B.Sc. in Computer Science and Engineering Thesis

Comparison Between Different Transport Layer Mobility Protocols

Submitted by

Chowdhury Nawrin Ferdous
201014047

Md. Asif-Ur-Rahaman
201014036

Md. Ahsan Ullah
201014058

Supervised by

Dr. Md. Shohrab Hossain
Assistant Professor,Dept of CSE,BUET

Department of Computer Science and Engineering
Military Institute of Science and Technology

CERTIFICATION

This thesis titled **"Comparison Between Different Transport Layer Mobility Protocols"**, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering on December 2013.

Group Members:

Chowdhury Nawrin Ferdous
Md. Asif-Ur-Rahaman
Md. Ahsan Ullah

Supervisor:

_____

Dr. Md. Shohrab Hossain
Assistant Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

CANDIDATES' DECLARATION

This is to certify that the work presented in this thesis is the outcome of the investigation and research carried out by the following students under the supervision of Dr. Md. Shohrab Hossain, Assistant Professor, Department of Computer Science and Engineering, BUET, Dhaka, Bangladesh.

It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

_____

Chowdhury Nawrin Ferdous
201014047

_____

Md. Asif-Ur-Rahaman
201014036

_____

Md. Ahsan Ullah
201014058

ACKNOWLEDGEMENT

We are thankful to Almighty Allah for his blessings for the successful completion of our thesis. Our heartiest gratitude, profound indebtedness and deep respect go to our supervisor Dr. Md. Shohrab Hossain, Assistant Professor, Dept of CSE, BUET, Dhaka, Bangladesh, for his constant supervision, affectionate guidance and great encouragement and motivation. His keen interest on the topic and valuable advices throughout the study was of great help in completing thesis.

We are especially grateful to the Department of Computer Science and Engineering (CSE) of Military Institute of Science and Technology (MIST) for providing their all out support during the thesis work.

Finally, we would like to thank our families and our course mates for their appreciable assistance, patience and suggestions during the course of our thesis.

Dhaka
December 2013

Chowdhury Nawrin Ferdous
Md. Asif-Ur-Rahaman
Md. Ahsan Ullah

ABSTRACT

Mobility of Internet hosts allows computing nodes to move between subnets. In order to provide seamless connectivity to the roaming users several mobility protocols have been developed at different layers. Mobile IP has been developed to handle mobility of Internet hosts at the network layer. Transport layer mobility can overcome many of the limitations of network layer schemes. Various approaches have been proposed to implement mobility in the transport layer. In this thesis, we discuss a number of transport layer mobility protocols, classify them according to their approach, and compare them based on a number of evaluation criteria. The components of a complete mobility management scheme consist of handoff, connection migration, and location management. Evaluation criteria have been developed to determine and compare the effectiveness of mobility schemes. The criteria include handoff, packet loss and delay, fault tolerance, requirement for change in network infrastructure, mobility type, support for IP diversity, security, scalability, etc. In this thesis, we use the above criteria to classify the proposed mobility schemes.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATION

ACK : Acknowledgement

API : Application Programming Interface

BARWAN : Bay Area Research Wireless Access Network

BS : Base Station

CN : corresponding node

CWND : Congestion Window

DHCP : Dynamic Host Configuration Protocol

DOS : Disk Operating System

FTP : File Transfer Protocol

HTTP : Hypertext Transfer Protocol

I-TCP : Indirect TCP

IPsec : Internet Protocol Security

MH : Mobile host

M-TCP : Migrate TCP

MMSP : Mobile Multimedia Streaming Protocol

M-UDP : Mobile UDP

mSCTP : Mobile SCTP

RCP : Receiption Control Protocol

R^2CP : Radial Reception Control Protocol

RTT : Round-Trip Time

SACK : Selective Acknowledgment

SCTP : Stream Control Transmission Protocol

SIGMA : Seamless IP diversity based Generalized Mobility Architecture

TCP : Tranmission Control Protocol

ZWA : Zero Window Advertisement

ZWP : Zero Window Probes

CHAPTER 1

INTRODUCTION

1.1 Introduction

The Internet is based on five layer architecture: physical, data link, network, transport and application layers, with each layer having specific responsibilities. Since mobility can be managed at different layers, a natural question to answer is the layer at which mobility should be managed. Several works have done on weakness and strengths of mobility management at the different layers. Mobility can be handled at different layers of the protocol stack, with network and transport layer mobility being the most widely studied. Transport layer mobility can overcome many of the limitations of network layer schemes like Mobile IP.

1.2 Overview

Mobile nodes of the future will be equipped with multiple network interfaces to take advantage of overlay networks, yet no current mobility systems provide full support for the simultaneous use of multiple interfaces. The need for such support arises when multiple connectivity options are available with different cost, coverage, latency and bandwidth characteristics, and applications want their data to flow over the interface that best matches the characteristics of the data. We present an architecture called Transport Layer Mobility that allows mobile nodes to not only change their point of attachment to the Internet, but also to control which network interfaces are used for the different kinds of data leaving from and arriving at the mobile node.

The Internet was originally designed for static hosts connected through wired networks. Proliferation of wireless net works has given rise to an increasing demand for mobility of hosts,

resulting in various mobility management schemes. Mobility management consists of two fundamental operations Handoff and Location Management. Handoff occurs when a mobile device changes its point of attachment while still communicating with its peer. Handoff can be implemented a Location management refers to the task of locating (finding the IP address) a Mobile Host (MH) in order to initiate and establish a connection by a node. A good location management scheme should provide a valid address of the MH, and be transparent to its peers.

1.2.1 Different transport layer protocols

There are many transport layer mobility protocols with different criteria. While Mobile IP is a network layer scheme which makes mobility transparent to upper layers by increasing the burden and responsibility of the Internet infrastructure, transport layer schemes are based on an end-to-end approach to mobility that attempt to keep the Internet infrastructure unchanged by allowing the end hosts to take care of mobility. MSOCKS[**?**], SIGMA[**?**], RCP[**?**], Freeze-TCP[**?**], R^2CP, MMSP, I-TCP, M-TCP, M-UDP, BARWAN, TCP-R, mSCTP[**?**] etc are the different mobility protocols of transport layer.

1.2.2 Comparison Criteria

Handoff, connection migration, and location management are the main fundamentals of a complete mobility management scheme. Evaluation criteria have to be developed to determine and compare the effectiveness of mobility schemes. In this paper, we use handoff, packet loss and delay, fault tolerance, requirement for change in network infrastructure, mobility type, support for IP diversity, security, scalability, etc. to classify the proposed mobility schemes.

CHAPTER 2

LITERATURE BACKGROUND

The proliferation of laptops, hand-held computers, cellular phones, and other mobile computing platforms connected to the Internet has triggered much research on mobility support in IP networks. Modern mobile terminals are likely to be equipped with wireless communication devices allowing them to constantly reach the Internet and participate in it as normal end systems would. A mobile terminal, however, represents an ill fit with the traditional assumptions upon which the IP protocols is based: a classic end system does not move and has only a single point of attachment to the network. For such an end system, a single handle is sufficient to represent both the identity of a terminal as well as its location within the network; the IP address is this very handle. Such a permanent handle is not appropriate in mobile networks. On one hand, an identifier is necessary to distinguish among different terminals; on the other hand, information about the current location within a network has to be provided to ensure that packets destined to a certain terminal can still be routed towards this terminal. The fundamental mobility problem in IP-based networks is therefore the separation of identity and location.

The problem of mobility in IP[**?**] networks has traditionally been solved at the network layer. Now it is solved at the transport layer in an end-to-end fashion. Implementing this concept requires changes to existing transport layers. In previous work, TCP has been modified to support such an end-to-end mobility concept. While TCP is indeed the most often used transport protocol in the Internet, it might not be the perfect platform to experiment with unconventional ways of supporting mobility. The Internet is based on five layer architecture: physical, data link, network, transport and application layers, with each layer having specific responsibilities. Since mobility can be managed at different layers, but for handle in efficient way transport layer introduces some important TCP schemes. Here, efficiency needs for some important issues like, Packet loss and latency, congestion control, connection

migration, infrastructure, location management etc. The schemes for mobility management under transport layer are:

- MSOCKS
- SIGMA
- Migrate TCP
- Freeze TCP
- RCP
- R^2CP

Some other schemes also entrusted. Our main task is to compare the six protocols which are mentioned. We compare the protocols on basically some important criteria handoff process, transparency, loss or delay, scalability and fault tolerance, security, path diversity, infrastructure change etc.

MSOCKS uses TCP Splice for connection migration and supports multiple IP addresses for multiple interfaces. When an MH disconnects itself from a subnet during handoff, it obtains a new IP address from the new subnet using DHCP, and establishes a new connection with the proxy using its second interface. The communication between proxy and CN, however, remains unchanged. The data flow between MH and CN thus continues, with the CN being unaware of the mobility. So, its achievement are mobile node has freedom to send and receive from network interface of its choice and preservation of TCPs end-to-end reliability and correctness. So, MSOCKS provides applications with different control over their sessions.

SIGMA is a complete mobility management scheme of seamless IP diversity implemented at the transport layer, and can be used with any transport protocol that supports IP diversity. MSOCKS implement mobility as an end-to-end service without the requirement to change the network layer infrastructures; they, however, do not aim to reduce the high latency and packet loss resulting from handovers. High latency and packet loss control is the significance of SIGMA. MH moves into the overlapping region of two neighboring subnets, it obtains a

new IP address from the new subnet while still having the old one as its primary address. When the received signal at the MH from the old subnet goes below a certain threshold, the MH changes its primary address to the new one. When it leaves the overlapping area, it releases the old address and continues communicating with the new address. Location management in SIGMA is done using DNS as almost every Internet connection starts with a name lookup. Whenever an MH changes its address, the DNS entry is updated so that subsequent requests can be served with the new IP address.

Migrate TCP (M-TCP) is a transparent mobility management scheme which is based on connection migration and uses DNS for location management. In Migrate TCP, when an MH initiates a connection with a CN, the end nodes exchange a token to identify the particular connection. A hard handoff takes place when the MH reestablishes a previously established connection using the token, followed by migration of the connection. Similar to SIGMA[**?**], this scheme proposes to use DNS for location management. The main achievement of M-TCP is service continuity. The basic application of M-TCP is for long lived connections like multimedia streaming services or videos, end users expect both correctness and good response time like, Internet banking, e-commerce etc.

Freeze-TCP mechanism which is a true end-to-end scheme and does not require the involvement of any intermediaries (such as base stations) for flow control. Furthermore, this scheme does not require any changes on the "sender side"or intermediate routers; changes in TCP code are restricted to the mobile client side, making it possible to fully inter-operate with the existing infrastructure. Freeze-TCP lets the MH 'freeze'or stops an existing TCP connection during handoff by advertising a zero window size to the CN, and unfreezes the connection after handoff. This scheme reduces packet losses during handoff at the cost of higher delay.

RCP (Reception Control Protocol) is a receiver-centric transport protocol that is a TCP clone in its general behavior, but allows for better congestion control, loss recovery, and power management mechanisms compared to sender-centric approaches. In RCP, since the control of data transfer is shifted from the sender to the receiver, the DATA-ACK style of handshaking in TCP is no longer applicable. Instead, to mimic the self clocking characteristics of TCP, RCP uses the REQ-DATA handshake for data transfer, where any data transferred from the sender is preceded with an explicit request (REQ) from the receiver.

So, the achievement are receiver has full control on data and any loss recovery mechanism can be used that optimizes the wireless environment.

R^{2}CP (Radial Reception Control Protocol) is based on Reception Control Protocol, a TCP clone in its general behavior but moves the congestion control and reliability issues from sender to receiver on the assumption that the MH is the receiver and should be responsible for the network parameters. R^{2}CP has some added features over RCP like the support of accessing heterogeneous wireless connections and IP diversity that enables a soft handoff and bandwidth aggregation using multiple interfaces. A location management scheme might be integrated with R^{2}CP to deploy a complete scheme. R^{2}CP maintains the four key data structures for performing effective packet scheduling like, binding, pending, rank and active. The achievements are seamless handoffs between interfaces, server migration, bandwidth aggregation etc.

There are more other protocol schemes of mobility management. For example, MMSP, I-TCP, mSCTP, M-UDP, BARWAN etc.

MMSP (Mobile Multimedia Streaming Protocol) supports transparent soft handoff through IP diversity and uses bicasting to prevent losses during the handoff period. This protocol uses Forward Error Correction (FEC) and fragmentation to mitigate wireless errors and does not include location management.

I-TCP (Indirect TCP) is a mobility scheme that requires a gateway between the communication path of the CN and MH to enable mobility. In this scheme, a TCP connection between CN and gateway and a I-TCP connection between the gateway and MH is established to provide CN to MH communication. The TCP portion remains unchanged during the lifetime of the communication and remains unaware of the mobility of MH. In the I-TCP portion, when the MH moves from one subnet to another one, a new connection between MH and the gateway is established and the old one is replaced by the new one. Location management is not included in this scheme.

mSCTP (Mobile SCTP) supports IP diversity and soft handoff. The handoff is similar to the one of SIGMA. mSCTP[**?**] can maintain application transparency but it does not support location management. M-UDP (Mobile UDP) is an implementation of UDP protocol with

mobility support similar to I-TCP and M-TCP. Like M-TCP, M-UDP uses a gateway to split the connections between MH and CN to ensure one unbroken gateway to CN connection and continuously changing MN to gateway connection. This also does not include IP diversity or location management.

