical model organization 85
3.12 Organization of results showing the calculated
values in all cases 97
3.13 Buffer requirement at old BS in case of "with hint"
for all rerouting schemes with variations of number
of hops and states information size 98
3.14 Disruption time on the uplink path in case of "with hint" for all
rerouting schemes with variations of number of hops between
old and new BS's 98
3.15 Disruption time on the forward and new paths in case
of "with hint" and full rerouting with variations of
number of hops and states information size 99
3.16 Disruption time on the forward path in case of "with hint"

and cell forwarding with variations of number of hops and
states information size 100

3.17 Disruption time on the forward path in case of "with hint"
and tree rerouting with variations of number of hops and
states information size 100

3.18 Disruption time on the forward path in case of "with hint"
and cell forwarding with variations of number of hops
and states information size 101

3.19 Disruption time on the forward path in case of "with hint"
and partial rerouting with variations of number of hops
and states information size 101

3.20 Buffer requirement at old BS in case of "with-out hint"
for all rerouting schemes with variations of number of
hops and states information size 102

3.21 Disruption time on the uplink path in case of "with-out
hint" for all rerouting schemes with variations of
number of hops and states information size 102

3.22 Disruption time on the forward and new paths in case
of "with-out hint" and full rerouting with variations
of number of hops and states information size 103

3.23 Disruption time on the forward path in case of
"with hint" and cell forwarding with variations
of number of hops and states information sizes 103

3.24 Disruption time on the forward path in case of
"with-out hint" and cell forwarding with variations
of number of hops and states information size 104

3.25 Disruption time on the forward path in case of
"with-out hint" and tree rerouting with variations
of number of hops and states information sizes 104

4.1 A log graph showing the window size changes and

total data sent by a single flow in a network environment consisting of two flows with a bottleneck bandwidth of 32KB. Each line in the above graph shows the impact of the initial threshold value on the total data sent 113
4.2 Two senders S_1 and S_2 are in communication with receivers R_1 and R_2, respectively. The agents are only at the portal router 116
4.3 Router A will run the join protocol to combine traffic from sources S_1 and S_2. Router C will run the join protocol to combine traffic from S_3 and the combined traffic of S_1 and S_2 118
4.4 Static environment (a) TCP Tahoe (b) Join algorithm 122
4.5 TCP Tahoe goodput 124
4.6 TCP Tahoe throughput 124
4.7 TCP goodput - join algorithm 125
4.8 TCP throughput - join algorithm 125
4.9 Simulation topology configuration 126
4.10 TCP Tahoe goodput 127
4.11 Join goodput 127

5.1 Schematic representation of routing tables in Wi-Fi networks 131
5.2 Network topology (a) before and (b) after power tuning 137
5.3 Hierarchical relationship among PAPs 142
5.4 Inter-region data communication 143
5.5 RUP propagation between APA: (a) Central PAP (b) Non-Central PAP (c) Central-Tree PAP 145
5.6 Dividing the area into quadrants, (a) First level (b) Second level (c) Third level 147
5.7 Quad tree representation of the regions in Figure 5.5 148
5.8 Simulation topology 149

5.9 Number of tracking messages 151
5.10 Goodput for TCP sessions, fast mode, with variable RTO 153
5.11 Goodput for TCP sessions, fast mode, with RTO = 10 156
5.12 Goodput for TCP sessions, slow mode, with variable RTO 158

Abstract

The Transmission Control Protocol (TCP) provides end-to-end data reliability and is the primary transport layer protocol for many applications such as email, web access, and file transfer. There has been a plethora of research activity that aims to improve the performance of TCP both in wired and wireless networks. Protocols for the computer networks have been very structured and layered to allow for easier upgrades and maintenance. The network layer protocol (e.g IP) is independent and below the transport layer protocol (e.g TCP). Our main goal in this dissertation is to examine the interaction and dynamics between the network layer protocols and TCP in the wireless environment.

Towards this goal, we examined the network layer protocols in one-hop wireless (e.g. cellular networks) and multi-hop wireless, e.g. distributed Wi-Fi (Wireless Fidelity) networks. For each of these networks we, for the first time, propose transport layer protocols that take into account the interaction between the network layer and transport layer. For the one-hop wireless networks we have investigated analytical methods to determine the buffer requirements at base stations and estimate disruption time which is the time between two packet arrivals at the mobile host. We will show that the estimation of buffer requirements and disruption time is not only dependent on the wireless TCP scheme used, but also its interaction with the underlying network protocol. We also propose a comprehensive study of the effectiveness of wireless TCP and network protocols taking into account different networking environments that is decided on many factors such as mobility of senders and receivers, simplex and duplex communication among communicating peers, connection oriented and connection less communication at the network layer, rerouting schemes used after movement, and with and without hint handoff schemes.

We recognized two important shortcomings of the current TCP protocol: misinterpretation of delayed acknowledgements and competition among different TCP flows. In this dissertation, we propose to address these two issues by a use of novel protocol that uses *immediate and delayed acknowledgement schemes* and provides a *coordination mechanism* among independent TCP flows. We also address certain important issues that are related to the implementation of our proposed protocol: can we maintain the end-to-end semantics of TCP? Are there additional benefits that can be harvested if intermediate nodes with TCP protocol can be used? We will show that our protocol can be implemented on a portal router or a node attached to the portal router of a network administrative domain. We can extend this approach to the Internet and we provide protocols for this extension. The simulation results show that our protocol achieved almost perfect fair sharing of network resources among competing flows. We also observed that throughput increases 9000 percent due to the use of immediate acknowledgment, while goodput increases between 40 and 70 percent.

Finally, we consider protocols to track mobile users in a wireless network that consists of non-mobile nodes forming a wireless backbone. We call such a network as the distributed Wi-Fi architecture. Our protocols are region based we exploit station locality to perform controlled flooding. We evaluate our protocols and fine tune them taking into account the mobility rates and application characteristics. The performance of TCP is evaluated for various tracking mechanisms and it has been observed that our protocol adapts to different mobility rates much more efficiently in comparison with pure flooding.

Chapter 1

Introduction

1.1 Mobile vs. Fixed networks

Mobile networks take different shapes and forms depending on the size of the coverage area and the type of physical layer used for connecting the nodes. The connection between the nodes in the network can be pure wireline, pure wireless and mobile, or a mixture of the two. An example of the pure wireline network is the conventional Internet where the stations' mobility is not supported. In the pure wireless and mobile network architecture, mobile station(s) (MS) are free to move around within the coverage area without losing connections. Users can send emails, browse the Internet, FTP a file, and conduct a video conference meeting, all while moving. Another aspect to pure wireless networks is that all links between the sender and the receiver are wireless radio links wherein the communication medium is the radio spectrum. These types of networks are called *Ad Hoc networks* and it introduces new challenges to the network research community.

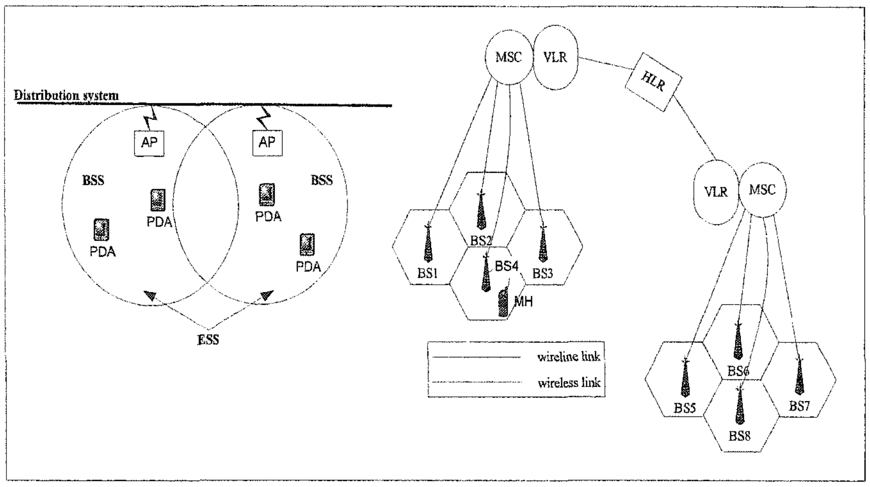


Figure 1.1: IEEE 802.11 and Cellular Networks

Finally, *cellular networks* used for telephone and other data services for large geographical areas are a type of one-hop wireless networks. The antenna that is responsible for receiving the signals from the mobile phone is called as the base station (BS). The BSs are connected to the wireline backbone network. The network formed using the pervasive IEEE 802.11 protocol is also a one-hop wireless network although IEEE 802.11 protocol provides a mechanism for multi-hop wireless communication. The

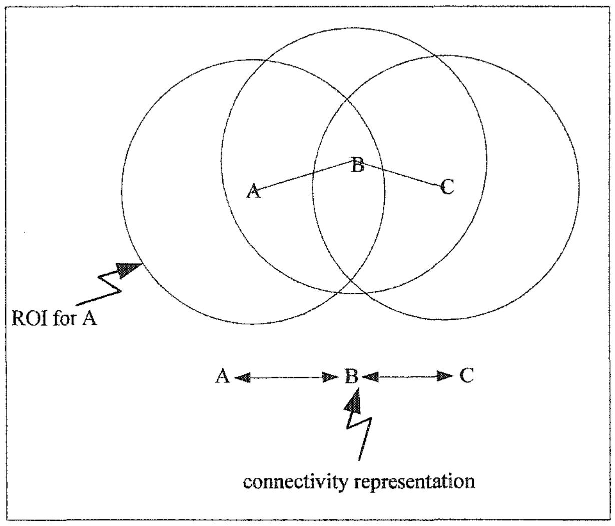


Figure 1.2: Region of influence

wireless units in the IEEE 802.11 protocol communicate with the immobile Access Point (AP). In these types of networks the MS can be of any type of portable computer or handheld device achieving true mobility by using radio signals to send or receive data. The BS or AP is a special fixed station that is a mobile-aware station and acts as an intermediary between the wireline backbone network and the MS.

Each BS, AP, or MS has an area of influence, which is called a *cell* or a *region-of-influence* (ROI). This area indicates the geographical region wherein the signals from the particular system can be detected. The size of the cell depends on the transmission power of each of the systems. Two systems MS-BS or MS-AP or MS-MS can communicate with each other if and only if their ROIs intersect.

1.2 Connectionless vs. connection-oriented networks

Services offered by the network layer can be of two types: connectionless and connection-oriented. In a simple way, connectionless is modeled after the postal service. Each envelope is self contained. It contains the material needed to be mailed and also, it contains the full address of source and destination. Therefore, each envelope is routing independently of all other envelops inside the mailing system. When two packages are sent from the same source to the same destination, there is no guarantee that the package sent first will reach first due to the fact that they might consume different paths from the source to destination.

The connection-oriented service is modeled after the telephone system. In telephone system, a complete connection from the caller and the callee must be established before a conversation can take place. After a successful connection establishment, the system guarantees in-order delivery and, some times, certain level of quality of the offered service. At the end of the call, the connection is torn down and the reserved resources are freed form future usage by other customers.

In data networks, IP (Internet protocol) is one of the most popular connectionless network layer protocols in use while ATM is the famous connection-oriented network protocol. A data unit in IP is called segment. A header is attached to each segment sent out that contains all information needed to route the segment from source to destination nodes. The information stored in the header is sufficient for intermediate IP routers to route the segment to the direction of the destination. Each router has a routing table that it uses to decide which out-port to use to send a segment at to reach its destination. The data stored at the routing table are dynamic and changes with the change of the level of congestion at the output link. In ATM, a sender must first acquire a virtual circuit to be used to carry connection establishment messaging. The sender sends a cell containing a request for a virtual circuit on virtual path 0 and virtual circuit 5. Once the request is approved, a new virtual circuit is created and opened for the sender and receiver to use to exchange the connection establishment messages. Basically, there are six messages to establish a connection: setup, call proceeding, connect, connect ACK, release, and release complete [29].

1.3 Mobile networks: Applications

The mobile network technology is reshaping enterprise connectivity worldwide. Government offices and private enterprises need information mobility for better customer interactions, and improving their work processes by offering anywhere, anytime access to data. This eventually will improve employee's productivity as well as customer satisfaction since real-time access to information can be critical to serving customers. The number of wireless users is said to reach one billion within 6 months.

A mobile ad hoc network (MANET) can be formed without any network infrastructure support using the devices and a wireless transceiver. MANETs are useful when a temporary network is needed in locations where infrastructure support is absent or expensive to build. For example, a disaster groups can form a temporary network to provide network support during the search and rescue operations. Military personal can use these networks in non-friendly environments for logistics planning.

Application for one-hop wireless networks includes cellular phone data and voice exchange, video conferencing, and live updates for critical information such as news, financial, or medical information all while moving. For example, a paramedic at the seen wants to transfer patients' medical information, such as heart rate, x-ray results, or blood test results to the hospital on way to the hospital. In such cases, fast response and right diagnoses depends in large on the speed and reliability of transfer of data.

1.4 Issues for Protocol Layers in Mobile Networks

In this section, we look at the various issues for the OSI reference model layers for wired networks and briefly describe the functionality of the layers as they apply to mobile and wireless networks (see Figure 1.3 for an overview). We pay special attention to the application, transport, network, data link, and physical layers. The OSI reference model contains presentation and session layers, which are not described in great detail in this chapter for the sake of simplicity. In the next subsections we explain the issues relating to application, transport, and routing layer. The data link layer, medium access layer, and the physical layer descriptions are omitted as they are not the focus of this dissertation. A more detailed explanation of the OSI reference model and its layers can be found in Tanenbaum [29].

1.4.1 Application Layer

The application layer provides network access to applications and protocols commonly used by end users. These applications and protocols include multimedia (audio/video, file system, and print services), file transfer protocol (FTP), electronic mail (SMTP), telnet, domain name service (DNS), and Web page retrieval (HTTP). Other higher-level issues such as security, privacy, user profiles, authentication, and data encryption also are handled by the application layer. In the case of *ad hoc* networks, the application layer also is responsible for providing location-based services. The presentation layer is responsible for data representation as it appears to the end user, including character sets (ASCII/EBCIDIC), syntax, and formatting. The protocols associated with the presentation layer include Network Virtual Terminal (NVT), AppleTalk Filing Protocol (AFP), and Server Message Block (SMB). The session layer is responsible for data exchange between application processes, including session flow control and error checking.

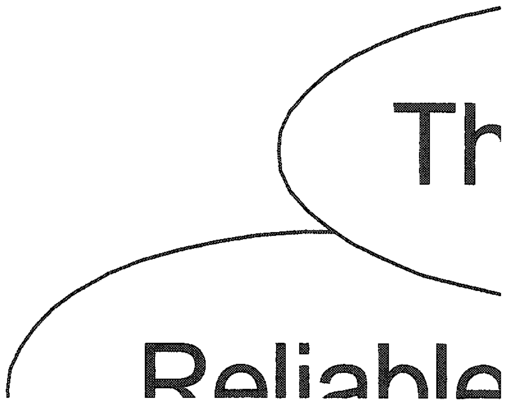


Figure 1.3: Illustrating some of the issues that need to be addressed by each layer of the protocol stack.

1.4.2 Transport Layer

The purpose of the transport layer is to support integrity of data packets from the source node to the destination node (end-to-end). Transport protocols can be either connection-oriented or connectionless. Connection-oriented transport protocols are needed for ensuring sequenced data delivery. In order to ensure reliable sequenced delivery, the transport layer performs multiplexing, segmenting, blocking, concatenating, error detection and recovery, flow control, and expedited data transfer. Connectionless protocols are used if reliability and sequenced data can be traded in exchange for fast data delivery. The transport layer assumes that the network layer is inherently unreliable, as the network layer can drop or lose packets, duplicate packets, and deliver packets out of order.

The transport layer ensures reliable delivery by the use of acknowledgments and retransmissions. The destination node, after it receives a packet, sends the acknowledgment back to the sender. The destination node sometime uses cumulative acknowledgment wherein a single acknowledgment is used to acknowledge a group of packets received. The sender, rather than sending a single segment at a time, sometimes sends a group of segments. This group size is referred to as the windows size. The sender increases its window size as the acknowledgments arrive. The Transmission Control Protocol (TCP), which is the most commonly used reliable transport layer protocol, also takes care of congestion avoidance and control. The TCP protocol increases its throughput as acknowledgments arrive within a time period called the TCP time interval. If the acknowledgments arrive late, it assumes that the network is overloaded and reduces its throughput to avoid congesting the network. In contrast, User Datagram Protocol (UDP) is a connectionless transport protocol used for applications such as voice-and-video transport and DNS lookup.

Transport layer protocols in wireless and mobile networks should address the following issues:

- Window size adjustments have to be made that not only take into account the channel errors and the end-to-end delays, but also should adjust based on the mobility dynamics of the network nodes.
- Cumulative acknowledgments can be both good and bad. Given the packet losses expected in mobile and wireless networks, the loss of a single acknowledgment packet will result in retransmission of a large number of packets. Mechanisms to

adjust the acknowledgment schemes based on the dynamics of the network should be taken into consideration.

- The timeout interval, which dictates the length of time the protocol waits before beginning the retransmission should be adjusted, based on the dynamics of the network. Clearly, a shorter time-out interval will increase the number of retransmissions, while longer time-out intervals decrease the throughput.
- The original TCP congestion control is purely based on acknowledgment delays. This does not necessarily work well in the case of wireless and mobile networks, where the delays are attributed to channel errors, broken links caused by mobility, and the contention at the medium access control (MAC) layer that is not only dependent on the traffic in the network, but also on the degree (number of neighbors) of nodes in the network.

A detailed description of the issues relating to TCP in wireless and mobile networks is discussed in section 1.4.

1.4.3 Routing Layer

The routing algorithms for wireless and mobile networks have received most attention in recent years, and many techniques have been proposed to find a feasible path between source and destination node pairs. The algorithms for finding paths depend on the particular network environment: pure wireless as in the case of ad hoc network or one-hop wireless network as in the cases of cellular and IEEE 802.11 networks.

In the case of a one-hop wireless mobile environment the sender, the receiver or both can be mobile. In such environment, mobile hosts require connecting to the base station to communicate with either other mobile hosts or fixed terminals. The process of registering with a new BS after moving is called a *handoff*. After the handoff, packets destined for the MH have to be rerouted to the MH at its new position through the new BS. Figure 1.4 shows the rerouting process after the handoff from the cell of the old BS to the cell of the new BS. In our study, the architecture of our mobile network consist of static communication backbone using wired means, a set of static host (routers), MH's, and base stations capable of sending and receiving signal from the MH's in there cells. The underline routing algorithm depends on the rerouting algorithm in consideration.

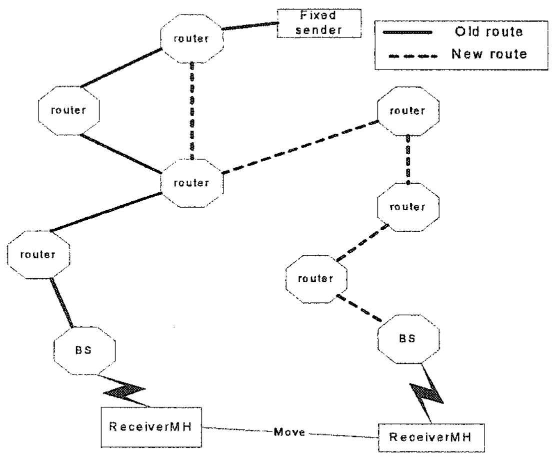


Figure 1.4: Rerouting after handoff.

Finding and establishing the new route is named as the *rerouting process* that has been studied extensively in the literature. The proposed schemes fit into one of four categories [8]: full rerouting, partial rerouting, cell forwarding, and tree rerouting. In *full rerouting* a new path is established from the fixed host or fixed sender (FS) to the new base station. *Partial rerouting* finds the first shared node (call it *crossover* node) between the old path (path from fixed host to the old base station) and the new path (path from fixed host to the new base station), and finds a new path from such a node to the new BS. This scheme will reuse the part of the old path from the sender down to the crossover node. The partial rerouting is of value in connection-oriented networks where the paths from source to destination remain the same for the entire session duration and resources are reserved along the paths. In *tree rerouting* the sender assumes the root of tree with base stations as leaves and the routers are the internal nodes. The root of the tree multicasts the packet to all its leaves and this allows for the packet to be available to the MH's after a handoff. In the *cell forwarding* data from the old BS is forwarded to the new one. A chain of forward connections will result in connecting the BS's which MH visits and the chain ends at the BS to which the MH is currently connected to.

Routing in MANETs involves two important problems: (1) finding a route from the source node to the destination node, and (2) maintaining routes when there is at least one session using the route. MANET routing protocols described in the literature can be either reactive, proactive, hybrid (combination of reactive and proactive), or location based. In a proactive protocol, the nodes in the system continuously monitor the topology changes and update the routing tables, similar to the link state and distance vector algorithms. There is a significant route management overhead in the case or proactive schemes, but a new session can begin as soon as the request arrives. A reactive protocol, on the other hand, discovers a route as a request arrives (on-demand). The route discovery process is performed either on a per-packet basis or a per-session basis. When routes are discovered on a per-packet basis, the routing algorithm has a high probability of sending the packet to the destination in the presence of high mobility. A routing algorithm that uses the route discovery process for a session must perform local maintenance of severed or broken paths. Location-based routing protocols use the geographical location information about each node to perform intelligent routing. Many proposals were suggested to improve the routing performance in MANET [Scalable routing protocols for mobile Ad Hoc Networks, Xiaoyan Hong et al]

1.5 TCP in Mobile Networks

TCP is a reliable, in-order delivery protocol, which uses a sliding window mechanism to control the flow between the communicating peers and to manage any congestion that might occur in the network during the communication period. The problem with TCP in mobile networks is that it can't distinguish between packet drop due to errors in the wireless links, overflow, or mobility. TCP assumes that all types of packet drops are due to congestion. Therefore, it will drop its transmission, enter a slow start period, and update the retransmission timer. This will result in unnecessary degradation in system performance [1].

Due to the importance of TCP in data communication and also due to the expanding in using mobile networks, many brilliant proposals were published to improve the performance of TCP in wireless and mobile networks. These proposals can be divided into two groups: modified and direct TCP and split TCP. In the first group, an attempt is made to change the some of TCP characteristics to work better in a mobile-wireless environment while the send group focuses on splitting a TCP connection into two segments to hide the impact of mobility and wireless errors from the sender.

1.5.1 Organization of Mobile TCP schemes

The categorization of all wireless schemes is illustrated in Figure 1.5. These different categories divide the wireless TCP schemes suggested in the literature into two main categories depending upon the node that sends the acknowledgement to the sender and the type of the reliable protocol used at the wireless link. In the first instance called the *WTCP*, the sender establishes a normal TCP connection to the mobile host (the receiver in the session in communication). In this case, the ends of the TCP sockets are the sender and the mobile host. Packets travel through the wired network from the sender to the base station (the node in the wired network with whom the mobile host is currently registered and in communication with), which in turn uses a standard data link layer protocol to forward the packets to the mobile host. The acknowledgements from the mobile host are sent to the sender directly from the mobile host through the base station. Again a simple data link layer protocol is used to send the acknowledgement from the mobile host to the base station. In the *WTCP* scheme, all intermediate nodes, including the base station, perform only operations at the network layer and below, while the end nodes (sender and mobile host) perform operations at the transport layer and below.

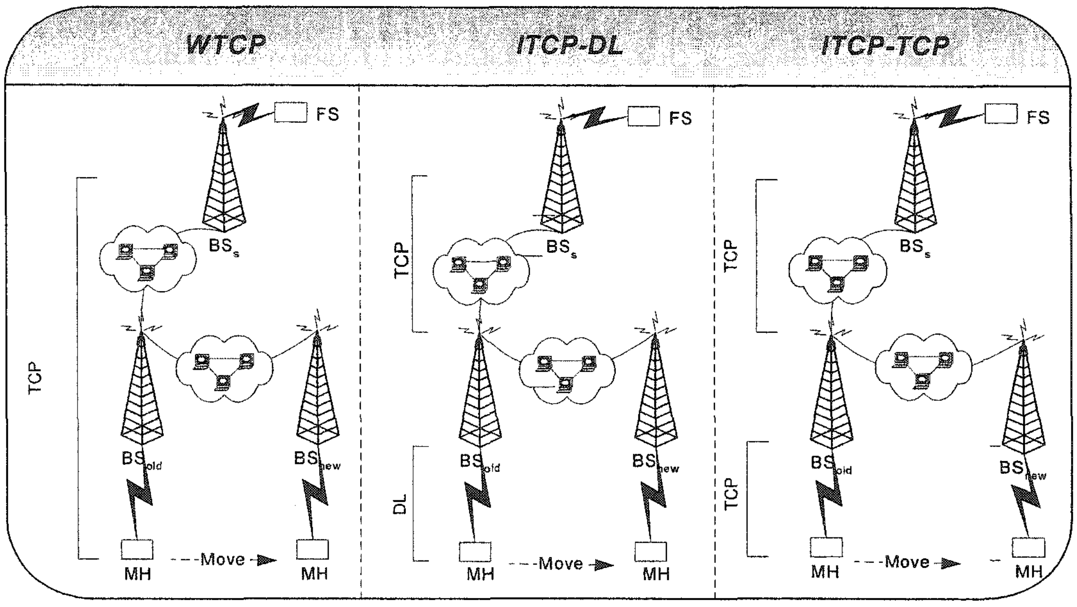


Figure 1.5: Various wireless TCP schemes under consideration.

In the second scheme, named *Indirect TCP* (*ITCP*), the sender establishes a TCP connection to the base station of the mobile host, and the base station can either establish another TCP connection to the mobile host (call this as *ITCP-TCP*) or it can send the packets using a data link layer protocol (call this as *ITCP-DL*). In these schemes, one has two options of sending the acknowledgement back to the sender. In the first option, the base station buffers the packets and sends the acknowledgement as soon as it receives the packet from the sender. We will suffix the schemes with an "*-I*" for immediate acknowledgement. For the second option, the base station sends the acknowledgement only after it receives the acknowledgment from the mobile host. A suffix of "*-D*" is given to the schemes to indicate delayed acknowledgement. After the arrival of mobile host, MH, the new base station, BS, will request old BS to transfer the state information of the TCP connections opened on behalf of the MH. It is obvious that this transformation need to be reliable. After successfully transferred all state information, the old BS will not acknowledge any packet destined to the MH and it will delete such connection information from its local buffer in order to save its resources such as buffer space and port numbers. It is clear that this state transformation is necessary only in case of split TCP because the end point of connection with the sender is the base station.

1.5.2 Mobile TCP proposals

Many proposals have been suggested in the letrutual to improve TCP's performance in mobile and wireless networks. Below, we present an examples of suggeted proposals that covers all categorization of mobile TCP that we define in brevouse section.

Modified and direct TCP

Proposals in this category uses the assumption that the wireless link is one hope only. Therefore, using local retransmission at the link-level and forward error correction over the wireless link to hide the packet loss for the above layers. Two main approach were adopted by these proposals. Forward error correction (FEC) and retransmission in response to Automatic Repeat Request (ARQ). Some examples are discussed below.

Snoop protocol [2]

Snoop is a module reside at the base station that monitors all packets passing through the TCP connection in both direction. It caches all unacknowledge packets that were sent by the fixed host to perform local retransmission, as needed. The module detect a packet loss by receiving duplicate acknowledgement from the receiver. When it detects a packet loss, the module perform a local retrasmission from its cach, if the packet is availabe, and suppresses the duplicate acknowledgement. Like any other data link layer proposal, it can not completely shield the sender from wireless losses but it has the advantage of suppressing duplicate acknowledgments and perform retransmission locally. Also it have to maintain a state information for each connection passes through the base station.

Each mobile host uses a multicast address as its care-of address. The mobile host invites nearby base station that it expects to move to in the near future to join his multicast address. All packets destined to the mobile host are ecnasulated by the home agent and sent to the multicast address using tunneling. Only one base station from the multicast group forwards packets to the mobile host, while the others buffer the packets. Most likelly, MH will hand off to a BS member of the multicast group and new BS will start forwarding packets immediately. algorithm is shown below.

- *When snoop receives a packet destined to MH.*
    - If it is new packet in the normal TCP sequence
        - Add the packet to snoop cache.
        - Forward it to the MH.
        - Add time stamp of some chosen packet in a transmitted window to estimate the RTT.
    - If it is an out of sequence packet that has been cached earlier
        - If the packet is greater than the last acknowledgment seen
            - It is forwarded to the MH.
        - Else

            - Snoop sends an ACK corresponding to the last ACK seen at the BS to the FH.
    - If it is an out of sequence packet that has not been cached earlier
        - Forward it to the MH.
        - Mark it as have been retransmitted by the sender. (to be used later).
- *If an acknowledgment is received from the mobile host.*
    - If it is a new ACK
        - All acknowledged packets are freed from snoop cache.
        - Update the RTT estimate for the wireless link.
    - (this is done for only one packet in each window with no retransmitted packets).
        - Forwarded it to the FH.
    - If it is less than the last acknowledgment seen by snoop
        - Discard it.
    - If it is a duplicate acknowledgement
        - If it is for a packet not in snoop cache or has been market as retransmitted by the sender
            - Route it to the FH
        - If it is an unexpected duplicate
    - (base on estimating the number of packets sent after the packet which its
    - number appear in the duplicate)
            - The lost packet is send with higher priority (to minimize number of

            - duplicate acknowledgments)
            - If it is an expected duplicate
                - Discard it. (most likely the packet has been sent)

- *If snoop receives a packet from MH destined for FH*
    - Cache the packet, for any retransmission in the future, and transmit it to the FH.
    - Every while (threshold No. of received packets or a time interval with no packets from MH)
        - Send NACK for the missing packet in each transmitted window
    - (as a bit vector, to ensure good utilization of the wireless bandwidth.)
- *If snoop receives an ACK from the FH*
    - free the acknowledged packets from its cache.

Kazunori and Nakase[3]

In this proposal there is no need to change the conventional TCP protocol. The authors propose two error control scheme that minimize the degradation in radio link throughput due to TCP retransmission. A configuration figuer shown in Figure 1.6 discribes the system moder of the proposal.

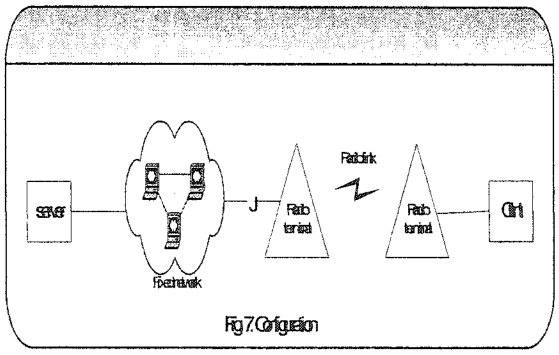


Figure 1.6: Kazunori and Nakase proposal

The first method is about discarding any TCP retransmission of packets and uses ARQ algorithm in the radio link. If the server (fixed sender) sends a packet to a client (mobile reveiver) on a TCP connection and the packet reached the client then an ACK is sent back to the server before it times out. If the server timesout then it will retransmit the un-acknowledged packet. Once the radio terminal (base station) recongnize that the packet is retransmitted then it discard it because it assumes the ARQ algorithm in the Link layer realize error free link.

The second scheme tries to deal with the rapid change in radio link condition due to fading and shadowing. Thuse, the time it takes to send a packet successfully in the first time (without retransmission) vary. The idea behind this scheme is to make the sender TCP sees a constant RTT and hence no need to invoke unnecessery retransmission. For this reason the author suggest that the radio terminal holds the TCP ACK in its buffer for Constant Round Trip Time (CRTT). CRTT equals minimum RTT plus some delay margin to allow for adverse radio link conditions. After waiting CRTT, radio terminal sends the acknowledgement to the sender. In this case the sender will see constant RTT and hence no retransmission and congestion avoidance algorithms invoked. Two algorithms are shown below to more discribe the two schemes.

Scheme 1: Discarding TCP Retransmission Packets

- Server sends TCP segment to client
- A radio terminal in the server-client path receive the packet
- The radio terminal uses ARQ to transmit the segment (as frames) to radio terminal 2
- which will send it to client upon successful receive of all frames.
    - ARQ will keep sending frames and increase the RTT with every unacknowledged
    - frame until all frames acknowledge.
    - If RTT exceeds RTO of the TCP entity on server
        - Server timeout and retransmits the segment again.
        - Radio terminal discover that the segment is a retransmission and discard it.

Scheme 2: Suppressing Transmission Delay Fluctuation

- If the radio terminal receives back acknowledgements for all sent frames
    - Hold the T TCP acknowledgement for the server for suitable duration.
    - Send the TCP acknowledgement to the server.

Wong & Leung [4]

The authors study the impact of sequincing and non-sequincing data link protocols on TCP's throughput and delay. A sequincing protocol needs some buffer requirments while non-sequincing protocol dosent. In the simulations conducted by the authors the non-sequicing protocol have a degradation in TCP throughput up to 2.5% in the worst case. In most cases there were no different in TCP throuput. Thus, non-sequinceing is prefered in TCP/IP environment.

Normally a data link layer protocol will try to retransmite a packet over the wireless link for some number of times predefined and if the transmission was not successful the packet will be droped and all the parameters will be reset.

The authors propose that the data link protocol make a best effort to recovver from lossses occure at the wireless link and TCP do the rest. They study the impact of number of retransmitting and find out that when the number of retries increase the RTO increases. When the number of retransmissions reached 5 there were no changes in the RTO.

They suggest using a non-sequincing data link protocol with retries equals to 4. No change to TCP is required.

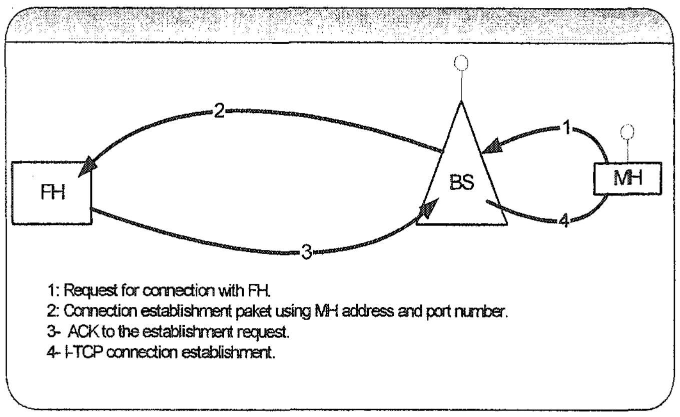
Split TCP proposals

In this category the proposals tend to splite the TCP connection into two connections at the base station – one TCP connection between the sender and the base station and one between the base station and the receiver. This approach is trying to split the error recovery over the wirless link from that across the wirline network

I-TCP [5]

The connection is asuumed to be between a fixed host and a mobile one. This connection is established as two separate connections – one over the fixed network and one over the wireless link and the base station serve as an intermideate point. The connection establishment by the MH to the FH, Figure 1.7, is done in the following steps. The MH sends a requests to establish an I-TCP connection with the FH to the BS. Then the BS establishes a regular TCP connection with the FH intended using the IP address and port number of the MH for local endpoint parameters. Another connection is established between BS and MH. BS uses its own IP address and choose some sutable port number to identfiy the end points of the connection.

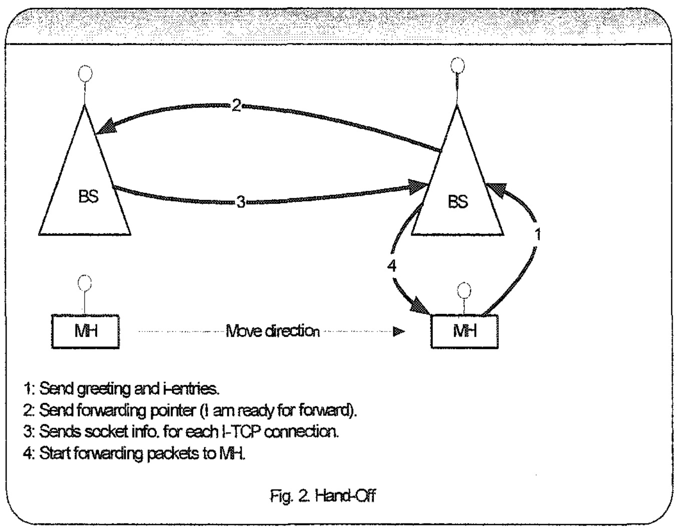
If the FH wants to create a connection with the MH, it sends a request to the MH. The BS intercept and accepts the request. Then BS sends a request to the MH to establish a modefied TCP connection using its own address and some approperiate port number. If the MH accepts the connection, BS creates a connection with the MH using its own IP and some port number. Subsequently, BS creates a socket using the MH address and port number. Using MH's address helps the BS to grab the TCP segments that are sent by the FH to the MH addrss and port number. At BS, data are copied from one socket buffer to the other one.



1: Request for connection with FH.
2: Connection establishment paket using MH address and port number.
3- ACK to the establishment request.
4- I-TCP connection establishment.

Figure 1.7: Indirect TCP algorithm

As shown in Figure 1.8, when a Mobile host move to another base statio the fixed host is unaware of the movement and the new base station become the new intermideate point. The old base station hand over the state of the two sockets opend for the connection to the new base station. Then the new base station creates two sockets with the same endpoint parameters as the sockets at the old base station have. No need to re-establish the connection sicnce the two endpoints are the same. An algorithm below describes the protocol.



1: Send greeting and i-entries.
2: Send forwarding pointer (I am ready for forward).
3: Sends socket info. for each I-TCP connection.
4: Start forwarding packets to MH.

