Socketless TCP

A Connection Identification Philosophy for End to End Mobility

Thesis submitted in accordance with the requirements of the University of Cape Town for the degree of Master of Science in Electrical Engineering

Prepared by: Eric Beda Mutagahywa

Supervised by: Neco Ventura

February 2006

DECLARATION

I declare that this dissertation is my own work. It is being submitted for the degree of Master of Science in Engineering at the University of Cape Town. It has not been submitted before for any degree or examination in any other university.

Signature of author .....

Eric Beda Mutagahywa

February 2006

ACKNOWLEDGEMENTS

I can not hope to enumerate, let alone repay, all those to whom I am indebted for assistance not only in preparing this manuscript, but in helping me survive and prosper during my years at UCT. Despite the unavoidable omissions, I still insist on mentioning a few in print.

First I thank my supervisor, Neco Ventura, for the assistance he provided throughout the execution of this project. I am especially grateful for the funding he acquired for me that made this research possible. I extend my thanks to David Waiting for all constructive suggestions and reviews of my thesis. I thank all members of the STAR and CRG lab for providing an exciting and energizing work environment, with special thanks to Vitalis Gavole and Nicholas Zulu. I also extend thanks to Ntanzi Carrilho, my brothers Raymond, Paul, Edwin and all who showed me support when I needed them most. I am profoundly grateful to my parents for their relentless support and encouragement throughout the years. But, above all, I am eternally grateful to the One who watches over my soul. To Him belongs all the glory, honor and praise.

ABSTRACT

Wireless networking is becoming an increasingly important and popular way of providing global information access to users on the move. A computer is no longer an immobile, gargantuan machine that remains static for the lifetime of its operations. Today's personal computing devices are portable, and Internet access is becoming ubiquitous. Hence there is increasing pressure on Telecoms and Internet Service Providers to supply their customers with access to their customized services anywhere, on any terminal via any access technology. The pressure is trickled down to software programmers to provide innovative and advanced applications to fit this new environment. This will require network protocol architects to provide an Internet framework that will give programmers more control and flexibility to create mobile aware applications.

To fulfill such requirements, network protocol architects need to shift their mobility perspective from the mobile terminal to a finer grained model; finely grained in respect to allowing individual transport connections to seamless switch between network terminals, controlled dynamically and/or manually by applications or users. The key issue of this vision is *how* to support this model in TCP/IP networks. We argue that the TCP/IP socket pair connection identification model restricts this granularity and flexibility of mobility. We present *Socketless TCP (SL-TCP)*, an architectural concept based on reconsidering the socket pair for connection identification. SL-TCP introduces a new self-sufficient, connection identifier within the transport layer packet header in-place of the port number pair. This identifier is used instead of the socket pair for multiplexing and forwarding packets to the network and applications layer respectively; eliminating the dependency of IP addresses and port numbers for connection management. This will allow mobile hosts and/or transport connections to switch

terminals (IP addresses) seamlessly without terminating ongoing transport connections.

We describe our SL-TCP concept and demonstrate handover mechanism with a prototype implementation. We show that this connection identification scheme is flexible, robust and efficient way to manage seamless connectivity for mobile end points. In addition, we demonstrate two analytical models that show handover cost and power consumption efficiency gains of SL-TCP over popular mobility schemes. Experimental results show that handover latency induced by SL-TCP is approximately equal to the return time trip of a packet.

CONTENTS

LIST OF FIGURE

LIST OF TABLES

LIST OF ACRONYMS

3G	3rd Generation Networks
4G	4th Generation Networks
BUD	Binding Updates
CD	Communicating Device
CID	Connection Identifier
CN	Corresponding Node
CRG	Communication Research Group
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Service
FA	Foreign Agent
GID	Global Identifier
HA	Home Agent
HIP	Host Identity Protocol
HOD	Handover Delay
IEEE	Institute of Electric and Electronic Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
LAN	Local Area Network
LID	Local Identifier
MAC	Media Access Control
MIP	Mobile Internet Protocol
MN	Mobile Node
MSM-IP	Mobility Support Using Multicast in IP
NAT	Network Address Translation
NIC	Network Interface Card
OS	Operating System
PCB	Protocol Control Block
POA	Point of Attachment

QoS	Quality of Service
RSIP	Realm Specific Internet Protocol
RTT	Return Time Trip
SCTP	Stream Control Transmission Protocol
SIP	Session Initiated Protocol
SL-TCP	Socketless TCP
STAR	Speech Technologies Research Group
TCB	TCP Control Block
TCP	Transmission Control Protocol
UCT	University of Cape Town
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
URL	Unified Resource Locator
VNAT	Virtual Network Address Translation
WLAN	Wireless Local Area Network
XDP	Extended Datagram Protocol

Chapter 1

1 INTRODUCTION

Wireless networking is becoming an increasingly important and popular way of providing global information access to users on the move. A computer is no longer an immobile, gargantuan machine that remains static for the lifetime of its operations. Today's personal computing devices are portable, and Internet access is becoming ubiquitous. A well traveled laptop user might visit half a dozen of different networks throughout the course of a day: a cable modem from home, wide-area wireless on the commute, wired Ethernet at the office, a Bluetooth network in the car, and a wireless, local-area network at the airport or the neighborhood coffee shop.