BARWAN (The Bay Area Research Wireless Access Network) is a solution to heterogeneous wireless overlay network. It has a gateway centric architecture on an assumption that the wireless networks are built around the gateways. Diverse overlapping networks are integrated through software that operates between the MH and the network. This supports the MH to move among multiple wireless networks. BARWAN requires the application to be aware of mobility as the decision to make a handoff is taken by the application. This scheme does not specify a location manager.

CHAPTER 3

TRANSPORT LAYER MOBILITY PROTOCOLS

3.1 Introduction

In this chapter, we will discuss about six transport layer mobility protocols: MSOCKS, SIGMA, RCP, R^{2}CP, Freeze TCP, Migrate TCP.

3.2 MSOCKS

MSOCKS is built around a proxy that is inserted into the communication path between a mobile node and its correspondent hosts. For each data stream from a mobile node to a correspondent host, the proxy is able to maintain one stable data stream to the correspondent host, isolating the correspondent host from any mobility issues. Meanwhile the proxy can simultaneously make and break connections to the mobile node as needed to migrate data streams between network interfaces or subnets. The proxy can then mediate the communication between server and client, and provide services on behalf of either. Proxies can:

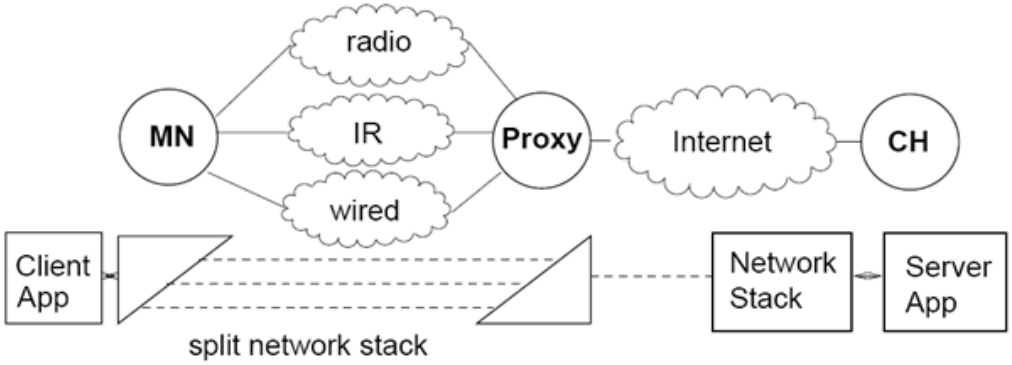


Figure 3.1: Common network topology showing the location of a proxy between the mobile node and correspondent host.

provide processing resources the client may not have; reformat information from the server to fit the mobile node, such as resizing GIF images for small screens; or use compression to reduce the bandwidth required between the mobile node and proxy, which is frequently a low quality link. In one word, MSOCKS is a flexible system that mobile nodes can continue connections between different interfaces.

3.2.1 MSOCKS Architecture

MSOCKS, is built around a technique we call TCP Splice. TCP Splice allows the machine where two independent TCP connections terminate to splice the two connections together, effectively forming a single end-to-end TCP connection between the endpoints of the two original connections. MSOCKS architecture consists of three pieces: a user level MSOCKS proxy process running on a proxy machine; an in-kernel modification on the proxy machine to provide the TCP Splice service; and a MSOCKS library that runs under the application of the mobile node.

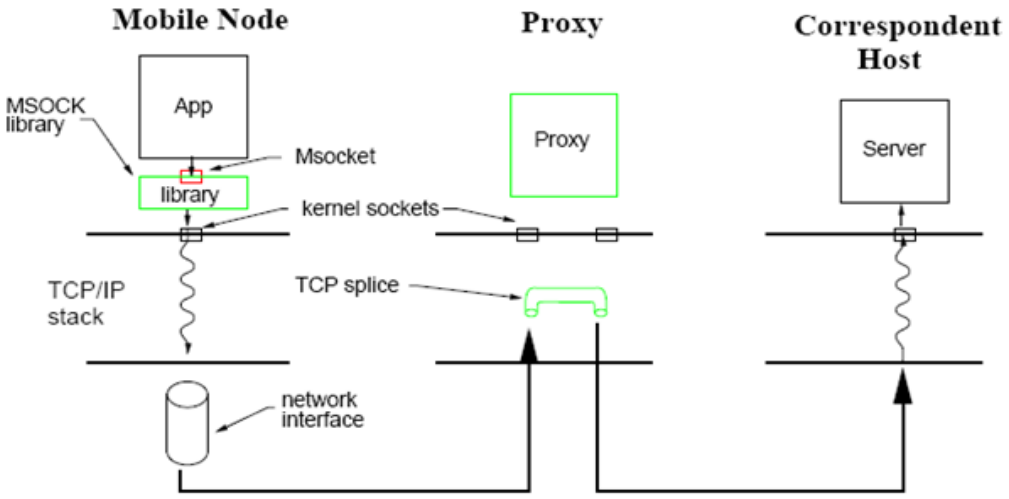


Figure 3.2: Parts shown in green are where MSOCKS alterations are made to the standard parts of proxy based client-server system.

3.2.2 MSOCKS Protocol

Connection establishment

Figure 3.3 shows the packets exchanged when an MSOCKS client application connects to a server on a correspondent host. The applications connect() call is intercepted by the MSOCKS library and turned into a call to Mconnect(). Mconnect first uses the mobile nodes normal TCP stack to make a connection to the proxy, using whatever addresses are appropriate to the data the connection will carry. Over this connection, the library sends the proxy the servers address and port number that the application gave as arguments to Mconnect() along with any authentication information the proxy requires.

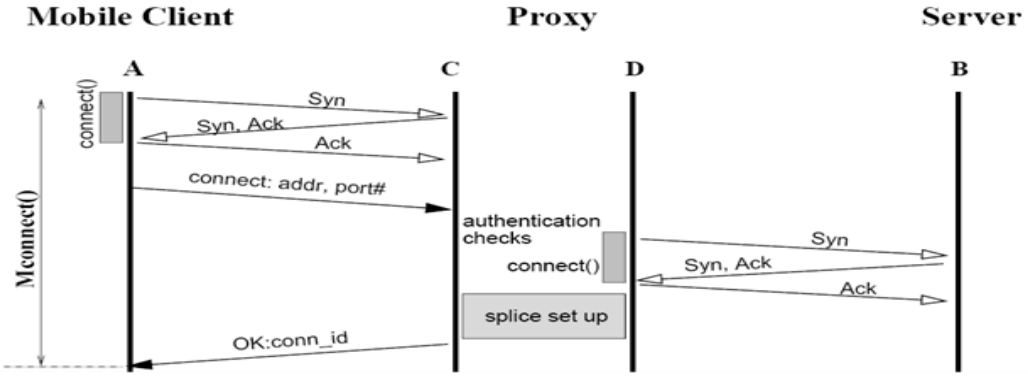


Figure 3.3: Connection establishment between a MSOCK client and a corresponding host via a MSOCK Proxy.

The splicing technique supports an arbitrary authentication negotiation with packets sent both from and to the proxy, After authenticating the mobile node, the proxy connects to the desired serve and then splices the mobile-proxy and proxy-server connection together. When the splice is set up, the proxy transmits a final OK message to the mobile node to synchronize the MSOCKS library. The OK message contains the connection identifier of the proxy has assigned to this session for use should the mobile node later want to reconnect it.

Mobile node reconnection

The splicing technique allows us to perform reconnections even when there is data in flight between correspondent host and mobile node, or when there is no warning the mobile node will need to change addresses, such as during hard hand-offs. Without care, these packets in flight may be lost or duplicated; the MSOCKS RECONNECT protocol together with TCP

Splice ensures that the end-to-end reliable, in-sequence semantics of TCP are maintained. Figure 3.4 shows the packets exchanged when a connection between a mobile node and proxy breaks for some reason (e.g., the mobile node moves and obtains a new IP address, or it wishes to switch the session from one network interface to another). After the connection to the proxy is broken, the MSOCKS library opens a new socket, labeled E in the figure, and connects to the proxy using it. The MSOCKS library transmits a reconnect message to

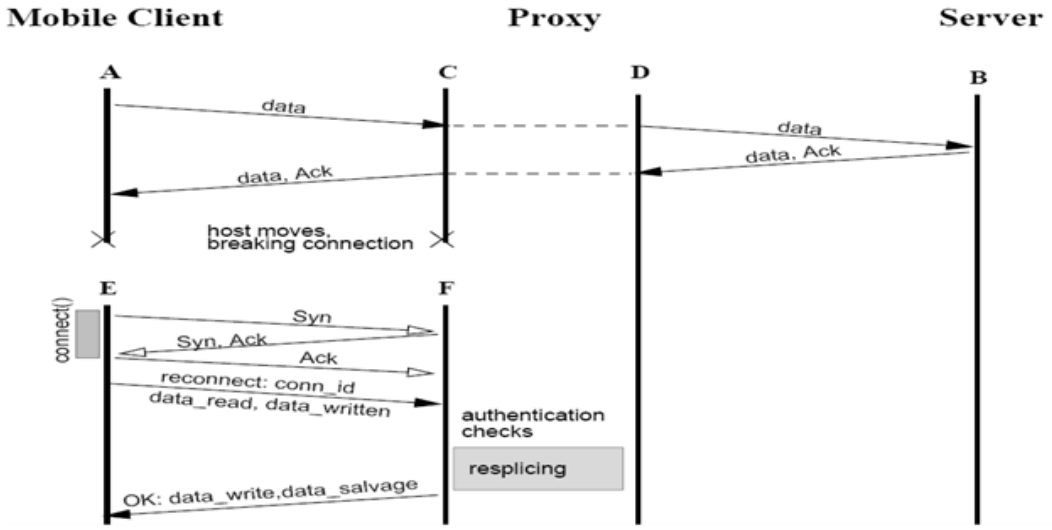


Figure 3.4: Packet exchange datagram for a mobile node reconnecting to an existing connection.

the proxy giving the connection identifier of the old connection to the server, along with a data-read counter, telling the proxy how many bytes of data the application has read from the connection, and a data-written counter, telling the proxy how many bytes of data the application has written to the connection. The proxy then splices the new connection to the proxy-server connection in place of the old mobile-proxy connection and closes the old connection. Once the splice is setup, the proxy sends an OK message to the MSOCKS library, along with data write and data salvage counters directing the MSOCKS library how to complete the splice at the mobile node's end. The application and server are completely unaware the switch has happened.

3.2.3 Performance

In addition to supporting many connections, an ideal MSOCKS proxy would add minimal latency to the path of packets traveling to or from mobile nodes.

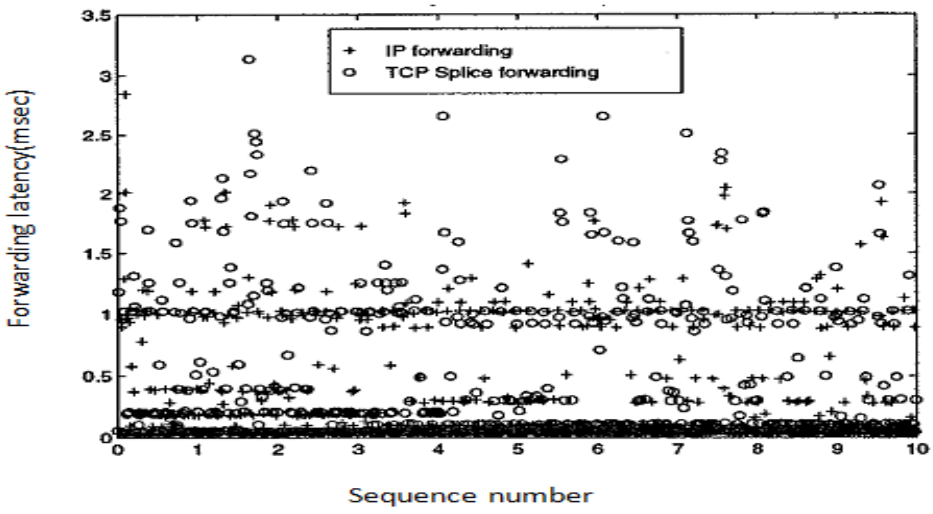


Figure 3.5: Comparison of the latency of IP forwarding with the latency of TCP Splice forwarding

Another issue of concern is how quickly MSOCKS will be able to reconnect TCP connections after the decision to reroute a connection has been made. The time taken by the proxy to resplice two connections is insignificant. The greatest latency in reconnection results from the time required to establish the new TCP connection and transmit the RECONNECT message, which in turn depends critically on the roundtrip time (RTT) of the particular network technology being switched to. A protocol level analysis shows the greatest rate at which a mobile node can reasonably reconnect TCP sessions is limited to once per 2.5 round trip times: 1.5 RTT for the connection establishment, and 0.5 for transmission of the RECONNECT OK message, and 0.5 RTT for the data.