Fig. 2. Hand-Off

Figure 1.8: Indirect TCP hand-off

Connection establishment

- If MH wants to communicate with FH
  - MH sends a request to inform BS that it wants to establish a connection with FH.
  - When BS receives the request
    - Establish a TCP connection with the FH using MH IP address and port number.
    - Establish a modified TCP connection with the MH using its own IP and port number
- if FH wants to communicate with MH
  - FH sends a connection establishment packet to MH.
  - if the BS receives the packet
    - Save the packet
    - Establish a connection with the MH using its own IP and port number.
    - if the connection with the MH established
      - BS reply to FH request for connection using MH IP address and port number.

Handoff procedure

- When the MH move from one cell to another
  - mhmicp process at the MH losses contact with old BS.
  - mhmicp process at the MH hears a beacon in the cell of the new BS.

- o mhmicp process at the MH sends a greeting message to the new BS
    - ▪ That includes a list of MICP i-entries and the address of the old BS.
- o If the msrmicp process (the micp at the BS) receives the greeting from MH
    - ▪ Sends a copy of the greeting message to the local I-TCP daemon.
    - ▪ I-TCP daemon establishes skeleton sockets for each I-TCP connection from the
        - o i-entries using SIOCCREATE calls.
    - ▪ I-TCP daemon sends a forwarding pointer to the old BS.
    - ▪ I-TCP daemon sends an ACK to the msrmicp daemon.
    - ▪ If the msrmicp daemon at the old BS receives the forwarding pointer
        - o It updates its data structures to reflect the new location of the MH.
        - o It sends a copy of the forwarding pointer to its local I- TCP.
        - o If the local I-TCP receives the forward pointer
            - ● It establishes a handoff connection with the I-TCP daemon at the new BS.
            - ● It sends the state of each I-TCP connection to the I-TCP daemon at the New BS using SIOCGETSTATE calls.
            - ● The I-TCP daemon at the new BS executes SIOCSETSTATE calls to receive the sockets for, and restart, each I-TCP connection.

Mobile-TCP [6]

As in any split connection scheme the connection is devided into two segments: wireline segment, the connection between the local fixed host (mobile gateway) and the corresponding host and wireless segment, the connection between the mobile host and a local fixed host. Regular TCP is used in the wireline segment while a new variation of TCP called M-TCP is used in the wireless segment. This protocol was possible due to the observation that the wireless segment of a transport connection is a single-hop connection. For this reason we could list this protocol under the link-Level protocols section.

What is new in this solution is that the wireless segment was implemented with mimimum communication overhead on the mobile host by employing the asymmetrically-based protocol design. This design is applied in connection control, retransmission and timer management, flow control, and error control as follow.
In connection control there is no need to communicate the full TCP-layer source and destination addresses.Thus, a connection ID is assigned to each direction and it is used in any future exchange of data over the wireless segment. Both ends of the connection over the wireless segment stores the following informaion for each connection ID: source and destination IP adresses and the corresponding port numbers.

Different retransmission schemes used in the two directions of the over the wireless segment to ease the work needed by the wireless host. They placed timers on the base station only and not using any timers on the mobile host. To achieve this they suggest using Go-Back-N scheme for retransmission of packets by the base station and using Selective Reject in opposite direction. The choice minimizes the use of the wireless transmission.

Base station maintain two thresholds: low and hi. ON/OFF scheme sends to the mobile host a single bit signal. ON signal means keep sending while OFF means stop. To make sure mobile host gets the signals correct it repeates the ON/OFF signal in all its packets sent to the base station. The flow in the revers direction (MH to BS) is controled by the base station. The base station estimated the availble buffer at the mobile host and stop sending when the available buffer space reache some predefined level. The mobile host helps base station to make its estemation accurate by sending two bits, Buffer Occupancy (BO), in each packet which indicate the buffer level. Figure 1.9 discribe this process.

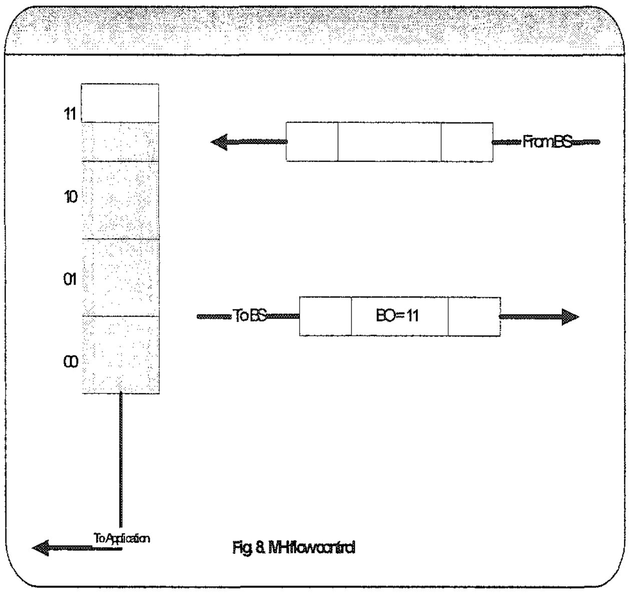


Figure 1.9: Estimating Buffer Occupancey at MH

In this proposal they suggest to place error detection function at the mobile host and the error correction algorithm at the base station. Congestion control function is not needed in M-TCP due to the single-hop nature of the wireless segment. An algrithm discribing the model is shown below.

Connection control

- If the MH (mobile host) wants to execute an active open
  - It sends an SYN packet contains some information about the connection including
    - the destination address.
  - if the BS receive the SYN packet
- Sends back an ACK to the MH.
- Assign a unique connection id (CID) to the connection.
- Cache the connection information and use the CID as a key to look for the connection
  - information in future transmission.
  - if the MH receives the ACK for its SYN packet
    - Set a CID for the connection with BS.
    - Cache the connection information locally.
    - Sends back an ACK to the BS.
- if the MH wants to execute a passive open
  - inform the BS that it is ready to accept connection
  - if BS receives a connection request to MH
    - accept the request without communicating with MH

Retransmission & timer management

- if BS wants to send a packet to the MH
  - Sends packets and start a timer. (timer value exceeds RTD)
  - When MH receives a packet
    - If it is in order

            - Send an ACK back to the BS
            - If BS receives the ACK
                - Remove the packet from its buffer.
        - if it is not in order
            - discard the packet
            - sets the Retransmission bit and include the lost packet sequence number in the Acknowledgement field in every ACK BS sends, until it receives the lost packet.
        - if BS receives a retransmission request
            - BS retransmits all packets again starting with the lost one.

- if MH wants to send a packet to the BS
    - Send the packets to the BS.
    - if the BS receive the packets
        - sends back a cumulative ACK
        - If the MH receives the ACK, it removes acknowledged packets from its buffer.
    - if BS receive an out of order packet
        - In only one ACK, BS sets Retransmission bit and include the sequence number of the lost packet in the Acknowledgement field.
        - Start a timer to protect from losing the retransmission request.

BS flow control

- if the occupancy level at the BS buffer reach *high.threshold*
    - BS sets ON/OFF bit to OFF in the next packet or ACK it sends to MH.

o When MH receives a packet with OFF signal, it stops sending.

o MH repeats the same ON/OFF signal in its entire packet to BS.

- ELSE

o BS sets the ON/OFF bit code to ON.

o When MH receives a packet with ON signal, it continues sending.

o MH repeats the same ON/OFF signal in its entire packet to BS.

MH flow control

- MH communicates its buffer occupancy in each packet it sends by setting the BO bits to refer to the level of occupancy in MH buffer.
- Upon receiving the BO information, BS decide to send or stop sending based on many parameters including connection settings and MH buffer size.

METP (Mobile-End Transport Protocol) [7]

In this paper the authors propose three techniques to improve TCP connections in wireless and mobiel environments: to replace TCP/IP protocol over the wireless link with a simpler protocol with smaller headers, shifts functions needed to connunicate with an internet host using TCP/IP from the mobile host to the base station, and finally exploits link-layer acknowledgments and retransmissions to quickly recover losses over the wireless link. Figure 1.10 shows an METP protocol in progress.

Due to the fact that the hop between the mobile host and its base station is either the first or last hope hence only part of the IP functioalty, not all, have to be performed on the base station. Any packet destined for the mobile host is accepted by the base station as it is destined for itself. Then it strips the datagram of its IP header and delivers it to the higher layer. Header check sum is replaced by the link-layer CRC. If the IP datagram is contains a UDP datagram then METP strips the UDP/IP header and delivers it to the mobile host. If IP datagram contain a TCP segment then METP puts it in the receiving vuffer and sends an acknowledgment back to the source. To guarantee reliable and in-order delivery over the wireliess link METP realize the CSMA/CA mechanizem at the link layer. This mechanizem ensure proper retransmission and reliable delivery.

When the mobile host wants to send data it sends the data to the base station. The base station stores the data at the sending buffer and uses the information in the METP heaser to send a regular TCP segment to the receiver.

Since the wireless band width is limited, METP header is made as small as possible to utalize the link more effeciently. This new header is suffecent due to the fact that the link is hidden from the outside internet. The only header information excahnged between the mobile host and the base station is the port and IP numbers for the source and the destination entities. Figure 1.11 shows the header of METP packet. Thus, the METP header size is only 12 bytes compared to 40 bytes for TCP/IP header. The authors uses the idea of header compression by adding 4 bytes of information. The information include the connection ID, sequence number, and some flags. The full header is exchanged during the connection establishment and afterwards only the added 4 bytes are used. An algorithm below discribes the protocol.

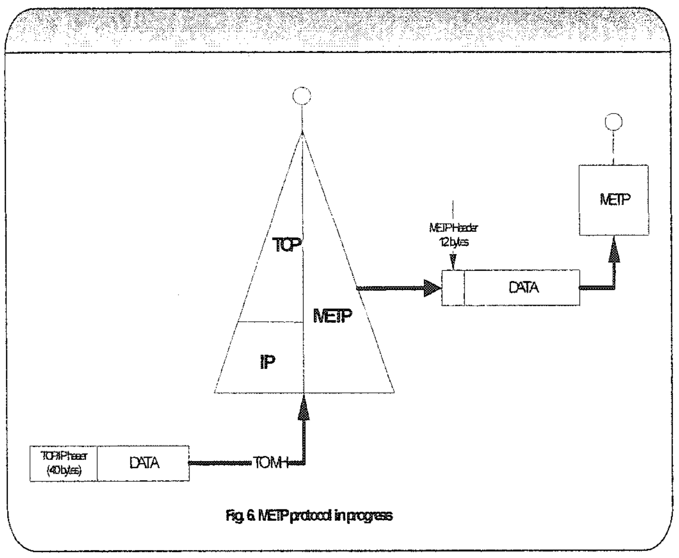


Figure 1.10: METP protocol

- If base station receives a packet destined to the mobile host
  - o If it is a UDP datagram (know at the time of connection establishment)
    - ▪ Strip its UDP/IP header and deliver it to the MH
  - o If it is a TCP segment
    - ▪ METP puts it in the receiving buffer of the connection.
    - ▪ Acknowledge the source.
      - o Using CSMA/CA with priority ACK, METP try to send the segment.
      - o Segment is not freed until the data it contains received by MH.
- If base station receives an METP packet from the MH
  - o If it is a UDP datagram
    - ▪ BS adds UDP/IP header and sent out the datagram.
  - o If it is a TCP segment
    - ▪ Put it in the sending buffer of the connection.
    - ▪ Send it out as a TCP segment to the destination.

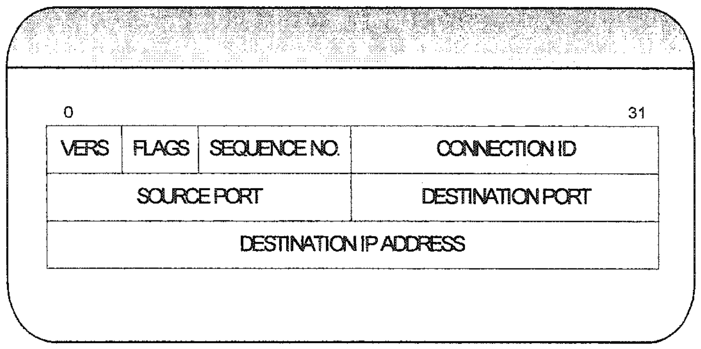


Figure 1.11: Header of METP packet

After the mobile host moves to a new base station and request to register with it, the new base station sends a request to the home base station and notify the old base station of the movement of the mobile host. The old base station opens a TCP connection with the new base station and sends all information about all TCP connections opened on behalf the mobile host. Old connection ID's need to be invalidate and a full METP header need to be used the first time a packet of a connection is sent.

1.6 Dissertation overview and Organization

The dissertation can be divided into four main parts. The first and second parts study, intensively, the behavior of TCP in mobile networks and organize the possible solution of TCP in mobile networks. In addition, it studies the rerouting algorithms proposed in the literature and their interaction relationship with TCP. These studies were conduct in both connection-less and connection-oriented environments. New protocols were developed that details the interaction behavior between all possible mobile TCP scheme suggested by us and the rerouting schemes. Also, Different mobile TCP scheme were tested on top of mobile IP as the network layer protocol. In this area, two of our papers were accepted and published in two different conferences [36, 37].

The third part of the dissertation focuses on improving the performance of TCP in conventional networks such as the Internet where nodes are connected by wireline cables. We studied the competition relationship among many TCP flows sharing the same path and there effects on TCP performance. Also, we discussed the fairness problems among TCP flows is a best-effort networks. We describe a *join* protocol that combines the TCP flows to lessen competition and improve individual TCP performance. Using our protocol we achieve fairness and increase throughput and goodput of TCP flows. The join protocol is an end-to-end protocol and can be extended network wide to overcome congested regions. Part of our work in this area were accepted for publication in ICN04 conference [38] which is technically co-sponsored by IEEE/Comsoc CSIM committee, IEEE/Computer, and by IEE.

Finally, we Introduces the challenge of tracking system complexity in Wi-Fi environment. Then, we developed three tracking protocols to make all nodes in the network position-aware of all other nodes in the topology. We compared there performance using simulation to evaluate the time it takes to propagate tracking information, number of overhead tracking messages exchanged, and resources required to provide connectivity between mobile stations. Our protocols for tracking updates will exploit station locality information to perform controlled flooding, use backward learning, and forward addressing techniques. We will evaluate the protocols and fine tune the protocols taking into account mobility rates and application characteristics.

The rest of this dissertation is organized as follows. Chapters 2 and 3 discuss the categorization of mobile TCP solutions and the interaction between mobile TCP and different rerouting schemes in connection-less and connection-oriented environments, respectively. Analytical models and evaluation results were presented. Chapter 4 explains the competition and fairness problems associated with TCP and explain the *join TCP* protocol. Tracking system at Wi-Fi and their impact on TCP performance were presented in chapter 5. Next, our conclusions and future work are presented on chapter 6.

Chapter 2

Mobile TCP and Rerouting in Connection-less Environment

2.1 Introduction

The performance of wireless Transmission Control Protocol (TCP) schemes are directly tied to the underlying network protocol and the rerouting strategies used after a handoff. Analytical models are proposed for the various interactions and verified by simulation to determine the buffer requirements at base stations (BS) and to estimate disruption time, which is the time between the arrival of the last packet to the mobile receiver while at the cell of the old BS and the arrival of the first packet to the mobile receiver while at the cell of the new BS. This evaluation will aid a protocol designer to pick the right combination of wireless TCP scheme and rerouting strategy based on the network properties including mobility patterns, network topology, and others. Towards this, we provide for the first time protocols for reliable communication that takes into account every combination of wireless TCP schemes and rerouting strategies.

The transmission control protocol (TCP) recognizes the presence of congestion by observing the amount of delay in the acknowledgements it receives. If the delay exceeds a threshold, TCP will assume that the packet is lost due to congestion in the network. In [11], the authors propose to enhance TCP's error recovery mechanism by eliminating the retransmission ambiguity, which will solve the problems caused by spurious timeouts and spurious fast retransmits. In the mobile and wireless environment the delay is not only attributed to the presence of congestion but also due to packet losses and routing delays caused by mobility of one or more of the hosts. Upon recognizing the delay, TCP decreases its throughput to avoid congesting the network. Such decreases due to mistaken congestion leads to unnecessary degradation of system's performance [7]. In the literature, many proposals [1-7] have been made to improve TCP's performance in wireless networks. These proposals provide clever techniques to distinguish delays due to packet losses and delays due to other events. Even if the error is detected, TCP Tahoe treats all different types of errors the same. Many proposals [12] were suggested to improve TCP's error recovery process depending on the type of the error. As an example, one of the most important techniques used to improve the quality of the wireless link is to retransmit lost data at the data link layer. In this paper, we are not trying to improve TCP's error detection nor the TCP's error recovery process; rather we are more interested in designing and evaluating the interaction between protocols at the transport layer and protocols at the network layer.

At the network level, there are many proposals that define the rerouting schemes (routing packets after handoff). In this paper, we provide a fine grain analysis of the interaction between wireless TCP schemes and rerouting techniques. Such a study is important because the wireless TCP schemes and rerouting techniques belong to different layers of the network protocol stack and in an ideal case they should be independent of each other. This study will enable a protocol designer to find the right combination of rerouting and wireless TCP schemes based on different network properties including mobility patterns of users and applications that will be used. We provide analytical models to estimate the buffer size required and disruption time for various proposed wireless TCP schemes and rerouting methods for connectionless networks. These analytical models are verified using extensive simulation using the network simulator ns-2 [11]. We added additional patches to ns-2 to implement different rerouting strategies and wireless TCP schemes. The wireless TCP schemes proposed in the literature have not provided performance factors that take into account the interaction between the wireless TCP schemes and many routing and rerouting schemes [9-10] proposed in the literature.

In a recent paper by Racherla et al [8], a limited study was conducted to evaluate performance of different TCP schemes that take into account different rerouting schemes and focused on the estimation of the sender's throughput. Wang and Tripathi [6] propose a scheme to improve both the transport protocol and the underlying rerouting scheme at the last hop, but it is not intended to study and evaluate different combinations of TCP and rerouting schemes.

The presence of mobility will affect at least two aspects of the communication between either two mobile hosts (MHs) or between fixed end (or fixed host) and a mobile host: buffer requirement at different base stations and mobile hosts, and the disruption time on the uplink and down link sides of the communication at the mobile host (MH). The type of transport and network layer protocols, mobility environment, acknowledgement process and the presence or absence of advance hint process affects both of these aspects during a handoff. The advance hint informs the MH (or base station, or both) about the time and direction of next move.

2.2 Interaction between mobile TCP's and rerouting schemes

In this section, we will show how the type of wireless TCP schemes, rerouting mechanism, and different acknowledgement process affects the length of the disruption time at the mobile receiver and the buffer requirement at old and new base stations. For the first time, we provide protocols required to establish different wireless TCP connections in conjunction with all possible rerouting schemes.

In Figures 2.1-2.3, we detail the steps that take place to complete the seamless wireless TCP communication after the mobile receiver moves from the cell of the old base station to the cell of the new base station. These descriptions explain the steps for all different combinations of TCP and rerouting schemes, with the hint process. When the MH moves from its current position, the TCP connection between the source and the MH cannot continue correctly unless the connection states at the source, base station, and MH are updated reliably to ensure correctness.

The type of rerouting scheme used at the network layer governs the route that the update packets take to reach the source. After successfully transferring all states and update information, the TCP connection opened between the two base stations is no longer needed and can be closed; as a result all resources used up can be freed. Note that the type of TCP scheme determines the source and content of such updates. In the case of Indirect TCP, the source is the new base station, and the update packet contains the new IP and port number. On the other hand, in case of WTCP, it is clear that the MH itself sends such an update packet, which contains only the new IP address, as the port number used is the same.

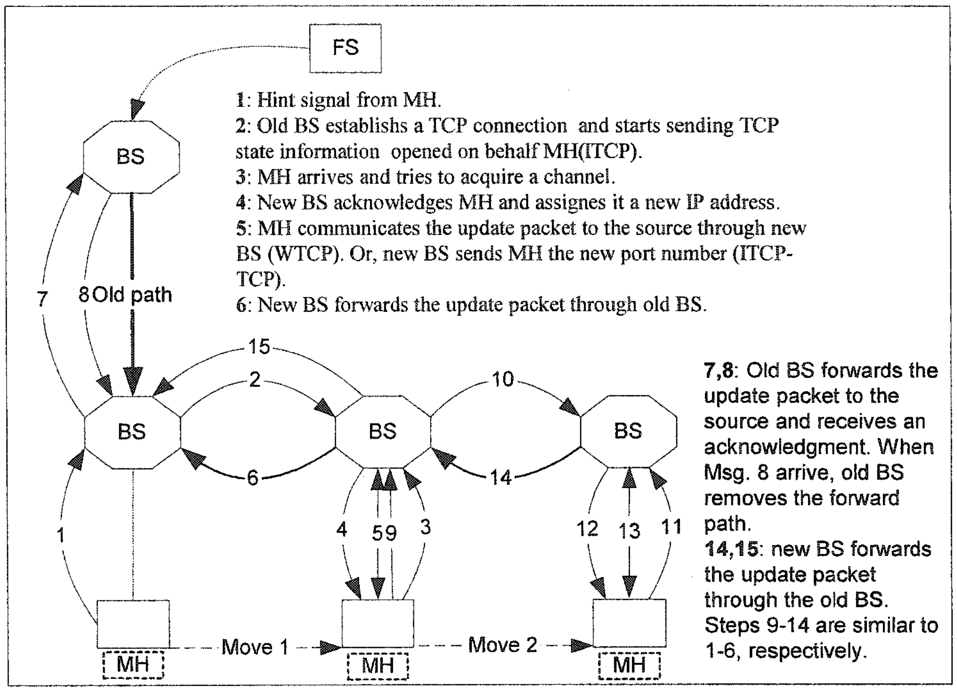
In all rerouting schemes for the ITCP case, we must establish a TCP connection between the last two base stations the MH visited. This is necessary to transfer the state of the TCP connection opened at the old base station on behalf of the MH to the new base station. On the other hand, the path that update packet takes from either the new base station or MH to the source depends on the rerouting scheme used at the network layer. During cell forwarding, the update packet will travel backward through the old base stations until it reaches the source. Then the source will return an acknowledgment to the sender, which will assume that the source has updated its connection state, enabling it to resume normal TCP communication. In the case of full rerouting, the update packet and its acknowledgement will utilize a new path between the new base station and the source. In partial rerouting, the new base station must wait for the crossover discovery algorithm [10] to finish and the subsequent receipt of a notification message. Then the update packet is sent through the crossover node.

For tree rerouting, there are two possible scenarios for updating the TCP connection. The first possibility is to have only one TCP connection between the source and the MH (or the base station). In this case, the new base station includes the updated state information inside the join request message sent to the root of the multicast group. This scenario will offer more bandwidth and greater resource utilization, because there is only one TCP connection (or two for ITCP) established between the two communicating peers. Unfortunately, it will take more time to establish the new TCP connection at the new location, which in turn will increase the disruption time at MH as well as the buffer requirements at base stations. The second scenario involves establishing a TCP connection between the root and every base station that is a member of the broadcast group, and only one base station will transfer packets to the MH.

In this case, the MH can start communication immediately after moving and registering with the new base station. There is no need for state transfer or update packets. Obviously, this method will waste bandwidth along the extra connections; it will also waste resources at the source and all inactive base stations, that is base stations to which MH has not visited.

2.2.1 Cell forwarding interaction protocol

In this section we will explain the detail steps of the cell forwarding and mobile TCP interaction protocol. In this protocol we will show how different mobile TCP schemes perform connection establishment, sending packet, and sending acknowledgment. Further, we will show how different mobile TCP schemes would react to packet drop due to mobility.



FS
BS
1: Hint signal from MH.
2: Old BS establishs a TCP connection and starts sending TCP state information opened on behalf MH(ITCP).
3: MH arrives and tries to acquire a channel.
4: New BS acknowledges MH and assignes it a new IP address.
5: MH communicates the update packet to the source through new BS (WTCP). Or, new BS sends MH the new port number (ITCP-TCP).
6: New BS forwards the update packet through old BS.
7
8 Old path
15
2
10
BS
BS
BS
6
14
7,8: Old BS forwards the update packet to the source and receives an acknowledgment. When Msg. 8 arrive, old BS removes the forward path.
14,15: new BS forwards the update packet through the old BS. Steps 9-14 are similar to 1-6, respectively.
1
4 59 3
12 13 11
Move 1
Move 2
MH
MH
MH

Figure 2.1: Cell forwarding and mobile TCP interaction protocol.

We are assuming a communication between a fix host (FH) sender and a mobile host (MH) receiver. The FH does not move while the MH moves while communicating with the FH. The last hop of the path is a wireless connection between a base station (BS) and the MH while the connection between the FH and the BS can be wireless or wireline. In both cases, we will ignore the error transmission between the FH and the BS.

The communication started when the FH sends a connection establishment request to MH. It will utilize the normal TCP hand shake process as in TCP Reno except for split TCP where the BS intercept the packet and responds to the FH by accepting the connection establishment on behalf of the MH. Then, the BS will establish another connection with the MH.

While the connection between the FH and MH is active, MH keeps moving toward another BS. When the signal strength received from the new BS is stronger than the signal strength received from the old BS, MH inform the old BS about a possible hand off to the new BS by sending a hint packet. This hint packet contains the identity of the new BS.

In case of split TCP, old BS establishes a TCP connection with the new BS immediately after receiving the hint packet from the MH. After successful connection establishment with the new BS, old BS sends the state information of all the TCP connections opened on behalf of the MH to the new BS.

When MH reached the cell of the new BS, it sends a packet requesting a channel and includes its identity information in such a packet. MH waits because it is unable to send any acknowledgements back to the sender until a respond is back from the new BS. The respond acknowledges the request for a channel and include a new IP address to the MH. At this point, the MH is assigned a channel and it knows its new IP.

The FH must be updated about the new address of the MH. In case of WTCP, the MH will communicate its new identity (IP) to the FH by sending an update packet through the new BS, which will forward it to the old BS because the underline rerouting protocol is cell forwarding. The old BS will forward the update message to the FH. If FH receives the update packet correctly, it will acknowledge the old BS. As a result, old BS is confidence that the FH know about the new address of MH and the responsibility is shifted to the new BS. Thus, old BS removes the TCP forward connection with the new BS.

In ITCP-TCP, the new BS already knows the new IP address of the MH, and hence it sends it directly to the FH. The old port number that MH uses while communicating with the old BS might not be available at new BS. Therefore, the new BS communicates an available port number that MH can use to communicate with it.

2.2.2 Full rerouting Interaction protocol

After FH establishes a TCP connection with the MH as explained earlier it will start moving toward the cell of the new BS. When the signal strength of new BS increases enough, the MH sends a hint message to the old BS that contains the identity of the new BS. In case of ITCP-TCP, similar to previous protocol, old BS establishes a forward TCP connection with the new BS in order to transfer the states of all the TCP connections establish on behalf of MH.

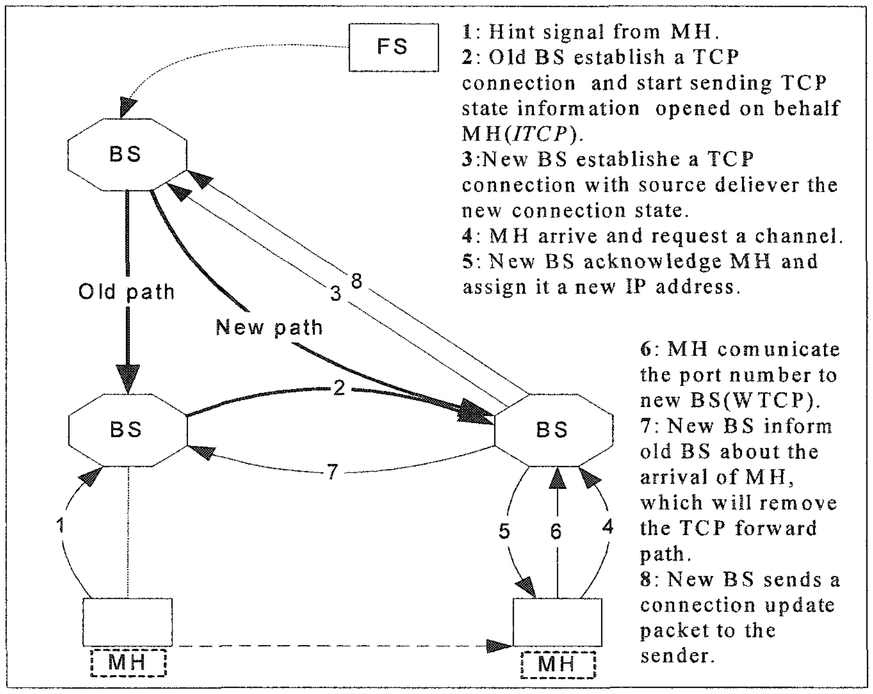

FS
BS
Old path
New path
3
8
2
BS
BS
7
1
5
6
4
MH
MH
1: Hint signal from MH.
2: Old BS establish a TCP connection and start sending TCP state information opened on behalf MH(*ITCP*).
3:New BS establishe a TCP connection with source deliever the new connection state.
4: MH arrive and request a channel.
5: New BS acknowledge MH and assign it a new IP address.
6: MH comunicate the port number to new BS(WTCP).
7: New BS inform old BS about the arrival of MH, which will remove the TCP forward path.
8: New BS sends a connection update packet to the sender.

Figure 2.2: Full rerouting and mobile TCP interaction protocol

In ITCP-TCP, new BS establishes a TCP connection with the FH and sends a packet to inform it about its IP and new port number (if needed to change it). The FH will buffer the new information for future use and will keep communicating with the MH through the old TCP connection.

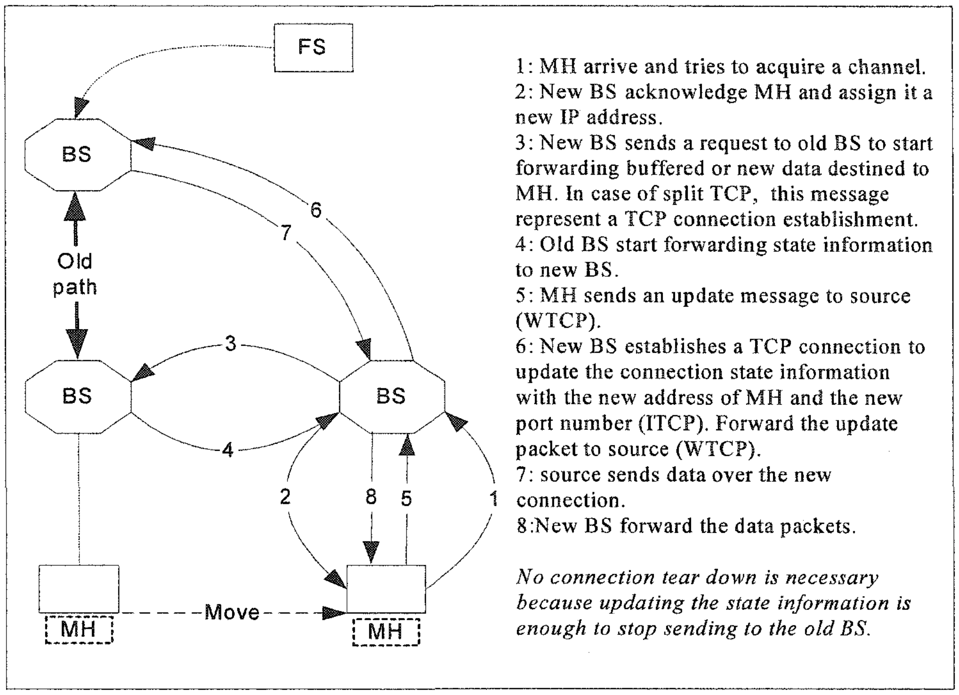
When MH reached the cell of the new BS, it sends a packet requesting a channel and includes its identity information in such a packet. MH waits because it is unable to send any acknowledgements back to the sender until a respond is back from the new BS. The respond acknowledges the request for a channel and include a new IP address to the MH. At this point, the MH is assigned a channel and it knows its new IP.

In WTCP, the MH will communicate the desired port number that new BS can use to communicate with as the new BS knows its new IP. Next, the new BS sends a notification messages to the old BS about the arrival of the MH. As a result, old BS tears down the TCP forward path with the new BS.

The new BS sends an update packet to the FH. In case of ITCP-TCP, the packet only signal the FH to use the new connection information already stored at its local buffer in place of the old connection. In WTCP, the packet includes the new IP address of the MH and the port number to use for communication.

2.2.3 Tree rerouting Interaction protocol

In this protocol, the hint process is not necessary due to the fact that the new BS already receiving a copy of all TCP packets destined to MH. When MH moves to the cell of the new BS, it requests a channel from the BS. If the new BS has enough resources, it will acknowledge MH's request for a channel and assign it a new IP. Immediately, new BS sends a request to the old BS to forward any new or buffered packets destined to MH.



1: MH arrive and tries to acquire a channel.
2: New BS acknowledge MH and assign it a new IP address.
3: New BS sends a request to old BS to start forwarding buffered or new data destined to MH. In case of split TCP, this message represent a TCP connection establishment.
4: Old BS start forwarding state information to new BS.
5: MH sends an update message to source (WTCP).
6: New BS establishes a TCP connection to update the connection state information with the new address of MH and the new port number (ITCP). Forward the update packet to source (WTCP).
7: source sends data over the new connection.
8: New BS forward the data packets.

No connection tear down is necessary because updating the state information is enough to stop sending to the old BS.

Figure 2.3: Tree rerouting and mobile TCP interaction protocol

In case of ITCP-TCP, the new BS established a TCP connection with the old BS to receive the state information for all the sockets open at old BS on behalf of MH. After a successful establishment of the forward TCP connection, the old BS starts sending the state information.

When the MH moves to the new BS and changing its IP address, it needs to inform the FH. In WTCP, the MH sends an update packet through the new BS. The new BS establish a TCP connection with the FH to update the connection state information with the new address of MH and the new port number in case of ITCP-TCP or to forward the update packet to FH in case of WTCP. After receiving the update information, the FH start sending new data destined to the MH through the newly established connection.

2.3 Evaluation metrics

Depending on the direction of the communication, there are two types of disruption times. The first type is the *downlink disruption time* (DT_D), which is the time between the arrival of the last packet to the mobile receiver while at the cell of the old BS and the arrival of the first packet to the mobile receiver while at the cell of the new BS. The length of this disruption time affects the throughput of the TCP connection due to the fact that the MH will delay sending acknowledgments until it receives the first packet at the new location. In order to receive data, the MH needs to acquire a channel from the new BS and then establish all necessary connections.

The second disruption time is the *uplink disruption time* DT-U, which is defined as the time between sending the last acknowledgment by the mobile host while at the cell of the old BS and sending the first acknowledgment while at the cell of the new base station. The length of such disruption time also clearly affects the throughput of the TCP connection because the sender uses a window mechanism to control the data flow with the receiver. Hence, the sender's ability to send new data depends on the arrival of acknowledgments from the destination.

The uplink disruption time has an extra requirement over the downlink disruption time. This extra requirement is the dependence on the timing of the arrival of the first packet to the mobile host at the new cell. The measurement of disruption time enables us to determine the length of time interval it will take the mobile receiver to start receiving data, as it is the pause time the mobile user will experience in real applications. Calculating this type of disruption time will also provide more accurate measurement of the time it will take the sender to receive the first acknowledgement after a handoff. The importance of DT-U is that the chances of the sender entering a congestion avoidance period lower when the DT-U is shorter.

We calculated the downlink buffer required at the old base stations for all different cases of rerouting and TCP schemes. When the old BS uses immediate acknowledgement, the buffer required to support the handoff process will depend on the length of the time between when the MH moves from the cell of the old BS until the old BS receives the forward request from the new BS. Despite whether the MH is a sender or receiver, the old BS does not need to buffer any data on the uplink side, because either the old path or the forward path is available for transmission. On the other hand, the buffer requirement at the new BS and MH can be ignored since the MH is only sending acknowledgements and losing one acknowledgment can be recovered by the next one. In future work we are considering duplex communication between the two end points. At the end, the simulation results show the throughput of the TCP sender for all different cases.

Our main goal is to present the interaction protocols between all different wireless TCP schemes and rerouting schemes. We present our results to help protocol designers to choose the best combination that fits their need. It is beyond the scope of the paper to achieve optimal performance of any of the metrics mentioned in this section. The design of cooperate protocols at the TCP level and at the network level is still an open for more improvement.

2.4 Analytical model and simulation

In this section, we present our analytical model and derive equations for various metrics introduced in section 2.4 as the mobile host moves from the old to the new cell. The calculations used in the analytical models assume that the sender always has data to be sent. The notations used in our analytical models are presented below in table 2.3.