Armed with a portable computing device and readily-available Internet access, today's user expects seamless *'always on'* operation for network applications. In addition she would want to take advantage of the best, inexpensive bandwidth available at anytime without the burden of network

reconnections. Hence there is increasing pressure on Telecoms and Internet Service providers to supply their customers with access to their customized services anywhere, on any terminal via any access technology. To fulfill such high expectations for unified and personalized service provisioning, a global solution for mobility management is of primary goal.

This dissertation recognizes the role of seamless handover (both horizontal and vertical) in the Convergence, Integration and Inter-working of existing and emerging fixed and mobile networks, and introduces a concept (philosophy) to assist in dealing with these challenges. We propose a solution based on *address* abstraction that eliminates *transport layer connection* dependency on *network layer* attributes for connection identification. The transport layer role is to maintain reliable, durable and long-term relationships between application end points that may span multiple network connections and application transactions; today's network connections (network layer), on the other hand, are ephemeral relationships between network attachment points. Everyday examples of applications that require long term relationships with corresponding nodes include interactive logins by users of remote hosts, multimedia conferences between remote peers, sets of web transactions between browsers and servers, etc. These applications and many more require a continuous connection session between end points regardless of the relationship between network attachment points. In particular, address abstraction affords the transport layer connection continuity irrespective of the Operating System (OS) network interface preference; hence it gives applications the opportunity to influence network interface selection. Unfortunately the current implementation of TCP/IP cannot entertain a change in network interface and/or terminal without terminating current transport connection sessions.

We propose to redefine how the transport layer identifies a connection, and eliminate the dual functionality of an IP address as an end point identifier and locator. We replace the traditional 32 bit port numbers with a new field in the TCP header, a connection identifier (CID). This field instead of the traditional TCP 4-tuple is then used for identifying different connections. This philosophy supports the end-to-end argument [2] and is inspired by Christian Huitema's [19] 1995 Internet draft. We present *Socketless TCP*, an end-to-end mobility concept that can support *Simultaneous multi-access, Terminal* mobility and *fine grained* mobility, a mechanism that allows mobile terminals and/or transport connections to change their network point of attachment or simultaneously use multiple attachment points without terminating current transport connection sessions.

1.1. IP Mobility Challenges

Regarding IP mobility support two issues need to be distinguished separately. The first issue deals with the question 1) how to locate a mobile host when establishing a *transport* connection. The second issue raises the question 2) how we can maintain continuity of a connection when it moves into another subnet and changes its point of attachment. Both issues are closely related to each other, as a solution to one has to make provision for the other for it to be considered feasible. Therefore, there exist two main directions for research, one is to support easy location management and hence *continuous reachability* while the other direction of research is to investigate solutions to handle seamless handover, be it horizontal or vertical, and hence *seamless connectivity*. Compatible solutions for the technical problems in 1 have already been addressed in the literature [6, 14, 15, 18] and are discussed in this dissertation for completeness only. However, the focus of this thesis addresses question 2 only. Seamless connectivity has 3 (three) aspects to it: *continuous connectivity, handover latency* and *fine grained mobility.*

1.1.1. Continuous Connectivity

Continuous connectivity refers to the ability of a mobile terminal to sustain a transport connection during a topological change in a network interface (IP address). Seamless connectivity is achieved when all ongoing communication sessions avoid termination during this change. For a mobile host to have seamless connectivity it must retain some form of identification while changing its location. Currently Internet protocols do not retain any identification, as the identification is physically bounded to the attachment point. Hence, the challenge is to remove this dependency between transport layer identification and network attachment points.

1.1.2. Reducing Handover Latency

Handoff is defined as the time elapsed between the last received packet (by the mobile node (MN) or the corresponding node (CN)) when the MN was attached to the old network, i.e. prior to the move, and the first received packet (by the MN or CN) when the MN is attached to the new network, i.e. after the move. There are two aspects in handover that contribute to the delay. These are a) mobility detection and b) registration /rebinding delay. Mobility detection is specific to the underlying link protocol, and not much can be done to reduce the resulting latencies. There have been a handful of successful proposals attacking these aspects of handover like [30] and [31], however we feel that the reconnection process contributes bulk of these latencies and hence focus on it. Latencies induced by reconnection are directly proportional to the number of signals used to automate the process. The higher number of signals required to be sent during handover the higher the handover latency, thus most mobility schemes focus on reducing the signaling overhead to reduce the overall handover latency.