Figure 3.5 compares the latency seen by packets in a TCP connection routed via IP forwarding through our test machine with the latency seen by packets in a TCP connection spliced at our test machine. The X-axis in the figure is the sequence number of the packet. The data from a larger run is summarized in table 3.1

Table 3.1: Summary of forwarding latencies created bt TCP Splice And IP Routing

Criteria	mean(msecs)	median(msecs)
IP Forwrding	0.4038	0.0960
TCP Splice Forwarding	0.4444	0.1120

3.3 SIGMA

SIGMA stands for Seamless IP diversity based Generalized Mobility Architecture. SIGMA[**?**] provides seamless handover for mobile hosts and is based on SCTP, which is a new reliable transport protocol introduced by IETF to transport SS7 signaling messages over IP network. It can greatly reduce the handover latency, packet loss, signaling costs and improve the whole systems throughput compared to the popular Mobile IP[**?**] based handover schemes.

3.3.1 Motivation of SIGMA

Other transport layer mobility protocols implement mobility as an end-to-end service without the requirement to change the network layer infrastructures; many transport layer protocols do not aim to reduce the high latency and packet loss resulting from handovers. But Sigma deals with high latency and packet loss. Seamless means low latency and low packet loss. The basic idea of SIGMA is to decouple location management from data transfer, and achieve seamless handover by exploiting IP diversity to keep the old path alive during the process of setting up the new path during handover.

3.3.2 Detailed Handover Procedure of SIGMA

STEP 1: Obtain new IP address

Refer to Figure 3.6 as an example, the handover preparation procedure begins when MH moves into the overlapping radio coverage area of two adjacent subnets. Once the MH receives the router advertisement from the new access router (AR2), it should begin to obtain a new IP address (IP2 in Figure 3.6). This can be accomplished through several methods: DHCP, DHCPv6, or IPv6 stateless address auto-configuration (SAA).

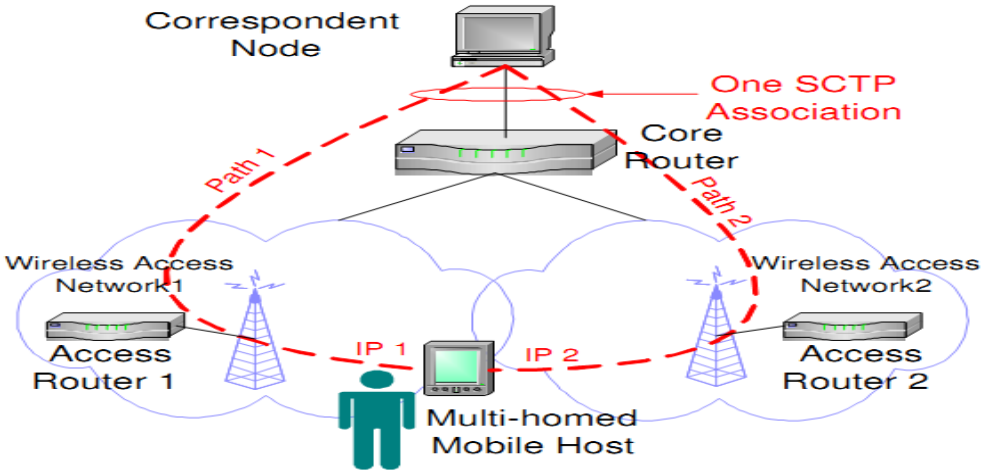


Figure 3.6: An SCTP association with multi-homed mobile host

STEP 2: Add IP addresses into the association

After the MH obtained the IP address IP2 by STEP 1, MH should notify CN about the availability of the new IP address through SCTP Address Dynamic Reconfiguration option. This option defines two new chunk types (ASCONF andASCONF-ACK) and several parameter types (Add IP Address,Delete IP address, and Set Primary Address etc.).

STEP 3: Redirect data packets to new IP address

When MH moves further into the coverage area of wireless access network2, CN can redirect data traffic to new IP address IP2 to increase the possibility that data can be delivered successfully to the MH. This task can be accomplished by sending an ASCONF from MH to CN, through which CN set its primary destination address to MHs IP2.

STEP 4: Update location manager (LM)

SIGMA supports location management by employing a location manager which maintains a database recording the correspondence between MHs identity and MHs current primary IP address. MH can use any unique information as its identity such as home address like MIP, or domain name, or a public key defined in Public Key Infrastructure(PKI). We can observe an important difference between SIGMA and MIP: the location management and data traffic forwarding functions are coupled together in MIP[**?**], while in SIGMA they are decoupled to speedup handover and make the deployment more flexible.

STEP 5: Delete or deactivate obsolete IP address

When MH moves out of the coverage of wireless access network1, no new or retransmitted data should be directed to address IP1. In SIGMA, MH notifies CN that IP1 is out of service for data transmission by sending an ASCONF chunk to CN to delete IP1 from CNs available destination IP list. A less aggressive way to prevent CN from sending data to IP1 is MH advertising a zero receiver window (corresponding to IP1) to CN. By deactivating, instead of deleting, the IP address,SIGMA can adapt more gracefully to MHs zigzag movement patterns and reuse the previously obtained IP address (IP1) as long as the IP1s lifetime is not expired. This will reduce the latency and signalling traffic caused by obtaining a new IP address.

3.3.3 Timing Diagram

Figure 3.7 summarizes the signaling sequences involved in SIGMA. Here we assume IPv6

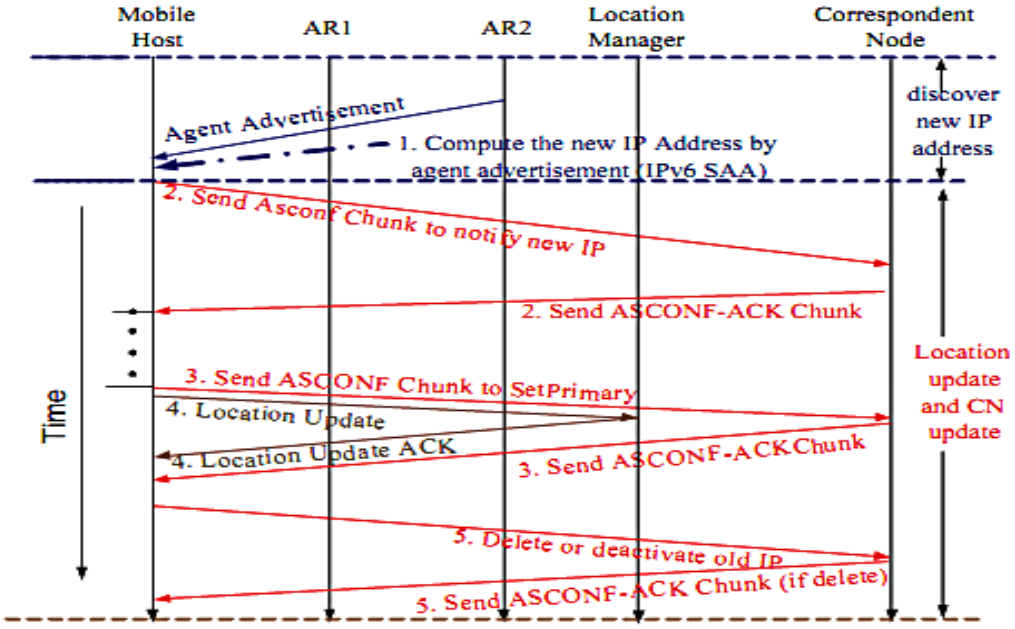


Figure 3.7: Timing Diagram of Sigma

SAA is used by MH to get new IP address. It should be noted that before the old IP is deleted at CN, it can always receive data packets (not shown in the figure) in parallel with the exchange of signaling packets.

3.4 Receiption Control Protocol

RCP moves the responsibility for performing reliability and congestion control from the sender to the receiver. Receiver-centric transport protocol called RCP (Reception Control Protocol) that is a TCP clone in its general behavior, but allows for better congestion control, loss recovery, and power management mechanisms compared to sender-centric approaches. More importantly, in the context of recent trends where mobile hosts are increasingly being equipped with multiple interfaces providing access to heterogeneous wireless networks, we show that a receiver-centric protocol such as RCP can enable a powerful and comprehensive transport layer solution for such multi-homed hosts. We evaluate RCP both to demonstrate its TCP-friendliness, and to highlight its unique benefits when compared to sender-centric transport protocols.

3.4.1 Transposition of Functionalities

TCP is a connection-oriented transport layer protocol that provides reliable in sequence data delivery to the application. Its protocol operation mainly consists of the following four functionalities: connection management, flow control, congestion control, and reliability. Figure 3.8 shows a schematic view of the sender receiver interaction in TCP, along with several state variables.

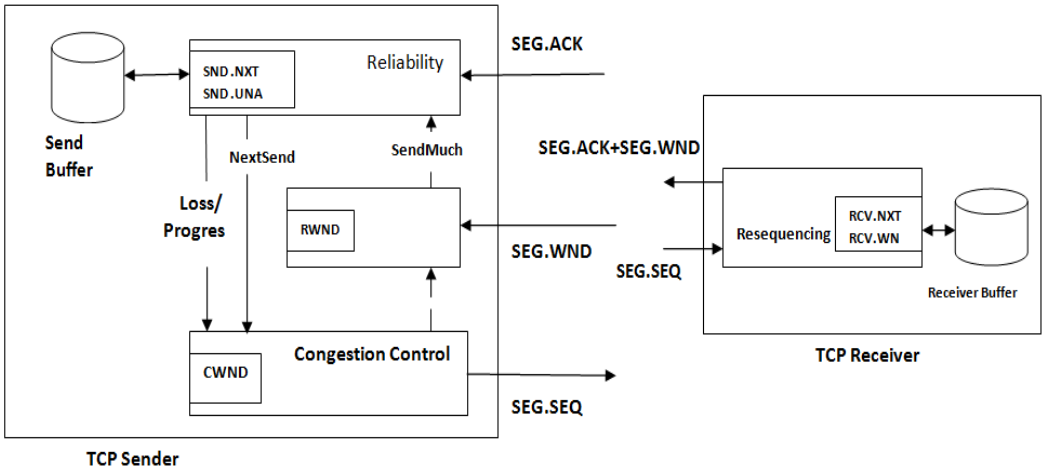


Figure 3.8: TCP sender centric

The connection management is required by any connection-oriented protocol to synchronize connection states between the communicating peers. After the connection is established, the sender in TCP controls the progress of data transfer. The sender drains data from its buffer based on the amount of data that the receiver can accept (flow control), and the amount of data that the network can sustain (congestion control). The receiver performs resequencing and acknowledges data received. Reliable data transfer is achieved through loss detection and loss recovery performed at the sender. It is clear that the connection management cannot be implemented only at one side of the connection, but needs participation of both the sender and the receiver. For the other functionalities, while TCP uses a sender-centric approach, RCP delegates the responsibility to the receiver as shown in Figure 3.9

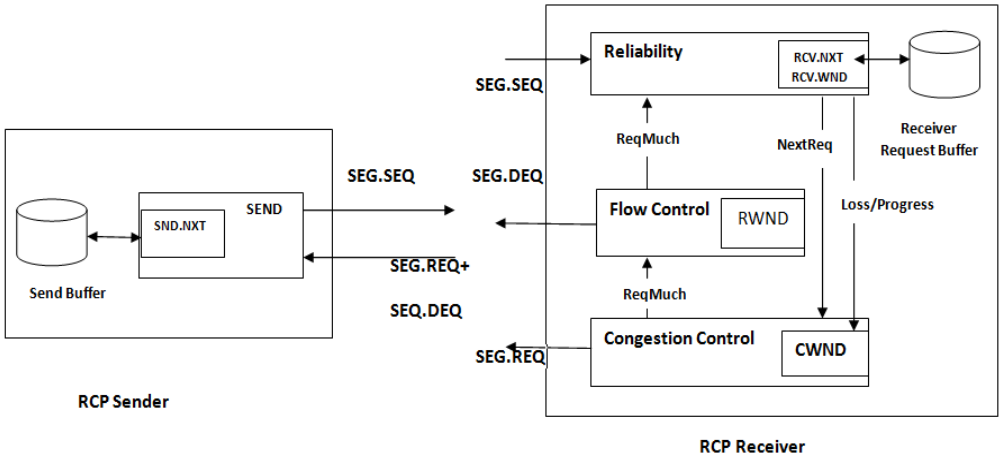


Figure 3.9: TCP receiver centric

While the receiver in TCP merely sends back ACKs with no control over which and in what sequence data is transmitted by the sender, in RCP the receiver explicitly controls these factors and the reliable delivery of data. Moreover, the RCP receiver also assumes total control over the bandwidth the connection can consume, using the same window based algorithm employed by the TCP sender. Finally, although flow control in TCP involves the sender, it is performed solely by the receiver in RCP. Therefore, the receiver in RCP determines how much data the sender can send (via congestion control and flow control), and which data the sender should send (via reliability).

3.4.2 Overview of RCP

In RCP, since the control of data transfer is shifted from the sender to the receiver, the DATA-ACK style of handshaking in TCP is no longer applicable. Instead, to mimic the self-clocking characteristics of TCP, RCP uses the REQ-DATA handshake for data transfer, where any data transferred from the sender is preceded with an explicit request (REQ) from the receiver. Equivalently, RCP uses the incoming data to clock the request for new data. The sender simply maintains the send buffer with one pointer (SND.NXT) indicating the maximum sequence number sent thus far. After the connection is established, the receiver requests data from the sender based on the size of the initial congestion window. The progression of its congestion window follows the slow start, congestion avoidance, fast retransmit, and fast recovery phases just like in TCP. The key difference in the operation is that any trigger for performing congestion control is inferred based on the arrival (or non-arrival) of data segments. For example, a loss is inferred upon the arrivals of three out-of-order data segments instead of ACKs. Upon detection of a segment loss, RCP cuts down its congestion window, and retransmits the corresponding REQ asking for the lost segment. Finally, the receiver performs data resequencing, and gives in-sequence data to the application.