$N_{X\text{-}Y}$	Number of nodes on the path between the nodes X and Y.
B_X	The buffer space required at node X.
S_{ctrl}	Size of a control packet.
S_{datal}	Size of a data packet.
$T_{chk\text{-}RT}$	Time needed to check a nodes' *routing* table in order to find the crossover point.
$T_{DT\text{-}U}$	Disruption time at mobile host on the uplink side.
$T_{DT\text{-}D\text{-}X}$	Disruption time at mobile host on the downlink side through X path: X=Nor F, (N=new, F=forward).
$T_{TCP\text{-}X\text{-}Y}\,(Z)$	The total time needed to establish a TCP connection between nodes X and Y. If Z is present in the notation, it means "while connecting to Z".
$T_{TCP\text{-}MH\text{-}Nbs}$	$3\left(\frac{Sctrl}{BW_{wl}} + L_{wl}\right)$
$T_{TCP\text{-}oBS\text{-}nBS}$	$3\left(\left(\frac{S_{ctrl}}{BW_{w}} + L_{w}\right)\times N_{oBS-nBS}\right)$
$T_{TCP\text{-}MH\text{-}S}(nBS)$	$3\left(\frac{S_{ctrl}}{BW_{wl}} + L_{wl} + \left(\frac{S_{ctrl}}{BW_{w}} + L_{w}\right)\times N_{nBS-src}\right)$
$T_{TCP\text{-}MH\text{-}S}\,(oBS)$	$3\left[\frac{S_{ctrl}}{BW_{wl}} + \left(\frac{S_{ctrl}}{BW_{w}}\right)\times N_{oBS-src}\right]$

T₁[11]	$T_{acqr} + \frac{Sctrl}{BWwl} + L_{wl}$
T₂[11]	$\frac{Sctrl}{BWwl} + L_{wl}$
T₃=T₄[11]	$\left(\frac{Sctrl}{BWw} + L_w\right)N_{oBS-nBS} = \left(\frac{Sctrl}{BWw} + L_w\right)N_{nBS-oBS}$

Table 2.3: Notations used in the analytical model

As far as the new BS is concerned, it is obvious that there is no buffer overhead related to handoff either on the up or down links. The reason is that it will start receiving data destined to the MH only when the communication channel with the MH is established. The only exception is in the case of tree rerouting for which we will calculate such a buffer requirement when we present the evaluation of tree rerouting.

In order to calculate the buffer requirements we will follow the steps in Figures 2.1-2.13. The maximum buffer requirement at the old base station is the total size of all packets arriving at the old base station after the MH moves until the forward path is established. There is no buffer requirement at the old BS in case of full rerouting and cell forwarding because the forward path is established before the MH leaves its old cell. In the case of tree rerouting, the old base station will keep buffering packets until it receives a request from new BS to remove itself from the multicast group (message 6). This buffer requirement is shown in (1).

$$B_{BSold} = \left(T_1 + \frac{S_{ctrl}}{BW_{wl}} + T_3\right) \times BW_w \qquad (1)$$

The length of DT_D-N, during full rerouting and ITCP, equals the sum of the lengths of messages 4, 5, 7, 8, and the time it takes the first packet to arrive at the MH and is shown in equation (2). For WTCP, we added message 6 ($\frac{S_{ctrl}}{BW_{wl}}+L_{wl}$) as it is part of establishing the new connection and is shown in (3). Following the same procedure, equation (4) shows the disruption time in the case of cell forwarding when the transport layer uses the WTCP scheme. For cell forwarding combined with ITCP, the update packet is sent by the new BS and not by the MH. Therefore, message 5 will not be included as shown in (5). For tree rerouting, the only difference between the case of WTCP and ITCP is message 3. Message 3 is merely a request to join the multicast group at the network level in the case of WTCP, as presented in equation (6), whereas for ITCP it represent TCP connection establishment and shown in equation (7).

$$\begin{aligned} DT\_D(FR-ITCP) &= T_{TCP-oBS-nBS}+T_1+\frac{S_{ctrl}}{BW_{wl}}+ \\ &\left(\frac{S_{ctrl}}{BW_w}+L_w\right)\times\left(N_{nBS-oBS}+N_{nBS-src}\right)+\frac{S_{data}}{BW_w}+L_w \end{aligned} \tag{2}$$

$$\begin{aligned} DT\_D(FR-WTCP) &= T_{TCP-oBS-nBS}+T_1+\frac{S_{ctrl}}{BW_{wl}}+\left(\frac{S_{ctrl}}{BW_w}+L_w\right) \\ &\times\left(N_{nBS-oBS}+N_{nBS-src}\right)+\frac{S_{data}}{BW_w}+L_w+\frac{S_{ctrl}}{BW_{wl}}+L_{wl} \end{aligned} \tag{3}$$

$$\begin{aligned} DT\_D(CF-WTCP) &= T_1+2T_2+\left(N_{nBS-oBS}\right)\times\left(\frac{S_{data}}{BW_w}+L_w\right)+ \\ &\left(\frac{S_{data}}{BW_w}+L_w\right)\left(N_{src-oBS}+N_{oldBS-nBS}\right)+\frac{S_{data}}{BW_{wl}}+L_{wl} \end{aligned} \tag{4}$$

$$\begin{aligned} DT\_D(CF-ITCP) &= T_1+2T_2+\left(N_{nBS-oBS}\right)\times\left(\frac{S_{data}}{BW_w}+L_w\right)+ \\ &\left(\frac{S_{data}}{BW_w}+L_w\right)\left(N_{src-oBS}+N_{oldBS-nBS}\right) \end{aligned} \tag{5}$$

$$DT\_D(TR-WTCP) = T_1+\frac{S_{ctrl}}{BW_{wl}}+\left(\frac{S_{ctrl}}{BW_w}+L_w\right)N_{oBS-src} \tag{6}$$

$$DT\_T(TR-ITCP) = T_1+\frac{S_{ctrl}}{BW_{wl}}+T_{TCP-oBS-src} \tag{7}$$

The length of DT_D-F depends on the location of the new packet that is sent to the MH. The packet can either be located at the local buffer of the old BS, or at the sender's buffer, or in between. We based our calculation on the first assumption, because the connection is active before the MH moves, and most likely, packets will arrive at the old

BS before the MH finishes registration with the new BS. The length of this disruption time is the same in the case of full rerouting and cell forwarding and in the case of tree rerouting this disruption time is zero.

We used network simulator *ns-2* to conduct our experiments. We added additional patches to the simulator that includes modification of the routing module to include the rerouting schemes and creation of the Indirect TCP agent.

In all the Figures generated using the analytical equations below, we set the wireline bandwidth to 1Gbps, wireless bandwidth to 11 Mbps, and the time to acquire a channel to 50ms. Even though, 11 Mbps is higher than the current wireless technology bandwidth but we expect substantial increase in wireless networks bandwidth. Hence, we are interested more in evaluating the impact of the different combination protocols on future networks rather than current one. Packets exchanged during the communication process are of two types: control and data packets. For all the analytical equations and their corresponding Figures the size of the control packet and data packet was set to 48 and 65536 bytes, respectively. Figure 2.5 shows that the longest disruption time experienced by the MH is in the case of full rerouting as per the analytical equations. Disruption time was fixed in the case of tree rerouting. The buffer requirement based on the analytical equation is presented in Figure 2.4.

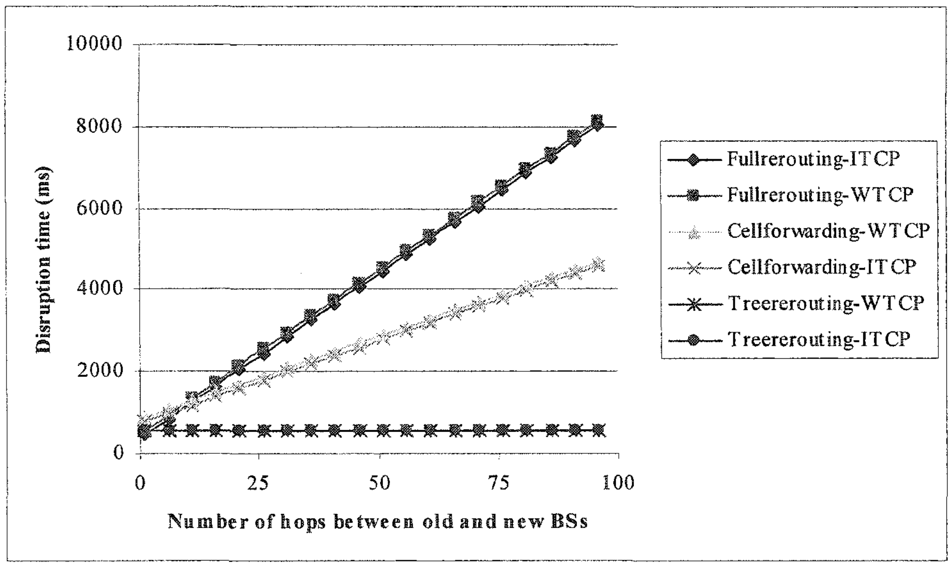


Figure 2.4: Disruption time evaluation.

Simulation results are based on a mesh network configuration. The MH moves once between old and new base stations and the file transfer protocol (*ftp*) was used as the senders application. The packet size of the *ftp* application is 1000 bytes. At a predefined time, the MH will hand off to the new BS and rerouting scheme will be executed. Figure 2.6 presents the buffer requirement at old BS in all different combinations of wireless TCP schemes and rerouting methods. It is clear that the size of the buffer needed at old BS is increasing with the increase of the number of hops at the forwarding path.

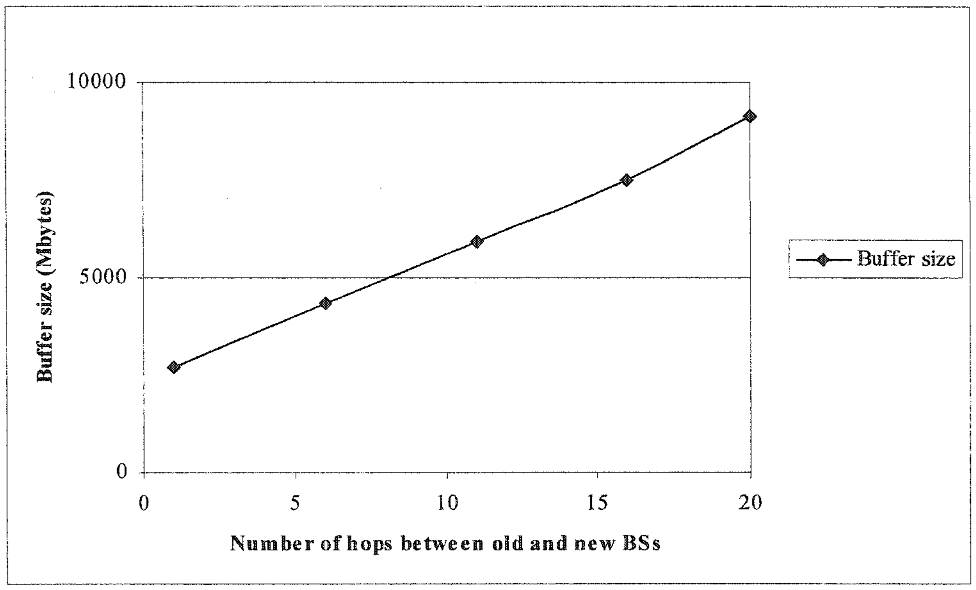


Figure 2.5: Buffer requirement at old BS.

The Figure shows one line that represents three very close lines. It suggests that the buffer requirements are very close for all three cases. The buffer requirements shown in the Figure present an extremely large buffer requirement in the magnitude of Gigabits. The reason is that the values of wireline and wireless bandwidths plugged into the analytical equations were very high.

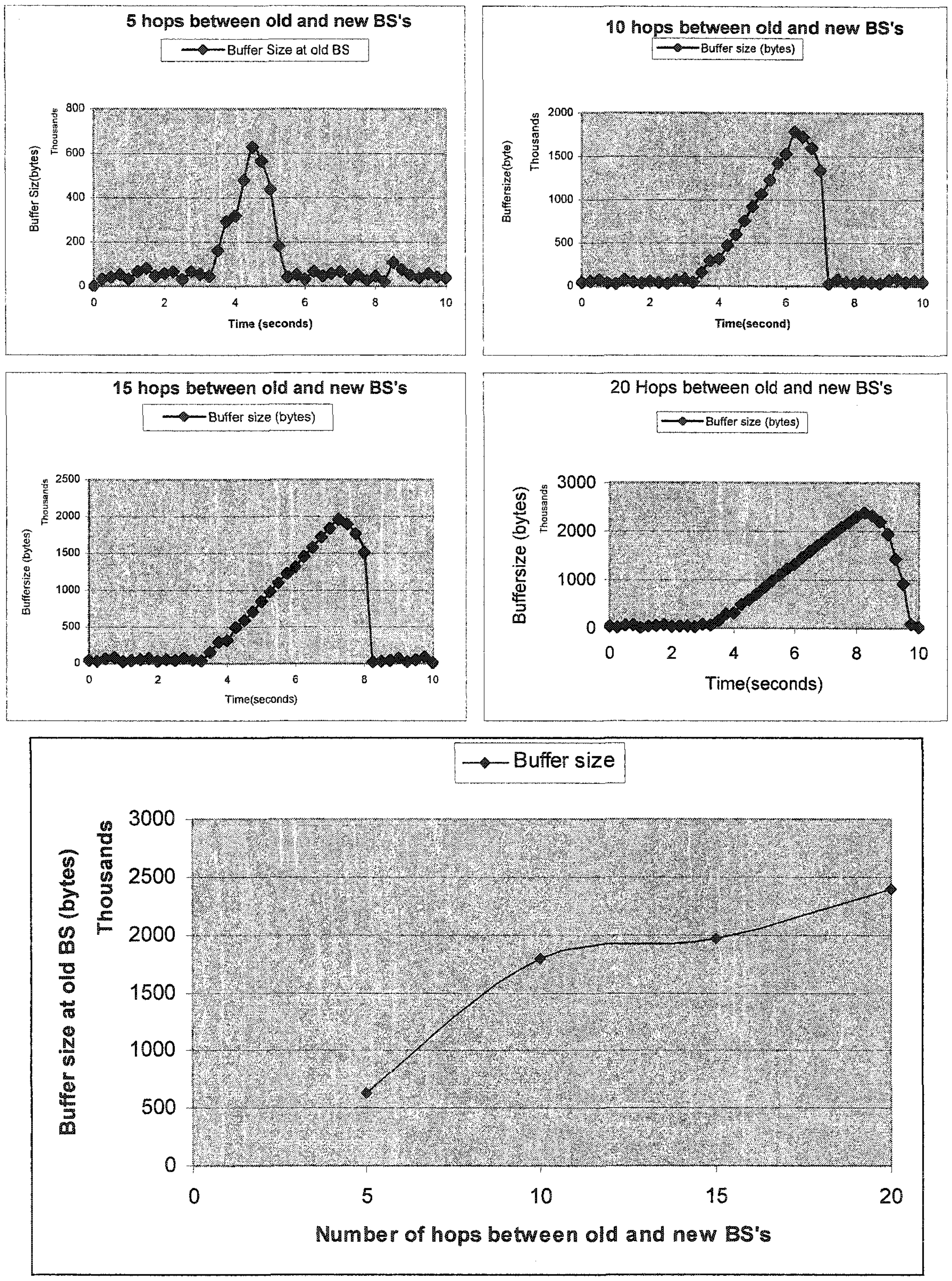


Figure 2.6: Simulation results for Buffer requirement at old BS for cell forwarding with different values of $N_{oBS\text{-}nBS}$.

We like to point out that in the future when the underlying network bandwidths are very high the buffer requirements at the base stations will be enormous. In Figure 2.6, we simulated the interaction protocol of cell forwarding with ITCP-TCP scheme. We ran the simulator for 4 different values of number of hops between old and new BS's.

We chose values for effective bandwidth of 5 Mbps between wired nodes, including the base stations. The maximum buffer requirement for each run is used to plot the last chart in Figure 2.6 to show the impact of number of hops between old and new BS's on the buffer requirement at old BS using the simulation.

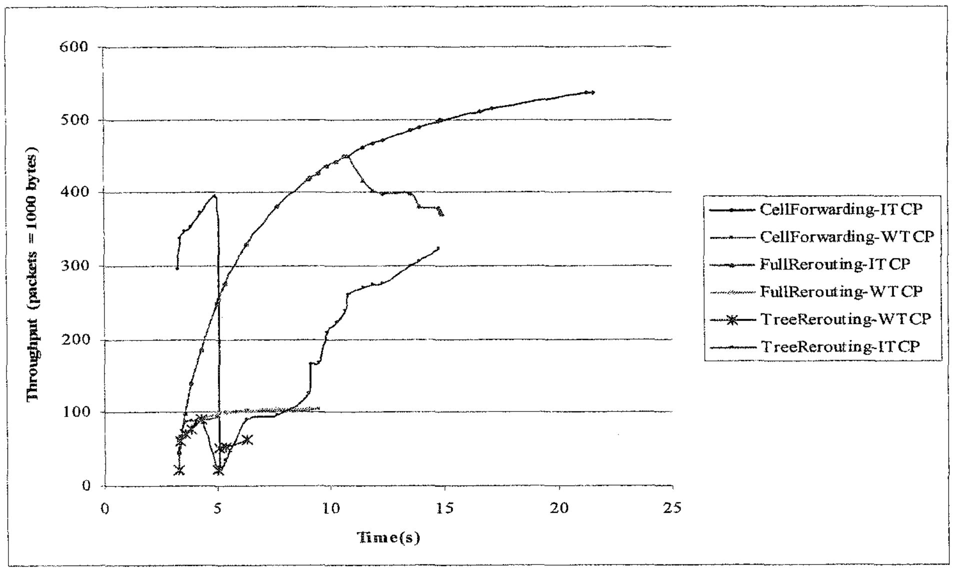


Figure 2.7: Sender's throughput.

The throughput of the sender is the number of packets sent per unit time. Figure 2.7 shows the throughput of the sender for all cases of combination of transport and rerouting procedures. For the ITCP schemes we use the immediate acknowledgment scheme (ITCP-I). From Figure 2.7, it can be noticed that cell forwarding with ITCP-I and full rerouting with ITCP-I have similar performance up to a particular time in the simulation after which the performance of full rerouting with ITCP-I degrades. It is clear that ITCP-I achieves a better throughput than WTCP on top of every rerouting scheme.

2.5 Case study: Mobile IP and mobile TCP schemes

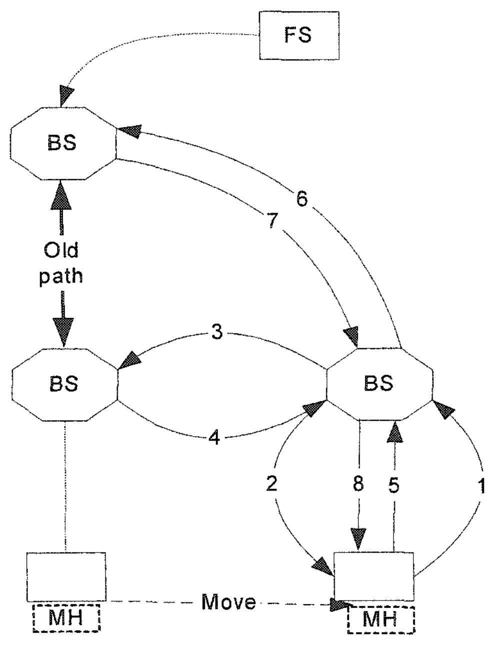
Mobile IP protocol [13] consists of three procedures: Registering the COA of the MH with the home agent (binding the permanent IP address of the MH with the current COA at the HA by adding one entry in the HA binding table), encapsulating/tunneling, and decapsulation. The movement of MH initiates the registration process. After reaching the new FA, the MH will request a channel to start communication. The foreign agent at the new location will make sure it is not the home agent of the MH before assigning a COA to the MH. Afterward, the foreign agent will communicate the COA address to the home agent who will use it to tunnel any packet addressed to the permanent address of the MH by adding a new IP header to the packet and replacing the destination address with the COA of the MH. This tunneling will deliver the packets to the foreign agent who will decapsulate them and delivers the inner datagram to the mobile host. The above scenario is very similar to the general case of full rerouting where the old route used by the packets before the handoff is completely independent of the new route the packet uses after handoff.

A MH is permitted to have simultaneous bindings at the home agent. This means that the home agent will have more than one entry in its binding table for the MH's permanent address. If this is the case, the home agent makes multiple copies of packets destined for the mobile host, and tunnels a copy to each care-of address and, hence, mobile IP fits under the general case of tree rerouting.

2.5.1 Analytical model

The downlink disruption time (DT_D) is the time between the arrival of the last packet to the mobile receiver while at the cell of the old FA and the arrival of the first packet to the mobile receiver while at the cell of the new FA. The length of this disruption time affects the throughput of the TCP connection due to the fact that the MH will delay sending acknowledgments until it receives the first packet at the new location. In order to be able to receive data, the MH needs to acquire a channel from the new FA and then establish all necessary transport layer connections.

We define disruption time differently from [9,10], because we are more interested in assessing how long it will take the mobile receiver to start receiving data, as it more accurately measures the pause time the mobile user will experience in real applications. Calculating this type of disruption time will provide more accurate measurement of the time it will take the sender to receive the first acknowledgement after handoff.



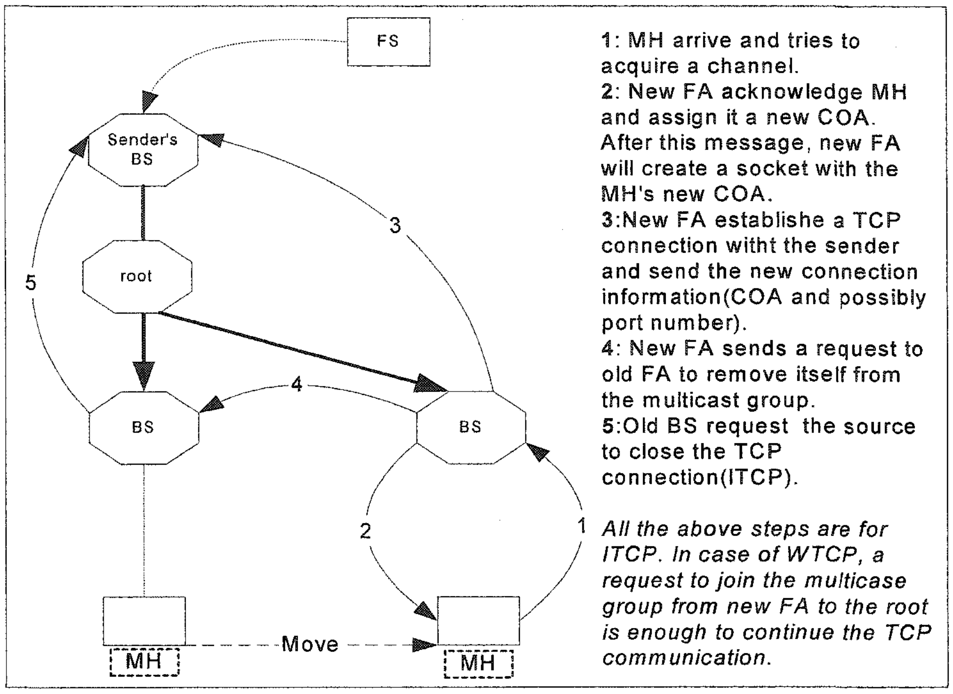
1: MH arrive and tries to acquire a channel.
2: New FA acknowledge MH and assign it a new COA.
3: New FA sends a request to old FA to start forwarding buffered or new data destined to MH. In case of split TCP, this message represent a TCP connection establishment.
4: Old FA start forwarding state information to new FA.
5: MH sends an update message to source (WTCP).
6: New FA establishs a TCP connection to update the connection state information with the new COA and the new port number (ITCP). Forward the update packet to source (WTCP).
7: source sends data over the new connection.
8:New FA forward the data packets.

No connection tear down is necessary because updating the state information is enough to stop sending to the old BS.

Figure 2.8: Different TCP schemes on top of mobile IP, without simultaneous bindings.

We calculated the downlink buffer required at the old FA for both cases of mobile IP under different TCP schemes. When the old FA uses immediate acknowledging the buffer required to support the hand-off process depends on the length of the time from when the MH moves from the cell of the old FA until the old FA receives the forward request from the new FA. Despite whether the MH is a sender or receiver, the old FA does not need to buffer any data on the uplink side, because either the old path or the forward path is available for transmission. On the other hand, the buffer requirement at the new FA and MH can be ignored since the MH is only sending acknowledgements and losing one acknowledgment can be recovered by the next one.

The MH will not be able to receive any TCP packets through the new FA until it finishes registration and the forward or new paths are established. The length of disruption time at MH, through the old FA depends on the type of the forward connection that needs to be established. In case of WTCP, message 3 represents sending a control



1: MH arrive and tries to acquire a channel.
2: New FA acknowledge MH and assign it a new COA. After this message, new FA will create a socket with the MH's new COA.
3:New FA establishe a TCP connection witht the sender and send the new connection information(COA and possibly port number).
4: New FA sends a request to old FA to remove itself from the multicast group.
5:Old BS request the source to close the TCP connection(ITCP).

All the above steps are for ITCP. In case of WTCP, a request to join the multicase group from new FA to the root is enough to continue the TCP communication.

Figure 2.9: Different TCP schemes on top of mobile IP, with simultaneous bindings.

packet at the network level which requires no acknowledgment while in case of ITCP, it represents a TCP connection establishment. Equations (8) and (9) show this disruption time in case of WTCP and ITCP, respectively.

$$T_{DT-D-F} = T1 + \left(\tfrac{S_{ctrl}}{BW_{wl}} + L_w\right) N_{nBS-oBS} + \left(\tfrac{S_{data}}{BW_w} + L_w\right)_{NoBS-nBS} + \tfrac{S_{data}}{BW_{wl}} + L_{wl} \qquad (8)$$

$$T_{DT-D-F} = T1 + T_{tcp-oBS-nBS} + \left(\tfrac{S_{data}}{BW_w} + L_w\right) N_{oBS-nBS} + \tfrac{S_{data}}{BW_{wl}} + L_{wl} \qquad (9)$$

The time it will take the first packet to arrive to MH through the newly established path, T_{DT-D-N}, depends on the underline network protocol and the type of transport protocol in use. We show the disruption time in case of WTCP and ITCP on top of single binding mobile IP in (10). In (11) we show the case of ITCP on top of both cases of mobile IP. In (12) we present the length of disruption time in case of simultaneous binding with WTCP.

$$\begin{aligned} T_{DT-D\_N} &= T_1 + \tfrac{S_{ctrl}}{BW_{wl}} + T_{TCP-oBS-src} + \\ &\left(\tfrac{S_{data}}{BW_w} + L_w\right) N_{src-oBS} + \tfrac{S_{data}}{BW_{wl}} + L_{wl} \end{aligned} \qquad (10)$$

$$\begin{aligned} T_{DT-D\_N} &= T_1 + \tfrac{S_{ctrl}}{BW_{wl}} + T_{TCP-nBS-oBS} + T_3 + \\ &T_{TCP-oBS-src} + \left(\tfrac{S_{data}}{BW_w} + L_w\right) N_{src-oBS} + \tfrac{S_{data}}{BW_{wl}} + L_{wl} \end{aligned} \qquad (11)$$

$$\begin{aligned} T_{DT-D\_N} &= T_1 + \tfrac{S_{ctrl}}{BW_{wl}} + \\ &\left(\tfrac{S_{data} + S_{ctrl}}{BW_w} + 2L_w\right) N_{oBS-src} \end{aligned} \qquad (12)$$

The buffer requirement at the old FA for all both cases of mobile IP depends on the total sum of the length of the time it takes the MH to register with the new FA and the time it takes the new FA to build the forward link. The two mobile IP schemes will vary in the way they build the forward path. These variations will lead to different time requirements for building the forward path, and ultimately will require different buffer sizes at the old FA. In both two cases, the old FA will stop buffering when it receives either the request to forward message or the request to leave the multicast group message from the new FA. In the case of single binding, the message from the new FA will be a request to establish a forward path. On the other hand, it will represent a request to leave the multicast group. In the first case, the note message will arrive after the MH finishes registration with new FA and the request message travels from new to old FA. In the other case, the request to leave message sent to the old FA must wait until the new path is established and the new FA become a new member of the multicast group (messages 1-4 in Figure 2.9). In (6) and (7) we showed the size of buffering at old FA in the cases of single and simultaneous binding.

$$B_{oBS} = \left(T1 + \tfrac{S_{ctrl}}{BW_{wl}} + T_{tcp-oBS-nBS}\right) \times BW_{w} \qquad (6)$$

$$B_{oBS} = \begin{pmatrix} T1 + \tfrac{S_{ctrl}}{BW_{wl}} + T_{tcp-nBS-src} \\ + \tfrac{S_{ctrl}}{BW_{w}} \times N_{nBS-oBS} \end{pmatrix} \times BW_{w} \qquad (7)$$

2.6 Summary

In this chapter, we explain the interaction relationship between wireless TCP protocols and rerouting schemes. In addition, we presented comprehensive analytical methods to determine the buffer requirements at base stations and to estimate the disruption time at the MH. Also, we studied the interaction relationship between mobile TCP schemes and mobile IP and presented analytical results.

Future work involves investigating the impact of data link layer on the performance of wireless TCP and rerouting schemes. Also, the communication between two mobile peers will introduce new challenges and need to be study future.

Chapter 3

Interaction between Mobile TCP and Rerouting schemes in Connection-Oriented Networks

3.1 Introduction

In this chapter we studied and proposed protocols for the interaction relationship between all possible mobile TCP protocols and different rerouting schemes in a connection oriented network environment. In a connection oriented network, a network channel must be established and resources must be reserved before any data exchange can take place. We proposed protocols for such interaction and built an analytical model to evaluate different metrics of importance including the buffer requirements at old and new BS's and the disruption time experienced by MH caused by mobility.

In our protocols, we considered two cases: with hint and without hint where the hint process was defined in chapter 2. With-hint means that MH is able to predict the direction of its future move by using different techniques to measure the strength of signals received from different base stations and in the absence of such a signal it is the without-hint case wherein the MH establishes connection with new BS after it disconnects suddenly from old BS and enters the cell of new BS. In connection-oriented networks, partial rerouting scheme is distinguished from full rerouting and hence it will be considered as the fourth rerouting scheme. For all protocols, we assume that there is a control channel between old and new BS's and it is used to send control messages and not data. The control messages include:

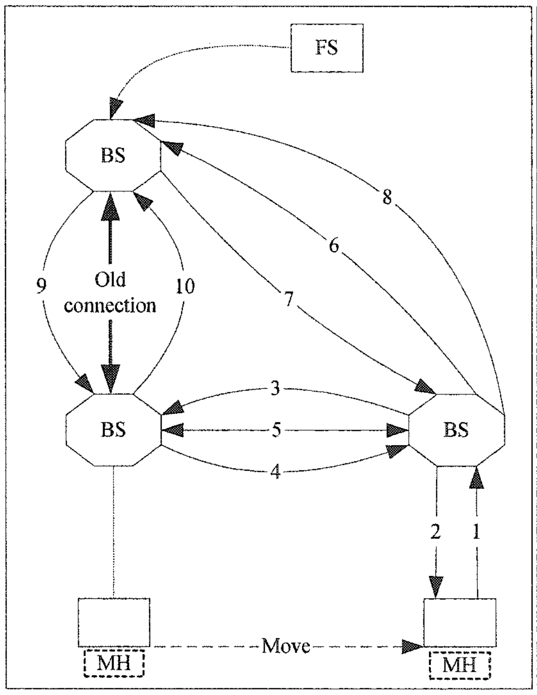
- $M_{O\text{-}N}$: A message from old BS to new BS about possible future movement of MH (in case of "with hint").
- $M_{N\text{-}O}$: A message from new BS to old BS about the arrival of MH (in case of "without hint").
- M_{tear}: A message from new BS to inform old BS about successful establishment of new path and a request to tear down old network channel.
- $M_{group\text{-}leave}$: A message from new BS to old BS requesting the later to remove it self from the multicast group.

For sending data reliably between old and new BS's, we need to establish a network channel and a transport connection. In case of Tree Rerouting, we do not need to establish forward connection (connection between old and new BS) at any layer because we need only to send one control message, which is either notification of future arrival (in case "with-hint") or notification message about arrival (in case of "without-hint").

3.2 Interaction between mobile TCP's and rerouting schemes

3.2.1 Mobile TCP and Full rerouting

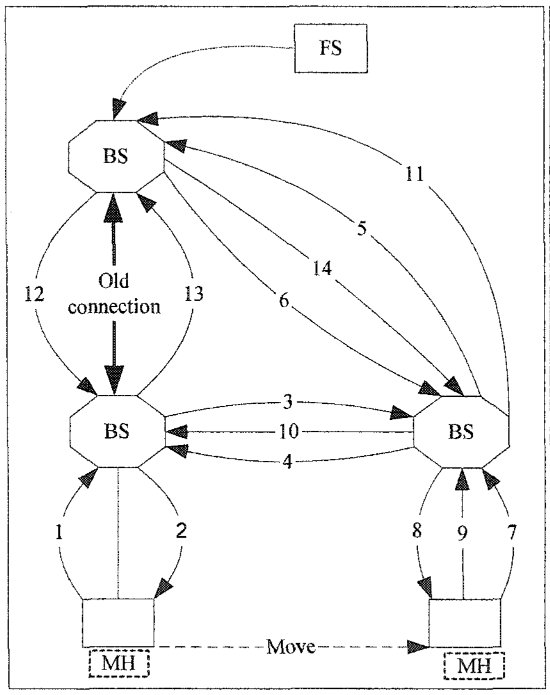
In this protocol, we propose detailed steps of the interaction protocol between mobile TCP protocols and full rerouting. First, we are going to detail the protocol steps to complete the interaction process between mobile TCP and full rerouting in case of no-hint and then we will explain the case when hint process is used.



1: MH arrives and tries to acquire a channel.
2: New BS acknowledge MH.
3: New BS sends a request to old BS to start forwarding buffered or new data destined to MH (this message include request to setup channel).
4: Old BS acknowledg the request and may start forwarding.
5: In case of ITCP, 3-way handshake process to establish a TCP connection between old and new BS's.
6: Set up a connection channel with sourece
7: Establish the connection channel.
8: New BS sends a request to the sender to redirect all data to the new path(s) and tear down the old one.
9 and 10: source tears the old connection down.

Figure 3.1: Interaction protocol between Mobile TCP and Full Rerouting – without hint

While MH is moving toward the cell of new BS, it suddenly looses communication with old BS. Next, MH tries to connect with new BS by sending a registration request shown in Figure 3.1 as step 1. New BS responds to the request by assigning MH a channel that enables MH to communicate again. During this process, old BS could receive packet(s) destined to MH. In case of WTCP, old BS can drop the packet(s) and the source has to recover those dropped packets by timing out and resending them again. This relieves old BS from any extra buffering or processing but causes degradation in TCP sending rate.



1: MH sends a hint signal to old BS.
2: In case of ITCP, old BS sends sockets states information opened on behalf of MH.
3 and 4: Old BS requests to setup a netowrk channel with new BS and new BS establishes a connection channel.
5: Setup a network channel with sourece.
6: Establish the connection channel.
7: MH arrives and request a channel.
8: New BS assign MH a channel.
9: MH trnasmit a copy of sockets states information to new BS.
10: New BS request that packet forwarding start.
11: New BS sends a request to the sender to redirect all data to the new path and tear down the old one.
12 and 13: Tear down the old and forward connections .
14: Source notify new BS about successful tear down.

Figure 3.2: Interaction protocol between Mobile TCP and Full Rerouting– with hint

The buffer size needed at old BS to support handoff increases and decrease with the increase and decrease of two factors. The first is the length of the handoff process and the other is the level of packet arrival rate at old BS. When the handoff process finished, old BS can forward arriving packets (destined to MH) to new BS. We suggest a method that falls in between, where old BS buffers the packets for some time hoping that the handoff process will finish soon so old BS can forward the packets to new BS. If the hand off process takes longer time than expected, old BS will drop incoming packets. Next, new BS sends a request to old BS to start forwarding buffered or new data destined to MH. This step includes a request to setup a network channel with old BS. After receiving the request, old BS acknowledges the request which also creates a network channel, and may start forwarding.

Only in case of ITCP, new BS started a 3-way handshake process to establish a TCP connection with old BS to transfer socket state reliably as we explained above. Next, the new BS sets up a network channel with the source and the source establishes the network channel as shown in steps 6 and 7 in Figure 3.1.

After establishing the connection with old BS and after packets start arriving, new BS sends a request to source to redirect all packets destined to MH to the new path and tear down the old connection. The order in which connections were established above maintains in-order arrival of packets. Packets arriving from old BS reached first and then packets from the source reached second. Steps 10 and 11 take care of tearing down the old network channel opened between source and old BS to save network resources.

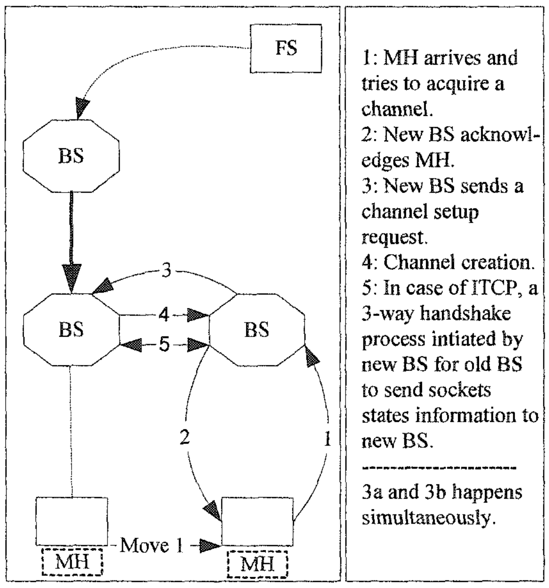
Figure 3.2, shows a representation of the interaction protocol between different mobile TCP protocols and full rerouting in case when the MH is capable of using the hint process. When MH moves close to the cell of new BS, the strength of the signals received from new BS is stronger than the signals received from old BS. The MH then sends a hint signal to old BS about possible move to another cell and includes the address of new BS in the hint signal. After receiving the hint signal, old BS establish a network connection with new BS as shown in steps 2 and 3 in Figure 3.2. In case of ITCP, old BS already created sockets locally to support communication with MH and therefore, when MH moves to the cell of new BS, MH needs to maintain up to date socket(s) state(s) information for any connection opened at old BS on behalf of MH. A reliable connection between old and new BS is required to transfer the socket(s) state(s) information reliably. Also, using this TCP connection, old BS is required to send a copy of each packet sent to MH to new BS.