1.1.3. Fine Grained Mobility

Fine-grained mobility refers to the ability of transport connections to migrate its current state from one device to another. From a mobility

perspective, the relevant state information are attributes used to maintain current transport connection session on the network i.e. IP address and Port numbers. These attributes have strict dependencies on the network point of attachment (IP address) and locality of kernel space structure (Port numbers). Hence, the need for new paradigm of end point addressing, that will require each transport connection to be identified uniquely independent of network attachment points and kernel resource data structures. From a research point of view, protocol architects need to focus on providing application programmers with the necessary APIs to take advantage of influencing mobility, and give programmers more freedom to control data flows to the application end points. We believe that providing protocol support for a fine-grained mobility framework is the next step to new exciting innovations in the 4G era.

1.2. Problem Statement

The Internet's popularity is leading it to its ultimate failure; the architects never envisaged that this in house lab technology will have this impact to society as we see today. Originally when host names were being deployed, it was implicitly assumed that the name to address binding will remain static. Hence instead of referring to hosts by name, protocols were developed that referred to hosts through their addresses. A standard example is a TCP connection identified by a 4-tuple (refer to Figure 1-1)

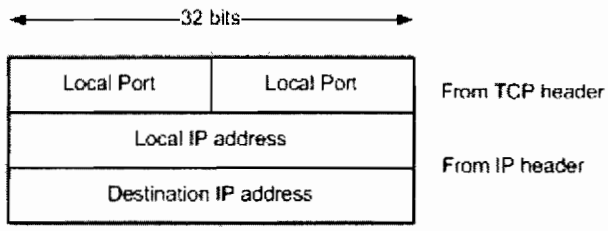


Figure 1-1 TCP 4-tuple

If the host remains static, all components of the connection identifier will remain fixed; thus, a continuous TCP session can be maintained between the two hosts. If either ends of the connection moves, we run into the following problems:

1. If the mobile host acquires a new IP address, then its associated TCP connection identifier also changes. This causes all TCP connections involving the mobile host to be broken.
2. If the mobile host retains its address, then the routing system cannot forward packets to its new location.

This fundamental IP mobility problem is also considered by Bhagwat et al. [1] noting that the IP address actually has two meanings, on the network level, it identifies a location on the network topology but transport layer protocols use it to identify a host.

1.2.1. A Motivating Example

The motivation for hybrid networks arises from the fact that no one technology or service can provide ubiquitous connectivity coverage at the best data rates available; and it will be necessary for a mobile terminal to employ various points of attachment to maintain connectivity to the network at all times. One need look no further than real time applications like Streaming Video or Internet Radio Broadcast applications, for a practical example of lack of support for mobility in the Internet. Imagine a user with both IEEE 802.11b WLAN and 3G UMTS interface cards on her PDA (refer to Figure 1-2), in a car en-route to the office, currently participating in a Video Conference session initialized by the UMTS network. As she approaches the office she notices that she is within WLAN coverage (faster and cheaper) two ideas come into mind.

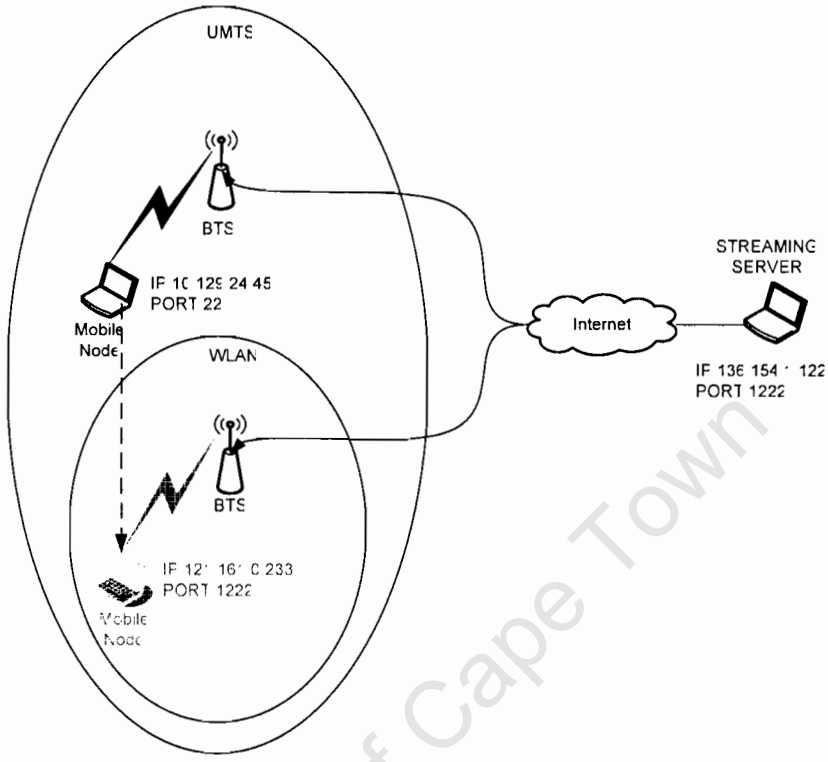


Figure 1-2 Movement from UMTS to WLAN network

1. She wants to connect through her WLAN interface so as to achieve cheaper and better quality videoconferencing experience.
2. She also notices another computer terminal (Larger monitor, better microphone and video capture device) and she prefers to switch to this terminal.