3.4.3 REQ-DATA Handshake

In the DATA-ACK handshake, TCP uses the cumulative acknowledgment for achieving robustness to losses. To emulate this behavior and tolerate loss in the reverse path, RCP allows the receiver to send request either in a cumulative mode or in a pull mode, by appropriately setting the pull flag (PUL) in the packet header. The receiver by default uses the cumulative mode to requests for new data, and uses the pull mode only for retransmission of requests. When the sender receives a request with the pull flag set, it sends only the data segment indicated in the packet header. Otherwise, the sender cumulatively transmits data from SND.NXT that has not been sent yet. Hence, the loss of REQ in cumulative mode has similar impact to that of ACK loss in TCP. To protect REQ in the pull mode from losses, RCP also uses a similar mechanism used by TCP for protecting SACK from losses. The receiver puts the most recent blocks of sequence numbers (we use three blocks as proposed in the SACK) it requested in the REQ header. The sender, in addition to maintaining the

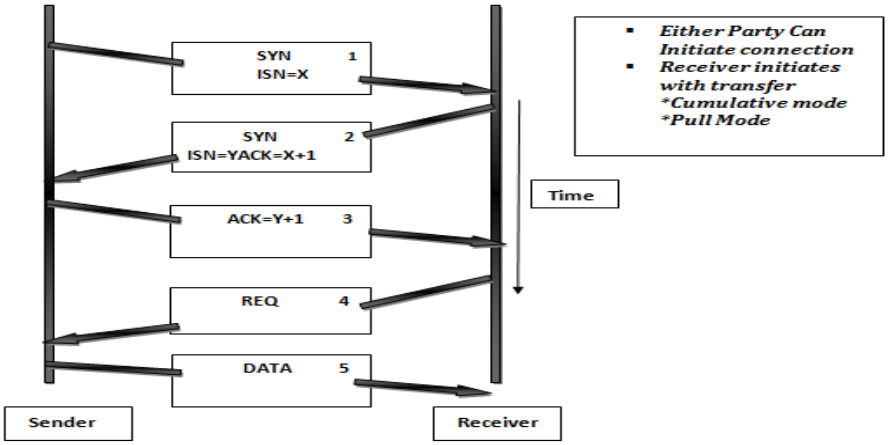


Figure 3.10: Transposition of TCP

send buffer, also maintains a cyclic buffer consisting of the most recent blocks of sequence numbers (three blocks) it sent out. Upon receiving the request from the receiver, the sender checks the consistency between the blocks in REQ and its cyclic buffer. Any mismatch is an indication of REQ losses, and will be recovered by the sender. Note that a request in the pull mode for a specific data segment will be carried in at least four REQs.

3.4.4 Connection Management

Just like in TCP, either the RCP sender or the receiver can initiate the connection setup. The setup process consists of the same SYN-SYN+ACK-ACK handshake as in TCP. However, once the connection is established, instead of the sender sending the first data segment, the RCP receiver transmits the first REQ with the initial sequence number. The sender then transmits the first data segment upon receiving the REQ. The connection teardown in RCP also follows that in TCP.

3.4.5 Congestion Control

In RCP, the receiver performs congestion control and maintains the congestion control parameters including the congestion window CWND and round-trip time information. Since RCP is a TCP clone, it adopts the window based congestion control used in TCP. The slow

start, congestion avoidance, fast retransmits and fast recovery phases are triggered and exited in the same fashion as in TCP. While the same window adaptation algorithm (additive increase, multiplicative decrease) can be implemented either at the sender or at the receiver for performing congestion control, the semantics of the congestion window and the trigger for window increase or cut down are different. In TCP, the size of the congestion window limits the amount of unacknowledged DATA in the network, and the sender uses the return of ACKs to trigger the progression of the congestion window. In RCP, the size of the congestion window limits the amount of outstanding REQs in the network, and the receiver uses the return of DATA to trigger the progression of the congestion window.

3.4.6 Flow Control

Flow control allows the receiver to limit the amount of in-transit data to the available buffer space at the receiverwhen waiting for the application to read (and purge) in-sequence data, or waiting for the arrivals of out-of-order data. In RCP, a request is sent out only if the corresponding data, once received, does not cause buffer overflow at the receiver. This can be achieved by creating a dummy (that does not contain any data) in the receive buffer for each data segment requested. New requests are issued as long as new space is created in the buffer. Unlike TCP in RCP, since the receiver maintains the receive buffer, and has total control over how much data the sender can send, flow control is internal to the receiver. Interestingly, RCP also needs a window field (SEG.DEQ) in the packet header to inform the sender of the highest in-sequence data received so far (which can be calculated at the sender using SEG.REQ-SEG.DEQ), thus allowing the sender to purge such data from its send buffer. The window scale option used in TCP can also be applied to RCP in the same fashion.

3.4.7 Reliability

As Figure 3.9 shows, in RCP the resequencing and reliability functionalities are collocated at the receiver. Upon receiving a data segment from the sender, the receiver enqueues the data in the corresponding dummy and updates RCV.NXT after the resequencing process. In TCP, since reliability is performed at the sender while resequencing is performed at the receiver,

RCV.NXT is conveyed as the cumulative ACK to the sender for it to perform loss detection. However, RCV.NXT conveys limited information about the state of the receive buffer, and hence early implementations of TCP that rely on the cumulative ACK for performing loss detection, suffer from recovering at most one loss per round-trip time, in addition to incurring frequent timeouts . The SACK option is proposed to address this limitation, using which the TCP sender aims to construct the bitmap of the receive buffer in the "scoreboard"data structure. However, in RCP the receiver has direct access to the receive buffer, and hence it can timely and accurately perform loss detection and loss recovery without relying on the use of SACK.

3.4.8 Supporting Heterogeneous Interfaces

Seamless Handoffs

When the coverage areas of different access technologies overlap, it is possible to achieve seamless handoffs at the link layer. However, such link layer handoffs do not necessarily translate into seamless handoffs at the transport layer. Specifically, when mobile host handoffs from one interface to another with an IP address change handled by Mobile IP, the prolonged delay for registration with the home agent can potentially introduce packet losses after the link layer handoff has completed. To prevent TCP from having adverse reactions due to packet losses during handoffs, the mobile host needs to inform the sender of the handoff decision. Whenever feedback information is required, a receiver-centric protocol has advantages over a sender-centric one due to the locality of information needed. However, while it is possible to freeze TCP during handoffs such a stall causes connection disruption and prevents users from enjoying seamless handoffs. One solution to avoid the handoff latency without relying on infrastructure support is to use a mobility-enabled transport protocol for achieving end-to-end host mobility. When the mobile host decides to perform a vertical handoff, it can create a new "data stream"for data transfer through the new address, as soon as the new interface becomes active. With an approach like the mobile host can use multiple TCP pipes (streams) simultaneously without experiencing any connection stall as long as the link layer supports seamless handoffs. A receiver-centric transport protocol thus has advantages over a sender-centric one in such a scenario, since the receiver can accurately control which and how much data to send through each pipe based on the status

(say, signal strength) of each interface. Moreover, when the receiver decides to switch to another interface specific congestion control mechanism after handoffs, such decision does not need to involve the sender, which otherwise would be tasked with, in addition to supporting a plethora of congestion control mechanisms, the seamless transition from one congestion control mechanism to another for a live connection.

3.5 R^{2}CP

R^{2}CP stands for RADIAL RECEPTION CONTROL PROTOCOL. When a mobile host handoffs from one interface to another during a live connection, it can benefit from the following functionalities the transport protocol supports:

1. seamless handoffs without relying on infrastructure support,
2. server migration for achieving service continuity, and
3. bandwidth aggregation using multiple active interfaces.

3.5.1 Receiver-Centric Operation

To achieve optimal performance, a mobile host may need to use network (or interface) specific congestion control. When the mobile host is equipped with heterogeneous wireless interfaces, a receiver-centric protocol allows it to freely use the desired congestion control mechanism depending on the interface it chooses, or the access network it migrates to, without involving the remote server. In addition, during periods of mobility, the mobile host may need to handoff from one server to another (for service continuity), or change the number of servers it connects to (for bandwidth aggregation). It is thus advantageous for the mobile host to use a receiver-centric protocol with a simple sender design, allowing the mobile host to have control over the reliable delivery of data from the sender(s). Being a receiver a receiver-centric protocol that allows the mobile host to drive the protocol operation such as congestion control and reliability, hence turns out to be an ideal protocol for the target environment.

3.5.2 Maintaining Multiple States

Existing transport protocols suffer from performance degradation during handoffs across heterogeneous networks due to the prolonged handoff latency Mobile IP introduces. While end-to-end host mobility without relying on the support from the infrastructure has been proposed, it does not fully address this problem due to the single-state design in TCP that maintains only one TCB per connection. When link layer handoffs invalidate the state maintained at the transport layer (e.g. due to the change in IP addresses), the transport layer protocol needs to modify its state accordingly for achieving transport layer mobility. Although intelligently perform connection migration, it introduces packet losses by overwriting the old state right after the new one is created. An ideal solution for achieving state migration, however, should allow the two states to co-exist in the connection for as long as it takes to handoff the states (considering packets in transit). Therefore, to support transparent host mobility without infrastructure support, a transport layer protocol should be able to handle multiple states. We hence build it as a multi-state extension of RCP.

It dynamically creates and deletes RCP states according to the number of active interfaces in use. It effectively maintains multiple states at the mobile host without requiring explicit support from the remote server. No change is necessary at the RCP sender to support the multistate operation at the receiver. Thus, it is different from related approaches that require changing both ends to support the multi-state operation. Since it is a receiver-only extension of RCP, it allows the mobile host to establish a multipoint-to-point connection to communicate with multiple servers, while in related work multiple states are confined to within a unicast connection.

3.5.3 Overview

Figure 3.11 presents an architectural overview of R^2CP and its key data structures. An R^2CP connection consists of one receiver, and one or multiple senders. Different senders of an R^2CP connection can be located at one or multiple hosts. While a unicast R^2CP connection is in fact equivalent to an RCP connection, a multipoint-to-point R^2CP connection can be considered as an aggregation of multiple RCP connections whose receiving ends are coordinated by an R^2CP engine at the receiver using the interface functions shown in the

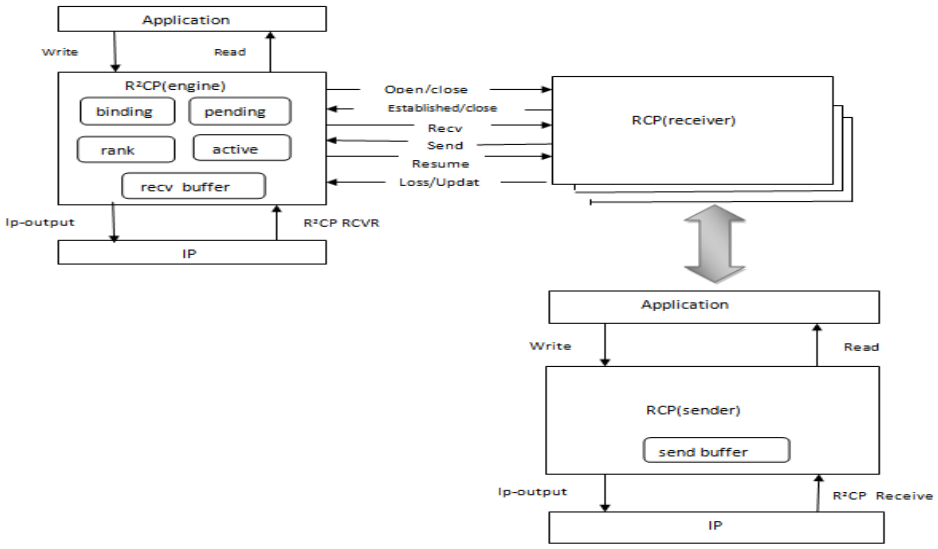


Figure 3.11: R^{2}CP Architecture

figure. We refer to the virtual connections that exist between the R^{2}CP receiver and individual senders as RCP pipes, and focus on the receiver for the following discussions. When the application at the mobile host opens an R^{2}CP connection, initially one RCP pipe is created between the active interface and the remote server. When the mobile host handoffs from one interface to another, a new RCP pipe between the newly active interface and the server is created, after which the old RCP pipe is deleted. However, if bandwidth aggregation is possible (the old interface remains active after handoffs) and desirable (instructed by the application through a socket option), the old pipe is not deleted. If server migration is required when the mobile host handoffs to the new interface, the new RCP pipe is created between the newly active interface and the new server. The application can use a socket option to convey the address of the new server to R^{2}CP. Whenever multiple RCP pipes co-exist in an R^{2}CP connection, the R^{2}CP engine performs transmission scheduling using the data structures, to minimize out-of-order arrivals due to data requested through different RCP pipes. Since multiple RCP pipes collaboratively request data for the same connection, it is possible that data requested through individual pipes is non-contiguous, depending on the transmission schedule used by the R^{2}CP engine. Hence, in R^2 CP the request is always transmitted in the pull mode, such that the sender can transmit only the data requested. However, to facilitate loss detection and loss recovery, at the receiver each RCP pipe internally maintains a local sequence number space. Since the R^{2}CP engine controls the packet I/O (to and from the IP layer), it converts the local sequence number used by each RCP pipe to the global sequence

number used by the aggregate connection before sending out the packet, and vice versa.