Steps 5 and 6 represent a network channel setup between new BS and source initiated by new BS. New BS knows that MH is going to move to its cell in the near future. So, in order to reduce the disruption time experienced by MH, new BS establishes a network channel with the source prior to the arrival of MH as shown in steps 5 and 6 but the source will not send any data to new BS until it receives a signal message announcing the arrival of MH to the cell of new BS. The connection pre-establishment between new BS and source reduces the disruption time at MH but at the same time it wastes network resources because it is reserved and not used until MH arrives.

Next, MH arrives at the cell of new BS and requests a channel with new BS. If resources are available then the new BS acknowledges the request and assigns MH a channel as represented by step 7 and 8 in the Figure 3.2. Next the new BS sends a request to the source to redirect all sent data to the new path which was established in steps 5 and 6, and tears down the one established with old BS. The source responds to the request and tears down the old connection. Immediately after that the source informs the new BS about the successful connection tear down, which in turn signals new BS to remove the TCP and network connection established with old BS. These actions will free network resources at each BS's and on the path between the two BS's (since such a path can be multi-hop path).

3.2.2 Mobile TCP and Cell forwarding

In this section we explain in detail the steps of the interaction protocol between mobile TCP and cell forwarding scheme. Figure 3.3 represents the interaction protocol when MH does not use the hint process. The protocol starts when MH arrives at the cell of new BS and sends a request to new BS to acquire a channel. If new BS has enough resources it will acknowledge MH's request and assign it a channel. The request and the acknowledgment messages are shown in Figure 3.3 as messages 1 and 2.



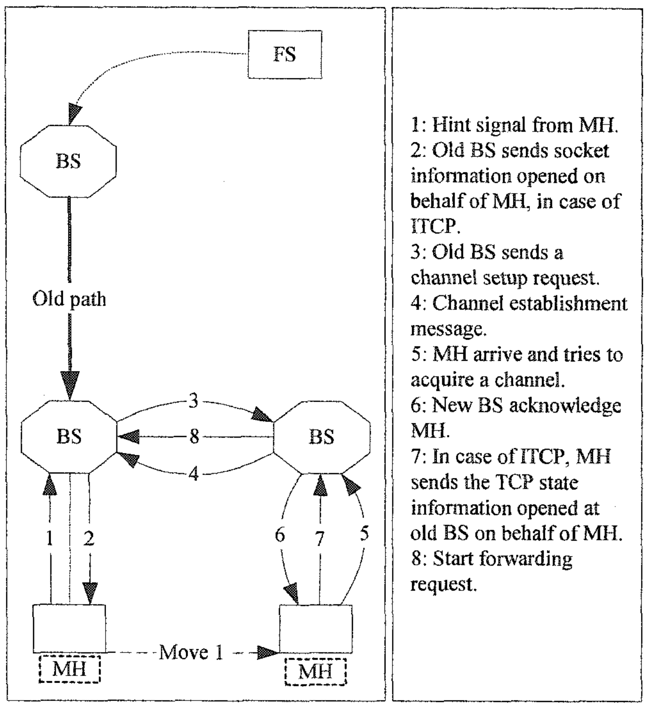
1: MH arrives and tries to acquire a channel.
2: New BS acknowledges MH.
3: New BS sends a channel setup request.
4: Channel creation.
5: In case of ITCP, a 3-way handshake process intiated by new BS for old BS to send sockets states information to new BS.

3a and 3b happens simultaneously.

Figure 3.3: Interaction protocol between Mobile TCP and Cell Forwarding – without hint

Next the new BS sends a channel setup request over the path to old BS and old BS responds by confirming the creation of the channel as shown in steps 3 and 4 in Figure 3.3. In case of ITCP, old BS already opened two sockets for each connection established with MH. The state information for all sockets opened at old BS on behalf MH needs to be transferred reliably to new BS in order to continue a seamless communication between the source and MH. Therefore, in step 5, new BS establishes a TCP connection with old BS using the 3-way handshake process. After a successful connection establishment, old BS sends the sockets state to new BS over the newly established TCP connection. The TCP connection between old and new BS's is removed as soon as old BS finishes from transferring the sockets state information to free memory spaces and reduce processing load at both ends.

Figure 3.4 show the details of the interaction protocol between mobile TCP and cell forwarding when MH uses hint process. The main difference between this protocol and the previous protocol shown in Figure 3.3 is in the way sockets state information is transferred to new BS in case of ITCP. In this protocol, when old BS receives the hint signal from MH, it sends a copy of the socket state to MH as shown in step 2 in Figure 3.4. When MH moves to the cell of new BS, it will establish a TCP connection with new BS as part of ITCP protocol. In this case, MH can use the reliable connection already opened with new BS to transfer the sockets state reliably. This method of sockets' states transfer saves old and new BS's from establishing and tearing down a TCP connection, which will save the two BS's time and network resources.



1: Hint signal from MH.
2: Old BS sends socket information opened on behalf of MH, in case of ITCP.
3: Old BS sends a channel setup request.
4: Channel establishment message.
5: MH arrive and tries to acquire a channel.
6: New BS acknowledge MH.
7: In case of ITCP, MH sends the TCP state information opened at old BS on behalf of MH.
8: Start forwarding request.

Figure 3.4: Interaction protocol between Mobile TCP and Cell Forwarding with hint.

3.2.3 Mobile TCP and Partial rerouting

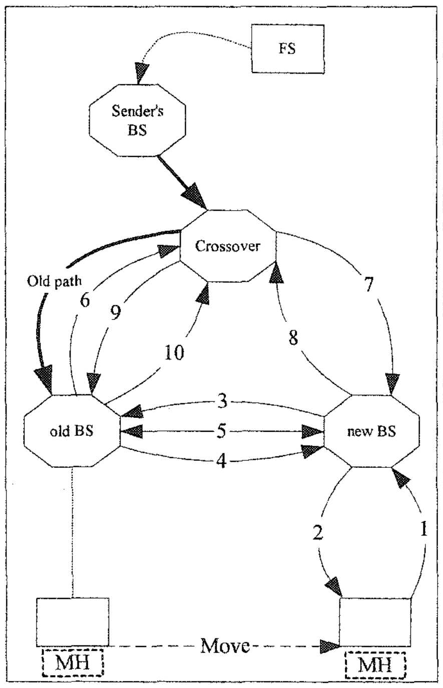
In this protocol, part of the old path between source and MH while at old BS is reused. The resources at the reused part are maintained and data stored or passing by these resources are not dropped and hence need not to be resent by the sender. In the first case, the hint process is not available and MH knows about its move only after it reaches the cell of new BS. The steps of the protocol are presented briefly in Figure 3.5. Below, more discussion is presented for each step.

Steps (1 and 2): A request for a connection establishment and channel allocation sent by MH and responded to by the new BS if it has enough resources.

Step 3: New BS sends a request to old BS to establish a network channel and request old BS to forward data to itself. Also, new BS asks old BS to invoke crossover point search algorithm to find the crossover point.

Step 4: Old BS acknowledges the requests and the acknowledgment confirms establishing a network channel between the two BS's.

Step 5: In case of ITCP, new BS establishes a TCP connection with old BS to transfer socket state information opened on behalf of MH from old BS to new BS.



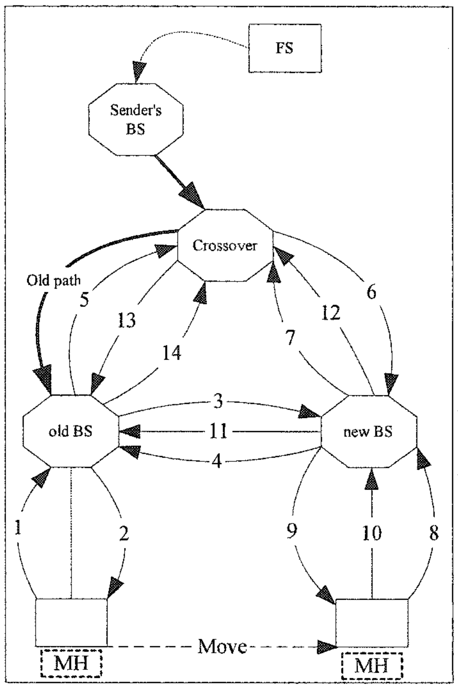
1: MH arrive and tries to acquire a channel.
2: New BS acknowledge MH's request (registration).
3: New BS requests old BS to forward data to itself and asks old BS to invoke crossover point search algorithem.
4:Old BS acknowledges the request.
5: New BS establishes a TCP connection with old BS to transfer socket states information (ITCP)
6: Old BS execute the crossover algorithm.
7: Crossover point sends a request to establish a connection to the new BS.
8: New BS informs crossover point about succesful connection establishment and it receive the acknowledgment.
9,10: Tear down old connection.

Figure 3.5: Interaction protocol between Mobile TCP and Partial Rerouting – without hint.

Step 6: Old BS execute the crossover algorithm to find the crossover point and to inform crossover point about the identity of new BS.

Steps 7 and 8: Crossover point sends a request to establish a network channel to new BS and the new BS sends an acknowledgement to setup the channel along the path in between.

Steps 9 and 10: Cross over point tears down the old connection and frees network resources along the path.



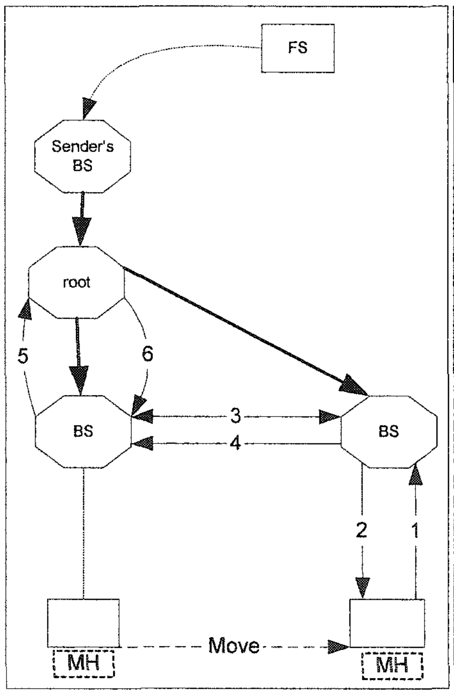
1: MH sends a hint signal to old BS.
2: In case of ITCP, old BS sends a copy of sockets states information to MH.
3: Old BS request channel setup.
4: New BS establish the channel.
5: Old BS execute the crossover algorithm.
6: Crossover node sends a request to establish a connection to the new BS.
7: Setup coonection with the cross over point.
8: MH arrive and tries to acquire a channel.
9: New BS acknowledge MH.
10: In case of ITCP, MH sends sockets states information to new BS.
11: New BS request the start of packet forwarding.
12: New BS requests the start of packet forwarding and torn down of old connection.
13 and 14: Torn down the old connection.

Figure 3.6: Interaction protocol between Mobile TCP and Partial Rerouting – with hint.

When hint process is used by MH to notify old BS about a possible move, the steps of interaction protocol shown in Figure 3.6 have some differences with the one shown in Figure 3.6. In case of with-hint and ITCP, the states of all sockets opened at old BS are sent to MH while it is in the cell of old BS using existing TCP connection between old BS and MH and there is no need to establish a TCP connection between old and new BS to transfer sockets state information. The second difference between hint and no-hint protocols is that in the first, old BS executes the cross-over discovery algorithm before MH moves and in the later the old BS executes the algorithm after MH moves.

3.2.4 Mobile TCP and Tree rerouting

Data communication between source and destination MH requires establishing two connections: one connection at the network layer and the other at the transport layer. When MH is at the cell of old BS and new BS is a member of the multicast group, there are three alternatives to connect new BS to the root of the multicast group in order to receive a copy of the data sent by the source.



1: MH arrive and tries to acquire a channel.
2: New BS acknowledge MH's request (registration).
3: New BS establishes a TCP connection with old BS to transfer sockets states reliably (ITCP).
4: New BS informs old BS to remove itself from the multicast group.
5 and 6: Old BS sends a request to leave the multicast group to the root, which will acknolwedg such a request.

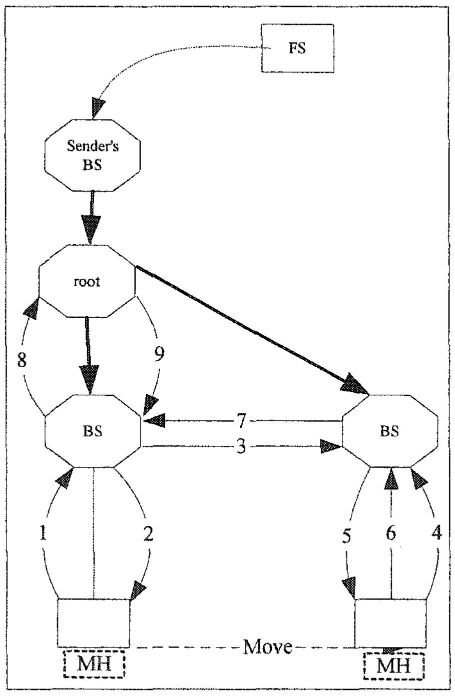
Figure 3.7: Interaction between mobile TCP schemes and Tree rerouting-without hint.

- Method 1: New BS establishes a network channel and transport layer connection when MH moves to its cell or when new BS receives a note about MHs possible move. In this method, resources along the path between the root of the multicast group and new BS are used only when MH moves, or about to move, to the cell of new BS and therefore it minimizes resources consumption. On the negative side, this approach increases the delay experienced by MH as new BS will start to establish the connection with the root only after MH moves to the new cell.

- Method 2: In this method, new BS opens a network channel with the root of the multicast group before the arrival of MH or the hint signal. After new BS receives a registration request from MH (in case of "without hint") or a notification (in case of "with hint") of possible MH movement to its cell, it initiates a 3-way hand shake process to establish a TCP connection with the root. In best case scenario, new BS will have all the packets destined to MH in its local buffer and it does not need the source TCP to retransmit any packets.

- Method 3: This is the most aggressive method that can be used by a new BS. Here new BS establishes both network and transport layers connections prior to the arrival of MH or the arrival of the hint signal from old BS. Resources along the path between the root and new BS are reserved and a copy of each TCP packet sent to MH is sent also to new BS. It is clear that this method consumes network resources much more than the above two methods. In this method, new BS has a copy of the most recent packets sent to MH and therefore protects the TCP sender from retransmitting any packet that arrives correctly at new BS. Also the MH can start communicating with nodes in the network immediately after successful channel allocation.

In this section, we consider the second method as it requires less resource than method 3 and at the same time provides higher probability that the packets sent by the sender and unacknowledged by MH are still in the local buffer of new BS. Hence, this scheme reduces the probability that the sender will time out and retransmit any packets.



1: Hint signal from MH.
2: In case of ITCP, old BS sends socket information opened on behalf of MH.
3: Old BS informs new BS about MH possilble move.
4: MH arrive and tries to acquire a channel.
5: New BS acknowledge MH's request (registration).
6: MH sends the socket information opend at old BS (ITCP).
7: New BS informs old BS to remove itself from the multicast group.
8 and 9: Old BS sends a request to leave the multicast group to the root, which will acknolwedg such a request.

Figure 3.8: Interaction between mobile TCP schemes and Tree rerouting-with hint.

Figure 3.7 shows the steps needed to complete the interaction protocol between different mobile TCP schemes on top of tree rerouting when MH does not use the hint process. Figure 3.8 describes the same protocol when MH utilizes the hint facility. In this section, we explain the detail steps of the interaction protocol while using Figures 3.7 and 3.8 as a summarized. When MH does not use the hint process, the steps of the interaction protocol are as follows:

Steps 1 and 2: These two steps are similar to previous protocols.

Step 3: In case of ITCP and with the information provided by MH about the identity of old BS, the new BS establishes a TCP connection with old BS to transfer sockets state reliably. After successful connection establishment, old BS sends the states information to new BS.

Step 4: After it receives all state information, new BS informs old BS to remove itself from the multicast group.

Steps 5 and 6: As a response to the request in step 4, old BS sends a request to leave the multicast group to the root, which will acknowledge the request and free all reserved resources along the path.

In case of "with-hint", the steps of the interaction protocol are similar to the above steps with one difference. In case of ITCP and when MH informs old BS about its possible move, old BS sends sockets state information to MH over the TCP connection that already exists. After MH moves to the cell of new BS, it will exchange the sockets information with new BS. Therefore, new BS need not establish a TCP connection with old BS to exchange the sockets information.

3.3 Protocol verification and proof of correctness

In this section we present a finite state machine model to proof the correctness of the interaction protocol between cell forwarding and mobile TCP. The interaction protocol is explained in more details in section 3.2.2.

Number	MH position states	MH states	Old BS states	New BS states	Wireless link states
0	Attached to old BS	Measuring the strength of beacon signal received from different BS's	Idle	Idle	Idle
1	Attached to new BS	Processing packet Pi (including generating ACK Ai)	Processing packet destined to MH	Processing channel setup request sent by old BS	Contains packet Pi
2		Decided to move and doesn't have sockets states information	Processing ACK destined to sender	Processing registration request sent by MH	Contains ACK Ai
3		Decided to move and have sockets states information	Expecting ACK (Ai-Ai+w)	Processing sockets states sent by MH	contains hint signal sent by MH
4		Waiting for registration ACK from new BS and doesn't have sockets states information	Expecting packet (Pi-Pi+w)		Contains sockets states information packet
5		Waiting for registration ACK from new BS and have sockets states information	Processing hint signal		Contains registration request packet sent by MH
6		Processing registration ACK received from new BS	Forwarding packets to MH directly		Contains registration ACK sent by new BS to MH
7			Forwarding packets to MH through new BS		

Table 3.1: Values of all Variables of the Protocol.

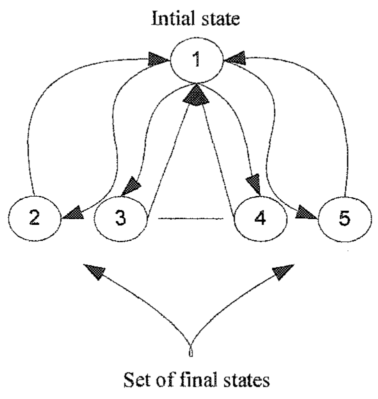


Figure 3.9: Protocol Correctness Proof

In tables 3.1 and 3.2 we presented the finite states and transitions possible in the protocol. A unique state called initial state. We will proof the correctness of the protocol by showing that starting from the initial state, the protocol goes to a subset of states called final states that signal the end of the interaction protocol. Also, we will show that there is a path from every final state to the initial state. Figure 3.9 illustrates the proof. In the Figure, state 1 is the initial state and the set of final states are 2,3,4, and 5.

We should point out that the protocol goes to the initial state when it receives the first packet through the new BS. In this study we assume that the control packets are neither lost nor duplicated and they arrive in a FIFO order.

The interaction protocol starts when the mobile receiver receives the first packet from source through old BS. At this point, the mobile host will be attached to the old BS and hence, the value of the variable "MH position states" is 0. The mobile receiver (MH) state will be "Processing packet Pi" which corresponds to the variable value 1 as shown in table 3.1. The old BS and new BS are idle, and hence, their state values are 0's. Finally, the wireless link does not contain any packet and this corresponds to the value 0. To put it all together, the initial state of the interaction protocol is (0, 1, 0, 0, 0).

We define the set of final state to be the subset of protocol's states where the protocol is complete. After the interaction protocol finished, receiving a packet by the mobile host though the current base station will take the protocol to the initial state again.

According to the definition of final state, the protocol is in a final state after mobile host receives the ACK for registration request from new BS as per the protocol description. At this point, the new BS is aware of mobile host's movement. As a result, when it receives a packet destined to MH, it will be able to pass it through to MH. The transition of sending a packet by new BS to MH will take the protocol to the initial state and signal the end of the interaction protocol. The status of the other 4 variables of the model does not affect the behavior of the protocol. We will use an asterisk as the variable value to indicate that the variable's value has no impact on the behavior of the protocol. Therefore, the final state of the protocol is (1, 0, *, *, *) and it means that MH has successfully registered with new BS and waiting for a packet to arrive. The states of old BS, new BS, and wireless link have no effect because the transition that will take the model to the initial state can not happen until the rerouting protocol is finished and the new BS knows the state information.

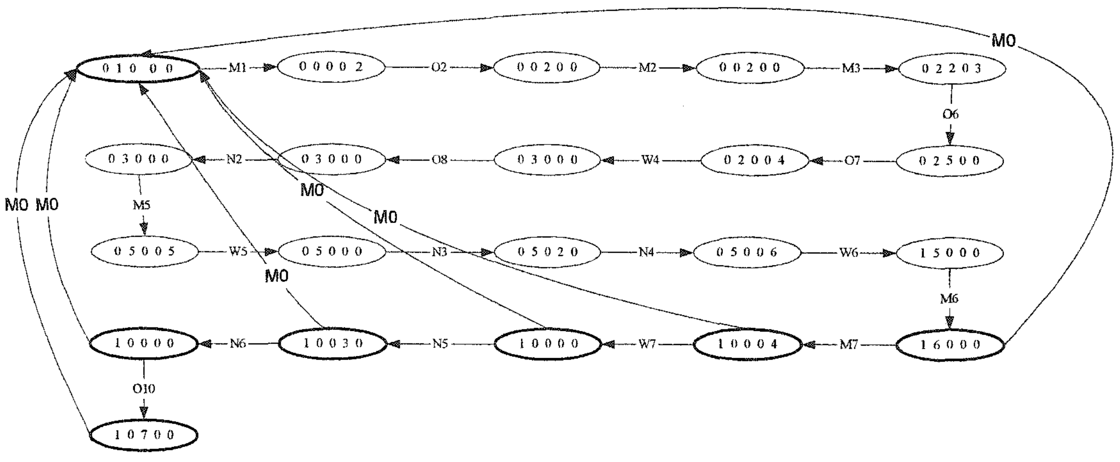


Figure 3.10: A finite state diagram for the interaction protocol of cell forwarding and mobile TCP in connection-oriented environment.

Now, we will show that our interaction protocol is able to transit successfully from the initial state to any final state and then goes again to the initial state. This cycle is repeated every time the mobile host moves. In Figure 3.10, the initial state is (0 1 0 0 0) and the set of final states are shown in the diagram as the bold states. Transition M0 takes the model from the final states back to the initial state.

Transition	Who is running	Action
M0	MH	Receive packet Pi
M1	MH	Sends ACK Ai
M2	MH	Receives stronger signal from new BS
M3	MH	Sends hint signal to old BS
M4	MH	Receives ACK from old BS for the hint signal that contains sockets states information.
M5	MH	Sends registration request message to new BS and start Timer for Request Retransmission (TRR).
M6	MH	Receives registration ACK from new BS
M7	MH	Sends sockets states information to new BS
M8	MH	TRR expires
O1	Old BS	Receives packet Pi from sender
O2	Old BS	Receives ACK Ai from MH
O3	Old BS	Receives corrupted packet Pi
O4	Old BS	Receives corrupted ACK Ai
O5	Old BS	Sends packet Pi to MH

O6	Old BS	Receives a hint signal from MH
O7	Old BS	Sends sockets states information to MH
O8	Old BS	Sends channel setup request to new BS
O9	Old BS	Receive channel setup conformation from new BS
O10	Old BS	Receives a forward request from new BS
N1	New BS	Receives a channel setup request from old BS
N2	New BS	Sends channel establishment conformation to old BS
N3	New BS	Receives a registration request from MH
N4	New BS	Sends registration ACK to MH
N5	New BS	Receives sockets states information packet from MH
N6	New BS	Sends a forwarding request to old BS
W1	wireless link	Delivers packet Pi to MH
W2	wireless link	Delivers ACK Ai to old BS
W3	wireless link	Delivers hint signal to old BS
W4	wireless link	Delivers sockets states information packet to MH
W5	wireless link	Delivers registration request to new BS
W6	wireless link	Delivers ACK for registration request to MH
W7	wireless link	Delivers sockets states information packet to new BS
W8	wireless link	Drops the packet it contains

Table 3.2: Possible transactions by all members of the protocol.

3.4 Evaluation metrics

When MH moves from its current cell, its connection will be disrupted until it establishes a connection with another BS. Depending on the direction of the communication, there are two types of disruption times DT_D and DT_U as explained in chapter 2. In case of "without-hint", DT_U can be further divided into two sub-types: DT_D-F and DT_D-N.

The mobility nature of the receiver introduces an excess buffer requirement on the old and new base stations. Therefore, with the increase of mobile devices usage, the base stations' buffer space will be a critical aspect to the performance of any communication taking place in mobile networks. Hence we calculated the buffer required on old and new base stations in different directions (up and down links) for all different cases of rerouting and TCP schemes. When old BS uses immediate acknowledgement the buffer required for supporting the hand-off process depends on the length of the time from when MH moves from the cell of old BS until old BS receives the forward request from new BS. On the other hand, the buffer requirement at new BS depends on the time it will take the forward or new path to be established. In the next section, we derive analytical equations for computing the exact values of buffer requirements.

The last metric is the buffer requirement at MH. When MH disconnects from its old base station, it cannot send out data and hence data coming from higher layers are stored at the network layer until a successful registration with the new BS is complete. Hence, the buffer requirement at MH equals the size of the packet received during the disconnect time. We assume that the cells of neighboring base stations have an overlap area maintaining signal reception, and hence, the length of the movement of MH between the two BS's is omitted for our models.

3.5 Analytical model

In this section, we present our analytical model to derive the metrics introduced in section 3.3. This section is further divided into two sub-sections: with-hint and without-hint as presented in Figure 3.9. In each subsection, we presented the calculation of buffer requirement at old BS and disruption time at MH. Since message numbering is not unified in Figures 3.1 – 3.8 of the interaction protocols, we added one table containing the meaning of common notations used in the analytical model in addition to one table for each one of the eight interaction protocols to explain the value of the messages. The calculations presented in this section assume that the sender always has data to be sent.

Below, we define the common notations for all interaction protocols used in our analytical model:

Symbol	Description
$N_{X\text{-}Y}$	Number of nodes on the path between the nodes X and Y.
B_X	The buffer space required at node X.
B_W, B_{wl}	Bandwidth on the wireline and wireless link, respectively.
L_w, L_{wl}	Latency on wired and wireless link, respectively.
S_{ctrl}	Size of a control packet.
S_{datal}	Size of a data packet.
S_{state}	Size of sockets state opened at old BS on behalf of MH.
State$_{old\text{-}new}$	$(\frac{S_{state}}{B_w} + L_w) \times N_{old-new}$
State$_{MH\text{-}new}$	$(\frac{S_{state}}{B_{wl}} + L_{wl})$
$TCP_{X\text{-}Y}$	The total time needed to establish a TCP connection between nodes X and Y.

Table 3.3: Description of analytical model symbols.

For all rerouting and TCP schemes, there is no buffer requirement at MH either on the uplink or on the downlink sides because MH is a receiver that will be sending out acknowledgments, and if one is lost, it can be recovered by the others. In case of old BS, there is buffer requirement on the down link only. Up-link communication going through old BS does not need to be buffered because either the old or the forward path is available. Similarly, new BS does not have any buffer requirement either on the up or down links. In case of down link communication, new BS starts receiving data destined to MH only when the communication channel with MH is established and, hence, new BS can forward the data immediately to MH. The only exception is in the case of tree rerouting as we explained below. On the uplink side, new BS does not need to buffer any data sent by MH because MH is a receiver and sends only acknowledgments. Losing one acknowledgment can be recovered by a later one as TCP uses a cumulative acknowledgment method. !

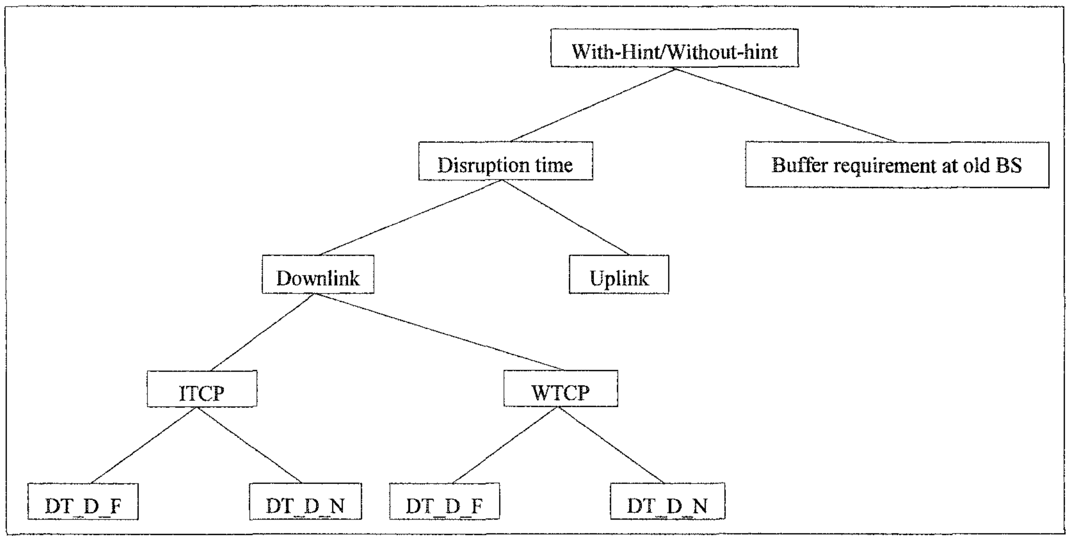


Figure 3.11: Analytical model organization

3.4.1 With-hint

Mobile Connection-oriented environment includes a mobile host, base stations, and a connection-oriented back bone. At any point of time, the mobile host is connected to only one base station which is connected to the back bone of the network through wire medium.

3.4.1.1 Buffer requirement

In ITCP, old BS acknowledges packets destined to MH before it receives one from MH. Therefore, in case of WTCP, old BS does not need to buffer any packets as it the sender's responsibility to recover any packet losses. After MH moves out of the cell of old BS, old BS will not be able to deliver down link packets destined to MH until it receives a forward request from new BS. In case of tree rerouting, old BS will keep buffering until it receives the request to leave the multicast group from new BS. This interval of the time between the two events decides the buffer requirement at old BS.

All rerouting schemes have the same buffer requirement since they all, except tree rerouting, use the same forwarding process. In tree rerouting, the length of buffering time is the same as other rerouting schemes but the type of messages exchanged are different. In tree rerouting, the message sent by new BS to old BS is a request to leave the multicast group. Upon recipient of such message, old BS will send a request to the root of the multicast group requesting a removal from the group. At this time, old BS knows that MH is registered with another BS and there no need to buffer any packets. This buffer requirement is shown in equation (3.1).

Notation	Description
M(1)	$(\frac{S_{ctrl}}{B_{wl}} + L_{wl})$
M(3)	$(\frac{S_{ctrl}}{B_{w}} + L_{w}) \times N_{old-new}$

Table 3.4: Values of notation used to calculate buffer requirement.

$$B_{oldBS} = [M(1) + M(2) + state_{mh-new} + M(3)] \times B_{w} \qquad (3.1)$$

3.4.1.2 Disruption time at MH!

Two types of disruption time are of important in our study. The first is the disruption time on the uplink (DT_U) and the other is disruption time on the downlink (DT_D). The DT_U is the time interval between sending the last ACK by MH while attached to old BS and sending the first ACK while attached to new BS. It is preferable and important that MH (the receiver) sends ACK soon enough to protect the sender from timing out. More importantly, DT_D is the time between receiving the last packet by MH while attached to old BS and receiving the first packet while attached to new BS. All applications running on top of TCP at MH will be affected severely when data packets are received slowly. In cell forwarding, we found out that DT_D_F = DT_D_N because the new path is the same as the forward path. In tree rerouting, DT_D_F does not exist because new BS already received copy of unacknowledged packets since it is a member of the multicast group.

First, we calculated DT_U. In case of WTCP, MH must acquire a wireless channel before it can send any segment out. The length of DT_U is the same for all rerouting schemes. Equation (3.2) shows the length of DT_U in case of WTCP.

$$DT\_U(WTCP) = (\frac{S_{ctrl}}{B_{wl}} + L_{wl}) \times 2 \qquad (3.2)$$

In case of ITCP, MH needs to perform one more operation in addition to the operations of WTCP before it can send any segments out. The sockets state opened at old BS on behalf of MH must be transferred reliably to new BS. We have two options to perform this task: either open a TCP connection between old and new BS or transfer the states information from old BS to MH and then to new BS after MH successfully registered with new BS. In our model, we adopted the second option because it does not impose extra connection establishment constraints. It is important to note that MH takes advantage of the fact that the link between MH and new BS is one hop and uses a reliable data link layer protocol to transfer the sockets state information. The length of DT_U in case of ITCP is shown in equation (3.3).

$$DT\_U(ITCP) = (\frac{S_{ctrl}}{B_{wl}} + L_{wl}) \times 2 + State_{MH-new} \quad (3.3)$$

We divided DT_D further into two types depending on the path the first packet takes to reach MH. The packet could arrive through the forward path or through the newly established path between new BS and source. We appended an identifier at the end of DT_D to distinguish between the two types of DT_D. The identifier F represents the first type while the letter N represents the later. The mathematical model for DT_D is shown below and is divided into four groups based on the type of rerouting scheme. Further, each group is divided based on the path used by the first packet to reach MH (F or N). Also the mathematical equations are further divided based on the type of mobile TCP scheme (WTCP or ITCP). In the equations below, M (i) refer to message i in the corresponding figure. For example, *M* (7) in case of full rerouting refer to message 7 in Figure 3.2. Table 2 shows the mathematical values of notations used in the equations below.

Below, we present the mathematical equations describing the length of the disruption time through both new and forward paths. In the case of ITCP, old BS must transfer the sockets state information opened on behalf of MH to new BS reliably. We can perform this task in two ways. First, old BS can establish a TCP connection with new BS then forward the state information. Second that we adopted in our protocols is that old BS transfer the state information to MH, which caries it with them to new BS. Upon arrival, MH transfers the information to new BS.

Notation	**Value**
Data(source->new)	$(\frac{S_{data}}{B_w} + L_w) \times N_{src-new}$

Data(old->new)	$(\frac{S_{data}}{B_w} + L_w) \times N_{old-new}$
Data(new->MH)	$(\frac{S_{data}}{B_{wl}} + L_{wl})$
TCP$_{\text{new-old}}$ = TCP$_{\text{old-new}}$	$3 \times (\frac{S_{ctrl}}{Bw} + L_w) \times N_{old-new}$

Table 3.5: Values of notations common to all rerouting schemes used in analytical model.

Full rerouting

Notation	Description
M(7) = M(8)	$(\frac{S_{ctrl}}{B_{wl}} + L_{wl})$
M(10)	$(\frac{S_{ctrl}}{B_w} + L_w) \times N_{old-new}$
M(11)	$(\frac{S_{ctrl}}{B_w} + L_w) \times N_{src-new}$

Table 3.6: Mathematical descriptions of messages for Full rerouting.

$$DT\_D\_F(WTCP) = M(7) + M(8) + \frac{S_{ctrl}}{B_{wl}} + M(10) + data(old -> new) + data(new -> MH)$$

$$DT\_D\_F(ITCP) = M(7) + M(8) + State_{MH-new} + M(10) + data(old -> new) + data(new -> MH)$$

$$DT\_D\_N(WTCP) = M(7) + M(8) + M(11) + data(source -> new) + data(new -> MH)$$

$$DT\_D\_N(ITCP) = M(7) + M(8) + State_{MH-new} + M(11) + data(source -> new) + data(new -> MH)$$

Partial rerouting

Notation	Description
M(8) = M(9)	$(\frac{S_{ctrl}}{B_{wl}} + L_{wl})$
M(10)	$(\frac{S_{state}}{B_{wl}} + L_{wl})$
M(11)	$(\frac{S_{ctrl}}{B_{w}} + L_{w}) \times N_{old-new}$
M(12)	$(\frac{S_{ctrl}}{B_{w}} + L_{w}) \times N_{cross-new}$

Table 3.7: Mathematical descriptions of messages for Partial Rerouting.

$$DT\_D\_F(WTCP) = M(8) + \frac{S_{ctrl}}{B_{wl}} + M(11) + data(old->new) + data(new->MH)$$

$$DT\_D\_F(ITCP) = M(8) + M(9) + M(10) + M(11) + data(old->new) + data(new->MH)$$

$$DT\_D\_N(WTCP) = M(8) + \frac{S_{ctrl}}{B_{wl}} + M(12) + data(cross->new) + data(new->MH)$$

$$DT\_D\_N(ITCP) = M(8) + Max[(M(9) + M(10), M(12) + data(cross->new)] + data(new->MH)$$

Cell forwarding

Notation	Description
M(5) = M(6)	$(\frac{S_{ctrl}}{B_{wl}} + L_{wl})$
M(7)	$(\frac{S_{state}}{B_{wl}} + L_{wl})$
M(8)	$(\frac{S_{ctrl}}{B_{w}} + L_{w}) \times N_{old-new}$

Table 3.8: Mathematical descriptions of messages for Cell Forwarding.

$$DT\_D(WTCP) = M(5) + \frac{S_{ctrl}}{B_{wl}} + L_{wl} + M(8) + data(old->new) + data(new->MH)$$

$$DT\_D(ITCP) = M(5) + M(6) + M(7) + M(8) + data(old->new) + data(new->MH$$

Tree rerouting

Notation	Description
M(4), M(5)	$(\frac{S_{ctrl}}{B_{wl}} + L_{wl})$
M(6)	$(\frac{S_{state}}{B_{wl}} + L_{wl})$

Table 3.9: Mathematical descriptions of messages for Tree rerouting.

$$DT\_D\_N(WTCP) = M(4) + \frac{S_{ctrl}}{B_{wl}} + data(new->MH)$$

$$DT\_D\_N(ITCP) = M(4) + M(5) + M(6) + data(new->MH)$$

3.4.2 Without-hint

This section concentrates on the interaction protocols when MH does not have the ability to discover its possible movement in the near future, which we call without-hint. As a result, new BS will not set any connections prior to MH's arrival.