She attempts to reconnect using the WLAN interface or the other computer terminal, expecting her videoconference session to continue seamlessly. Sadly, the user will find her session aborted, and she is unable to resume her session. This occurs because the remote end point (the Video Streaming server) was unable to determine the new attachment point and unsure that the session would continue.

1.2.2. Mobility Argument

For a mobile host to have seamless connectivity it must retain some form of identification while changing its location. Previous mobility proposals decouple this binding by introducing a fixed indirection point (e.g. Mobile IP [3]), virtualizing transport layer connections (e.g. VNAT [5]) or using indirection at the link layer (e.g. cellular mobility schemes). However, these proposals lack one or more of the following properties to fully realize the promise of ubiquitous and continuous connectivity: *Efficient routing, efficient handoffs, Session/application mobility, Simultaneous multi-access and Link layer independence.* Generally speaking, mobility management can be better-optimized the lower the protocol stack on which it is implemented [51]. However, the lower the layer, the more specialization is required, with ensuing increases in complexity and limitations in scope. Therefore, there is a trade-off between optimization, complexity, and functionality that has to be considered when deploying terminal mobility.

1.3. Proposed Solution

In this thesis, we propose (to the best of our knowledge) the first solution to make provision for *terminal mobility, application mobility and simultaneous multi-access* properties. Our solution called Socketless TCP is based on ideas originally presented by Christian Huitema [19] in his 1995 Internet draft. The general idea behind Socketless mobility support can be intuitively described by managing the mapping of connections and hence the process of demultiplexing packets from the network layer, without using the conventional *socket pair* that defines a TCP connection, hence the name *Socketless.* This is achieved by redefining the taxonomies used to identify end-to-end connections.

We believe that by introducing a concept of identifying all transport connections sessions independent of network layer attributes (IP addresses) and kernel entities (port numbers) will permit end points to move freely

between networks of different access technologies and/or between different devices (kernel spaces) seamlessly; Hence catering for handover (both vertical and horizontal) and application migration. This approach will require significant changes to end systems. However, it is noted that software updates are necessary in any case, and this is an ongoing process. If IPv6 is to be implemented or if RSIP (Realm Specific IP) [50] [52] is to be used, end hosts would have to be updated. Similarly, if QoS support will be offered in access networks, end hosts will have to be upgraded with QoS capability [53].

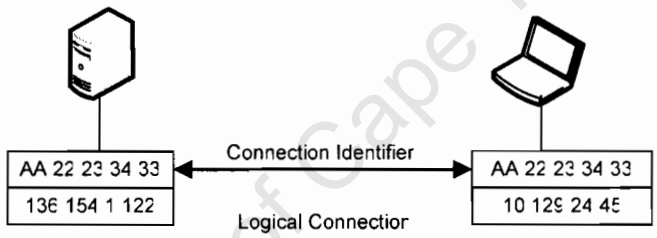


Figure 1-3 Network Layer abstracted logical connection

Figure 1-3 illustrates a logical connection between two end points using a parameter in layer 4 of the protocol stack to abstract the IP address of the attachment point. Note that the IP address is not associated with the connection. SL-TCP achieves this by introducing 3 (three) concepts core to its implementation.

1.3.1. Redefining network end points

SL-TCP philosophy redefines the how traditional end points are defined. Currently end points are referred to as the host terminal as an entity, thus even the connection end points are defined using the attributes of the host terminals identity (IP address) thus binding the IP address to the connection, thus causing a situation known as the mobility problem [8]. In

SL-TCP we move away from these semantics and define an end point as the smallest unit of communication, which is the network process, currently identified by the socket (IP address and Port Number). We move even further and choose to identify this network process by a 64bit connection identifier; this is discussed in detail in Section 4.4.