3.5.4 Connection Management

When R^{2}CP creates an RCP pipe, it uses the open() call to make the RCP pipe start the connection setup procedure. The connection setup procedure for each RCP pipe is same as RCP connection management. When the RCP pipe is established, it uses the established() call to notify R^{2}CP. The R^{2}CP connection is established when any of the RCP pipe returns with the established() call. On the other hand, when R^{2}CP deletes an RCP pipe, it uses the close() call to make the RCP pipe enter the closing handshake. When all RCP pipes return with the close() call, the R^{2}CP connection is closed.

3.5.5 Congestion Control

Congestion control in an R^{2}CP connection is performed on a per pipe basis, where each RCP pipe is responsible for controlling the amount of data transferred through the respective path. R^{2}CP decides the congestion control mechanism to use for each wireless interface by opening an appropriate RCP pipe. We assume the choice as to which congestion control scheme to use for each interface is an external decision, and is provided to R^{2}CP through a system configuration or a socket option.

3.5.6 Flow Control

Since R^{2}CP has control over the receive buffer, it is responsible for the flow control of the aggregate connection. R^{2}CP freezes a requesting RCP pipe if it finds that the number of outstanding data is equal to the available buffer space. It de-freezes concerned pipes through the resume() call when any space is created in the buffer. The flow control mechanism for individual RCP pipes will not kick in since they do not deal with the actual data segments. Note that R^{2}CP is also responsible for appropriately informing the senders about what data to purge using the SEG.DEQ field in the RCP header.

3.5.7 Reliability

R^2CP is primarily responsible for the reliable data transfer of the aggregate connection. It achieves this goal by maintaining the binding information for all data segments. Once a segment is bound to a particular RCP pipe, the concerned pipe will take over the responsibility (since RCP is a reliable protocol). However, note that when an RCP pipe detects a segment loss and reports to R^2CP using the loss() call, R^2 CP will unbind the corresponding data segment, and delegate the reliable transfer of the lost segment to the next available pipe (according to the rank). While the original RCP pipe will still strive to deliver the same segment (in terms of the RCP sequence number) via retransmissions, it will be assigned a different data segment by R^2CP

3.5.8 Seamless Handoffs

When mobile hosts handoff between heterogeneous wireless networks, a key challenge in supporting seamless handoffs is the problem associated with address change and prolonged registration delay. Conventional approaches for performing vertical handoffs suffer from connection disruptions due to this problem. The multi-state design in R^2CP allows it to open multiple connections (pipes) associated with the wireless interfaces that become active during handoffs. By retaining the old connection (for as long as the link layer supports) during the initial setup delay of the new connection, the application can continue transmitting and receiving data from either or both interfaces without being disrupted during handoffs. When the mobile host handoffs from one access network to another the mobile host is initially connected to Server-I through network A and hence one RCP pipe (RCP-1) is created in the R^2CP connection. After some time the mobile host decides to handoff to network B, so a second RCP pipe (RCP-2) is created (using the new network address). However, RCP-1 is not closed until some more time, and hence during this time two pipes co-exist in the connection to collaboratively deliver data for the application. Even if there is some setup or ramp-up (e.g. due to slow start) delay for the RCP-2 pipe, the existence of the RCP-1 pipe allows the aggregate connection to continue progressing without being disrupted. This is very different from related work that uses a single state transport protocol for handoffs. Since R^2CP is a multi-state transport protocol, it is capable of maintaining multiple (in-

terface specific) pipes effectively in a connection without suffering from problems due to packet reordering or duplicates. The redundant striping technique can also be used during handoffs for achieving better performance.

3.5.9 Server Migration

A key difference between R^{2}CP and other multi-state transport protocols is the ability to support end-point handoffs in R^{2}CP. By virtue of its receiver-centric design, the sender does not maintain any hard state (e.g. retransmission timers) of the connection. Since the mobile host controls which data to receive from the sender, handoffs from one server to another can be as simple as stop requesting data from the old server, and start from the new one. Server migration involves interaction between the transport layer and higher layer protocols. We focus in this section on the ability of R^{2}CP to facilitate server migration given sufficient support from the higher layers, and hence motivate its use as a valuable and effective building block for end-to-end mobility support frameworks.

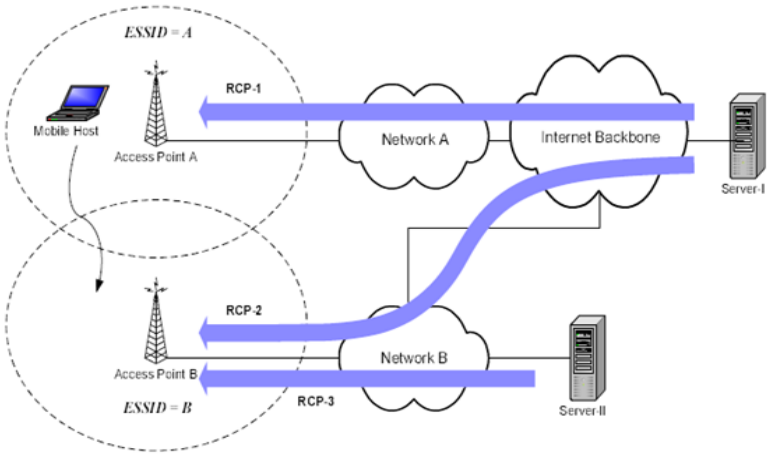


Figure 3.12: R^{2}CP Testbed Scenerio

As Figure 3.12 shows, when the mobile host moves to network B, it has access to a replicated server (Server-II). The end-to-end path from the mobile host (using interface B) to Server-II has a shorter round-trip time and a larger bandwidth, and hence the mobile host decides to perform server migration from Server-I to Server-II. Initially, the R^{2}CP connection creates an RCP pipe (RCP-1) using network address A and the address of Server-I. When the mobile

host moves to network B, R^{2}CP creates a new RCP pipe (RCP-3) using network address B and the address of Server-II. At first mobile host does not perform server migration, and hence the second RCP pipe created (RCP-2) is between network address B and the address of Server-I. After the new RCP pipe is established, the mobile host requests data that has not been delivered by Server-I, instead of requesting from the first byte of the data. Approaches used for achieving seamless handoffs discussed previously can also be used for achieving seamless server migration. Based on the content of its receive buffer, R^{2}CP may request non-contiguous data from Server-II. Hence server migration using R^{2}CP does not cause redundant transmissions compared to that using only TCP (the TCP sender delivers only in-sequence data stream). While support for selective pulling of data is provided by some applications (e.g. HTTP 1.1 Range Requests), it can be achieved in R^{2}CP with no support from the server side application.

3.5.10 Bandwidth Aggregation

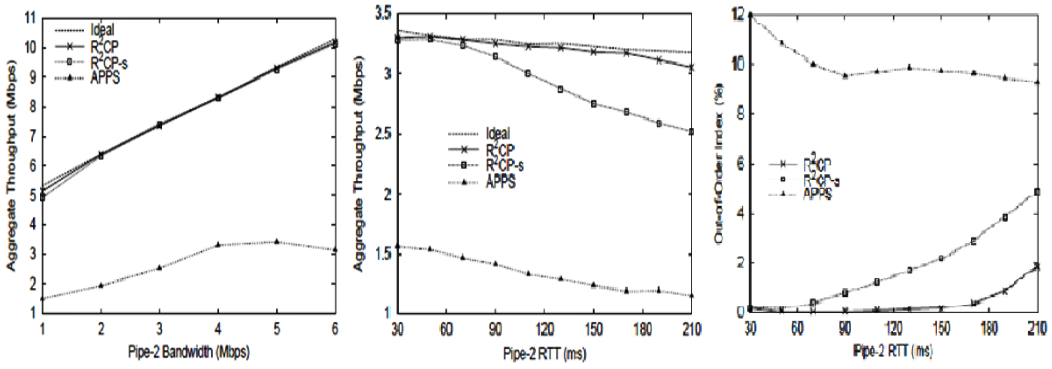
When a mobile host handoffs between heterogeneous wireless networks, it is possible that the old connection remains active after the handoff is complete. In such a case, it would be advantageous for the mobile host to achieve aggregate bandwidths by simultaneously using both interfaces. Since R^{2}CP allows multiple RCP pipes to co-exist in one connection, and performs effective transmission scheduling for striping across multiple pipes, a mobile host using R^{2}CP can easily achieve bandwidth aggregation if desired. While bandwidth aggregation can be achieved between the mobile host and one server (point-to-point), we consider a scenario where the two pipes connect to different servers (multipoint-to-point). The mobile host opens the RCP-1/RCP-2 pipe between network address A/B and the address of Server-I/Server-II respectively. However, instead of closing the RCP-1 pipe after RCP-2 is established, the mobile host keeps both pipes open during the period it is within the coverage of both WLANs. R^{2}CP can achieve the aggregate bandwidth of the two pipes. We now use simulation to evaluate the performance of R^{2}CP in achieving effective bandwidth aggregation under various network conditions. We use a network topology similar to the topology. The mobile host opens two pipes to aggregate bandwidths from different servers. We vary the characteristics of the two paths, in terms of the bandwidth of the bottleneck link, and the round-trip time of the entire path, to introduce bandwidth mismatches and

delay mismatches. We also introduce bandwidth fluctuations by using on/off traffic sources. We compare the performance of R^{2}CP against the following approaches:

1. Ideal: the ideal performance of bandwidth aggregation, where the aggregate bandwidth equals the sum of bandwidths along the two pipes;

2. APPS: an application layer striping approach, where the application stripes across multiple RCP connections without using R^{2}CP;

3. R^{2}CP-s: a simplified version of R^{2}CP, where the data request is assigned to individual pipes on a first-come-first-served basis without considering the round-trip times.



(a) Bandwidth Mismatch (b) RTT Mismatch (c) Out-of-Order Index

Figure 3.13: R^2 CP Performance

Due to lack of space, we present only a subset of the performance results in Figure 3.13. In Figure 3.13(a), we vary the bandwidth of the two pipes such that the bandwidth of the first pipe is fixed at 4Mbps, while that of the second pipe varies from 1Mbps to 6Mbps. We observe that both R^{2}CP and R^{2}CP-s achieve the ideal performance irrespective of the bandwidth mismatches. The application striping approach fails to achieve the desired performance for the same reason explained in. In Figure 3.13(b), we vary the roundtrip time of the two pipes such that the RTT of the first pipe is fixed at 30ms, while that of the second pipe varies from 30ms to 210ms. We find that while the performance of R^{2}CP still closely tracks the ideal performance, R^{2}CP. R^{2}CP-s fails to scale when the RTT mismatch increases beyond 3. The performance degradation of R^{2}CP-s is due to the scheduling used that does

not take into consideration the round-trip times of different pipes. While an FCFS style of striping policy works well when the round-trip times of different paths are comparable, as the RTT mismatches increase, it suffers from frequent out-of-order arrivals. Due to the limited space in the R^2CP receive buffer, head-of-line blocking eventually triggers the flow control of R^2CP and causes the progression of the aggregate connection to stall. We show in Figure 3.13(c) the percentage of packets that find the buffer 75% full upon arrivals, for three different striping approaches. The reason for the non-performance of the application striping approach is clear from the figure. While R^2CP-s manages to maintain a small queue size when the RTT mismatches are small, the queue builds up noticeably as the RTT mismatches increase. R^2CP, on the other hand, achieves better performance even with large RTT mismatches.

3.6 FREEZE TCP

With explosive growths in wireless services and their sub-scribers, as well as portable and affordable computing devices; it is natural that supporting user mobility in the Internet is a hot and exciting issue that has attracted extensive efforts. the basic mobile IP protocol is more or less standardized, researchers are beginning to focus on performance enhancing mechanisms at all layers of the networking stack in order to deliver high performance at the end-user level.

TCP is a vital component of the Transport layer of the Internet protocol suite. It is intended to provide connection oriented reliable service over an underlying unreliable network. It is there-fore not surprising that TCP has received a lot of attention and fairly large number of researchers have tried to optimize and improve TCP for different environments characterized by heterogeneous subnetworks with widely different bandwidths and latencies (for instance TCP over wireless links, satellite links, slow serial links, etc.).

In the following, we first outline the problems with TCP in mobile environments. Next, we summarize the proposed solutions, indicating their strengths and weaknesses, and the current status of TCP enhancements/modifications and our solution will be Freeze TCP.

3.6.1 TCP window management and mobile environment

TCP uses a sliding window mechanism to accomplish reliable, in-order delivery and flow/congestion control. Figure 3.14 shows this graphically, with the window sliding towards the right. The window size (W) is determined as the minimum of receiver's advertised buffer space, and the perceived network congestion. The sender allows up to W outstanding or unacknowledged packets at a time. This results in a "usable window "size equal to W minus the number of outstanding packets.