3.4.2.1 Buffer requirement

In all rerouting schemes, old BS will stop buffering when it receives a request to forward packets from new BS. The only exception is tree rerouting, where the message is going to be a request to leave the multicast group. Luckily, the impact of both messages on the buffer requirement is the same and is shown in equations (3.4) and (3.5). !

!

Notation	Description
M(1)	$(\frac{S_{ctrl}}{B_{wl}} + L_{wl})$
M(3)=M(4)	$(\frac{S_{ctrl}}{B_{w}} + L_{w}) \times N_{old-new}$

Table 3.10: Mathematical descriptions of notations used in equations 3.5 and 3.6.

$$\mathrm{B}_{\mathrm{oldBS}}(WTCP) = \left(M(1) + \frac{S_{ctrl}}{B_{wl}} + M(3) \right) \times B_{w} \tag{3.4}$$

$$\mathrm{B}_{\mathrm{oldBS}}(ITCP) = \left(M(1) + \frac{S_{ctrl}}{B_{wl}} + M(3) + M(4) + TCP_{new-old} \right) \times B_{w} \tag{3.5}$$

3.4.2.2 Disruption time at MH

In previous section dealing with the with-hint approach, the uplink disruption time in case of WTCP is different when compared with the disruption time in case of ITCP. The main difference is that, in the later MH must transfer sockets state information to new BS before it start sending any segment. In case of with-hint, the sockets state is transferred directly between old and new BS and, hence, DT_U is the same in both WTCP and ITCP and is shown in equation (3.6). It is also the same for all rerouting and mobile TCP schemes.

$$DT\_U(WTCP, ITCP) = (\frac{S_{ctrl}}{B_{wl}} + L_{wl}) \times 2 \tag{3.6}$$

The length of DT_D is shown below for each rerouting scheme. For each scheme, the length of DT_D depends on the mobile TCP scheme used. In case of ITCP, old BS must transfer sockets state information reliably to MH after it receives the hint signal. After arriving at the cell of new BS, MH must transfer the sockets state information to new BS to continue communicating without disconnection. Below, we present the mathematical equation for all cases.

Full rerouting

Notation	Description
M(1)	$(\frac{S_{ctrl}}{B_{wl}} + L_{wl})$
M(3)=M(4)	$(\frac{S_{ctrl}}{B_{w}} + L_{w}) \times N_{old-new}$
M(5)	$(\frac{S_{ctrl}}{B_{w}} + L_{w}) \times N_{old-new}$
M(7)=M(8)	$(\frac{S_{ctrl}}{B_{w}} + L_{w}) \times N_{src-new}$
M(9)	$(\frac{S_{ctrl}}{B_{w}} + L_{w}) \times N_{src-old}$

Table 3.11: Mathematical descriptions of messages for Full rerouting.

$$DT\_D\_F(WTCP) = M(1) + \frac{S_{ctrl}}{B_{wl}} + M(3) + (\frac{S_{ctrl}}{B_{w}} \times N_{old-new}) + data(old->new) + data(new->MH)$$

$$DT\_D\_F(ITCP) = M(1) + \frac{S_{ctrl}}{B_{wl}} + M(3) + M(4) + TCP_{new-old} + data(old->new) + data(new->MH)$$

$$DT\_D\_N(WTCP) = M(1) + \frac{S_{ctrl}}{B_{wl}} + M(7) + M(8) + M(9) + data(source->new) + data(new->MH)$$

$$DT\_D\_N(ITCP) = M(1) + \frac{S_{ctrl}}{B_{wl}} + M(3) + M(4) + TCP_{new-old} + State_{\text{old-new}} + M(7) + M(8)$$
$$+ data(source->new) + data(new->MH)$$

Partial rerouting

Notation	Description
M(1)	$(\frac{S_{ctrl}}{B_{wl}} + L_{wl})$
M(3)=M(4)	$(\frac{S_{ctrl}}{B_w} + L_w) \times N_{old-new}$
M(6)	$(\frac{S_{ctrl}}{B_w} + L_w + T_{chk\_RT}) \times N_{old-cross}$
M(7)=M(8)	$(\frac{S_{ctrl}}{B_w} + L_w) \times N_{new-cross}$

Table 3.12: Mathematical descriptions of messages for Partial rerouting.

$$DT\_D\_F(WTCP) = M(1) + \frac{S_{ctrl}}{B_{wl}} + M(3) + \frac{S_{ctrl}}{B_w} + data(old->new) + data(new->MH)$$

$$DT\_D\_F(ITCP) = M(1) + \frac{S_{ctrl}}{B_{wl}} + M(3) + M(4) + TCP_{new-old} + data(old->new) + data(new->MH)$$

$$DT\_D\_N(WTCP) = M(1) + \frac{S_{ctrl}}{B_{wl}} + M(3) + M(6) + M(7) + M(8) + data(Cross->new) + data(new->$$

$$DT\_D\_N(ITCP) = M(1) + \frac{S_{ctrl}}{B_{wl}} + M(3) + Max[M(6) + M(7) + M(8) + data(Cross->new),$$
$$M(4) + TCP_{new-old}] + State_{old-new} + data(new->MH)$$

Cell forwarding

Notation	Description
M(1)	$(\frac{S_{ctrl}}{B_{wl}} + L_{wl})$
M(3)	$(\frac{S_{ctrl}}{B_w} + L_w) \times N_{old-new}$

Table 3.13: Mathematical descriptions

of messages for Cell forwarding.

$$DT\_D(WTCP) = M(1) + \frac{S_{ctrl}}{B_{wl}} + M(3) + \frac{S_{ctrl}}{B_{w}} + data(old->new) + data(new->MH)$$

$$DT\_D(ITCP) = M(1) + \frac{S_{ctrl}}{B_{wl}} + M(3) + TCP_{new-old} + State_{old-new} + data(old->new) + data(new->MH)$$

Tree rerouting

Notation	Description
M(1)	$(\frac{S_{ctrl}}{B_{wl}} + L_{wl})$

Table 3.14: Mathematical descriptions of messages for Tree rerouting.

$$DT\_D\_N(WTCP) = M(1) + \frac{S_{ctrl}}{B_{wl}} + data(new->MH)$$

$$DT\_D\_N(ITCP) = M(1) + \frac{S_{ctrl}}{B_{wl}} + TCP_{new-old} + State_{\text{old-new}} + data(new->MH)$$

3.6 Results

In this section we calculated the buffer required at old BS to support mobility, disruption time on uplink, and disruption time on down link. We wanted to examine the protocols under the same network topology with different network conditions. Therefore, to fix the network topology, we fixed physical network topology and vary the error transmission on the wireless link and move MH up to 100 times. Table 12 represents the network setups for wireless and wireline parts.

Network parameters	**Value**
Effective wireless bandwidth (B_wl)	1000 bytes
Effective wireline bandwidth (B_w)	100 KB
Size of control packet (Sctrl)	48 bytes
Size of data packet (Sdata)	65536 bytes
Latency of wireless link (L_wl)	50 microsecond
Latency of wireline link (L_w)	2 millisecond

Table 3.15: Network parameters.

Varying the error rate on the wireless link affects the number of segments delivered to old BS when MH in move. Eventually, this variation will result in different size of stored packets that are not acknowledged, yet. Since the size of sockets states information is proportional to the size of unacknowledged packet, changing the size of S_state in our equation represent closely the fluctuation in error rate at wireless link. In our calculations, the size of S_state range between 60KB and 600KB representing 10 to 100 TCP data packets.

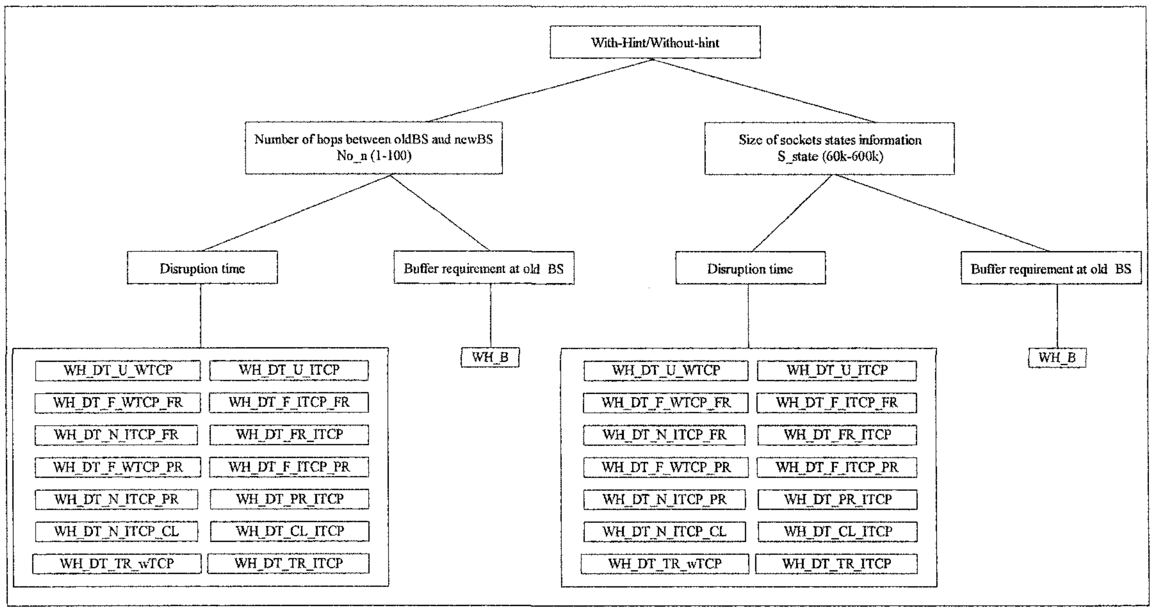


Figure 3.12: Organization of results showing the calculated values in all cases.

The other variable we used to generate our results is the number of hops between old and new base stations (No-n). It is obvious that the time needed to exchange information between old and new base stations depends on the distance and the number of hops on the path between the two base stations. Therefore, we calculated the buffer requirement and disruption time length for different values of No-n ranging between 1 and 100. The organization of the section, including the variables used is shown in Figure 3.10.

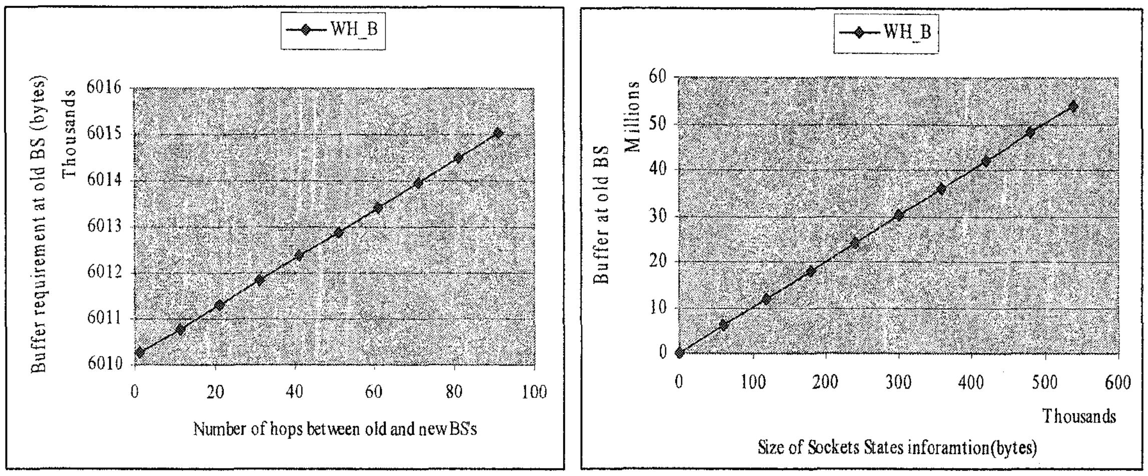


Figure 3.13: Buffer requirement at old BS in case of "with hint" for all rerouting schemes with variations of number of hops and states information size.

Figure 3.11 shows how the buffer requirement changes with the change of number of hops between old and new base stations and with the change of sockets states information. In the left graph, we fixed S_state to be 60K in order to concentrate on the impact of changing No-n. In the second graph at the right we fixed No-n to be 5 in order to evaluate the impact of different values of S_state on the buffer requirement at old BS.

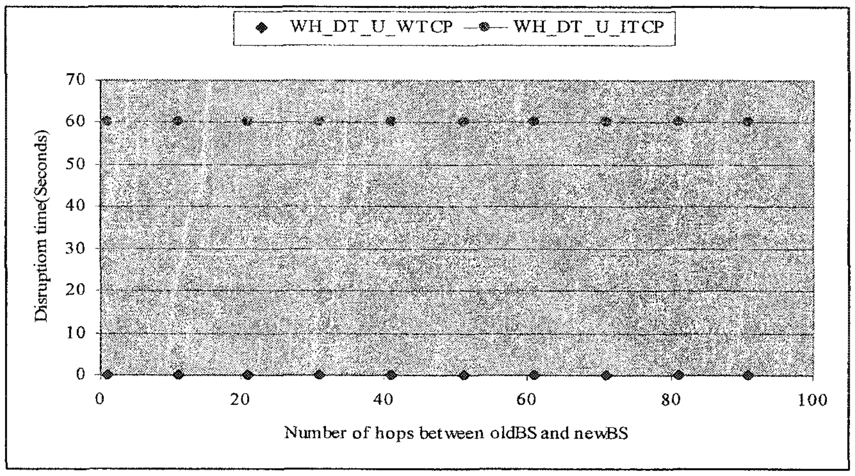


Figure 3.14: Disruption time on the uplink path in case of "with hint" for all rerouting schemes with variations of number of hops between old and new BS's.

Figure 3.12 shows the disruption time on the uplink path at MH. The type of rerouting schemes used at the network layer does not make any difference in the length of this disruption time. It is clear that this disruption time is longer in case of ITCP because of the time it takes to transfer the sockets states information from MH to new BS.

The rest of this section shows self explanatory figures for all different cases of combinations of mobile TCP and rerouting schemes. The Figures shows the disruption time on the uplink and downlink sides. Also, buffer requirement required at old BS to support hand off is presented. The first set of Figures (3.13-3.17) is for "with hint" case and the second set of Figures (3.18-3.23) are for "with-out hint" case.

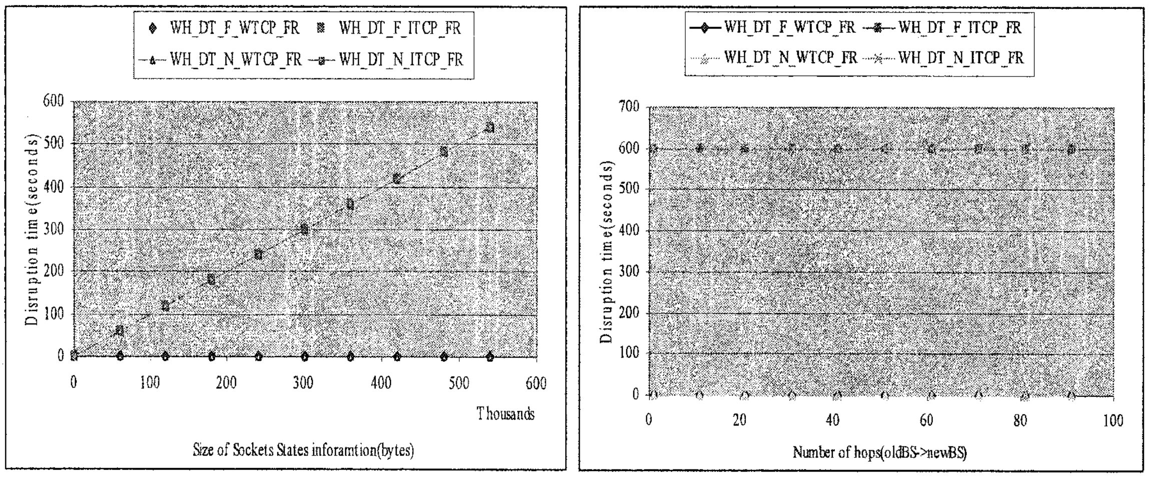


Figure 3.15: Disruption time on the forward and new paths in case of "with hint" and full rerouting with variations of number of hops and states information size.

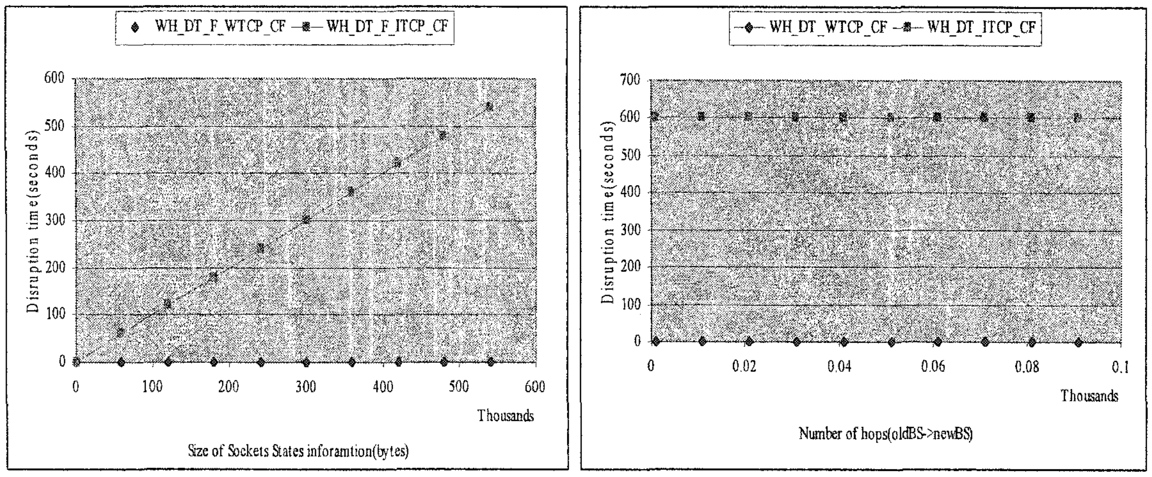


Figure 3.16: Disruption time on the forward path in case of "with hint" and cell forwarding with variations of number of hops and states information size.

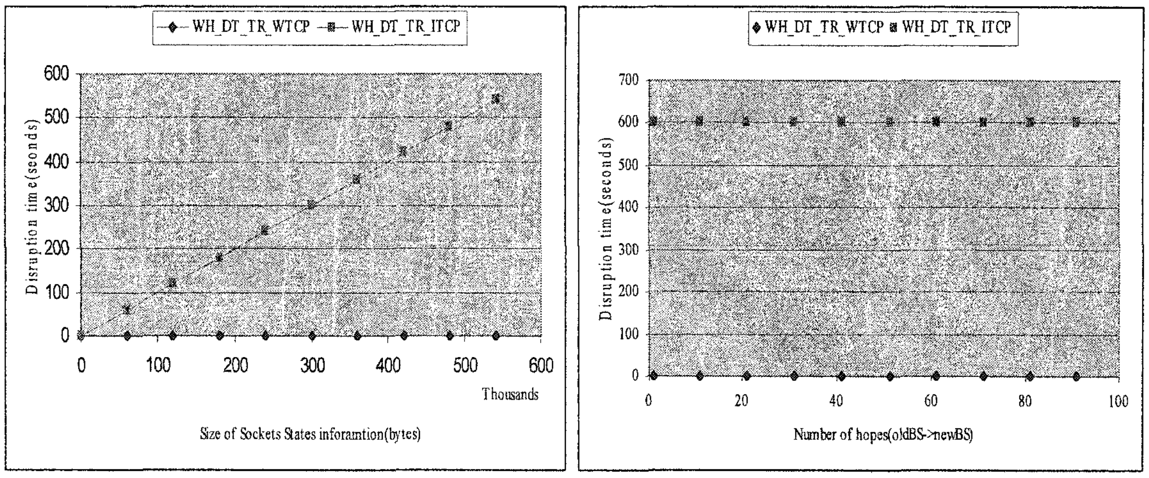


Figure 3.17: Disruption time on the forward path in case of "with hint" and tree rerouting with variations of number of hops and states information size.

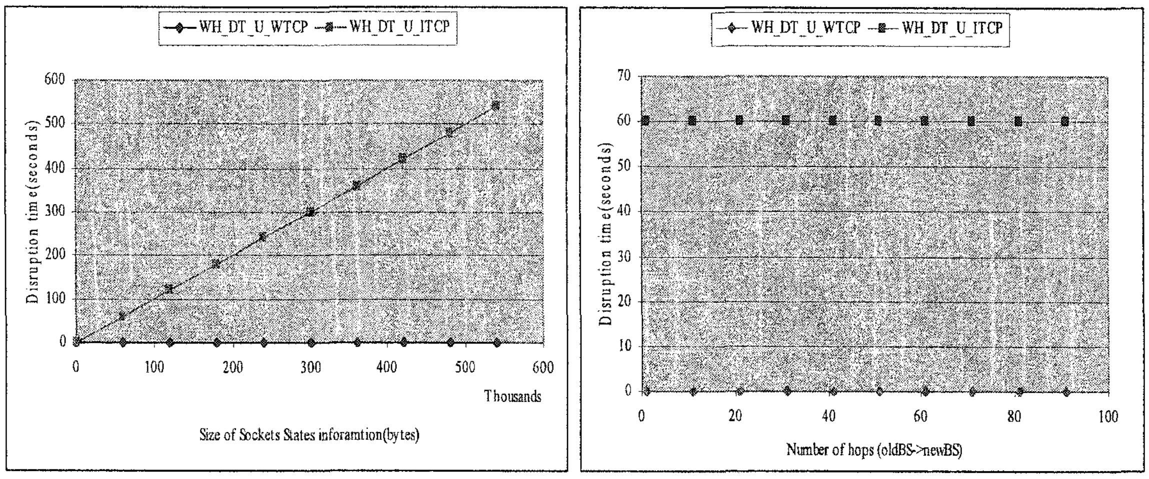


Figure 3.18: Disruption time on the forward path in case of "with hint" and cell forwarding with variations of number of hops and states information size.

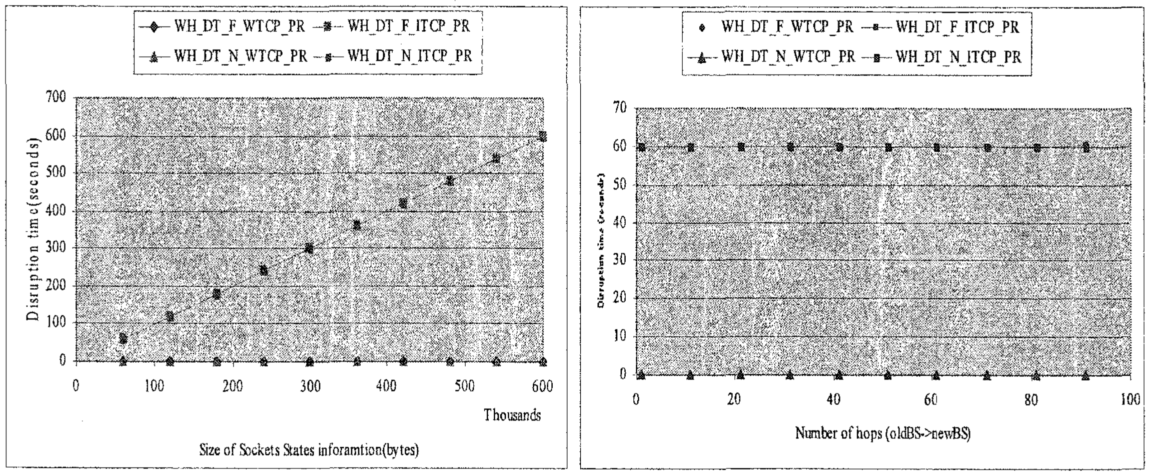
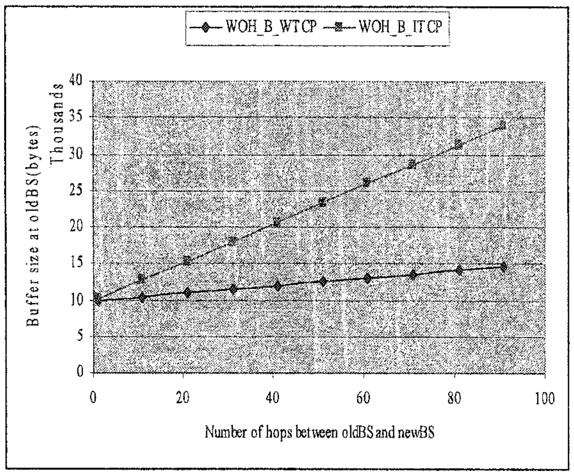


Figure 3.19: Disruption time on the forward path in case of "with hint" and partial rerouting with variations of number of hops and states information size.



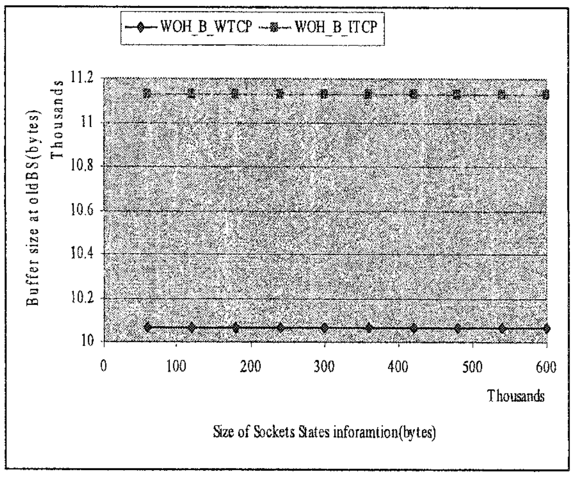


Figure 3.20: Buffer requirement at old BS in case of "with-out hint" for all rerouting schemes with variations of number of hops and states information size.

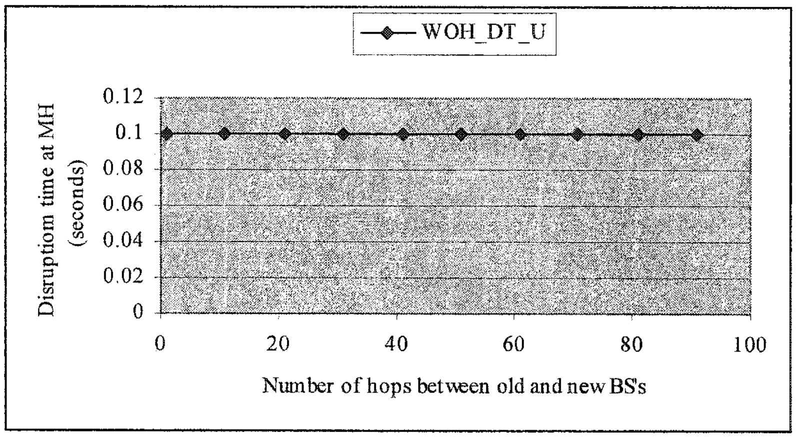


Figure 3.21: Disruption time on the uplink path in case of "with-out hint" for all rerouting schemes with variations of number of hops and states information size.

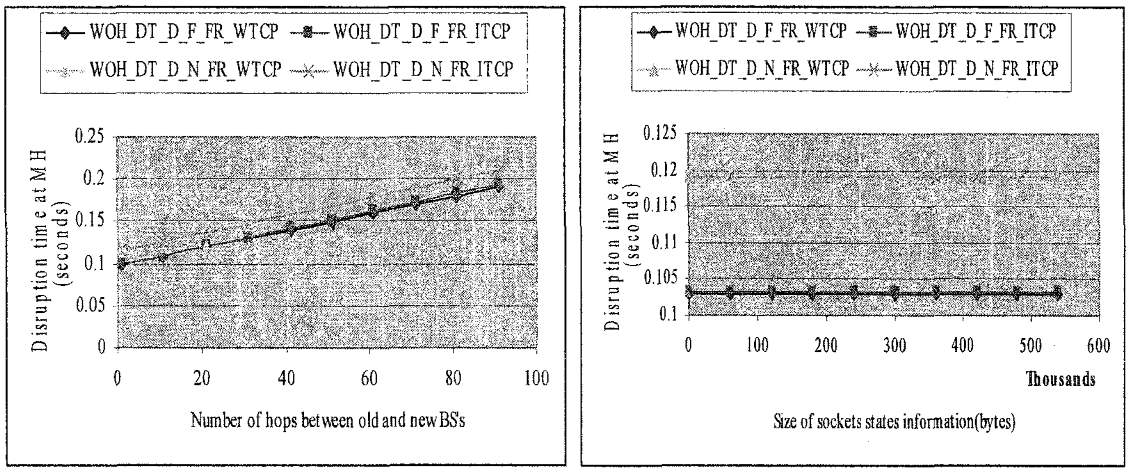


Figure 3.22: Disruption time on the forward and new paths in case of "with-out hint" and full rerouting with variations of number of hops and states information size.

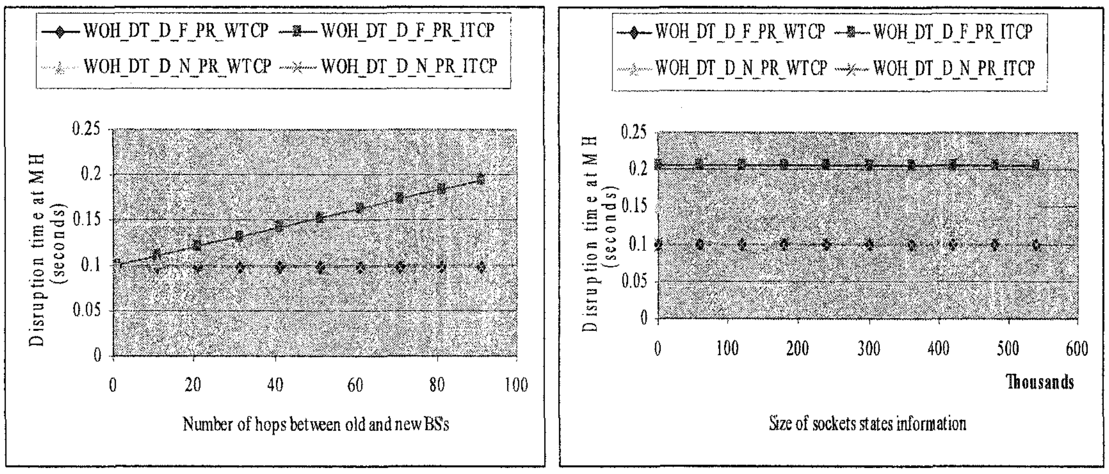


Figure 3.23: Disruption time on the forward path in case of "with hint" and cell forwarding with variations of number of hops and states information sizes.

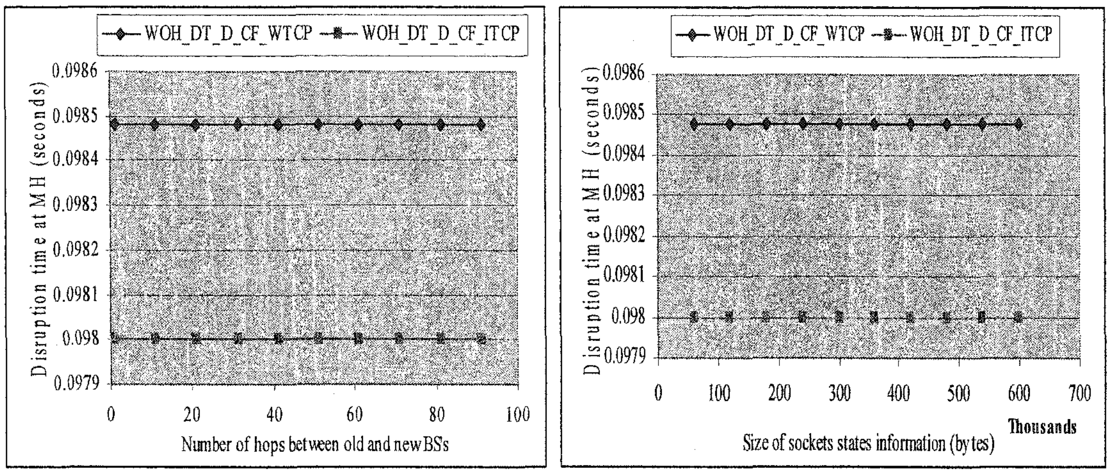


Figure 3.24: Disruption time on the forward path in case of "with-out hint" and cell forwarding with variations of number of hops and states information size.

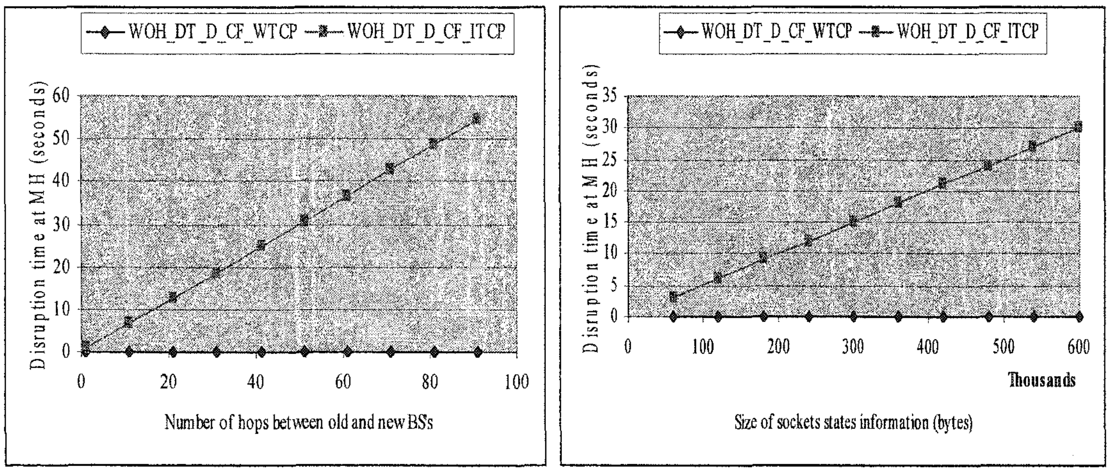


Figure 3.25: Disruption time on the forward path in case of "with-out hint" and tree rerouting with variations of number of hops and states information sizes.

3.7 Summary

In this chapter, we explain the interaction relationship between wireless TCP protocols and rerouting schemes in a connection-oriented environment where connection must be established prior to any communication. In addition, we presented comprehensive analytical methods to determine the buffer requirements at old base station and to estimate the disruption time at the MH on both the uplink and downlink paths. Future work involves investigating the impact of data link layer on the performance of mobile TCP and rerouting schemes. Also, the communication between two mobile peers will introduce new challenges and need to be study further.

Chapter 4

JTCP: Join Protocols for Improving Performance of TCP

4.1 Introduction

Individual TCP flow tries to maximize gain by increasing sending window size and filling up the available bandwidth on any link rapidly. Several TCP flows uncooperatively try to maximize their individual gain, and this leads to serious downgrading of TCP performance. We describe a *join* protocol that combines the TCP flows to lessen competition and improve individual TCP performance. Using our protocol we achieve fairness and increase throughput and goodput. The join protocol is an end-to-end protocol and can be extended network wide to overcome congested regions. We evaluate the effectiveness of our approach using extensive simulations with *network simulator* (*ns*) modified to implement our protocol.

The Transmission Control Protocol (TCP) is a widely used transport layer protocol that guarantees end to end reliable transfer of information. The efficiency of a TCP is measured by *throughput* which is the amount of data transferred reliably by the sender to the receiver. The throughput of the sender is controlled by two quantities: *receiver window* is the amount of acknowledged data the receiver can hold that have not been processed yet and *congestion window* is the amount of unacknowledged data the sender can send before waiting for an acknowledgement. The amount of actual data sent by the sender is the minimum of congestion and receiver window. Assuming a large receiver window, the congestion window grows (using a *slow start* algorithm) or shrinks depending on the arrival time of acknowledgements at the sender. The acknowledgements arrive late if either the data sent from the sender reaches the receiver late, or acknowledgements arrive late, or both. It is possible for the sender's data to arrive sooner at the receiver and the acknowledgements to arrive later because of the following: the receiver might delay sending the acknowledgments, the acknowledgements can use a different path to the sender that might be congested while it is in flight, or the acknowledgements might be piggybacked on a packets from receiver to sender and the delay may be attributed to the packet size. In all the above cases, the sender TCP incorrectly recognizes the state of the "network regions" along the path(s) from sender to receiver as being congested and shrinks the window size thereby decreasing the throughput.

The dynamics of TCP is further complicated by the presence of a large number of flows that share network resources: shared router and shared link. A single TCP flow in the absence of many flows that share the network resources keeps increasing the window size as the acknowledgements arrive on time. Thus each flow uncooperatively will try to increase its window size starting from a network condition that has few flows. This leads to packet losses at the congested link and all the flows will dramatically reduce their window size. When all the flows reduce their window size, the bandwidth on the shared links are inefficiently utilized and this cycle of increasing and decreasing window size continues. A noticeable decrease in throughput is observed by all the individual flows.

Based on the observations above we recognize two important shortcomings of the current TCP protocol: misinterpretation of delayed acknowledgements and competition among different TCP flows. In this research work, we propose to address these two issues by a use of novel protocol that uses *immediate and delayed acknowledgement schemes* and provides a *coordination mechanism* among independent TCP flows. We also address certain important issues that are related to the implementation of our proposed protocol: can we maintain the end-to-end semantics of TCP? Are there additional benefits that can be harvested if intermediate nodes with TCP protocol can be used? We will show that our protocol can be implemented on a portal router or a node attached to the portal router of a network administrative domain. We can extend this approach to the Internet and we provide protocols for this extension.