1.3.2. Connection Identification

The current Internet architecture is based on the TCP/IP model, with TCP being the most used transport layer protocol. TCP uses local and foreign IP addresses as part of its connection identification structure know as the TCP 4-tuple [7]. Originally IP addresses were good identifiers for hosts, because hosts were static and they could be identified on account of their topological location. However; the introduction of mobility and multi-homing has changed this situation. This implies that the binding between the four-tuple TCP connection identifier and socket structure defines an IP address dependency for TCP connections. Once a single address is changed within the four-tuple, the connection gets broken and the communication is temporarily terminated. If a TCP connection can be independent of its four-tuple identifier, then mobile terminals will be able to change their IP addresses (and hence network interfaces) at will without disrupting the TCP connection session. We believe that network layer independence from Transport Layer protocols can be achieved through the inclusion of a unique connection identifier (CID) within packet headers; this identifier will be sufficient to multiplex packets at the end hosts.

1.3.3. Connection Management

SL-TCP revolutionizes the manner in which connections are multiplexed and demultiplexed to the application layer. In SL-TCP packets are forwarded to applications based on a connection identifier instead of the 4-tuple hence removing physical dependencies from the connection. This aspect is core to the concept, since network layer attributes are abstracted

from the connection identifier, ongoing connections are not bound to the IP address, and hence a change in IP address does not influence the routing of packets to destination applications. Hence, applications can seamlessly and correctly receive packets using any network interface and from any IP address the corresponding host is using, as long as the packets are identified with the correct connection identifier.

1.4. Research Objectives

Our research focuses on providing continuous connectivity support and improving performance of an individual user switching between multiple IP subnets (handover) synonymous to a wireless overlay environment. In this work, the primary objective is to minimize the handoff latency for an individual user while keeping bandwidth and power overheads as low as possible. The technical objectives in the design of a seamless handoff system are:

- Achieve Latency Handoff: make the switch between networks as seamless as possible for disruption-intolerant applications and with as little data loss as possible.
- Minimize power overheads: minimize the power drain due to multiple simultaneously active network interfaces.
- Minimize bandwidth Overhead: minimize the amount of additional network traffic used to implement handoffs due to signaling overhead
- Support simultaneous Multi-access: to support applications to receive packets originating from different paths, with different IP addresses but destined to a single connections
- Support continuous connectivity: to avoid connection terminations during handover due to a change in IP address.

1.5. Organization

The remainder of this dissertation presents Socketless TCP model for continuous connectivity discussing the philosophy and architecture behind it. Chapter 2 motivates our work by describing related work in the fields of mobility categorizing current solutions by their implementation on the protocol stack. We describe the core philosophy behind SL-TCP in chapter 3, including characteristics of and guidelines in creating a new connection identifier for next generation networks. Traditional mobility semantics are redefined here. Chapter 4 introduces SL-TCP its design goals, describes its design and presents a migration model. We present the SL-TCP architecture in chapter 5, discussing relevant data structures, protocol processing, and data paths, we also illustrate how the protocol handles application mobility, terminal mobility and simultaneous multi-access. Furthermore, various security enhancements for SL-TCP are described and we illustrate how it guards against certain attacks. The prototype used to evaluate our concept is discussed in Chapter 6. We then evaluate SL-TCP in chapter 7 both analytically and by use of an evaluation platform. Finally, chapter 8 presents a summary of the dissertation and concludes with a discussion of both specific contributions and general principles that can be extracted from the work.

Chapter 2

2 BACKGROUND AND RELATED WORK

Mobility has been a fertile area of research for many years. In this chapter, we provide an introduction to basic networking concepts and discuss them in respect to issues pertaining Internet (or inter-network) mobility. By surveying previous approaches to the problem of mobility, we motivate a shift in philosophy to using a *self-sufficient* identifier rather than IP addresses and port numbers for connection identification, and how this shift helps to address the three themes of this thesis: handling transport connection sessions during changes in network attachment points, supporting a fine grained mobility and making provision for simultaneous multi-access.

We begin in Section 2.1 with a brief introduction on Internet concepts. Section 2.2 presents previous proposals implemented on the network layer in particular Mobile IP. We then explore previous proposals implemented on the transport layer in Section 2.3. In Section 2.4 solutions implemented

on layers other than network or transport are discussed. Finally, the chapter concludes in Section 2.5 with an analysis of how the connection identification scheme used in these proposals, restricts their ability to develop handover solutions that (a) have low signaling overhead and (b) support fine grained mobility. Desirable design criteria for IP based handover solutions are then identified, concluding with general remarks motivating the need for a new end to end approach to mobility implemented on the transport layer.

2.1. Internet Basics

In this Section we first describe basic Internet terminologies in respect to IP mobility and then we briefly describe 3 fundamental Internet concepts that influence IP mobility schemes namely: *names and addressing, end-to-end principle and protocol stack layering.*

2.1.1. Terminologies

The Internet is an inter-connection of *networks*. A network connects *hosts* and networks are interconnected via *routers*. Communication in the Internet is based on the *Internet Protocol* (IP), a network layer protocol defined by [10], referred to as IP-Version 4 (IPv4), and alternatively by [32], referred to as IP-Version 6 (IPv6). Users are equipped with communication devices (CD). The CD forms one end of a network communication and is known as an end point. Data is exchanged between end points by means of the Internet. In order for an end point to gain access to a network, it needs to acquire and IP address, which will uniquely identify it within the network, this IP address is assigned to its *Network Identification Card* (NIC), the NIC is uniquely identified by *Media Access Control Address* (MAC).