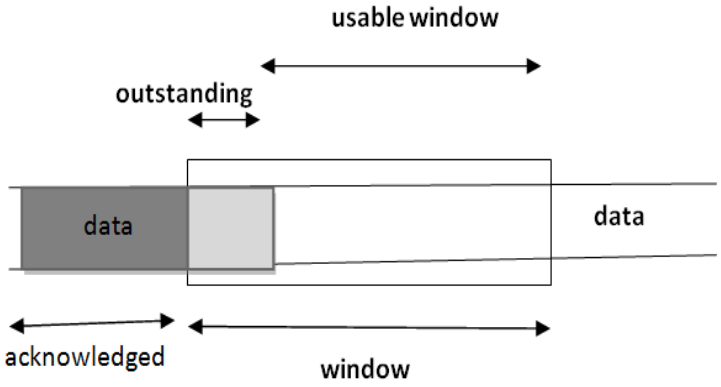


Figure 3.14: TCP Window Management

Under normal conditions, the right edge of the window stays fixed (when the packets in the current window remain unacknowledged), or advances to the right along with the left edge of the window, as packets are acknowledged. If the consuming process at the receiver end is slower than the sender, the receiver's buffers will begin to fill causing it to advertises progressively smaller and smaller window sizes. Eventually the receiver may run out of buffer space in which case it advertises a window size of zero.

Upon seeing an advertised window size of zero, the sender should freeze all re-transmit timers and enter a persist mode. This involves sending probes (called the Zero Window Probes or ZWPs) until the receiver's window opens up. In a strict sense, each ZWP should contain exactly one byte of data but many TCP implementations including those in Linux and FreeBSD do not include any data in their ZWPs. The interval between successive probes grows exponentially (exponential back-off) until it reaches 1 minute, where it remains constant. Because these probes are not delivered reliably, the sender does not drop its congestion

window if a Zero Window Probe itself gets lost. Eventually the receiver responds to a ZWP with a non-zero window size, and the sender will continue transmission using a window size consistent with the advertised value.
An exception to this normal window management operation occur if the receiver "shrinks"its advertised window, that is moves the right edge towards the left. This can suddenly create a negative usable window size which might confuse the sender. While this behavior is discouraged, the sender must recover if it occurs. The sender is allowed to retransmit any outstanding packets (up to W), but should not send new data. Also, any lost packets from within the old window (and now to the right of the new window because the right edge moved leftward) should not cause the congestion window to drop. This means that if the receiver shrinks its window to zero, all outstanding packets can be lost without affecting the sender's congestion window and the sender should enter the persist mode described above.

3.6.2 Problems with TCP in mobile environments

TCP was conceived for wired, fixed topologies which are fairly reliable. Hence it operates on the assumption that any losses are due to congestion, which is reasonable for a reliable infrastructure. In mobile environments, however, losses are more often caused by

1. The inherently higher bit error rates of the wireless links, and
2. Temporary disconnections (due to signal fading or other link errors; or because a mobile node moves, etc).

To better illustrate the second item above, it should be noted that mobility is distinct from wireless connectivity. For instance, a user working in the office on a notebook wants to move (with the notebook) to a laboratory or a meeting room at the other end of a building or in the next building, where the IP addresses can be on different subnets; possibly across one or more firewalls. FTP, Telnet sessions and other connections can certainly remain alive for a few minutes it might take to go from one end of a building to another. The idea behind mobility is that such open connections should be retrieved seamlessly despite the move and a change of the underlying IP address.

Even if a single packet is dropped for any reason, the current standard implementation of TCP assumes that the loss was due to congestion and throttles the transmission by bringing the congestion window down to the minimum size. This, coupled with the TCP's slow-start mechanism means that the sender unnecessarily holds back, slowly growing the transmission rate, even though the receiver often recovers quickly form the temporary, short disconnection. This is illustrated in Figure 3.15. where it is seen that the network capacity can remain unutilized for a while even after a reconnection.

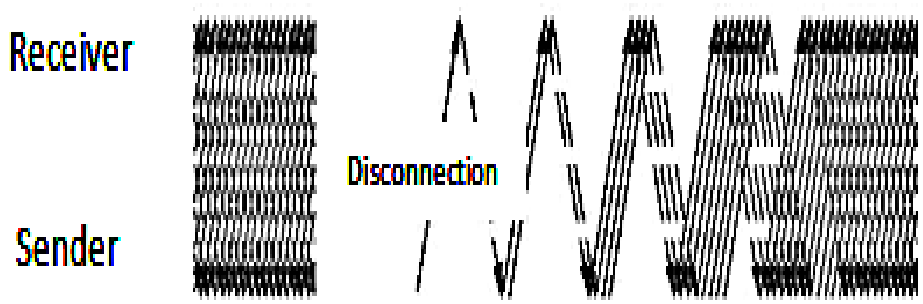


Figure 3.15: TCP Slow Start

3.6.3 Existing solutions

Several approaches have been proposed to overcome these shortcomings of standard TCP. The Berkeley Snoop module resides on an intermediate host (preferably the base station), near the mobile user. It caches packets from the sender and inspects their TCP headers. Using the snooped information, if the module determines that a packet has been lost, it retransmits a buffered copy to the mobile node. It maintains it's own timers for retransmission of buffered packets, implements selective retransmissions, etc.

Indirect TCP (I-TCP) proposes to split the connection between a fixed sender host (FS) and mobile host (MH) at a mobility support station. The data sent to MH is received, buffered and ACKed by BS. It is then the responsibility of BS to deliver the data to MH. On the link between BS and MH, it is not necessary to use TCP. One can use any other protocol optimized for wireless links. MTCP proposed in is similar to I-TCP and also splits a TCP connection into two: one from MH to BS and the other from BS to FH. The MH to BS

connection passes through a session layer protocol which can employ a selective repeat protocol (SRP) over the wireless link. In a method it is proposed to delay the duplicate ACKs for a missing packet (which could trigger a fast retransmission from the sender) in order to allow any special local retransmissions on the wireless links to work, before forcing the sender to fast-retransmit the missing packet(s) and more other method are exists.

3.6.4 Strengths and Drawbacks of Existing Solutions

Next we consider major factors (not necessarily in the order of importance) that should be considered in assessing any TCP enhancement scheme.

1. One of the main considerations is inter-operation with the existing infrastructure. To realize this goal, ideally, there should not be any change required at intermediate routers or the sender because these are likely to belong to other organizations, making them unavailable for modifications. All approaches that split the connection into two parts require substantial modification and processing at an intermediate node (BS).

2. The second important issue is encrypted traffic. As network security is taken more and more seriously, encryption is likely to be adopted very widely. For instance, IPSEC is becoming an integral part of IPv6, the next generation IP protocol. In such cases the whole IP payload is encrypted, so that the intermediate nodes (be it the base station or another router) may not even know that the traffic being carried in the payload is TCP. Any approach (such as SNOOP, I-TCP, MTCP, M-TCP) which depends on the base station doing a lot of mediation will fail when the traffic is encrypted.

3. Even more serious, sometimes data and ACKs can take different paths (for instance, in satellite networks). Schemes based on “intermediary”involvement will have serious problems such a case.

4. Yet another consideration is maintaining true end-to-end semantics. I-TCP and MTCP do not maintain true end-to-end semantics.

5. Even if one assumes that issues (1)-(4) above are not relevant, and that an intermediary (such as a base station) can be brought in for performance enhancements; there is

still a need to consider whether the intermediary will become the bottle-neck. It is clear that the base stations (BS) in SNOOP, I-TCP, MTCP, M-TCP will all have to buffer at least some amount of data and do some extra processing for each connection going through them. If hundreds or thousands of nodes are mobile in the domain of a base station, it could get overwhelmed with the processing of traffic associated with each connection. When a mobile node moves from the domain of one BS to another, the entire "state"of the connection (including any data that was buffered for retransmissions) needs to be handed over to the new base station. This can cause significant amount of overhead and might lead to the loss of some packets and the sender dropping congestion window, which would defeat the original purpose behind the whole endeavor.

3.6.5 Main Idea

The main idea behind Freeze-TCP is to move the onus of signaling an impending disconnection to the client. A mobile node can certainly monitor signal strengths in the wireless antennas and detect an impending handoff; and in certain cases, might even be able to predict a temporary disconnection (if the signal strength is fading, for instance). In such a case, it can advertise a zero window size, to force the sender into the ZWP mode and prevent it from dropping its congestion window. As mentioned earlier, even if one of the zero window probes is lost, the sender does not drop the congestion window. To implement this scheme, only the client's TCP code needs to change and there is no need for an intermediary (no code changes are required at the base station or the sender).

If the receiver can sense an impending disconnection, it should try to send out a few (at least one) acknowledgements, wherein it's window size is advertised as zero (let an ACK with a zero receiver window size be abbreviated "ZWA", i.e., Zero Window Advertisement). The question is: how much in advance of the disconnection should the receiver start advertising a window size of zero? This period is in a sense the "warning period"prior to disconnection. Ideally, the warning period should be long enough to ensure that exactly one ZWA gets across to the sender. If the warning period is any longer, the sender will be forced into Zero Window Probe mode prematurely, thereby leading to idle time prior to the disconnection.

If the warning period is too small, there might not be enough time for the receiver to send out a ZWA which will cause the sender's congestion window to drop due to packets lost during the disconnection (which, in turn leads to some idle-time/underutilization after the reconnection).

Given this, a reasonable warning period is the round-trip-time (RTT). During periods of continuous data transfer, this allows the sender to transmit a packet and then receive its acknowledgment. Experimental data corroborates this: warning periods longer or shorter than RTT led to worse average performance in most cases we tested. Note that Freeze-TCP is only useful if a disconnection occurs while data is being transferred (as op-posed to when the receiver is idle for some time and then gets disconnected), which is the most interesting case anyway.

Since the ZWPs are exponentially backed off, there is the possibility of substantial idle time after a reconnection. This could happen, for instance, if the disconnection period was long and the reconnection happened immediately after losing a ZWP from the sender. In that case, the sender will go into a long back-off before sending the next probe. Meantime the receiver has already reconnected, but the connection remains idle until the sender transmits its next probe. To avoid this idle time, we also implement the scheme. As soon as a connection is re-established, the receiver sends 3 copies of the ACK for the last data segment it received prior to the disconnection. This scheme is henceforth abbreviated as "TR-ACKs"(Triplicate Reconnection ACKs). Note that even in standard TCP, packet retransmissions are exponentially backed off. Therefore the post reconnection idle time can occur there as well. For a fair comparison, the Standard TCP on the receiver side was also modified to optionally send TR-ACKs. This way, the effect of only the Freeze-TCP mechanism (i.e., forcing the sender into ZWP mode prior to a disconnection) can be isolated. Unlike M-TCP, there is no advantage to holding back the ACK to the last byte. For M-TCP it was useful because even when the mobile client was disconnected, the base station could still signal the sender on behalf of the client. In the case of Freeze-TCP, since changes are restricted to the client end, holding back the ACK for the last byte does not help. Note that Freeze-TCP will avoid any repacketization penalty at the sender end (which M-TCP might incur because it holds back the ACK to the last byte). Figures 3.16 and 3.17 help estimate the performance gain possible due to the Freeze-TCP technique.

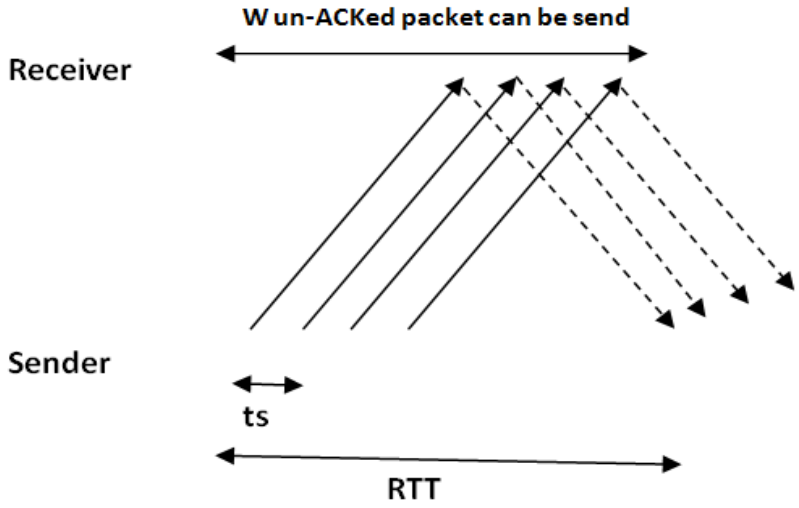


Figure 3.16: Relation between ts,RTT,W

In Figure 3.16, ts is the time required to "put or write the packet on the wire", RTT is the total round trip delay including the ts delays at sending, receiving as well as any intermediate nodes; and W is the senders window. From the figure, it is seen that if any idle periods are to be avoided:

$$W \cdot ts \geq RTT \ldots\ldots (1)$$

Since; ts = packet-size/bandwidth;
(ignoring processing/queuing delays internal to the host, collisions in case of shared medium, etc.) it is seen that the [delay bandwidth] product is important in determining how big the congestion windowW needs to be if underutilization of network capacity is to be avoided.

$$\text{Assuming; } RTT/ts \gg 1; W \gg 1 \ldots\ldots (2)$$

is required for (2) full network capacity utilization. Figure 3.17 pictorially illustrates the increased throughput under this condition, when Freeze-TCP prevents sender side window, W, from dropping and regrowing (due to packet losses). From the figure it can be seen that the (approximate) number of extra packets transferred by the Freeze-TCP scheme is given by

$$\text{Extra Segments} = W2/8+WlgW-5W/4+1 \ldots\ldots (3)$$

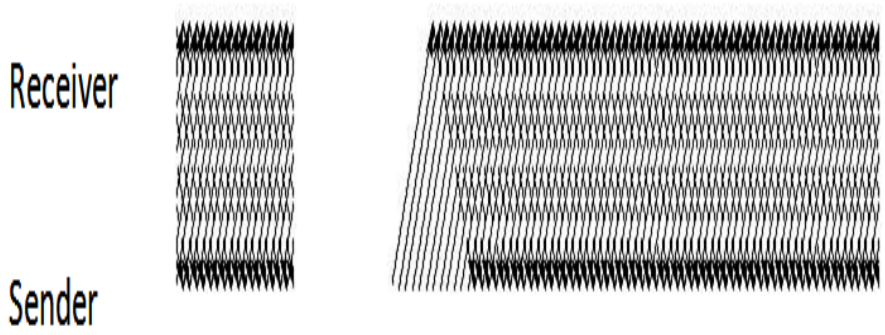


Figure 3.17: Freeze TCP

In addition to (2), the above expression (3) also assumes that upon a disconnection (and the loss of packets), regular TCP drops the congestion window all the way down to 1, and first grows it by a factor of 2 each time an ACK is received, until it reaches W=2. From there on, it is incremented by 1 each time an ACK is received until it reaches the same size W prior to disconnection. This congestion window growth mechanism is dubbed“slow-start congestion avoidance”. It should be noted that (3) is an approximate expression, ignoring collisions, and other factors that might affect the traffic.