Our protocol will combine all the incoming TCP flows at the portal router into one single flow in an operation called the *join* operation. The agent in the portal router examines the individual flows and can do the following: acknowledge the senders of each flow and it can take the responsibility of sending the data reliably to the receiver (*immediate acknowledgement*) and perform flow control and fairness control by adjusting the receiver's window size. The joining process will reduce the competition among different flows and data can be sent at a rate that fills up the entire bandwidth of the outgoing link from the portal router. Also, since immediate acknowledgment scheme is used, the server can free up its resources resulting in increases efficiency of servers. Of course as the number of flows increases the burden in clearly is on the portal router whose role is buffering and forwarding. The protocol presented in this paper does not add any additional information to the packets sent by the sender. We show by extensive simulations the effectiveness of the proposed protocol and also show how this protocol can be extended to wide area networks like the Internet. As an added bonus, we have show that the initial threshold value set for each flow can have a serious impact on the throughput of the TCP flow and especially more profound when there are multiple flows that share the bottleneck bandwidth.

4.2 Related work

In the last few years many congestion control mechanisms have been proposed to over come new challenges associated with the growth in the Internet usage and with the introduction of new application classes. For example, it is expected that the usage of application that are not TCP-friendly will increase in the near future, such as real-time sensor networks and audio/video streaming. The proposed congestion control mechanisms can be classified in many ways as shown in [18]. One aspect is whether the congestion control is an end-to-end or router-supported protocol. The problem with the first approach is its reliance on the collaboration of the end systems that might not be TCP-friendly and cause great unfairness sharing of network resources. On the other hand, router-supported mechanisms require modification to existing routers and amenable for implementation in Intranets and Grid Networks [21].

Apart from congestion control algorithms, our scheme shows excellent results in lowering competition among flows and increasing their throughputs in a special class of applications called Cluster-to-Cluster [22]. The application class consists of many independent, but semantically related flows that share a common path in between the communicating entities. The authors point out that the lack of coordination among competing flows as a main weakness in the current transport layer protocols, especially for cluster-to-cluster applications. They address is problem by proposing a coordination protocol (CP) that tunnels the packets passing through the portal router and the tunneled packet contain probe information to evaluate the network for delay and loss. Using this information, the CP calculates the available bandwidth using the algorithm of Floyd et al [20]. This information is sent to the receiver, which in turn communicates with the sender to adjust the sending rate. Our proposed protocol does not introduce any additional layers in the protocol and requires participation of the portal at the sender only.

Balakrishnan et al [23] propose a congestion manager (CM) module that monitors the network conditions and informs each flow to modify their flow conditions. Each individual TCP flow upon their discretion modifies their individual flow. The major pitfall of this approach is that the flows can be uncooperative and requires modification to the TCP/IP stack at each end point to process messages sent by CM. Kung and Wang [24] propose a mechanism to apply TCP congestion control on aggregating flows between sender and receiver pair. They propose a separate TCP management connection each sender and receiver. It probes for the network congestion condition and this information is used to regulate the aggregated TCP data flows. The scheme does not provide the applications the power to make inter stream tradeoffs. The management flows, continuously generate management packets, which are a significant overhead. Karbhari et al [19] presented a novel protocol to address the issue of fairness in an environment that consists of multiple senders and one receiver. They define two types of fairness techniques: Inter-session and Intra-session. The intra-session deals with fairness among flows in the same session while inter-session handles fairness between sessions. The algorithms in [19] require strict coordination among all the flows before they begin sending data. It also assumes that no other flows exists path through which the flows are routed.

Kalampoukas et al [27] provide a protocol that dictates the size of the sender's window for each TCP flow based on the available bandwidth. This accomplished by modifying the advertised window size in the acknowledgment packet of each flow. As observed in [27] their proposed scheme of explicit window adaptation works only in environments where the available bandwidth can be precisely estimated and the number of bandwidth links for flows that share them is limited to one.

4.3 Flow Competition and Throughput

Congestion occurs when the aggregate throughput of the flows exceed the outgoing link capacity at a router. Active queue management methods at routers monitor the queue lengths and drop packets with certain probabilities belonging to flows. At higher layers, the individual flow with dropped packets does not get acknowledgements and hence they reduce their window size, which results in decreased throughput. The probability that a packet belonging to a flow is dropped is calculated using schemes such as RED and BLUE [28].

Assume that two TCP flows f_1 and f_2 share a bottleneck link with a bandwidth of B. As they both start the slow-start phase, they quickly fill up the bandwidth B resulting in packet losses and either one or both timing out and starting the slow-start phase. Assuming both the flows timeout, the process of growing and shrinking continues and this result in a decreased throughput. In Figure 4.1, the window size changes of two TCP flows are shown along with total data sent by each flow. We assume that the flows timeout (both at the same time) when the sum of the window sizes of the flows is greater than the bandwidth of the outgoing link. The outgoing link bandwidth is assumed to be 32KB (low bandwidth connection). The legend of Figure 4.1 should be interpreted as explained in the next example. The legend "A-2-32-8-WS" indicates that there are two flows and both timeout at the same time, the initial threshold value for both the flows are set to be 8, and WS is the window size. The suffix "DS" in the legend stands of data sent by each of flow. Also, in Figure 4.1, we show the window size and data sent for only one of the flows since it is same for both the flows as they both timeout at the same time.

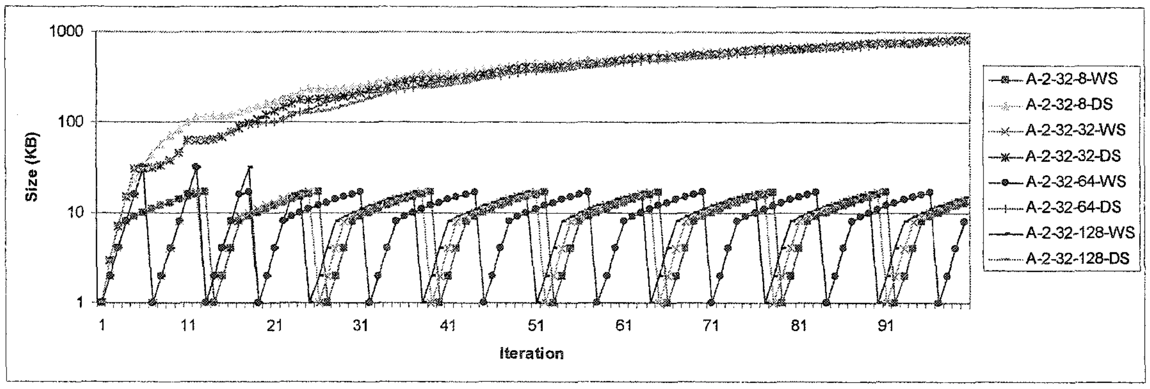


Figure 4.1: A log graph showing the window size changes and total data sent by a single flow in a network environment consisting of two flows with a bottleneck bandwidth of 32KB. Each line in the above graph shows the impact of the initial threshold value on the total data sent.

From Figure 4.1, we can make several important observations. First, due to competition the number of timeouts increases resulting in decreases throughput. Constant fluctuations in per port buffer requirements at the bottleneck router would require that the router allocate and dispose memory causing degradation of the router's performance. Second, the initial threshold set by the each of the flows is an important factor that determines the throughput of the individual TCP flow in the presence of other flows. For example, in Figure 4.1, the initial threshold value of 8 is the most optimal one in terms of total data sent and setting the value greater than 8 results in smaller throughput. We have observed similar phenomenon for the initial threshold values when the number of flows is greater than two. Third, when the initial threshold is very high, the frequency of timeouts is initially higher and overtime the frequency of timeouts converge to a constant number. Observe this in Figure 4.1 when the initial threshold value is 128.

4.4 The Join TCP Protocol

In this section, we will explain the Join TCP protocols for cluster-to-cluster and Internet environments.

The join protocol has two processes: the join process and acknowledgment process. The join process combines the different TCP flows into a single flow and the acknowledgement process decide when to acknowledge the sources of the TCP flows. These two processes will be discussed in the context of both network environments.

Definitions

In this subsection, we will define *portal router* and *join buffer* that are main components of our protocol. A portal router joins packets of all incoming flows to form one larger flow. The join and acknowledging processes take place at the portal router, as we will explain later. The second component is the *join buffer*, where the joining router buffers all un-acknowledged packets for each flow. Every incoming flow is assigned a separate *join buffer*. Similar to a TCP receiver, a *join buffer* maintains and advertises a receiving window used in controlling the data flow with the sender. The difference between a *join buffer* and a normal TCP receiver is that a *join buffer* does not send the received packet to the upper layers; rather it stores the packet until it is sent to the destination. Below we will explain in more detail the functionality of each one of the two components defined in this section and how each one affects the performance of our protocol.

Cluster-to-Cluster Environment

In the cluster-to-cluster environment, the senders are interrelated in their functionality at the application level where they service one mutual purpose, e.g., office of the future [22]. At the communication level, all flows share the same end-to-end path; packets sent by all senders traverse the same path from the source to the destination except may be the first and last hops. As a result the RTT (Round Trip Time), and therefore, the congestion window size for all flows will be almost the same[1], and hence the flows will execute their slow start and congestion avoidance phases closely. This will be similar in nature as demonstrated in Figure 4.1.

Implementing a join protocol requires modifications to the *portal routers*, where joining process and the acknowledgement process takes place. During the connection establishment stage of each TCP flow the portal router intercepts the messages (of the 3-way handshake protocol). Once the receiver accepts the TCP connection (the second message in the 3-way handshake protocol) the portal router creates a join buffer and an agent to control the buffer. The agent keeps track of the source, destination, and port numbers of the end points. Each packet sent by the source is intercepted and the portal router hands the packet to the appropriate agent (shown in Figure 4.2). The agent places the packet in the outgoing buffer that is shared by all the flows through the portal router and sends an acknowledgement to the sender (as if it were the receiver). This process is called as *immediate acknowledgement*. The portal router intercepts the acknowledgment packets and sends them to the appropriate agent which releases the appropriate packets from the buffer.

[1] The size of the packet might affect the value of RTT, which we prefer to ignore in our initial study.

The throughput of all senders in our simulations improves dramatically because join process shifted the process of acknowledging from the end point of the connection to the closest router to the sender. Therefore, a TCP sender enjoys faster arrival rate of acknowledgments, which leads to faster increase in sending rate, and hence an increase in sender's throughput.

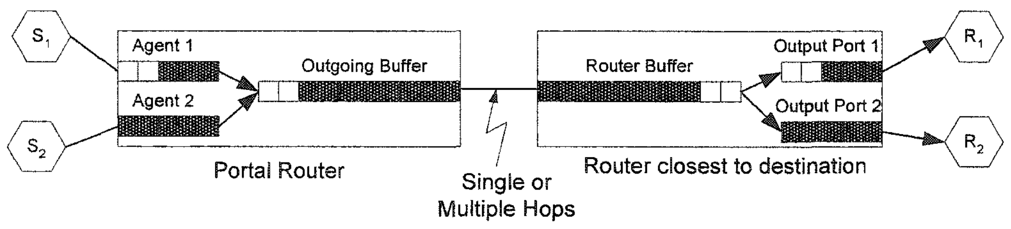


Figure 4.2: Two senders S_1 and S_2 are in communication with receivers R_1 and R_2, respectively. The agents are only at the portal router.

Fairness protocols can be implemented at the portal router where the different agents are scheduled based on the protocol to deliver the packets to the outgoing buffer. Another advantage of our proposed protocol is that the packets in the outgoing link can be sent at the full bandwidth of the outgoing link. Also the proposed agent scheme allows one to design a *delayed acknowledgement* scheme wherein the agent at the portal router waits for the acknowledgement to arrive from the receiver before acknowledging the sender.

Internet environment

The Internet environment is highly dynamic where connections are created and terminated independently. In such an environment, executing the join protocol at the time of connection establishment can cause unnecessary overhead. For example, if there is only one flow traversing a link or there is no congestion in the network, executing the protocol causes additional delay. Therefore, it is important to know when and where to execute the protocol to conserve resources and increase efficiency.

Consider the Internet environment given in Figure 4.3. Router A will serve as the portal router for TCP streams S_1 and S_2 since both share the outgoing link from A. Router C will be a portal router for stream S_3. The combined stream of S_1 and S_2 from router A and the stream S_3 share the same outgoing link to router D and hence the streams have to be joined. The join protocol on router A performs similar function to the ones discussed for cluster-to-cluster applications. Note that, as mentioned previously, it uses the message intercepts of the three-way handshake protocol to establish the agents. When the stream S_3 joins the network, the router C should create an agent for both S_3 and the combined flow from router A. If we assume that every flow entering a router is either a combined flow or direct flow from the application, then looking at the ports from which the packets are obtained at the router, we can create an agent from each port.

There are many cases in which the join protocol need not be executed and the flows can remain independent that is without having to be joined. This is true, especially in cases where the flow volume is small and the bandwidth is very high. In these cases a router may get packets belonging to different flows on the same port. When we decide to join the flows due to increased flow data or addition of other flows from other ports on the router, the join protocol requires that agents be created for each flow. The agents can recognize each flow (the IP address of the source and its port number) from the TCP header and perform necessary action to join the flows as mentioned for the cluster-to-cluster applications. For example, with the addition of flow S_3, three agents will be created, one for each of the three flows. Upon receiving the packet from say flow S_1 the agent will send an acknowledgement packet to the source which will be intercepted by router A's agent corresponding to S_1. This agent will release the acknowledged packet. In some sense, as new join protocols are executed down stream, the responsibility of packet delivery to the receivers is delegated to those routers.

Router C may need to additionally change the traffic rate for the combined flow (generally decrease). It may inform router A of this using the acknowledgement packets that are sent from the receiver to the sources S_1 and S_2. Note that, when router A decreases its outgoing rate its buffer may be filled up and packet losses could occur. But, the flow control mechanism that can be implemented in router A will inform the respective agents to adjust the window size of the senders by decreasing the acknowledging rate to the sender.

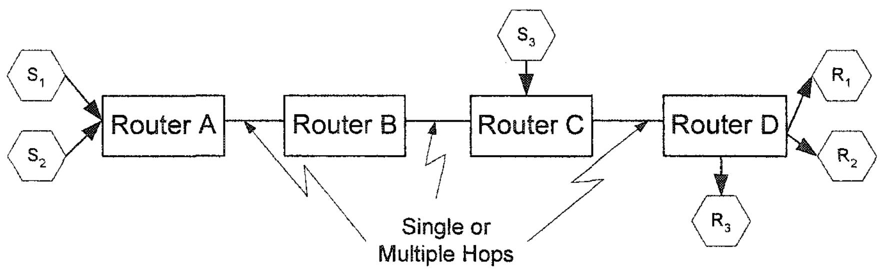


Figure 4.3: Router A will run the join protocol to combine traffic from sources S_1 and S_2. Router C will run the join protocol to combine traffic from S_3 and the combined traffic of S_1 and S_2.

4.4.1 When to Join

In this section we will explain why the timing of executing the join process affects the efficiency of our protocol and propose a method to ensure that a router executes the split-and-join process only when it is necessary. The decision of when to execute the algorithm is essential because executing the algorithm incurs processing and buffer over head at the *portal routers*.

Due to its complicated nature, a dynamic Internet environment imposes some difficulties when deciding when and how to implement the join protocol. The goal is to make sure that the improvement gained from join exceeds the cost (overhead) of implementing the algorithm. With this requirement in mind, the first, and most obvious, condition is that there must be more than one flow sharing the same link. A single flow traversing the network will be able to utilize the available network resources efficiently, as there is no fluctuation in the available bandwidth, which is usually caused by competition with other flows. This can be accomplished by monitoring the connection establishment and disconnection packets that are required as part of the transport layer protocol. Second, the available network resources at the shared link must be insufficient to meet the total requirement of all flows. Without this condition, the bottleneck will be the application sending the data and not the network or transport layer protocols. Routers can monitor the queue lengths and number of packet drops as a good indication of the available network resources.

Monitoring the flows

The data passing through the joining router are either data packets or acknowledgment packets depending on their direction. The router maintains an array of pointers for all flows. Each distinguish TCP flow (IP address, port number) is assigned an entry in the array. One entry is a pointer to a doubly link list of packets and the other is a counter that stores the sequence number of the most recent acknowledged packet.

When a data packet arrives at the router for the first time, the router creates a new entry in the array associated to the source IP address and port number of the packet. Then, it buffers the packet and makes the pointer points to. When an acknowledgment arrives, the router updates the most recent sequence number acknowledged and removes the packet from the buffer.

Automatic Join

As we explain above, a router decides to split-and-join when it experience congestion, which means a packet or more have been dropped from an outgoing link. Then, the joining router looks up the IP address and port number of each packet and adds them to the array and adds the packet at the end of the doubly linked list. After that, the router uses the data stored at the array to create an acknowledgment to send it immediately to the sender. Next, the router swaps the packets from the incoming buffers to the outgoing buffer fairly and then forwarded them with a FIFO discipline on the outgoing link. Because the router uses immediate acknowledging, it is responsible for making sure that the packet reached the destination with no errors. We suggest that the router uses a retransmission timer similar to the retransmission timer used in TCP. The advantage in our case is that the fluctuation in RTT value is less compared to the case of normal TCP.

The value of RTT depends on three components: buffering time, processing time, and link delay. In case of one flow traversing the path, the sender will maintain its sending rate at some accepted level. Every packet will experience the same level of delay at every one of the three components. Unless another flow comes in and start pumping data into the network, the buffer occupancy level will remain the same. Therefore, the RTT experience by every packet is equal and the joining router measure it once at the beginning of the joining process, which simplified the flow control.

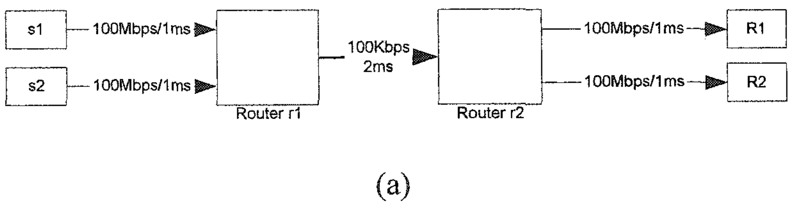
After receiving the packet, the TCP receiver sends acknowledgement back to the sender. When the acknowledgment arrives at the router, it will examine the TCP header and free the acknowledged packet(s) from the array. Since the router already acknowledged the sender, no need to forward the packet again to the sender.

4.5 Improvements to the algorithm

The most important advantage of the algorithm is that it joins all flows into one flow. Therefore, one flow is traversing the network. Due to this fact, we can add an important modification to improve the performance of our algorithm. It is about setting up the sending window size to be equal to the link bandwidth. It is un-usual that the link does not change its bandwidth capacity nor the router changes its processing time. Hence, the only variable is the router's buffer delay, which can be fixed by fixing the sending rate at the sender side. So, if the router knows the bottle neck bandwidth, its buffer size, and its processing speed, it can fix its sending rate enough to utilize the network resources efficiently without causing congestion. As a result, the joining router sends packets out at high rate immediately and does not need to go through TCP's AIMD probing process.

4.6 Simulation and results

In our experiments, we used the network simulator *ns* [25] to conduct our simulations. We used a modified version of TCP receiver, which we call *split sink*, in order to simulate the immediate acknowledging process by the joining router. Also, at the outgoing port of the router, we used a modified version of TCP Tahoe, that we call forwarder, in order to simulate the process of sending the packets reliably via the joining router. Using a complete TCP stack to simulate the reliable forwarding process at the router requires extra processing activities than our algorithm. Two processes are executed by the full implementation of TCP Tahoe are not required by our algorithm. These two processes are the 3-way hand shake process and the updates of RTT with the arrival of each acknowledgment. Therefore, with careful fine grain implementation, our algorithm should perform better than the results presented in this paper.



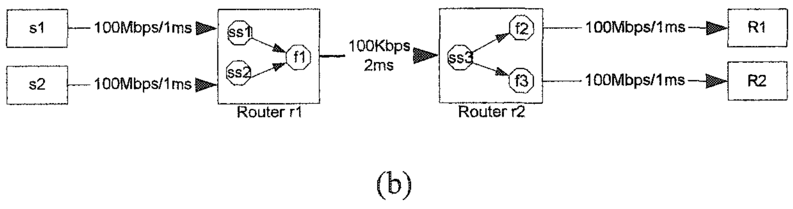


Figure 4.4: static environment (a) TCP Tahoe (b) Join algorithm.

The first set of experiments simulates the Tahoe TCP and joining algorithm in a network model similar to the one in [17] and is shown in Figure 4.4. TCP flows initiated at sources S1 and S2 traverse links that have similar bandwidths and delays of 100 Mbps and 1 ms, respectively. This high bandwidth and low delay settings are used to represent a local area connection between the senders and r1. In case of TCP Tahoe, as shown in Figure 4.4a, the sources sends TCP packet normally to the nearest router, which will route the packet to the destination. In joining algorithm experiment, Figure 4.4b, at the first router, r1, the two flows are split and two TCP split sinks are created: *ss1* and *ss2*. At the same router, a *forwarder, f1* and an associated buffer are created and packets from the *ss1* and *ss2* are sent to the buffer of *f1*. At the last router, r2, the reverse procedure is executed using one split sink, ss3, and two *forwarders*, *f2* and *f3*. When ss3 receives packets belong to different flows, it needs to look at the packet header to learn about the sender. Depending on the source information, the split sink will direct the packets to the appropriate forwarder. For example, in our simulations the end split sink (*ss3*) distinguish the packets of two flows using a flag we impeded at the packet header and named it *agent id*. When *ss3* receives a packet it examines the *agent id* flag. If it equals 1 then ss3 will hand the packet to *f2* otherwise it will hand it to *f3*. After receiving the packets, the two forwarders have only one way to send the packets as each forwarder is attached to only one TCP sink.

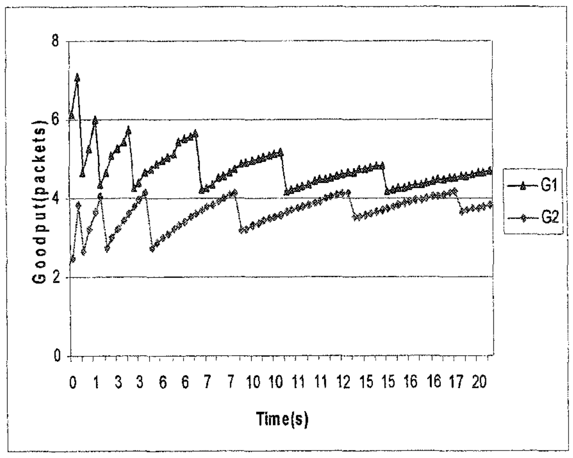


Figure 4.5: TCP Tahoe goodput

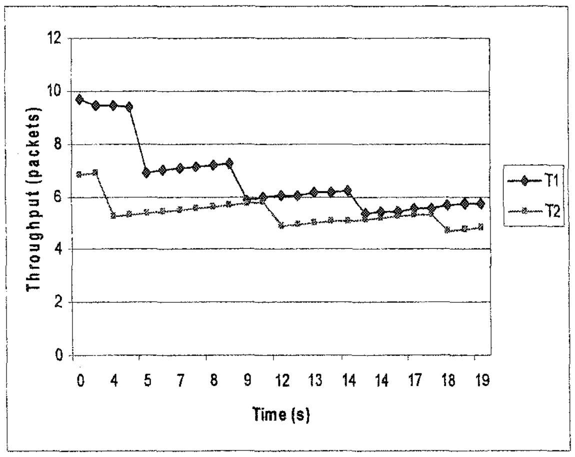


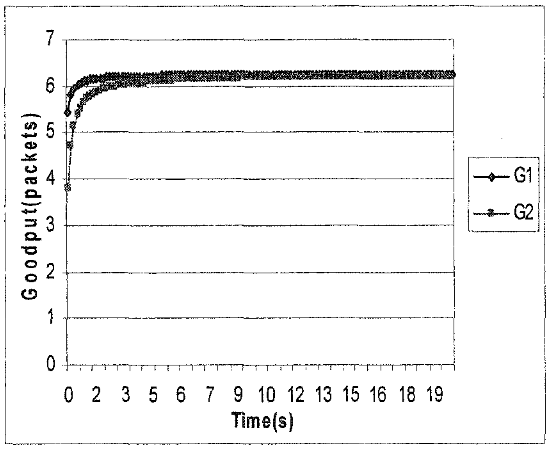
Figure 4.6: TCP Tahoe throughput

The path between r1 and r2 is the main source for congestion. The path is shown as one hope link, which has the smallest MTU among the links along the path between r1 and r2. In the simulation, we set up the bandwidth of this link to 100Kbps and the delay to be 2ms to represent the effective bandwidth assigned to the flows initiated by S1 and S2.

The results using our algorithm show that there is a substantial increase in both throughput and goodput compared to using TCP Tahoe. The level of competition among the two flows sharing the same path is eliminated in our algorithm as shown in Figures 4.7 and 4.8 while it is clearly active in the case of TCP Tahoe as shown in Figures 4.5 and 4.6.

The goodput achieved by using the joining algorithm have two advantages. The first is its fluctuation-free performance as shown in Figure 4.7, which led to better over all goodput compared to the TCP Tahoe, which is shown in Figure 4.5. Also, in the case of joining algorithm, all receivers receive equal amount of data during the life of the communication as depicted in Figure 4.7. In the Figure, both flows have almost identical level of goodput during their life time. In contrast, in case of TCP Tahoe, flow 2 achieved lower goodput performance compared to flow 1 due to the competition nature as shown in Figure 4.5. The performance achieved by the joining algorithm ranges between 33 and 79 percent better than TCP Tahoe.

We calculated the throughput; which is the size of data sent by the sender per time unit, of the communicating flows in both cases of TCP Tahoe and joining TCP. Figures 4.6 and 4.8 showed the throughput of all flows in case of TCP Tahoe and joining algorithm, respectively. The joining algorithm achieved performance of 9600 percent, which is due mainly to the immediate acknowledging by the first joining router. It does not mean that the destination received the data but it will allow the sender to send more data.



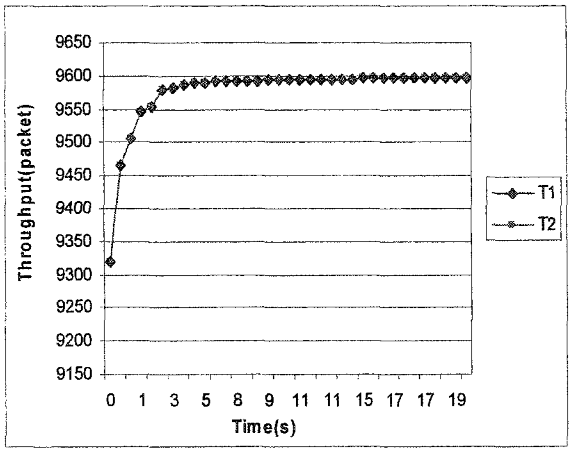


Figure 4.7: TCP goodput - join algorithm **Figure 4.8:** TCP throughput - join algorithm

The next set of experiments is intended to measure the performance of the joining algorithm on the Internet. These experiments were conducted on a system model similar to the one shown on Figure 4.9. In this environment, there are no constraints neither on the position of the senders nor their inter-relationship. The communication bottle neck of the network shown in Figure 4.9 is the link connecting the last routers and it has a bandwidth of 100 kbps and propagation delay of 2 ms. We assigned all other links high bandwidths of 100 Mbps and low propagation delay of 1 ms to make sure there is competition among the flows at the bottle neck link.

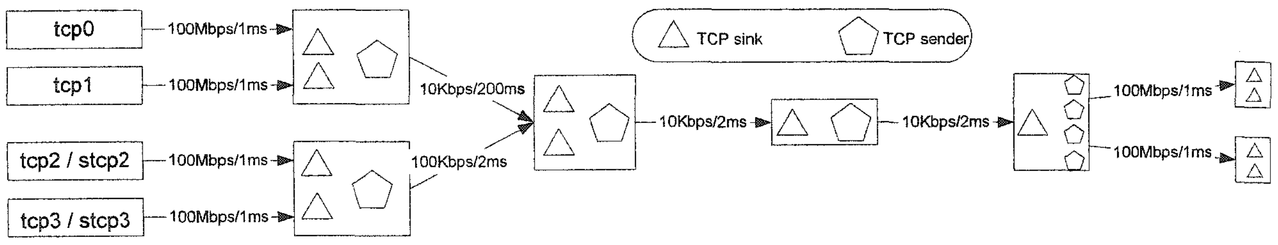


Figure 4.9: Simulation topology configuration.

In Figures 4.10 and 4.11, the performance of TCP Tahoe is compared to the case of joining algorithm. The comparison metric is the total goodput achieved by all flows traversing the network. In the experiment, all flows started an FTP session at the same time and ended at the same time. All FTP applications have unlimited input of data to send.

As shown in the Figures, our algorithm achieves improvement ranges between 48 percent and 77 percent over TCP Tahoe. The main sources of improvement are the absence of competition among flows, and the immediate acknowledge process that enables the application layer at the sender to send packets down to the transport layer faster. This is due to the fact that the transport layer is freeing packets very fast from its buffer as it receives acknowledgment fast from a close by joining router and not from the end receiver.

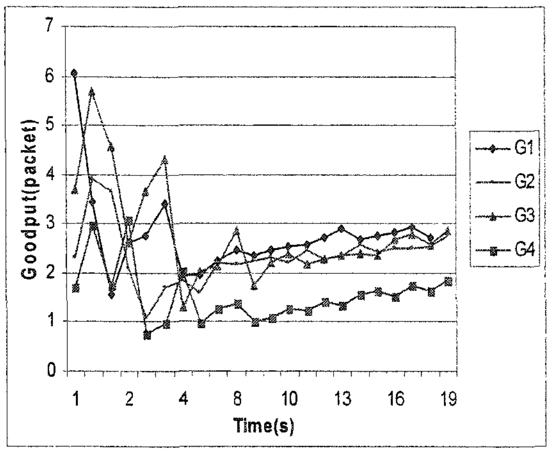


Figure 4.10: TCP Tahoe goodput

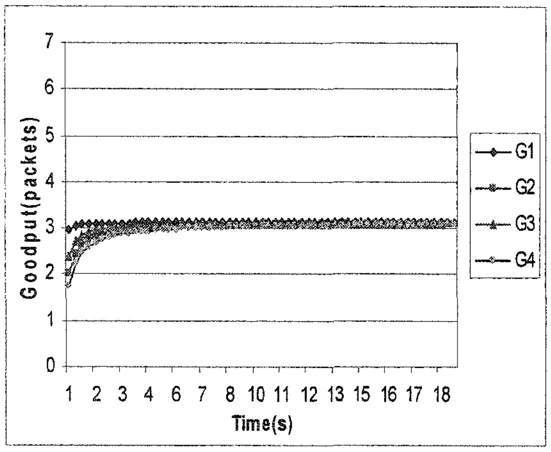


Figure 4.11: Join goodput

The results shown in this section do not include the improvement discussed in previous section. We believe if they are implemented and tested it will improve the performance of the algorithm more. Also, in our experiments we used a complete TCP stack including the connection establishment and congestion control. We believe that a careful design of the algorithm that eliminate these processes will lead to less over head and therefore better performance for the joining algorithm.

4.7 Summary and conclusions

TCP flows tend to compete with other flows for the available resources in the network. In this article, we identified the impact of such competition on the performance of TCP flows and we propose a *join* protocol that combines many TCP flows into one flow so to reduce the level of competition in the network, which lead to substantial improvement in the performance of individual TCP flow and as a result increases the level of utilization of network resources.

The *join* protocol achieves fairness among all flows sharing a link or a path in the network. This fairness is a result of a fair queuing algorithm at the joining router as well as the *join* agent, which join all flows into one large flow by the join agent. Using modified *network simulator* (*ns*), simulation results prove the increase of data throughput and goodput and also show a fair utilization of network resources among communicating flows.

Chapter 5

Tracking Updates in Distributed Wi-Fi Architectures: Flooding Protocols and Application Fine Tuning

5.1 Introduction

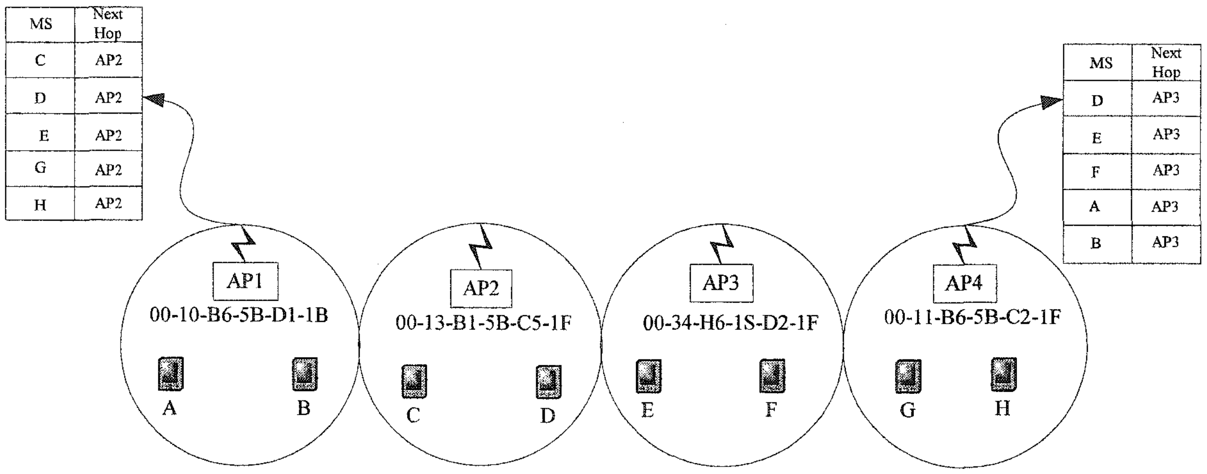
The distributed Wi-Fi architecture consist of a set of wireless access points (AP's) distributed in a metropolitan area and mobile stations that communicate with other fixed and mobile stations using the access points. The communications between AP's and between stations and AP's are done using the IEEE 802.11b protocol. Each station is recognized using a unique MAC address. If two stations *a* and *b* are *attached* to access points AP_1 and AP_2, respectively, then *a* and *b* can communicate with each other after finding a routing path between access points AP_1 and AP_2. Thus any two nodes can communicate with each other provided the access points to which they are attached are known. If the stations are mobile, then the station to which they are attached at any given time is unknown to all the other stations and access points. Determining the access points to which a given mobile station is attached is called *tracking*.

Constant tracking of mobile users is essential to maintain connectivity at all times otherwise, packets will be routed to wrong addresses leading to loss of connectivity and termination of TCP sessions. Most tracking systems proposed in the literature use the flooding as the fundamental technique to determine locations of destination stations. This flooding is either performed on demand when a packet is to be routed or performed in the background for updating the routing table at each access point. Communication between two nodes in the distributed Wi-Fi architecture uses the IEEE 802.11b protocol whose performance is directly proportional to the amount of contention in the channel that increases with the number of users. The level of contention is high especially due to flooding packets and this in turn increases communication delays for many applications. Large the number of mobile users larger will be number of tracking information that needs to be flooded causing severe degradation in the performance of the entire network.

In this research, we will develop tracking protocols and evaluate these protocols in terms of time and resources required to provide connectivity between mobile stations. Our protocols for tracking updates will exploit station locality information to perform controlled flooding, use backward learning, and forward addressing techniques. We will evaluate the protocols and fine tune the protocols taking into account mobility rates and application characteristics.

5.2 The Distributed Wi-Fi Architecture and Routing

Distributed Wi-Fi environment consist of mobile stations, access points, basic service sets, and distribution system. Mobile stations are laptops or palm held devices running different types of applications, which require communication with other stations. The mobile stations can move around the coverage area without losing on-going communication or the need to restart the connection establishment process.



MS	Next Hop
C	AP2
D	AP2
E	AP2
G	AP2
H	AP2

MS	Next Hop
D	AP3
E	AP3
F	AP3
A	AP3
B	AP3

Figure 5.1: Schematic representation of routing tables in Wi-Fi networks.

The coverage area is divided into cells or basic service set (BSS) where communication at each BSS is controlled by an access point (AP). A mobile station can communicate with stations inside the same BSS (intra-BSS communication) or with stations in other BSS (inter-BSS communication). Extended service set (ESS) is a collection of BSS. Access points in the same ESS cooperate to support the mobility of the stations roaming the covered area. As the mobile stations disassociate from and associate to access points, routing tables will contain invalid routing information. The process of updating the routing information is the responsibility of the *tracking* system. When a mobile station moves to a new BSS, it must register with the access point. First, the mobile station listens for a beacon from the new access point. After receiving the beacon containing the MAC address of the access point, the mobile station sends an association request that contains its MAC address. After accepting the registration request, the access point acknowledges the request, and the mobile station can start communicating with other mobile stations.

The routing table at each access point in the distributed Wi-Fi architecture consists of the addresses of the "known" (those mobile units it knows exists in the system) mobile units and the next hop MAC address. For example, when mobile unit B in Figure 5.1 wants to send a message to mobile unit E it uses the routing table at AP_1 which indicates that the packet has to be forwarded to AP_2. The goal of the tracking system is to keep up-to-date information on all mobile units as they move around. If we have available up-to-date information on each mobile unit including the access point to which they are currently associated with, then we can use a standard static routing algorithms to route the message between any two access points. This can be easily accomplished since the access points are non-mobile.