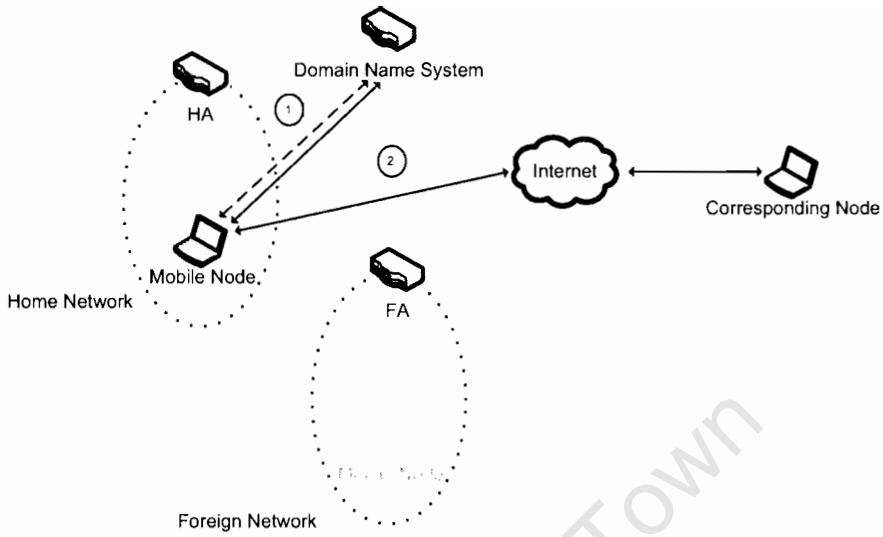


Figure 2-1 Common Internet components in mobility environment

An end point initializes a connection to a server by resolving its *unified resource locator* (URL) by use of *Domain Name Service* (DNS). It gets back the IP address of the server and initializes the connection. If the end point is able to move and change its *point of attachment* (POA) without the need of human interference we refer to it as a *Mobile Node* (MN). The primary POA of a MN is in its *Home Network* (HN), the visited network is termed as its *Foreign Network* (FN), and the node with which it will be communicating with is referred to as a *corresponding node* (CN).

2.1.2. Naming and Addressing

The name, address and route represent the fundamental components of the Internet architecture. Schoch [33] makes the following definitions

- The name of a resource indicates what we seek,
- An address indicates where it is, and
- A route tells us how to get there.

The notion here is that a name is something with which a human can deal comfortably, such as a character string (URL). The name is mapped into an IP address, which is something less comfortable to humans but more useful to machines, such as routers. The nature of IP addresses enables it to map into routes that defines a path from one host to another [47].

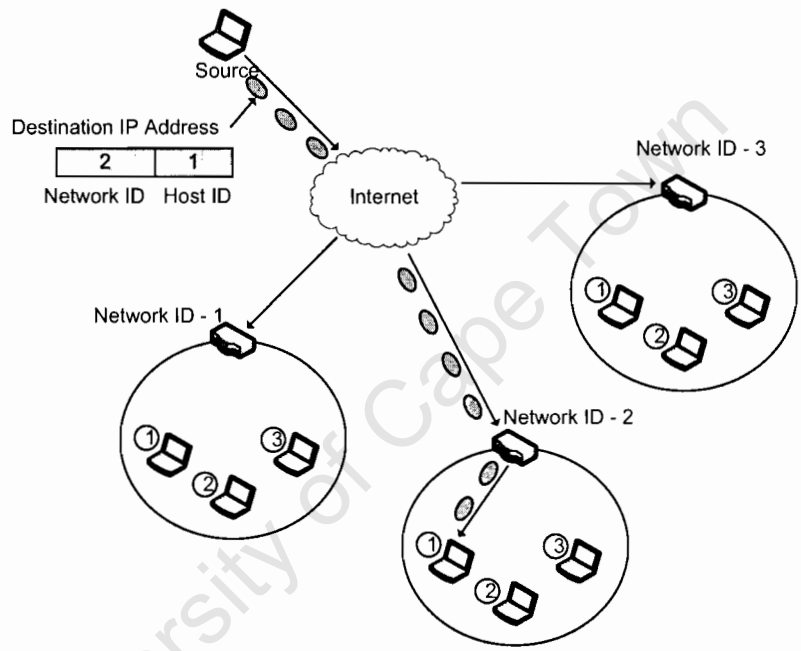


Figure 2-2 Internet address architecture:
Hierarchical routing

Figure 2-2 shows how an IP address is mapped into a path to route datagram's to the destination host. There are two components to an IP address 1) the Network component 2) the Host Component. The Network component represents the topological location of the host, known as subnets and the Host component uniquely identifies the host within this subnet. This model organizes the Internet address space into numerous subnets, each subnet successively narrowing an address until it points to a single end point. This organization is known as the hierarchical structure of the Internet. In this structure it is impossible for a host to change subnets and remain addressable

with the original IP address, thus forcing transport connections to end. Hence, the goal of mobility schemes is to maintain transport connections despite this mandatory change in IP address.