3.6.6 Final Words

Freeze-TCP is a connection migration scheme that lets the MH ‘freeze’or stop an existing TCP connection during handoff by advertising a zero window size to the CN, and unfreezes the connection after handoff. This scheme reduces packet losses during handoff at the cost of higher delay. Although it provides transparency to applications, It is a true end-to-end signaling scheme and does not require any intermediaries (such as base stations) to participate in the flow control. Freeze-TCP only deals with connection migration, but does not consider handoff or location management. It can be employed with some other schemes like Migrate to implement a complete mobility management scheme.

3.7 MIGRATE TCP

Todays Internet services are commonly built over TCP, the standard Internet connection-oriented reliable transport protocol. The endpoint naming scheme of TCP, based on network layer (IP) addresses, creates an implicit binding between a service and the IP address of a server providing it, throughout the lifetime of a client connection. This makes a TCP client prone to all adverse conditions that may affect the server endpoint or the internetwork in between, after the connection is established: congestion or failure in the network, server overloaded, failed or under DOS attack. Studies that quantify the effects of network stability and route availability demonstrate that connectivity failures can significantly impact Internet services. As a result, although highly available servers can be deployed, sustaining continuous service remains a problem. Service continuity can be defined as the uninterrupted delivery of a service, from an end users perspective. The TCPs ability to support it is limited by its error recovery scheme based on retransmissions to the same server endpoint of the connection (bound to a specific IP address). In practice, the end user might be more interested in receiving continuous service rather than statically binding to a given server. As server identity becomes less important than the service, it is desirable for a client to switch servers during a service session, e.g., if a server cannot sustain the service. We propose the cooperative service model, in which a pool of similar servers, possibly geographically distributed across the Internet, cooperate in sustaining a service by migration of client connections within the pool. The control traffic between servers, needed to support migrated connections, can be carried either over the Internet or over a private network. From clients viewpoint, at any point during the lifetime of its service session, the remote endpoint of its connection may transparently migrate between servers.

3.7.1 Migratory TCP

To enable the cooperative service model for service continuity we have designed Migratory TCP (M-TCP), a reliable connection-oriented transport layer protocol that supports efficient migration of live connections. The protocol enables stateful servers to seamlessly resume service on migrated connections by transferring an application controlled amount of specific state. Although fine-grained connection migration solutions have been proposed before by

exploiting features of application-level protocols like HTTP, to our best knowledge M-TCP is the first solution that provides generic migration support through a TCP-compatible transport protocol. The M-TCP design assumes that the state of the server application can be logically split among connections by defining fine-grained state associated with each connection. The M-TCP service interface can be best described as a contract between the server application and the transport protocol. According to this contract, the application must execute the following actions:

1. Export a state snapshot at the old server, when it is consistent with data sent/received on the connection;
2. Import the last state snapshot at the new server after migration, to resume service to client. In exchange, the protocol: transfers the per-connection state to the new server and synchronizes the per connection application state with the protocol state.

The migration mechanism of M-TCP (Figure 3.18) ensures that the new server resumes service while preserving the exactly-once delivery semantics across migration, without freezing or otherwise disrupting the traffic on the connection. The client application does not need to change. A client contacts the service through a connection Cid to a preferred server S1. At connection setup, S1 supplies the addresses of its cooperating servers, along with migration certificates. The client-side M-TCP initiates migration of Cid by opening a new connection to an alternate server S2, sending the migration certificate in a special option. To reincarnate Cid at S2, M-TCP transfers associated state (protocol state and the last snapshot) from S1.

Depending on the implementation, the state transfer can be either

1. Lazy (on-demand), i.e., it occurs at the time migration is initiated.
2. Eager, i.e., it occurs in anticipation of migration, e.g., when a new snapshot is taken. Figure 3.18 shows the lazy transfer version: S2 sends a request (b) to S1 and receives the state (c). If the migrating endpoint is reinstated successfully at S2, then C and S2 complete the handshake, which ends the migration (d). Upon accepting the migrated connection, the server application at S2 imports the state snapshot. It then resumes service using the snapshot as a restart point, and performs execution replay for a log-based recovery supported by the protocol. The execution replay restores the state of

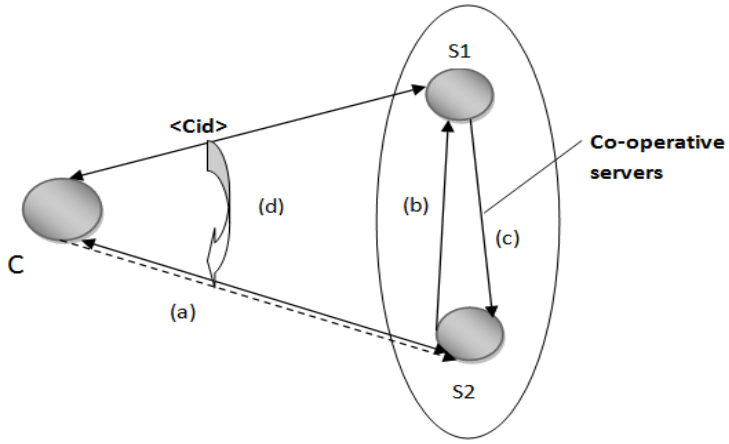


Figure 3.18: Migration mechanism in M-TCP

the service at the new server and synchronizes it with the protocol state. To support the replay, M-TCP logs and transfers from S1 data received and acknowledged since the last snapshot. It also transfers unacknowledged data sent before the last snapshot, for retransmission from S2.

3.7.2 Applications

We have implemented M-TCP in FreeBSD as an extension to the TCP/IP stack, compatible and inter-operable with the standard TCP. M-TCP is decoupled from and can work with various migration policies. We identify two classes of services that can benefit from M-TCP:

1. Applications that use long-lived connections, e.g., multimedia streaming services, applications in the Internet core etc.;

2. Critical applications from which end users expect both correctness and good response time, e.g., Internet banking, e-commerce, etc.

To demonstrate the potential of M-TCP in providing service continuity, we have implemented and evaluated two applications. The first one is a synthetic (generic) media streaming server, for which we use M-TCP in conjunction with a migration policy based on esti-

mated inbound data rate. Migration is triggered when the throughput perceived on the client side falls under a fraction of the maximum observed. We show that, for the same profile of performance degradation at a server, M-TCP can sustain effective throughput close to the average server profile by migrating the connection between servers. The second application is remote access over the Internet to a transactional database server. We have augmented a PostgreSQL database back-end with support for migratory front-end contexts and used M-TCP between clients and front-end hosts. The resulting system allows a client to start a sequence of transactions with one front-end, then migrate and continue the execution on other front-ends if necessary. The system ensures that ACID semantics are preserved and that the execution is deterministic across migration.

CHAPTER 4

COMPARATIVE STUDIES

4.1 Introduction

The components of a complete mobility management scheme consist of handoff, connection migration, and location management. Evaluation criteria have to be developed to compare the effectiveness of mobility schemes. The criteria could include handoff, packet loss and delay, fault tolerance, requirement for change in network infrastructure, mobility type, support for IP diversity, security, scalability, etc. In this paper, we use the above criteria to classify the proposed mobility schemes.

4.2 Fundametal of Mobility Management

Mobility management in data networks involves changing the point of attachment, and hence the IP addresses, of a mobile host (MH). A change in IP address gives rise to the challenges in maintaining an uninterrupted data flow while the MH is changing its address, minimizing loss of packets, maintaining security, identification of the newer location, etc. Some fundamentals are discussed below

4.2.1 Connection Migration

An MH acquires a new IP address when it changes its subnet. Since the old IP address is retained, a natural question to answer is how the CN will continue communicating with the MH which now has multiple IP addresses. Connection Migration, which involves notifying the CN about this change and migrating the connection from the old to the new address, is a possible scheme. To avoid data flow through the old address of MH, connection migration

may result in a temporary stop in the data flow during the migration process. A gateway in the middle of the connection may be used to handle the connection switching. Some protocols support multiple IP addresses for a single MH having multiple interfaces, thus enabling a smooth transition from one interface to another when changing subnets.

4.2.2 Packet Loss and Latency

When an MH acquires a new IP address, unless the MH and the underlying protocols support multiple addresses, the MH can only be contacted via the new address. Packets destined to the MH via the old address cannot reach the destination, resulting in packet loss, latency and wastage of internet bandwidth. Mobility schemes must come up with techniques to mitigate packet losses and latency during handoffs.

4.2.3 Infrastructure Requirement

The Internet was no initially designed with mobility in mind. Consequently many of the proposed schemes require changes in the existing Internet infrastructure, such as gateway or proxy in the middle of the connection, to support mobility.

4.2.4 Location Management

Following the change of IP address of an MH, a CN should be able to locate the MH. A location manager keeps track of the current IP address of an MH, and provides the current address to any entity trying to initiate communication with the MH.

4.3 Evaluation Criteria

When an MH decides to detach itself from one subnet and connect to another one based on the signal strength of neighboring subnets, the MH obtains a new IP address from the new subnet. The data already in transit to the MH's old IP address may be lost, resulting

in increased delay due to retransmission of the lost packets. The change in point of attachment may be confined to a single subnet or a group of neighboring subnets. The handoff may require applications running on MH and CN to be aware of mobility, thereby reducing application transparency. Additionally, handoff between subnets may also result in conflict with standard network security solutions, and may require additional hardware/software to be deployed in the existing network infrastructure.

4.3.1 Handoff Process

The performance of a mobility management scheme depends on the type of handoff which can be either soft or hard. Soft handoff (also called seamless handoff) permits a smooth handoff by allowing a mobile to communicate and exchange data with multiple interfaces simultaneously during handoff. Communication through the old interface is dropped when the signal strength from the corresponding access point drops below a certain threshold. On the contrary, hard handoff results in disconnecting from the old access point when the signal strength is below a threshold before connecting to the new access point.

4.3.2 Scalability and Fault Tolerance

Scalability refers to the ability of a mobility management scheme to handle a large number of MHs and CNs. A scheme is scalable when its performance does not drop with an increase in the size of the network size or the number of MHs and CNs. A system is said to be fault tolerant when it can function in the presence of system failures. For example, a scheme with a single point of failure in said to be faulting intolerant.

4.3.3 Application Transparency

A mobility scheme is transparent to an application when the application does not need to know about handoff taking place in the lower layers, and hence does not require any modification to the application.

4.3.4 Loss/Delay

Packets in flight may not be delivered to the MH during the handoff period. This may result in packet losses, packet delay, and a false indication of congestion in the network.

4.3.5 Security Solutions

Internet is vulnerable to many security threats. Many of the solutions, such as ingress filtering and firewalls, to the threats do not allow network entities to process packet headers as may be required by some of the mobility schemes.

4.3.6 Path Diversity/IP Diversity

Increasing number of mobile devices nowadays comes with multiple communication interfaces. During handoff, an MH may be able to take advantage of multiple IP addresses (called IP diversity), obtained from separate subnets, associated with the multiple interfaces.

4.3.7 Change in Infrastructure

A mobility management scheme may require additional software agents (such as Home/Foreign agents in the case of MIP) or hardware to be deployed in the existing network infrastructure. Such additional agents/hardware may result in scalability and deployment issues for the scheme to be implemented in the real world.

4.3.8 Change in Protocol

A transport layer mobility management scheme may require change in the transport protocol, or may require applications to use a new transport protocol or API.