The access point must inform other access points about the new location of the mobile station by sending a message that we will call as the Route Update Packet - RUP. Upon receiving the RUP, every access point updates its routing table to reflect the new association between the access point and the mobile station. As a result, any further packets destined to the mobile station will be forwarded to the new access point.

In order to maintain movement freedom for users and easy deployment for hotspots, Wi-Fi uses MAC 802.11b technology as its distribution medium for communication between access points. Therefore, access points must content with other access points for a channel before sending RUP's. The level of contention among access points increases with the increase in the number of access points and RUP's. Such increase in contention will result in increase of data packet drops by the MAC layer, which in turn will affect the upper layer protocols such as TCP.

5.3 Flooding Protocol and its variants

In mobile networks, the sender, the receiver, or both can be mobile. Therefore, it is not enough for routers to know the address of the destination in order to perform packet routing. Information about the current location of the mobile station is essential important to complete packet routing successfully. For this reason, many location management techniques proposed in the past for one hop wireless networks such as cellular networks, e.g., IS-54, IS-95, and GSM. Such technique depends on special backbone servers that receive location update messages from different base stations and store it locally. When a base station wants to communicate with a mobile station, it must find its location and hence sends a request to the backbone servers. The number and the size of the messages exchanged between base stations and backbone servers were not a bottleneck when designing such protocols because the medium connecting the base stations to the backbone servers were wireline that has high bandwidth and low delay characteristics.

On contrast, in distributed Wi-Fi environment, the links connecting the access points are wireless with low bandwidth and high bit error rate compared to wireline links. It is essential for any location management protocol to use less number of messages to support routing to mobile users. Therefore, in our protocols, we emphasizes our design heavily on informing the necessary number of nodes about any location changes of mobile stations in the network and in the same time keep the number of exchanged messages as small as possible.

In the area of mobile Ad-Hoc many routings protocols were proposed [33] to improve packet routing in such a dynamic environment. These protocols can be organized into three categories as in [33]. These categories are: flat routing, hierarchical routing, and position based routing. In such protocols, the environment is composed of many mobile nodes that keep moving at all the times and without central control. In our research, we assumed a Wi-Fi environment where access point are stationary which uses IEEE 802.11 MAC protocol to communicate with other access points as well as mobile stations. Therefore, we only have to keep track of the mobile nodes and information pertaining to their movement has to be distributed. Moreover, only access points need to exchange such information, as every access point is responsible for all mobile stations residing in its area of coverage. Also in mobile ad-hoc environments, when mobility is high, flooding is the only way to route packets [34]. In Wi-Fi environments routing information has to be exchanged only between the access points. Also in Wi-Fi environments a mobile station must register with an access point and it can be traced. After registration one can use some form of multicasting to deliver a packet to a fast moving mobile station. For example, assume mobile station is currently registered with access point AP1, which has AP2, AP3 and AP4 as its close neighbor. It is most likely that the mobile station will stay in the neighborhood of AP1 for longer time in comparison with the time it will stay attached to a particular access point. It is clear that multicasting the packets to all four access points will increase the chance of successful delivery while avoiding the expensive task of broadcasting to every node in the network.

In Wi-Fi environment, the communication between mobile stations and access points and between access points uses MAC IEEE802.11b protocol. The messages exchanged in this environment are of two types: data messages and control messages. The data messages, as its name implies, carries data from a sending station to one or more destination mobile stations and the control messages carry the association and authentication packets exchanged between an access point and a mobile station and the route update information between the access points in the network. When an access point wants to send a message to another access point, it must content for a channel. This contention uses the standard request to send (RTS) and clear to send (CTS) messaging protocol. Other neighboring access points must be idle during the reserved time by the communicating access points. Therefore, it is obvious that if the number of messages exchanged increases, then the waiting time experienced by non communicating access points for a channel will be longer and the chances of having a collision also increases. Our hypothesis is that if the over all number of messages exchanged in the network decreases, contention decreases, and hence, the idle time decreases, which will improve the performance of upper layer protocols such as TCP. In our research, we focused on reducing the number of control messages exchanged between access points used to update the routing information and all associated control messaging such as RTS and CTS control messages.

The purpose of a tracking system is to inform enough number of nodes in the network topology to make all mobile stations reachable. The current approach adopted by the IEEE 802.11b standard involves the creation and updating of a two columns tables. The first column contains the unique MAC address of every known station in the network while the second column contains the MAC address of the access point in control of the mobile station. Hence, if an AP wants to forward a frame to a station, say S_1, it looks up its table to find the MAC address of the controlling access point and sends the frame to it. Then, the controlling AP forwards the data frame to the destination mobile station. When a mobile station moves to a new BSS, it sends an RUP to other APs to ensure that the mobile station is reachable.

In the following subsections, we propose three tracking protocols suitable for Wi-Fi environment: Pure flooding, Tree flooding, and Spatial tree flooding.

5.3.1 Pure flooding

In pure flooding, the source access point sends an RUP to all access points within its wireless transmission range. Similarly, every receiver-AP broadcast the RUP to its neighbors and the neighbors to their neighbors and so fourth until all the nodes in the networks receives the RUP. In order to avoid cyclic forwarding, an access point drops any RUP packet seen before. A pair of source id and packet number is used at every access point to identify each RUP. It is possible that the access point uses broadcasting to send the RUP to every reachable (neighbor) access point in order to reduce the time needed to propagate the RUP to all access points in the network. For example, if the AP sending the RUP has 4 neighboring AP's and if it uses unicast, it will take 4 channel-times to send the RUP to all the neighbors. When using broadcasting, the AP will send 4 copies of the RUP to all the neighbors in one channel-time.

5.3.2 Tree flooding

In this protocol we will represent the network topology as a graph G = (V, E), where V is the set of vertices that represent the set of access points in the network and E is the set of edges that represent the set of links in the network. A directed link, e_l, which start at AP_j and ends at AP_k means that AP_j can transmit to AP_k. The goal of this protocol is to modify the transmission power of the access points in order to create a spanning tree T from graph G. The weight of each edge represents the physical distance between the two access points.

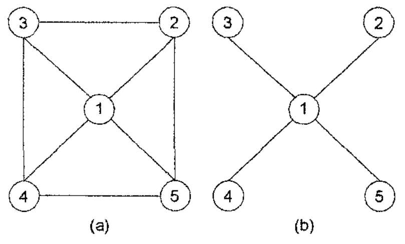


Figure 5.2: Network topology (a) before and (b) after power tuning.

The advantage of such architecture is to reduce the number of RUP's sent between access points to update the routing information. If the power transmission of the access points is not set carefully, the resulting topology might not resemble a tree shape. If this is the case, then some access points will receive more than one copy of the RUP by two or more neighboring access points. In Figure 5.2 (a), assume access point 1 is the source of the RUP. First, it will send 4 copies of the RUP to its neighbors; access points 2, 3, 4, and 5. Assume that the distance between access point 1 and other access points is proportional to the access point number, e.g., the distance between access point 2 and access point 1 is shorter than the distance between access point 1 and access point 3. With this assumption in place, the order in which other access points will receive the RUP is 2, 3, 4, and 5. When AP 2 receives the RUP, it will send 2 copies to its neighbors; AP 3 and AP 5. Similarly, AP 3 will send one copy to AP 4 and at the same time AP5 will send another copy to AP 4. Thus, the total number of RUP's exchanged is 8.

Tree flooding reduces the number of RUP messages exchanged between the access points to carry the routing table updates. As shown in Figure 5.2 (b), AP 1 broadcast 4 copies of the RUP to AP's 2, 3, 4, and 5. All access point in the network received a copy using only 4 copies of RUP. In this particular example, tree flooding uses half the number of RUP copies used in pure flooding to propagated routing table update to every AP in the network.

The basic idea of tree flooding is to send RUP messages over the edges of the tree instead of sending them over every edge in the network, as it the case in pure flooding. Since the number of edges in a tree is n-1 for an n node network the worst case number of RUP messages is n-1.

Once a tree is constructed the power at each access point should be adjusted in such as way that there is communication only between parent and children. We have to do this in order to reduce the number of collision resulting from simultaneous RUP transmissions. This also can be done at design time by carefully positioning the access points. Transmission scheduling schemes have been suggested in [30]. These scheduling schemes incorporate delays to avoid collisions and keep the total broadcasting time to a minimum. One of the approached to reduce the broadcasting time is to build a breadth first tree, but there are instances of the tree which could result in collisions due to simultaneous broadcast. Gandhi et. al. [30] has proposed a technique to schedule broadcasting on the tree to reduce the total the number of collisions.

Yet another scheme that uses the minimum connected dominating set has been proposed by Stojmenovic [31] to reduce the total number of transmissions of RUPs. A connected dominating set in a graph is a subset of vertices such that every vertex is either in the subset or adjacent to a vertex in the subset and the sub graph is connected. The minimum connected dominating set is such a vertex subset with minimum cardinality. One can construct a spanning tree of the sub graph induced by the minimum connected dominating set and perform routing on the tree. Radhakrishnan et al [32] proposed a technique to schedule transmissions on the minimum connected dominating set to reduce the total time for broadcasting.

5.3.3 Space partitioned tree flooding (SPTF)

When the network coverage area grows to cover a large terrain, mobile stations are likely to roam in one region of the network. The size of the region depends on the behavior of the users such as their speed of movement. If the intended users of the network are likely to travel on foot, the region is much smaller in comparison when the intended customers are using automobiles. Thus, deciding the size of the region is an engineering decision and careful study helps improving the performance of the communication in the network.

The main idea of SPTF is to limit the number of RUP's exchanged by the access points in the network while keeping all mobile stations reachable. In order to achieve such a goal, SPTF divides the network into smaller regions where each region contains a group of access points. Selected access points are of special type that has extra routing functionality. We call them Portal Access Points or PAPs. The distance between any access point and its PAP is shorter than the distance between the access point and any other PAP in the network.

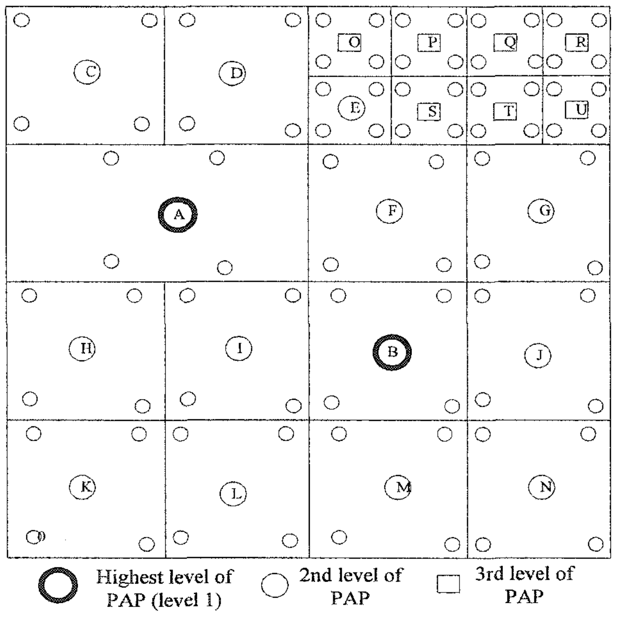
In our model, two types of communication can exist: *intra-region* and *inter-region* communications. In intra-region communication, the mobile sender and the mobile receiver are located in the same region while in region-to-region communication they are in two different regions.

Intra-region communication started when the mobile sender sends a request to begin communication to the receiver. It will send the packet out and the nearest AP will be responsible to forward it to the destination. Therefore, the AP must know the address of the controlling AP, which the mobile receiver is registered with. The PAP has updated tracking information for every mobile station in its region. The tracking updates are generated by local access point in the region as needed and then sent to the PAP. When an access point receives a message from a mobile sender, it consults its PAP about the current location of the mobile receiver by sending a request packet that includes the address of the mobile receiver. Since the mobile receiver is in the same region, PAP will have updated tracking information and it will reply to this request. The reply message contains the address of the AP which the mobile receiver is currently attached to. Once the sender AP, which is the closest AP to the mobile receiver, knows the address of the receiver AP, it can utilize existing routing protocols to route packets to the mobile receiver.

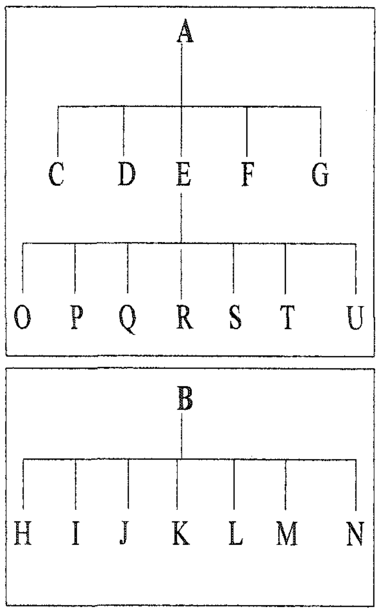
In inter-region communication, the local PAP of the sender AP does not have the updated tracking information about the receiver mobile station because it is located in another region. The process of sending packets from the mobile sender to the mobile receiver starts when mobile sender sends a packet destined to the mobile receiver. After receiving the packet, sender AP needs to know the address (identity) of the receiver AP. Immediately, sender AP consults its local PAP for the address of the receiver AP. Since it is in another region, local PAP does not have the location information. Next, the local PAP consults its parent PAP, and if the parent PAP does not have the requested information, it will repeat the process and consult its parent PAP and so on until an answer is found or all PAP in the PAP-chain is exhausted. The PAP who has the location information will send the requested information as a unicast packet backward from one PAP to its child, until it reached the requesting PAP.

In order to keep the number of RUPs as low as possible, we can take advantage of the assumption that mobile station will likely be roaming in the same region most of the time. When a mobile station (MS) disassociates from one AP and associates with another AP, the new AP sends an update message to only the local PAP. When an MS associates with an AP outside the old region, the new AP sends location update packet to its local PAP, which will inform the old PAP about the movement of the MS. Therefore, the old PAP knows that the MS is no more in its region and it is in the other region but it does not know the exact location of the MS. When a location request arrives from remote region, the parent PAP will send the request to old PAP, which will send an update message informing the parent PAP that new PAP is in charge of such MS. Upon receiving the update, parent PAP sends the request again, but to new PAP this time.

The PAP's are organized in a hierarchical structure. At the first level (the bottom most level in the hierarchy), each cluster is assigned a unique PAP. Depending of the level of traffic, the number of access points and mobile nodes, and the level of communication quality required. Next the individual clusters are grouped in one larger cluster. The PAP of the larger cluster will be one of the PAPs of the smaller clusters. We call the smaller clusters "lower-level clusters." As needed, more grouping is performed and one of the PAPs is chosen to be the leader PAP of the new group of clusters.



(a)



(b)

Figure 5.3: Hierarchical relationship among PAPs.

In Figure 5.3 (a), the network is divided into 21 clusters of different sizes. PAP A is at the top level of the PAP hierarchy of the upper part of the network topology while PAP B is the top level of the hierarchy of the lower part of the network. Figure 5.3 (b) shows the hierarchy relationship among all PAPs in the network.

The process of sending RUP is started when one of the mobile stations moves from one BSS to another. First, the mobile station will request to associate with the new access point. After that, an authentication processes is exchanged between the two. If the mobile station is authentic, the new access point sends a RUP to its local PAP.

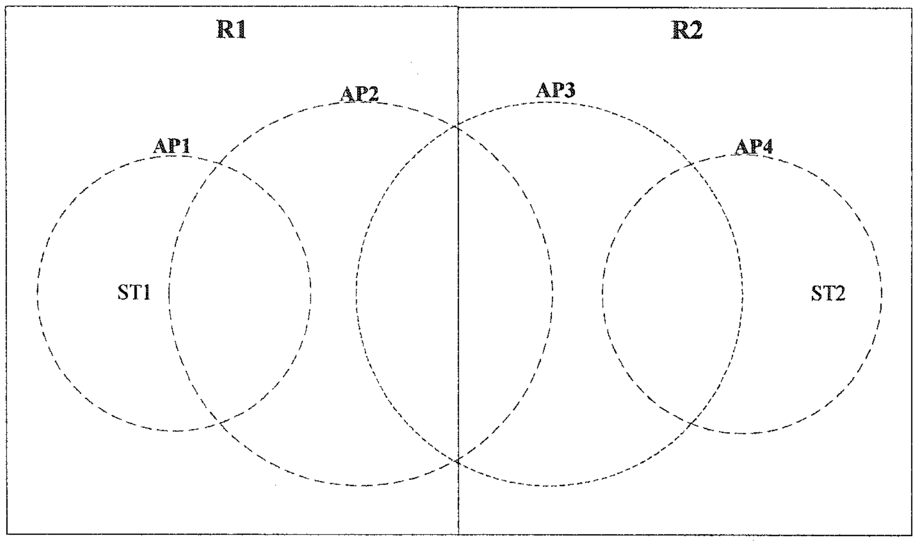


Figure 5.4: Inter-region data communication.

To explain this further, let us assume the ad hoc network in Figure 5.4. Further assume that access points AP_1 and AP_2 are in the same region, R_1, and access points AP_3 and AP_4 are in another region, R_2. Let station ST_1 be in BSS of AP_1 and station ST_2 in BSS of AP_4. We will choose AP_2 and AP_3 to be the PAP's of R1 and R_2, respectively. When ST_1 wants to send a message to ST_2, it will send the message to AP_1. The header of the message includes the MAC addresses of the source and destination mobile stations. When AP_1 receives the message it will not find a match for ST_2s' MAC address in its routing tables because ST_2 belongs to another group. Therefore, it is the job of the PAP to forward it to the controlling access point of ST_2, which is AP_4. Next, AP_1 sends a location update request to AP2 for ST2. After receiving the request, AP2 informs AP1 that ST2 is attached to it. Following that AP1 sends the packet to AP2 and it forwards the packet to ST1 after looking up its MAC address from the header of the packet.

At the design level, the number of PAP's in each region can vary. It could be as little as one PAP or as many as *r* PAP's, where *r* is the number of neighboring regions. The choice has to be made such that the number of messages exchanged in the network to propagate RUP's is as small as possible. Also, the position of the PAP must be chosen carefully to reduce the time it takes to update all other PAP's in the network.

In our research, we chose to use one PAP for each region and that is closest access point to the center of the region. We limited the number of PAP at each region to one PAP to reduce the number of messages exchanged inside the region if we have more than one PAP, which will lead to less-contention inside the region. Also, the position of the PAP inside the region affects the length of time it takes the RUP to reach every other PAP in the network. Figure 5.5 (a) and 5.5 (b) shows an example of how the position of the PAP affects the length of the time it takes the RUP to propagate to every other PAP in the network. The numbers inside the access points represent the time in which the messages was received.

In the Figure, the sender PAP receives a location update packet from one of the access points in its region. Immediately, it sends a unicast message to every neighboring PAP. When the sender PAP is placed at the center of the region, as shown in Figure 5.5 (a), the three other PAP's in the network received the RUP at times 8, 8, and 15, respectively. In contrast, when the sender-PAP is at the edge of the region, the other PAP's received the RUP at times 10, 10 and 17, respectively.

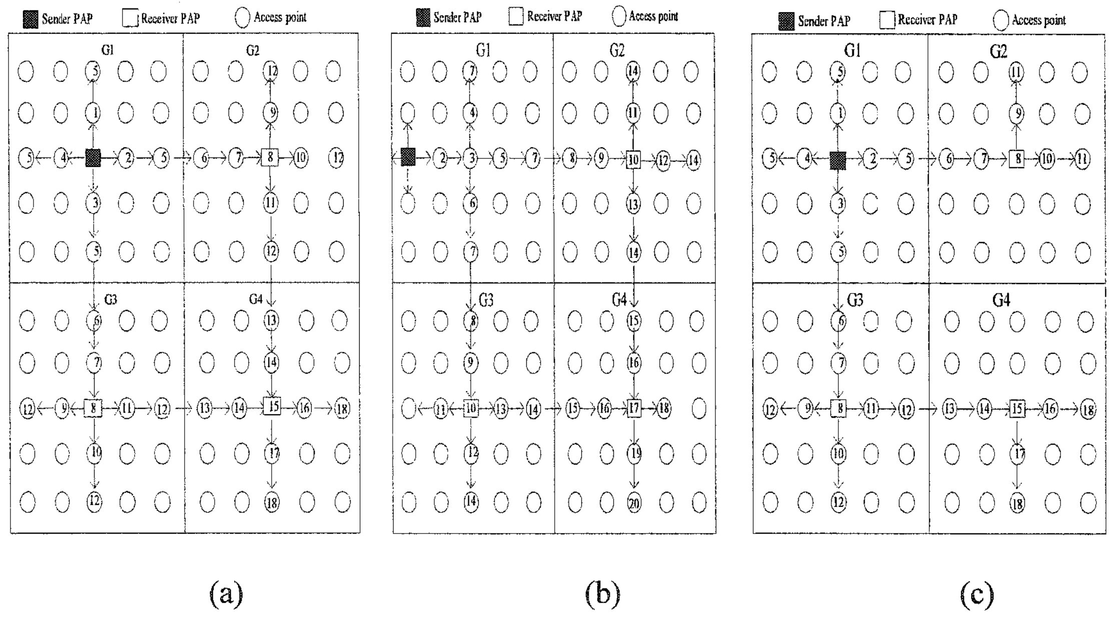


Figure 5.5: RUP propagation between APA: (a) Central PAP (b) Non-Central PAP (c) Central-Tree PAP.

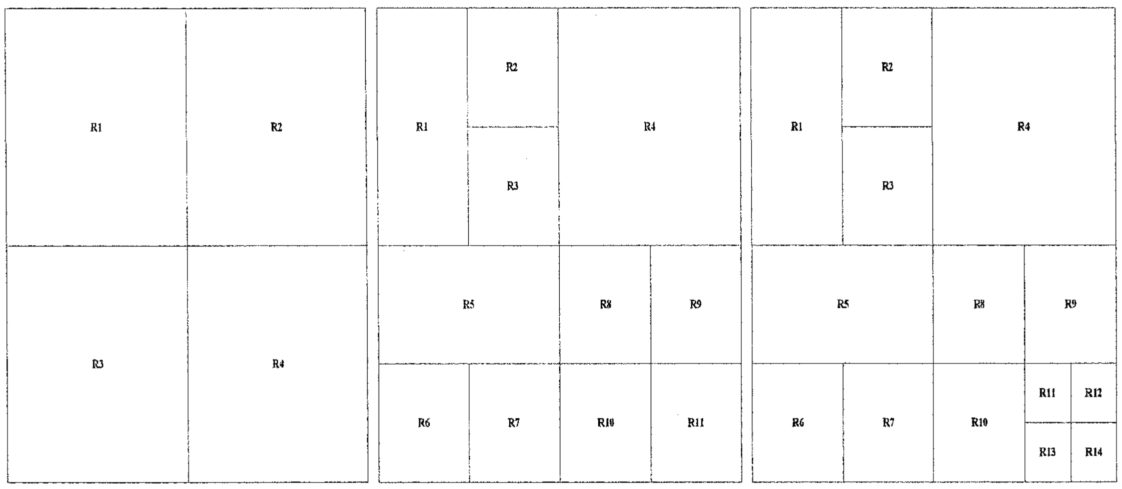
As seen in Figures 5.5 (a) and 5.5 (b), both PAP's of regions R2 and R3 sends location request packets to the PAP of region R4. This messaging redundancy between the PAP's in the network is not necessary as it consumes power, and increases contention. Therefore, we propose to organize the PAP's in a tree shape where the parent PAP sends the RUP's to its children. In such a tree we need to make sure that every child has only one parent to prevent sending redundant messages. As shown in Figure 5.5 (c), the number of messages exchanged to update the PAP's of the network decreases while the time it takes to propagate the route update information is still the same as in Figure 5.5 (a).

5.4 Intelligent Tree Flooding Algorithm

In this section, we will introduce our novel method on how to divide the network topology into smaller regions and construct a spanning tree of the access points inside each cluster. We adopted a hierarchical region representation named quadtree. Quadtree is a hierarchical data structures used to represent spatial data and they are based on the idea of recursive decomposition. It is one of numerous available hierarchical data structuring in use to represent spatial data. One of its applications is the representation of regions. The most quadtree approach used in region representation is named *region quadtree*. This approach is based on successive subdivision of a bounded region into four quadrants. All quadrants have the same size. If the quadrants are not consists of the desired data, they are subdivided into smaller quadrants. This process continuous until each quadrant contains the desired number and type of data [35].

We had the chance to use any type of tree representation such as binary tree or octree. But we chose to use quadtree for two main reasons. The first reason is that using quadtree is a trade off between the binary tree and the octree. The trade off is between the level of tracking traffic experienced by a PAP and the total number of PAP's in the network. In the case of binary tree, each parent PAP has two children PAP's. The tracking requests received by the parent PAP are originated from mobile stations residing in one of the two regions controlled by the two children. In octree, each PAP has 8 children and each child controls a region containing number of mobile stations. It is clear that the tracking traffic is more intensive in the case of octree compared to binary tree. On the other side of the coin, it is clear that the number of PAPs used in the case of binary tree is more. So, we used quadtree as a trade off between the two.

The second reason is that there are well-established algorithms for quadtree data structure. For example, many techniques were proposed for testing adjacency and finding neighbors. Examples of such algorithms are illustrated in more details in [35].



(a) (b) (c)

Figure 5.6: Dividing the area into quadrants, (a) First level (b) Second level (c) Third level.

We started by identifying the area to be covered by communication. After that, the area is divided into quadrants depending on the accessible stations density at every quad. Figure 5.6 shows an example of how a rectangular area, A, is divided into quadrants based on the stations density. In Figure 5.6a, the area is divided into four large and equal size regions, R_1, R_2, R_3, and R_4. Assuming the mobile stations' density is not equal for all groups and that it is higher at R_1, R_3, and R_4, which leads to more decomposition. Figure 5.6b shows a graphical representation of such secondary decomposition. Finally, we assume the mobile stations' density at group R_{11} in Figure 5.6b is high enough to require further decomposition. Therefore, in Figure 5.6c, R_{11} is divided into 4 smaller groups, R_{11}, R_{12}, R_{13}, and R_{14}. The quad tree representation of such decomposition is shown in Figure 5.7.

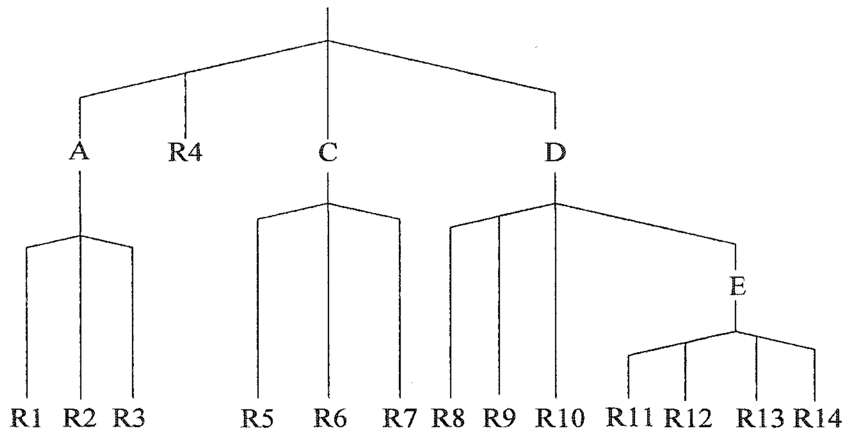


Figure 5.7: Quad tree representation of the regions in Figure 5.5.

5.5 Simulation and results

In our experiments we used Network Simulator NS [25]. We implemented pure flooding, tree flooding, and spatial tree flooding as the tracking protocols. We ran our simulation for 256 access points and 16 mobile stations. The access points are arranged on the form of a grid on an area of 1500X1500 meters. The movement frequencies of these mobile stations ranges from slow (every 200 seconds) to fast (every 25 seconds). Two mobile stations MS-257 and MS-271 cross the borders of two adjacent clusters while five mobile stations MS-269, MS-261, MS-265, MS-259, and MS-267 move around the area of the same cluster. The rest of the mobile stations MS-270, MS-256, MS-268, MS-263, MS-258, MS-260, MS-264, MS-262, and MS-266 are stationary. For all TCP sessions, the sender is stationary and the receiver is mobile except in the case of MS-256, which is a stationary destination.

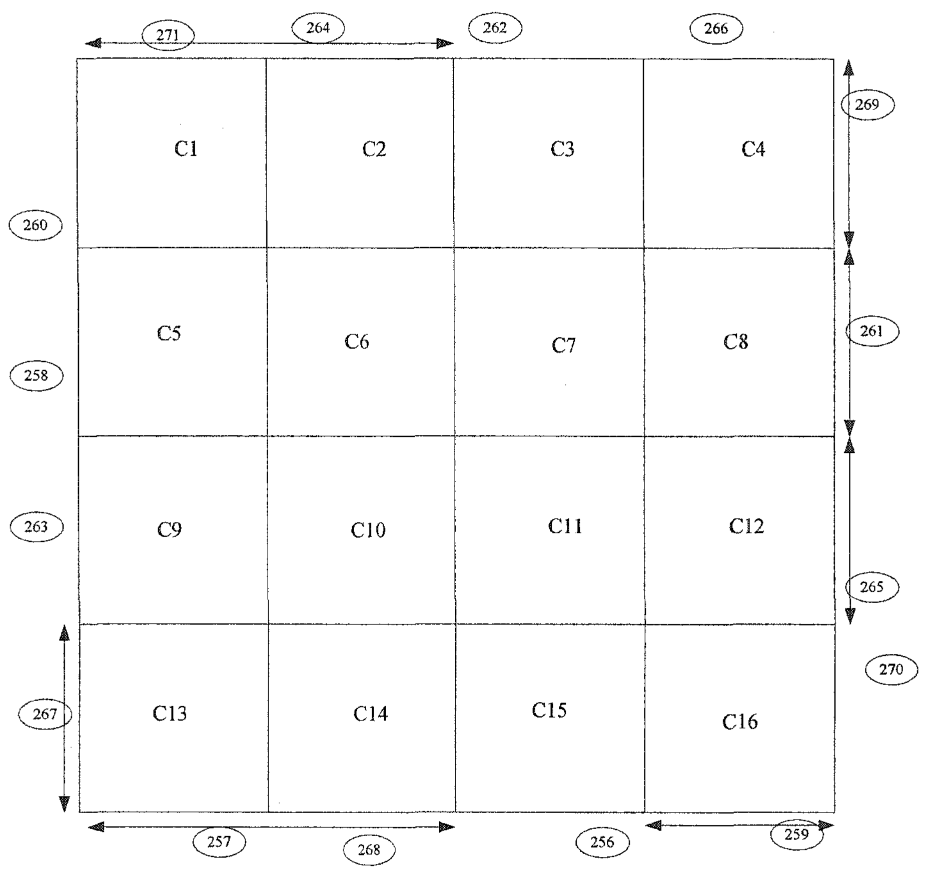
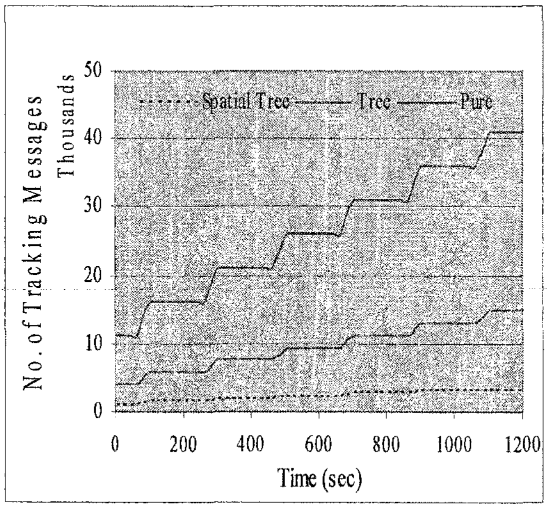


Figure 5.8: Simulation topology.

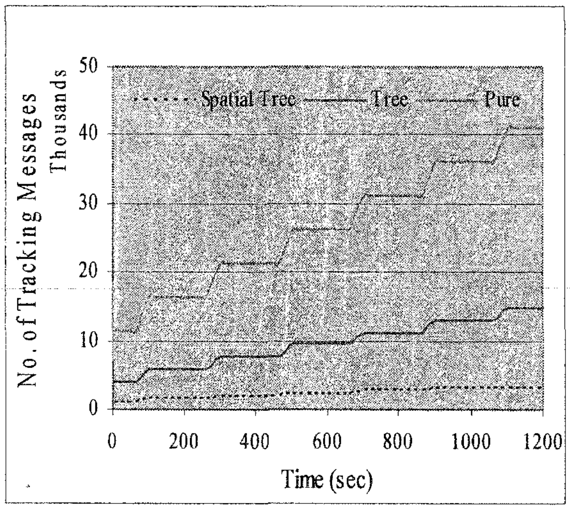
We ran the simulator for 4 different modes; *slow and variable*, *slow and fixed*, *fast and variable*, and *fast and fixed*. The name of each mode is consisted of two parts. The first part represents the movement mode of the mobile stations while the second part represents the algorithm used by the TCP protocol to calculate the retransmission timer. In our simulation experiments we have two movement modes: slow and fast. In slow mode, MS-257 starts moving at time 70 seconds. After that, the mobile station continues moving every 200 seconds. At time 73 seconds, MS-259 starts moving and continues moving every 200 seconds. Next, at time 76, MS-261 starts moving and continues to move every 200 seconds. Similarly, mobile stations MS-265, MS-267, MS-269, MS-271 start moving at 79, 82, 85, and 88 seconds, respectively, and continue to move every 200 seconds. Fast mode is similar to the slow mode except that moving mobile stations continue to move every 25 seconds.

For each movement mode, we used two algorithms to calculate the retransmission time out RTO. In one algorithm we used the normal calculation of RTO as explained in [29] and in the other algorithm we used a fixed value of RTO equals to 10 seconds. We wanted to study the impact of using a fixed value of RTO on the performance of TCP.

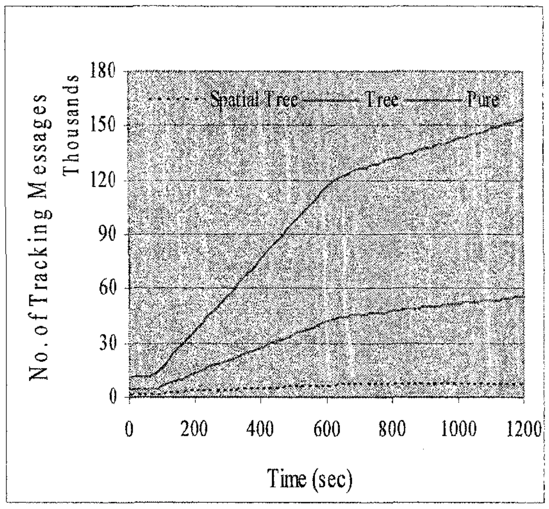
In both experiments, we have 8 TCP sessions between MS-268 and MS-269, MS-270 and MS-271, MS-263 and MS-261, MS-264 and MS-265, MS-262 and MS-256, MS-266 and MS-267, MS-260 and MS-257, and MS-258 and MS-259. The TCP sessions start sending data at the beginning of the simulation and they stop at the end of the simulation time.



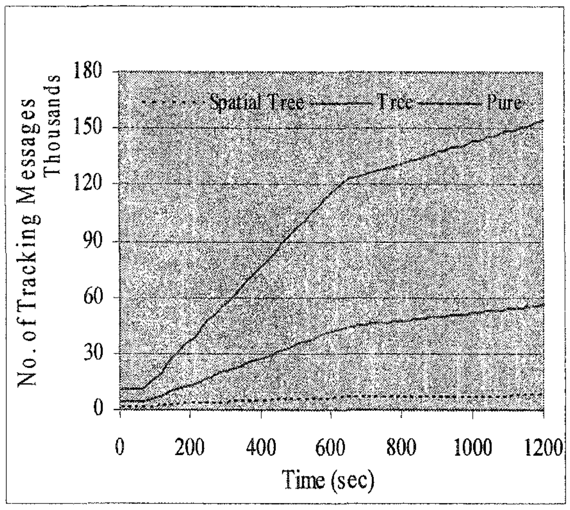
(a) Slow and fixed RTO



(b) Slow and variable RTO



(c) Fast and fixed RTO



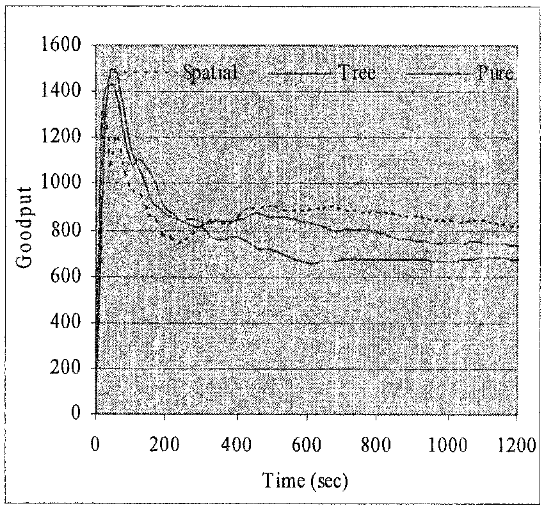
(d) Fast and variable RTO

Figure 5.9: Number of tracking messages.