2.1.3. End to end principle

One of the primary problems with IPv4 and consequently one of the principle drivers of IPv6 adoption was IPv4's lack of end-to-end connectivity; demonstrated by the influx of NAT [28] and firewalls as a consequence of IPv4 address space shortage. The end-to-end argument is discussed in depth by Saltzer [34]; briefly stated, end-to-end connectivity holds that any network elements between a source and a destination be transparent to the endpoints of a packet flow [26]. Future applications, particularly those based on peer to peer communications, can only be correctly executed at the endpoints and any attempt to modify packets by intermediate network elements limits the functionality and innovativeness of the application from the user and programmers perspective respectively.

2.1.4. Layering

The fundamental purpose of organizing networking as a protocol stack is to separate functionalities and related data and therefore make the architecture more modular and accessible. In the Internet architecture a number of protocols are run on the end hosts and routers, utilizing a four-layered architecture (refer to Figure 2-3) TCP and IP are fundamental elements in the architecture and hence it is called the TCP/IP protocol stack. The stack can be described as an hour glass structure, representing a variety of choices of network protocols on each end of the stack, all using IP as the routing and locating protocol.

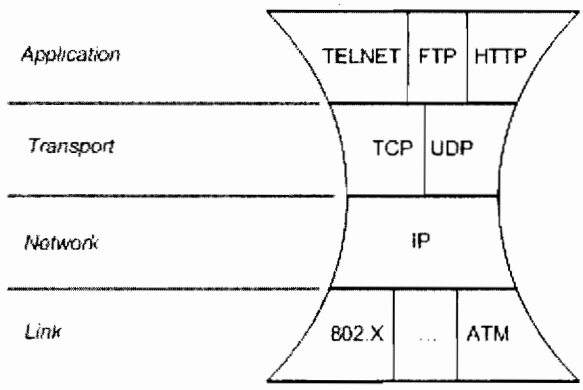


Figure 2-3 TCP/IP Protocol Stack

The ultimate goal of mobile solutions is to provide network applications with mobile communication services in the layers under the Application layer. Existing solutions can be distinguished based on the primary layer on which they provide mobility. Generally speaking, mobility management can be better-optimized lower in the protocol stack [51]. However, the lower the layer, the more specialization is required, with the ensuing increases in complexity and limitations in scope. Therefore, there is a trade-off between optimization, complexity, and functionality that has to be considered when deploying Internet mobility. Hence, the need for mobility proposals to address the question: '*at what layer does mobility belong?*'

2.2. Network layer mobility

Network layer mobility approach imposes the requirement on mobility schemes to be completely transparent to upper layer protocols and applications running on stationary hosts [1]. A stationary host should view a mobile host like any other stationary host connected to the Internet, as hosts are identified by their IP address. This imposes the requirement that even when mobile host change their point of attachment (IP address), the corresponding hosts should still be able to identify them by their original IP address. In addition, any changes in a mobile node's network attachment

point should be completely hidden from the protocols and applications running on stationary hosts.

Traditionally these schemes view mobility as an address translation problem, thus best resolved at the network layer. Since IP is at the waist of the protocol stack hourglass model, it is the one place where mobility support can benefit *every* higher layer [27]. The approach is centered on mechanisms to conceal the true identity (IP address) of the mobile host. This is accomplished by providing some sort of intelligent routing mechanism on the network layer, normally involving special routers. The architects of network layer mobility schemes argue that it is unfeasible to mastermind a complete redesign to networking protocols with goals of supporting mobility due to the current popularity of the Internet. Many proposals address the mobility problem with this paradigm. This Section describes proposals based on 4 different techniques, Mobile IPv4 [3], Mobile IPv6 [4], Internet Indirect Infrastructure (i^3) [29], and a Multicast approach [21].