4.4 Summary of different Transport Layer Schemes

4.4.1 MSOCKS

In MSOCK TCP Splice to split a TCP connection at a proxy by dividing the host-to-host communication into host-proxy and proxy-host communications. MSOCKS uses TCP Splice for connection migration. During handoff, it obtains a new IP address from the new subnet, and establishes a new connection with the proxy using its second interface. The handoff proceess is hard. The communication between proxy and CN, however, remains unchanged. The data low between MH and CN thus continues, with the CN being unaware of the mobility. Location management is done through the proxy who is always aware of the location of the MH; this limits the mobility within the coverage of the proxy. Only the flying packets are lost here. But a single point of failure, if the proxy fails then the whole system breaks. The distadvantage of this protocol is it needs to change the infrastructure of the existing network and as well as the protocol stack.

4.4.2 SIGMA

SIGMA is a complete mobility management scheme implemented at the transport layer, and can be used with any transport protocol that supports IP diversity. SIGMA supports IP diversity-based soft handoff. As an MH moves into the overlapping region of two neighboring subnets, it obtains a new IP address from the new subnet while still having the old one as its primary address. When the received signal at the MH from the old subnet goes below a certain threshold, the MH changes its primary address to the new one. When it leaves the overlapping area, it releases the old address and continues communicating with the new address thus achieving a smooth handoff across subnets. Location management in SIGMA is done using DNS as almost every Internet connection starts with a name lookup. Whenever an MH changes its address, the DNS entry is updated so that subsequent requests can be served with the new IP address. The handoff it supports is soft. There is less delay/loss of packets than the other protocols. New connection will fail if the location manager fails. Here, it is not needed to change the infrastucture but need to change in the protocol stack.

4.4.3 Migrate TCP

Migrate TCP is a transparent mobility management scheme which is based on connection migration using Migrate TCP, and uses DNS for location management . In Migrate TCP, when an MH initiates a connection with a CN, the end nodes exchange a token to identify the particular connection. A hard handoff takes place when the MH reestablishes a previously established connection using the token, followed by migration of the connection. Similar to SIGMA , this scheme proposes to use DNS for location management. The handoff it supports is soft. It avoids data transfer during handoff so that no packet is loss. Here, it is not needed to change the infrastructure. It needs to change the protocol stack in CN but not in MH.

4.4.4 Freeze-TCP

Freeze-TCP is a connection migration scheme that lets the MH 'freeze'or stop an existing TCP connection during handoff by advertising a zero window size to the CN, and unfreezes the connection after handoff. This scheme reduces packet losses during handoff at the cost of higher delay. Although it provides transparency to applications, Freeze TCP requires changes to the transport layer at the end nodes. Freeze-TCP only deals with connection migration, but does not consider handoff or location management. It can be employed with some other schemes like Migrate to implement a complete mobility management scheme. It supports hard hand-off. New connections would fail if location manager fails. Here, it is not needed to change the infrastucture but need to change in the protocol stack.

4.4.5 RCP

RCP moves the responsibility for performing reliability and congestion control from the sender to the receiver. It allows for better congestion control, loss recovery, and power management mechanisms compared to sender-centric approaches. The handoff is soft. It does not conflict with the security solution. It supports IP diversity. Here, it needs to change in the infrastructure. With further improvement next come R^2CP.

4.4.6 R^{2}CP

R^{2}CP is based on Reception Control Protocol (RCP), a TCP clone in its general behavior but moves the congestion control and reliability issues from sender to receiver on the assumption that the MH is the receiver and should be responsible for the network parameters. R^{2}CP has some added features over RCP like the support of accessing heterogeneous wireless connections and IP diversity that enables a soft handoff and bandwidth aggregation using multiple interfaces. A location management scheme might be integrated with R^{2}CP to deploy a complete scheme. The handoff is soft. It does not conflict with the security solution. It supports IP diversity. Here, it does not need to change in the infrastucture.

4.5 Classification of Transport Layer Schemes

The mobility management schemes described before can be classified, based on their approach towards mobility, into four groups as shown in Table 4.3 and described below.

4.5.1 Handoff Protocol

Rather than being complete mobility management schemes, schemes belonging to this class are enhancements of transport layer protocols that aim at improving the performance, such as low latency and reduced data loss, of mobile hosts during handoff. This class consists of R^{2}CP, MMSP, mSCTP each of which supports IP diversity and seamless handoff. They can aid handoff, but are not complete mobility management schemes because of their lack of mobility management components, such as location management.

4.5.2 Connection Migration Protocol

The mobility schemes in this class are based on migrating connections which have been stopped or put under waiting during handoff in order to ensure a single unbroken connection between CN and MH. They do not deal with handoff issues. Examples are Freeze-TCP, TCP-R which are enhancements of TCP to allow a connection to be stopped and restarted

Table 4.1: Transport Layer Mobility Schemes Classified By Approach

Class	Description	Example
Handoff protocol	Transport Layer Protocol that has features to support mobility	R^2CP,MMSP,mSCTP
Connection migration protocol	Transport Layer Protocol that can migrate multiple connections	Freeze TCP,TCP-R
Gateway-based mobility scheme	Provides mobility by putting a infrastructure between CN and MH and splitting the connection	MSOCKS, I-TCP,M-TCP, M-UDP, BARWAN
Mobility manager	Complete mobility schemes with handoff and location management	Migrate TCP and SIGMA

before and after a handoff, respectively.

4.5.3 Gateway based Mobility Scheme

Schemes in this class handle mobility with a special gateway in the Internet infrastructure. The connection between CN and MH is split at the gateway, with the connection between the gateway and CN being fixed while allowing the MH to roam and change its connection with the gateway. MSOCKS, I-TCP,M-TCP, M-UDP,BARWAN which belong to this class, requires special entities that split the connection between the MH and CN. The do not provide details about implementation of location managers, and hence are not complete mobility management schemes.

4.5.4 Mobility Management:

Schemes in this class provide complete end to end mobility management schemes at the transport layer. Migrate TCP and SIGMA , which belong this group, provide complete end-to-end mobility management schemes by implementing handoff and location management.

4.6 Comparison among the protocols based on different criteria

We, so far discussed about different transport layer mobility protocols and many evaluation criteria on which we will differentiate them. Hand-off, loss/delay, Fault tolerance, change in infrstructure, conflict with security solutions, IP diversity, change in protocol stack etc are the main evaluation criteria that we have used to differentiate between the protocols. MSOCKS, M-TCP supports hard hand-off, where RCP and R^2CP supports soft hand-off. There is minimum loss/delay in SIGMA, RCP and R^2CP but the flying packets are lost in MSOCKS. Freeze TCP avoids data transfer during hand-off to prevent loss. If the proxy fails then the whole connection fails in MSOCKS, new connection will fail if location manager fails in SIGMA, RCP and R^2CP have high fault tolerance. MSOCKS and R^2CP needs to change in the infrastructure but the other protocols need not to do so. SIGMA and RCP supports IP diversity but the others does not. Many other criteria are illustrated in details in table 4.2

Table 4.2: Comparison among the protocols based on different criteria

Criteria	MSOCK	SIGMA	FREEZE TCP	MIGRATE TCP	R^2CP	RCP
Handoff	Hard	Soft	N/A	Hard	Soft	Soft
Loss/Delay	Only the fly packets are lost	No	Avoids data transfer during hand-off to prevent loss	No, but stops transmission If MH is the server	No	No
Fault tolerance	Single point of failure: proxy	New connections would fail if location manager fails	Yes	would fail if location manager fails	Yes	Yes
Change in infrastructure	Yes	No	No	No	No	Yes
Transparency	Yes	Yes	Yes	Yes	Yes	Yes
Conflicts with security solution	Yes	No	No	No	No	No
IP Diversity	No	Yes	No	No	Yes	No
Change in protocol stack	Yes	Yes	No in CN, Yes in MH	Yes	Yes	Yes

CHAPTER 5

CONCLUSION

IP Mobility can be handled at different layers at the protocol stack. Previous solution like TCP sliding window has more drawbacks like, infrastructure, encrypted traffic, unsecure, erroneous data, confliction etc. To get rid of these problems, some mobility protocol at transport layer has been proposed and some are developed. But, it needs to compare the protocols to best choose and identify the lacking to more improve. We discussed six mobility scheme of transport layer.

At a glance, MSOCKS is built around a proxy that is inserted into the communication path between a mobile node and its correspondent hosts. An ideal MSOCKS proxy would add minimal latency to the path of packets traveling to or from mobile nodes. It is a flexible system that mobile nodes can continue connections between different interfaces.

SIGMA provides seamless handover for mobile hosts and it can greatly reduce the handover latency, packet loss, signaling costs and improve the whole systems throughput. Basic idea of SIGMA is to decouple location management from data transfer and achieve seamless handover by exploiting IP diversity to keep the old path alive during the process.

RCP performs reliability and congestion control from the sender to the receiver. The advantages are when the receiver decides to switch to another interface specific congestion control mechanism after handoffs, such decision does not need to involve the sender and at a time can control flow and seamless handoff.

R^2CP is the extension of RCP. When a mobile host handoffs from one interface to another during a live connection, it can benefit from functionalities like, seamless handoffs without relying on infrastructure support, server migration for achieving service continuity and bandwidth aggregation using multiple active interfaces.

Freeze-TCP is a connection migration scheme that lets the MH freeze or stop an existing TCP connection during handoff by advertising a zero window size to the CN, and unfreezes the connection after handoff. This scheme reduces packet losses during handoff at the cost

of higher delay.

Migratory TCP is a reliable connection-oriented transport layer protocol that supports efficient migration of live connections. The migration mechanism ensures that the new server resumes service while preserving the exactly once delivery semantics across migration, without freezing or otherwise disrupting the traffic on the connection. The client application does not need to change.

We also discussed some mobility management fundamentals which needs to change in IP address gives rise to the challenges in maintaining an uninterrupted data flow like, connection migration, packet loss and latency, location management, require infrastructure etc. We also discussed some other complete mobility scheme that supports IP diversity and soft handoff, behavior, transparency to applications, and can be deployed without any change in the network infrastructure is very suitable for handling mobility of hosts in the Internet. Finally we evaluate the schemes compare the schemes based on different criteria. The discussed evaluation criteria are handoff process, scalability and fault tolerance, transparency, loss and delay, path diversity, security, change in infrastructure, change in protocols etc.

REFERENCES

[1] Maltz, David, and P. Bhagwat, "Msocks: An architecture for transport layer mobility," *Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, vol. 3, 1998.

[2] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, *"SIGMA: A transport layer mobility management scheme for terrestrial and space networks"*. Kluwer Academic, 2005.

[3] Kevin and S. Singh, "A receiver-centric transport protocol for mobile hosts with heterogeneous wireless interfaces," *Wireless Networks 11*, no. 4, pp. 363–382, 2005.

[4] Goff, Tom, J. Moronski, D. S. Phatak, and V. Gupta, "Freeze-tcp: A true end-to-end tcp enhancement mechanism for mobile environments," *Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 1537–1545, 2000.

[5] Koh, S. Joo, M. J. Chang, and M. Lee, "msctp for soft handover in transport layer," *Communications Letters, IEEE 8.3*, 2004.

[6] C. E. Perkins, "IP mobility support," IETF RFC 3344, Aug 2002.

[7] Fu, Shaojian, Atiquzzaman, and Y. J. Lee, "Architecture and performance of sigma: A seamless mobility architecture for data networks," *Journal of High Speed Networks*, vol. 5, 2005.

[8] C. Perkins, D. Johnson, and J. Arkko, "Mobility support in IPv6," IETF RFC 6275, Jul 2011.

[9] Goff, J. Moronski, D. S. Phatak, and V. Gupta, "I-tcp: Indirect tcp for mobile hosts," *Distributed Computing Systems, Proceedings of the 15th International Conference*, 1995.

[10] Brown, Kevin, and S. Singh, "M-tcp: Tcp for mobile cellular networks," *ACM SIGCOMM Computer Communication Review 27*, no. 5, pp. 19–43, 1997.

[11] Atiquzzaman, Mohammed, and A. S. Reaz, "Survey and classification of transport layer mobility management schemes," *Personal, Indoor and Mobile Radio Communications*, vol. 4, 2005.

[12] Iyer, Y. Gopalan, S. Gandham, and S. Venkatesan, "Stcp: a generic transport layer protocol for wireless sensor networks," *14th International Conference on IEEE*, 2005.

[13] Phan, Thomas, K. Xu, R. Guy, and R. Bagrodia, "Handoff of application sessions across time and space," *IEEE International Conference*, vol. 5, 2001.

[14] Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil, "Proxy mobile ipv6," RFC 5213, Aug 2008.

[15] Eddy and W. M, "At what layer does mobility belong?" *Communications Magazine, IEEE, 42*, 2004.

[16] Ioannidis, D. Duchamp, and G. Q. M. Jr, "IP-based protocols for mobile internetworking," *ACM SIGCOMM Computer Communication Review*, vol. 21, no. 4, pp. 235–245, 1991.

[17] H. nd Peter J. Chidgey and F. Kaufhold, "A transport network layer based on optical network elements," *Lightwave Technology, Journal of 11*, no. 5, pp. 667–679, 1993.

[18] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, *Computer networking*. Pearson Education, 2012.

[19] S. Fu and M. Atiquzzaman, "Hierarchical location management for transport layer mobility," in *IEEE GLOBECOM*, San Francisco, CA, Nov 27-Dec 1 2006.