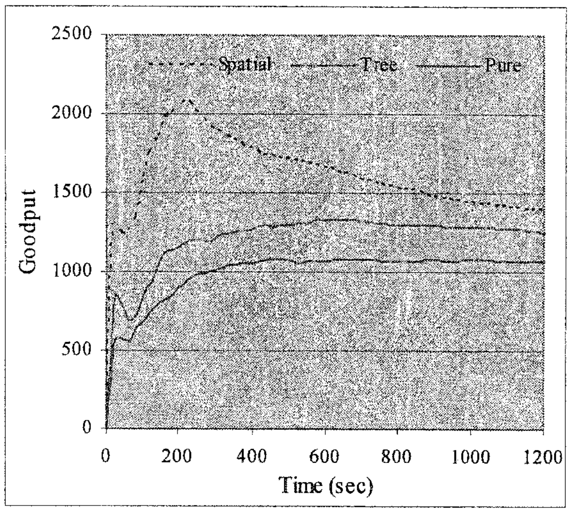
The simulation shows many important results. First, the protocol that uses less number of tracking update messages is spatial tree then comes tree flooding and then comes pure flooding as shown in Figure 5.9. In our simulation, we simulated 1200 seconds of communication time and the number of tracking messages for spatial tree in case of fast mode was close to 7k compared to close to 60k for tree flooding and 158k for pure flooding. As we expect, when mobile stations move slower, the number of tracking messages decreases in all cases. The number of tracking messages in case of slow mode for spatial tree, tree flooding, and pure flooding are 4k, 15k, and 40k, respectively. We ran the simulation for normal TCP and we ran it after changing TCP to use fixed time out timer and the number of tracking messages does not change.

As we mentioned above, using less number of tracking messages reduces the competition among mobile stations for a channel, and uses most of the channel time to send data, which increases the channel utilization. As a result, goodput increases for all

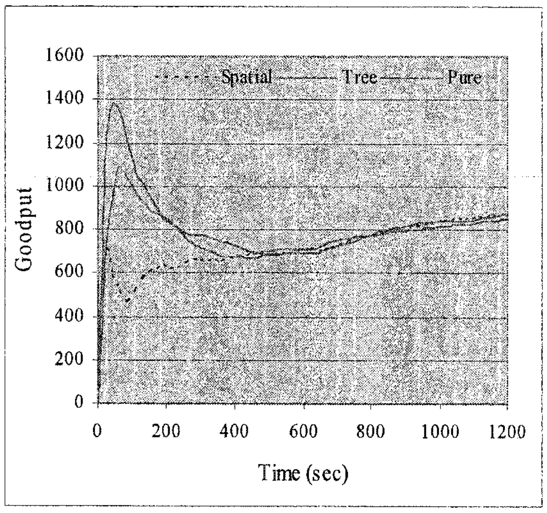
TCP sessions. Figure 5.10 shows the goodput for all TCP sessions in case of fast mode and variable RTO. In 5 sessions, spatial tree performs the best and for 3 sessions the performance of spatial tree and tree flooding are the same. In all sessions, spatial tree performs better than pure flooding.



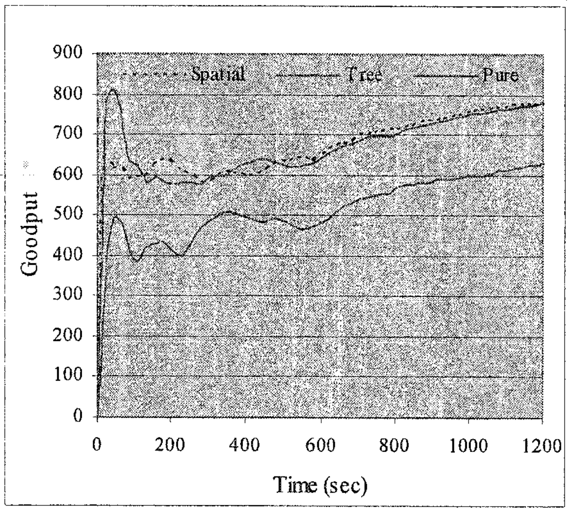
(a) 260-257



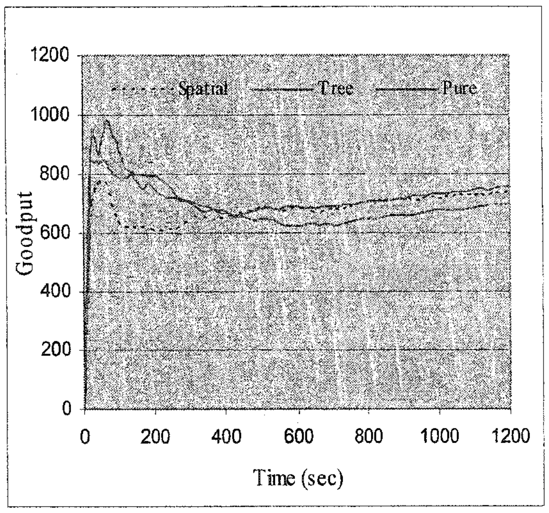
(b) 262-256 (fixed)



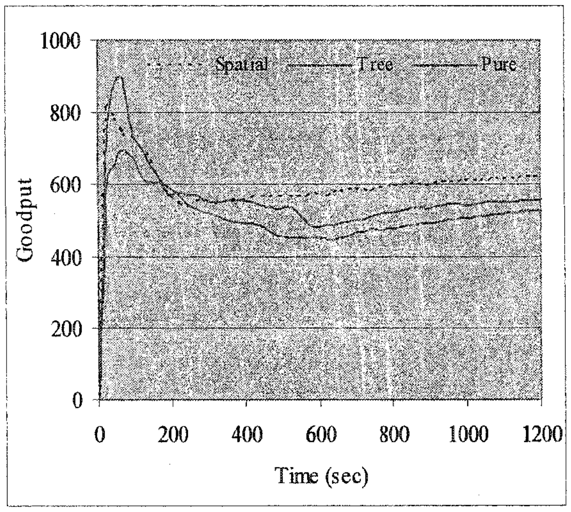
(c) 263-261



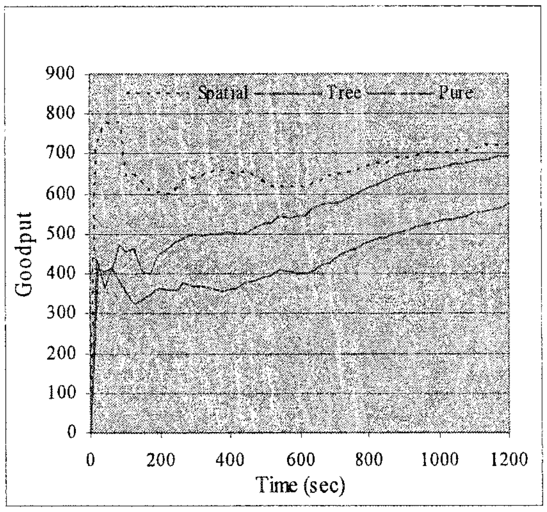
(d) 258-259



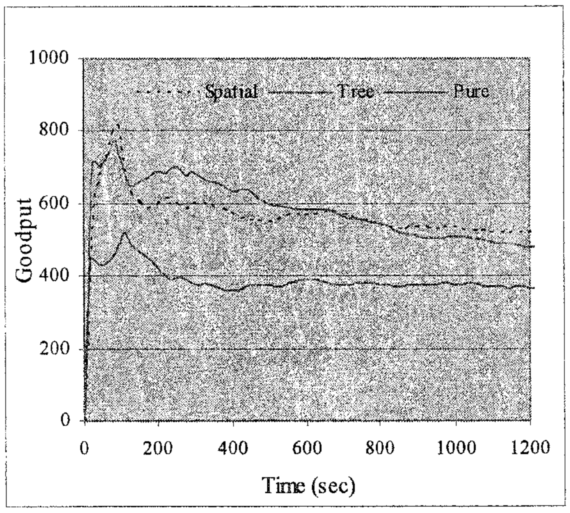
(e) 264-265



(f) 266-267



(g) 268-269



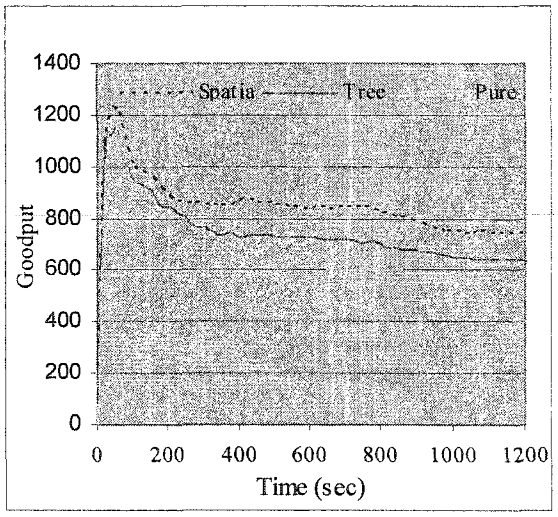
(h) 270-271

Figure 5.10: Goodput for TCP sessions, fast mode, with variable RTO.

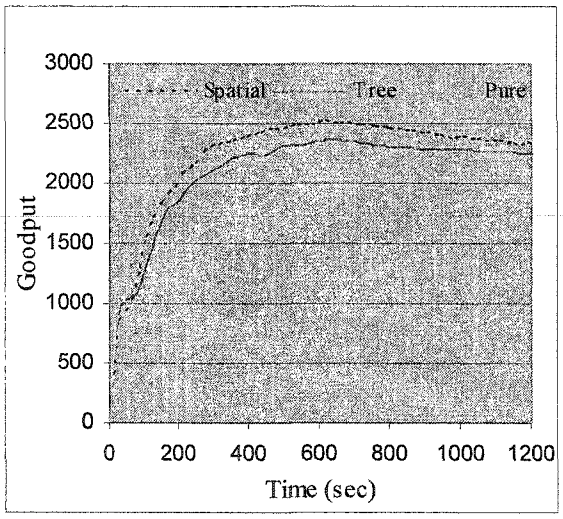
For the purpose of studying the impact of the length of TCP retransmission timer of the goodput of TCP sessions, we ran our simulation while fixing the length of the retransmission timer to 10 seconds. The goodput results for all TCP sessions are shown below in Figure 5.11.

When comparing the goodput of TCP sessions while using variable RTO to the goodput of the TCP sessions using fixed RTO we observed the following. In case of using variable RTO, the goodput of 7 out of 8 TCP sessions outperform the goodput of TCP sessions in case of fixed RTO. The only TCP session that performs better in case of using fixed RTO is shown in Figure 5.11b, which is the TCP session between MS-262 and MS-256. This is because in Fixed TCP, after the destination MS moves, the source MS times out, and starts sending data only after 10 seconds. During this time, the load on the network reduced. When the load is less, the MH-262 sends more data to MH-256 (MH-256 is stationary) and its goodput increases and increases the load on the network. Because of this, when the communication among the other sessions resumes (10 seconds after the movement), the congestion level at the network is higher, which results in lower goodput for other sessions.

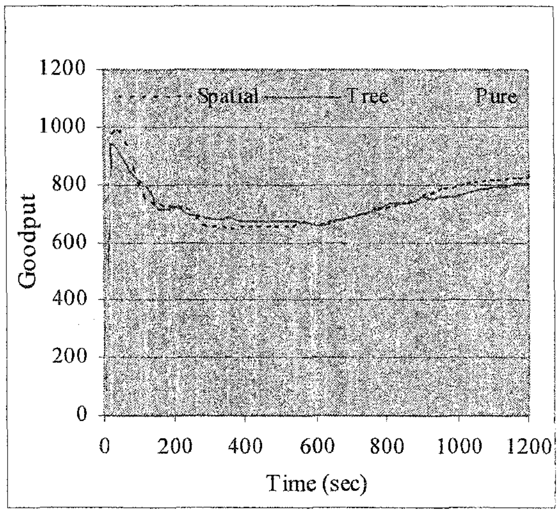
.



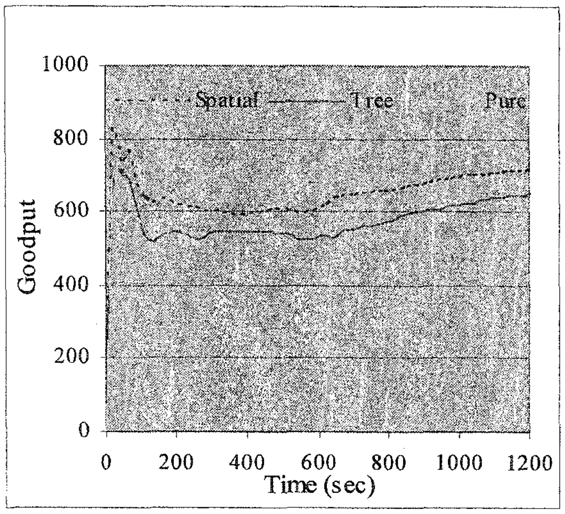
(a) 260-257



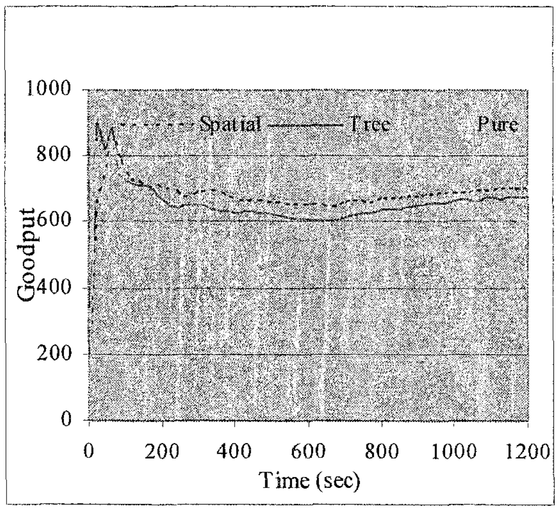
(b) 262-256 (fixed)



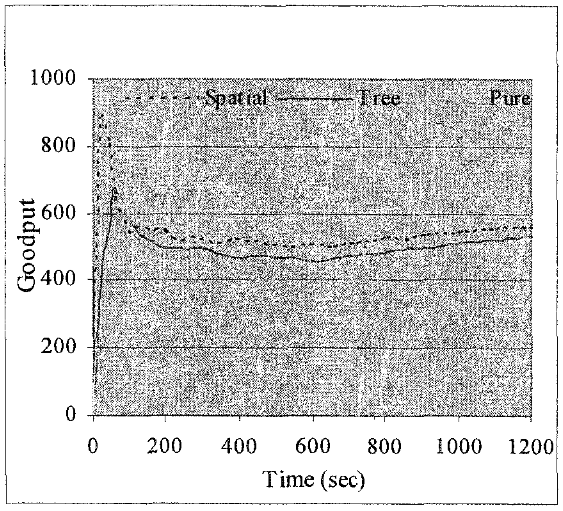
(c) 263-261



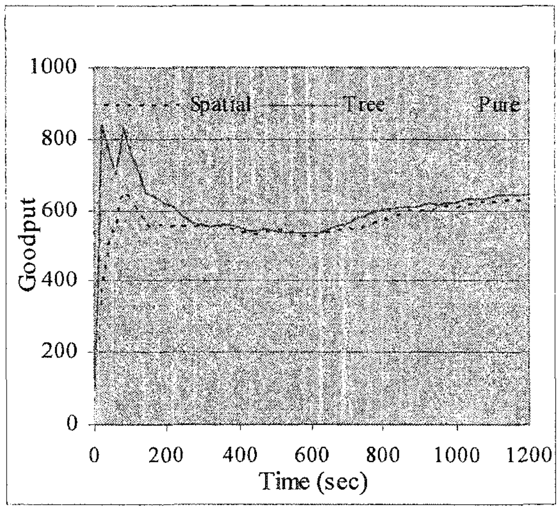
(d) 258-259



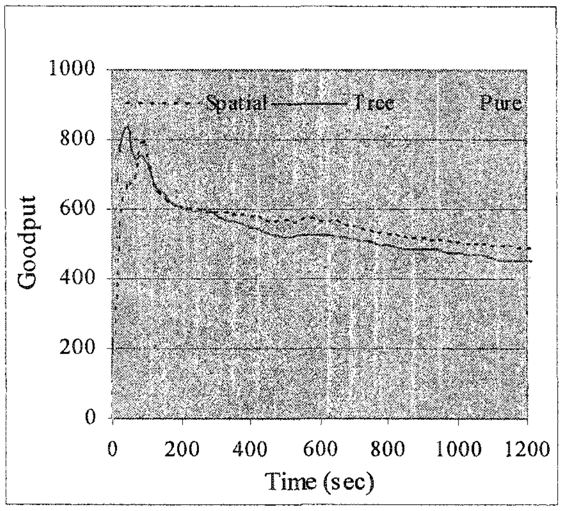
(e) 264-265



(f) 266-267



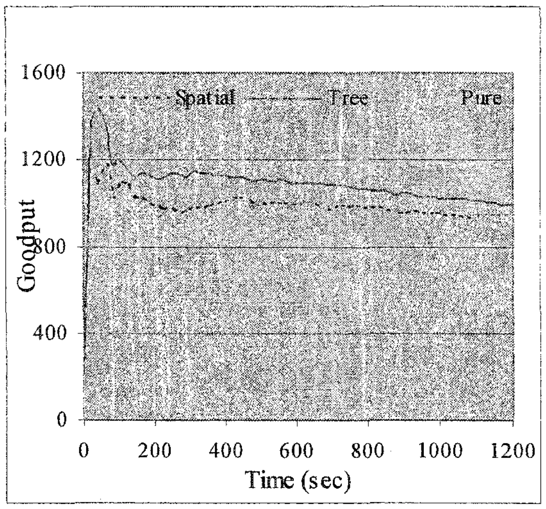
(g) 268-269



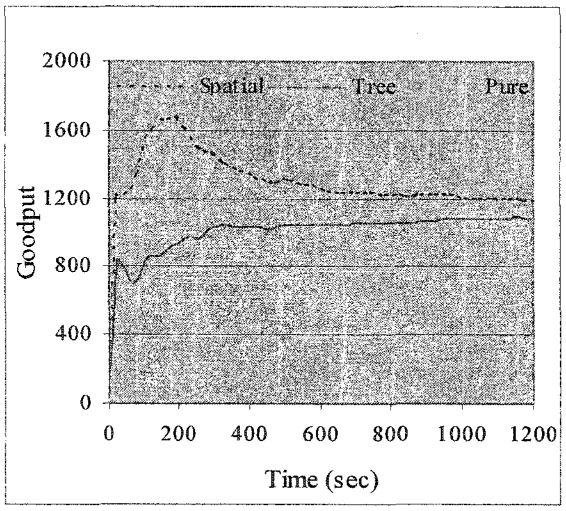
(h) 270-271

Figure 5.11: Goodput for TCP sessions, fast mode, with RTO = 10.

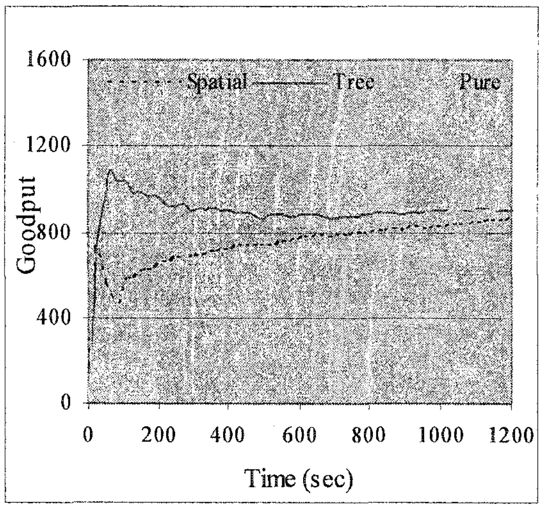
Finally, the importance of spatial tree algorithm is more obvious when the communication load on network resources is high. For example, when mobile stations moves faster, the number of tracking messages exchanged increases, which increases the load on the network. In this case, the overhead associated with spatial tree is minimal compared to the improvement on reducing the number of tracking messages exchanged inside the network. Therefore, in slow mode as shown in Figure 5.12, there are 3 TCP sessions out of 8 where spatial tree performs the best while in Figure 5.11, there are 5 TCP sessions where spatial tree performs the best.



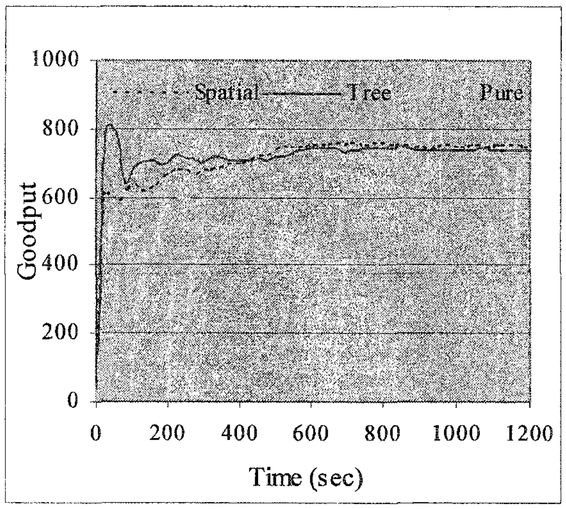
(a) 260-257



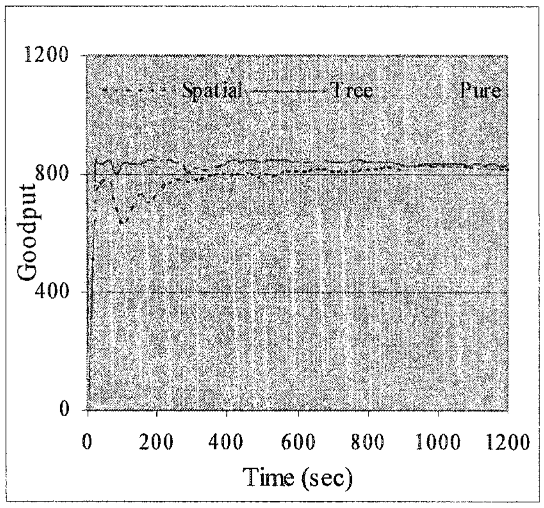
(b) 262-256

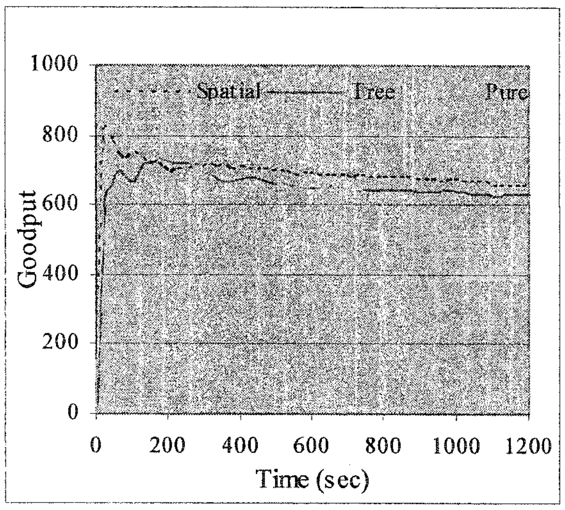


(c) 263-261



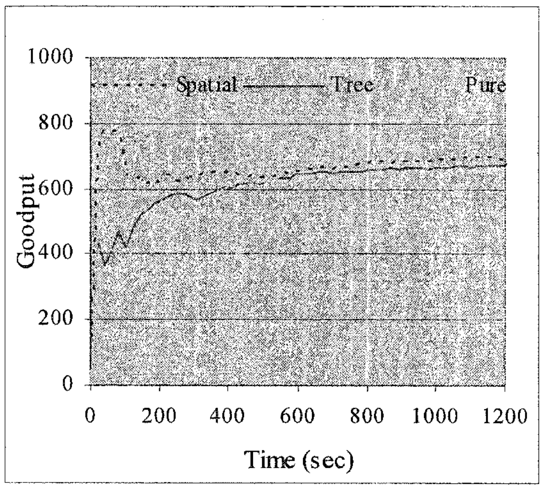
(d) 258-259

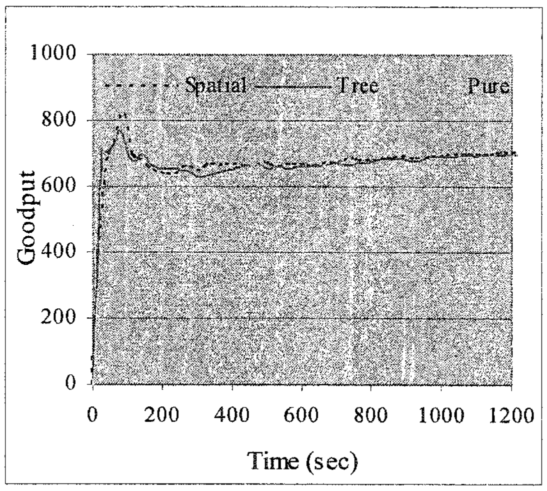




(e) 264-265

(f) 266-267





(g) 268-269

(h) 270-271

Figure 5.12: Goodput for TCP sessions, slow mode, with variable RTO.

5.6 Summary

In this chapter, we develop tracking protocols to support mobile stations location discovery. The protocols are pure flooding, tree flooding, and spatial tree flooding. We evaluate these protocols in terms of number of tracking messages used to support mobility transparent and TCP goodput. Our protocols for tracking updates will exploit station locality information to perform controlled flooding, use backward learning, and forward addressing techniques. We evaluate the protocols using different levels of mobility rates. Also, we exploit the impact of using fixed retransmission timer versus the current retransmission timer technique used in TCP. The results agree with our hypothesis that spatial tree flooding protocol uses the least number of tracking messages, and hence, TCP performs the best on top of it.

Chapter 6

Conclusions and future work

In this chapter, dissertation contributions as well as future work will be summarized. First, a brief introduction about the problem we are trying to solve is presented. Then, an organization of dissertation contributions is discussed followed by detailed description of each category. The chapters of the dissertation were placed under the appropriate category including a brief summary of each chapter. At the end, future work directions are presented and discussed.

6.1 The problem

The original TCP congestion control is purely based on acknowledgment delays. This does not necessarily work well in the case of wireless and mobile networks, where the delays are attributed to congestion as well as to broken links caused by mobility, channel errors, and the contention at the medium access control (MAC) layer that is not only dependent on the traffic in the network, but also on the degree (number of neighbors) of nodes in the network.

6.2 Proposals organization

Proposals found in this dissertation focus mainly on improving the performance of the network communication as a whole with extra emphasis on improving TCP performance. This improvement strategy took two directions. First, by modifying TCP to react more intelligently to packet drop and to work in harmony with other TCP flows in order to achieve better network utilization and more fairness among competing flows. We named this method *independent improvement*. We called the second direction *dependent improvement*, which takes into consideration the interaction relationship between TCP at the transport layer and other protocol at network layer and data link layer. Chapters 2,3, and 5 fits under the later categorization while chapter 4 fits under the first category.

6.3 Proposals of independent improvement

In chapter 4, we studied the dynamics of TCP and found that the presence of a large number of flows that share network resources further complicated the dynamics of TCP. A single TCP flow in the absence of many flows that share the network resources keeps increasing the window size as the acknowledgements arrive on time. Thus each flow uncooperatively will try to increase its window size starting from a network condition that has few flows. This leads to packet losses at the congested link and all the flows will dramatically reduce their window size. When all the flows reduce their window size, the bandwidth on the shared links is used inefficiently and this cycle of increasing and decreasing window size continues. A noticeable decrease in throughput is observed by all the individual flows.

Based on the observations above we recognize two important shortcomings of the current TCP protocol: misinterpretation of delayed acknowledgements and competition among different TCP flows. In this research work, we propose to address these two issues by a use of novel protocol that uses *immediate and delayed acknowledgement schemes* and provides a *coordination mechanism* among independent TCP flows. We also address certain important issues that are related to the implementation of our proposed protocol: can we maintain the end-to-end semantics of TCP? Are there additional benefits that can be harvested if intermediate nodes with TCP protocol can be used? We show that our protocol can be implemented on a portal router or a node attached to the portal router of a network administrative domain. We can extend this approach to the Internet and we provide protocols for this extension.

Our protocol combines all the incoming TCP flows at the portal router into one single flow in an operation called the *join* operation. The agent in the portal router examines the individual flows and can do the following: acknowledge the senders of each flow and it can take the responsibility of sending the data reliably to the receiver (*immediate acknowledgement*) and perform flow control and fairness control by adjusting the receiver's window size. The joining process will reduce the competition among different flows and data can be sent at a rate that fills up the entire bandwidth of the outgoing link from the portal router. Also, since immediate acknowledgment scheme is used, the server can free up its resources resulting in increases efficiency of servers. Of course as the number of flows increases the burden in clearly is on the portal router whose role is buffering and forwarding. The protocol presented in this paper does not add any additional information to the packets sent by the sender. We show by extensive simulations the effectiveness of the proposed protocol and also show how this protocol can be extended wide area networks like the Internet. As an added bonus, we have show that the initial threshold value set for each flow can have a serious impact on the throughput of the TCP flow and especially more profound then there are multiple flows that share the bottleneck bandwidth.

6.4 Proposals of dependent improvement

In Chapters 2 and 3, we proposed to improve TCP performance in mobile networks while taking into consideration the type of rerouting protocols at the network layer.

In Chapter 2, we explained the interaction relationship between wireless TCP protocols and rerouting schemes in a connection less environment. In addition, we present comprehensive analytical methods to determine the buffer requirements at base stations and to estimate the disruption time at the MH. Also, we studied the interaction relationship between mobile TCP schemes and mobile IP and presented analytical results.

In chapter 3, we explain the interaction relationship between wireless TCP protocols and rerouting schemes in a connection-oriented environment where connection must be established prior to any communication. In addition, we presented comprehensive analytical methods to determine the buffer requirements at old base station and to estimate the disruption time at the MH on both the uplink and downlink paths.

In chapter 5, we developed tracking protocols and evaluate these protocols in terms of time and resources required to provide connectivity between mobile stations. Our protocols for tracking updates will exploit station locality information to perform controlled flooding, use backward learning, and forward addressing techniques. We provide an agent based mechanism at the access points wherein each packet will be snooped to provide better service to applications.

6.5 Future work

The results and contributions motivate us to continue working in the area of improving reliable communication in a combination of more than two layers. For example, the impact of different data link layer protocols on the performance of TCP is an interesting subject to pursue and analyze. Further, tuning the application layer protocols to utilize the changes made at the lower layers seems an essential forward step for our area of research.

In chapter 2 and 3, our protocols and analysis assumes a communication between fixed sender and a mobile receiver. In the future, we are planning to study the behavior of our protocols in the case of communication is taking place between mobile sender and mobile receiver.

In the area of Wi-Fi, we are planning on applying changes to TCP to work best with the special tree tracking system. Modification includes modifying the handshake process as well as the retransmission timer behavior.

Bibliography

1. 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 (TON), vol.5, no.6, pp. 756-769, 12-01-1997.

2. H. Balakrishnan and S. Seshan et al. Improving TCP/IP Performance over Wireless Networks. In *Proceedings of MobiCom95,* November 1995.

3. K. Akabane, H. Nakase, "Retransmission Control Scheme for TCP Packet Radio System", IEICE Transactions in Commuincations vol. E82-B, No. 8, August 1999.

4. J. Wong and V. Leung. Improving end-to-end performance of TCP using link-layer retransmissions over mobile internetworks. In *IEEE International Conference on Communications*, page(s): 324 -328, vol. 1, 1999.

5. A. Bakre and B. Badrinath. I-TCP: Indirect TCP for mobile hosts. In *proceedings of 15th International Conference On Distributed Computing Systems*, Vancouver, Canada, June 1995.

6. K. Wang and S. Tripathi. Mobile-end transport protocol: an alternative to TCP/IP over wireless links. IN Proceedings of the Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, pages 1046 -1053 vol. 3, 1998.

7. Z. Haas and P. Agrawal. Mobile-TCP: an asymmetric transport protocol design for mobile systems. In *International Conference on Communications, Montreal, Quebec, Canada, June 1997.*

8. G. Racherla, S. Radhakrishnan and C. Sekharan. "Performance evaluation of wireless TCP schemes under different rerouting schemes in mobile networks," In *IEEE Region 10 International Conference on Global Connectivity in Energy, Computer, Communication and Control (TENCON '98)*, pages 93-96, 1998.

9. G. Racherla. Algorithms for routing and rerouting in mobile wireless and ad-hoc networks. PhD thesis, University of Oklahoma, Norman, 1999.

10. S. Seshan. *Low latency Handoff for Cellular Data Networks*. PhD thesis, University of California, Berkeley, 1995.

11. R. Ludwig and R. Katz, "The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions." In ACM SIGCOMM Computer Communication Review, vol. 30, issue 1, January 2000.

12. V. Tsaoussidis and I. Matta, "Open issues on TCP for mobile computing." In Wireless Communications and Mobile Computing, vol. 2, pp. 3-20, 2002.

13. C. Perkins, "Mobile IP," IEEE Communications Magazine, vol. 40, issue 5, pp 66-82, May 2002.

14. G. Hasegawa and M. Murata, "Survey on fairness issues in TCP congestion control mechanisms," *IEICE Trans. Commun.*, vol. E84-B, no. 6, pp. 1461-1472, June 2001.

15. M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round robin," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 375-385, June 1996.

16. G. Hasegawa, K. Kurata, and M. Murata, "Analysis and improvement of fairness between TCP Reno and Vegas for deployment of TCP Vegas to the Internet," *Proceedings of the International Conference on Network Protocols*, pp. 177 -186, 2000.

17. D. Ott and K. Mayer-Patel., "A Mechanism for TCP-Friendly Transport-level Protocol Coordination," *Proceedings of USENIX 2002*, pp. 147-159, Monterey, CA, June 2002.

18. J. Widmer, R. Denda, and M. Mauve, "A survey on TCP-friendly congestion control," *IEEE Network*, vol.15, no. 3, pp. 28 -37, May-June 2001.

19. P. Karbhari, E. Zegura, and M. Ammar, "Multipoint-to-Point Session Fairness in the Internet," *IEEE INFOCOM*, vol.1, pp. 207-217, March 30-April 3, 2003.

20. M. Handley, J. Padhye, S. Floyd, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification. IETF, May 2001. Internet Draft, work in progress.

21. I. Foster and C. Kesselman, *The Grid Blueprint for a New Computing Infrastructure.* Morgan Kaufmann Publishers, ISBN: 1-55860-475-8, July 1998.

22. D. Ott and K. Mayer-Patel, "Transport-level Protocol Coordination in Cluster-to-Cluster Applications," *Proceedings of IDMS*, pp. 10-22, September 4-7 2001.

23. H. Balakrishnan, H. S. Rahul, and S. Seshan, "An Integrated Congestion Management Architecture for Internet Hosts," *In Proc. ACM SIGCOMM*, pp. 175-187, Cambridge, MA, September 1999.

24. H. Kung and S. Wang, "TCP Trunking: Design, Implementation and Performance," *Proc. 7th International Conference on Network Protocols*, pp. 222-231, October 31 - November 03, 1999.

25. K. Fall and K. Varadhan, *The ns Manual*. The VINT Project, April 14, 2002. http://www.isi.edu/nsnam/ns/doc/ ns_doc.pdf.

26. I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: enabling scalable virtual organizations," *The International Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 200-222, November 2001.

27. L. Kalampoukas, Anujan Varma, and K.K. Ramakrishnan, "Explicit Window Adaptation: A Method to Enhance TCP Performance," *IEEE/ACM Transactions on Networking*, vol. 10, no. 3, pp. 338-350, June 2002.

28. E. Bowen, E., C. Jeffries, L. Kencl, A. Kind, and R. Pletka, "Bandwidth allocation for non-responsive flows with active queue management," *International Zurich Seminar on Broadband Communications, Access, Transmission, Networking*, pp. 13-1-13-6, February 2002.

29. Andrew Tanenbaum. Computer Networks. Third Edition. Prentice-Hall PTR, Upper Saddle River, New Jersey 07458, 1996.

30. R. Gandhi et al, "Minimizing Broadcast Latency and Redundancy in Ad Hoc Networks," MobiHoc '03, the fourth international symposium on mobile Ad Hoc networking and computing, June 1-3, 2003, Annapolis, Maryland, USA, pps 222-232.

31. Ivan Stojmenovic, Mahtab Seddigh, Jovisa Zunic, "Dominating Sets and Neighbor Elimination-Based Broadcasting Algorithms in Wireless Networks," IEEE transactions on parallel and distributed systems, vol. 12, No. 12, PPS 1-12, December 2001.

32. Sridhar Radhakrishnan, Shankar M Banik, Chandra N Sekharan, "Broadcast Scheduling in Ad Hoc wireless networks," (work in progress).

33. Xiaoyan Hong, Kaixin Xu, Mario Gerla, "Scalable Routing Protocols for Mobile Ad Hoc Networks," IEEE Network Magazine, pp. 11-21, July/August 2002.

34. Jie Wu and Fei Dai, "Broadcasting in Ad Hoc Networking Based on Self-Pruning," IEEE INFOCOM, Vol. 3, pp 2240-2250, San Francisco, March/April 2003.

35. Hanan Samet, *Applications of spatial data structures*, Addison-Wesley Publishing Company, 1990, ISBN 0-201-50300-X.

36. Waleed Al-Numay, Sridhar Radhakrishnan, "Analytical modeling for the interaction between wireless TCP schemes and Mobile IP," STCEX2002, Riyadh, Saudi Arabia, October 26-30, 2002.

37. Waleed Al-Numay, Sridhar Radhakrishnan, Chandra N Sekharan, "Interaction of Wireless TCP Schemes and Rerouting: Analytical Models and Simulation," 23rd International Conference on Distributed Computing Systems Workshops, pp. 883-889, Providence, Rhode Island, May 19-22, 2003.

38. Waleed Al-Numay, Sridhar Radhakrishnan, "JTCP: Join Protocols for Improving Performance of TCP," 3rd International Conference on Networking - ICN'04, Guadeloupe, French Caribbean, February 29 – March 4, 2004.