2.2.1. Mobile IPv4

Mobile IPv4 [3] is the current Internet Engineering Task Force (IETF) standard for supporting host mobility on the Internet. The standard introduces two new notions to the Internet architecture the *home network*-the network which the mobile node 'belongs' and *foreign network*-the network which the mobile node 'visits'. The standard requires that the mobile node has a permanent address within its home network, and a special case router - home agent located within the home network always knows how to reach (route packets to) mobile node at its most recent location. The home agent learns this by receiving a new care-of-address sent by the mobile node whenever it enters a new network. This CoA is the IP address assigned to the mobile node by a Foreign Agent – the administrative router in the foreign network. In this scheme all datagram's

destined to the mobile node are routed first to its home network, resulting to what is known as *triangular routing*. While mobile IP (MIP) is a widely accepted concept in both research and industry, MIP suffers from a number of problems, the most important ones as identified in literature [11,12,13]. These include: *(a) High handoff latency* as a result of signaling and time consuming steps until a mobile node is ready to resume communication after it migrates to a new network, *(b) Complications due to Ingress Filtering, (c) High Packet loss rate (d) Inefficient routing (e) Conflict with network security problems.* Furthermore, in context to fine grained mobility (application migration), in a MIP scheme each end point (application) will need to have its own end point identifier or home address to be managed by the routing infrastructure; this will severely stress MIP scalability.

2.2.2. Mobile IPv6

Mobile IPv6 [4] addresses most of the disadvantages that were visible in its previous version. The main goal in its design considered the fact that future IP mobility frameworks need to consider the QoS constraints of active connections more closely when handling the usual requests of handover and that in rerouting several optimizations can be done to improve the overall mobility performance. The designers acknowledged that it is beneficial to forward packets directly between the corresponding and mobile hosts, as it enables the resultant path to be optimized for quality of service. However, it is unclear if and when all corresponding hosts would become mobile aware to provide this service. The major enhancements can be summarized as: *Route Optimization, Frequent handover and fast location updates, Link layer assisted handover detection and tunneling across QoS domains*. Although MIPv6 promises route optimization that overcomes the problem of triangular routing as its ancestor experienced, the core of this protocol is based around depending on nodes that do not

partake in the actual communication (Home Agent). The added features, most notably *Binding Updates, Return Routability* and *Foreign Agent Discovery* contribute highly to handoff latencies due to the increase signaling between functional nodes and processing overheads at participating nodes. Hence, making deployment cumbersome and scalability dependent on the willing participation of foreign entities; furthermore, it requires modifications to the IP layer of all end hosts regardless of whether they are mobile or not.

2.2.3. Internet Indirection Infrastructure

Internet Indirection Infrastructure i3 [29], is implemented as an overlay network on top of IP, and provides a rendezvous-based communication abstraction [36]. With i^3, instead of explicitly sending a packet to a destination, each packet is associated with an identifier. This identifier defines an *indirection* point, and is used by the receiver (by means of a key termed *trigger*) to obtain the packet. In i^3, unlike MIP, the identifier can be bound to a host, session or person. The identifier is still used for routing and for identifying end points; but i^3 does not provide a clear enough separation between the two.

2.2.4. Multicast Mobility

Mysore and Bharghavan proposed mobility support using multicast in IP (MSM-IP) [21]. IP-multicast provides efficient algorithms for multiple packet delivery. It also provides location-independent group addressing. The receiver-initiated approach for IP-multicast enables new receivers to join to a nearby branch of an already established multicast tree. Hence, IP-multicast provides a scalable infrastructure for efficient, location-independent, packet delivery. However, it is noted that this approach whereby receivers join nearby branches of an already established tree is synonymous to MIP binding updates. Similarly, the home agent functionality is replaced by whatever entity is in charge of multicast tree

rendezvous [45]. Also, MSM-IP location service is a single point of failure and is vulnerable to overload, network faults, and host faults. Furthermore, as with MIP, in MSM-IP there is no separation between location names and end point identifiers, hence its unfeasibility in supporting application mobility.

2.3. Transport layer mobility

Transport layer mobility schemes view the mobility problem as a *connection management* problem rather than a routing problem. Solutions on this layer attempt to manage connections during an IP address change by equipping the end systems with intelligence to counter the network layer changes (IP addresses) in the TCP 4-tuple. The basic idea is to use indirection, migration, tunneling or multi-homing techniques to remove network layer dependencies. A number of researchers have proposed mechanisms to allow connections to adapt to changes in attachment points. The act of moving a connection from one attachment point to another is referred to as *connection migration*. Below we describe four current transport layer mobility proposals in literature, these are Virtual Network Address Translation (VNAT), Multi-homed TCP, TCP migrate and SCTP.

2.3.1. Virtual Network Address Translation

Virtual Network Address Translation (VNAT) [44] is based on the idea of introducing a virtual address to identify a connection end point. These virtual addresses are used to break the tie between the transport protocol and network protocol. As the transport end point identification is made independent of the network end point identification, the lifetime of a transport connection is no longer limited by changes in network end points. VNAT does not require transport protocol changes making it easier to deploy. However, for every TCP connection VNAT creates one new connection that abstracts it from topological changes in the network during handover. We argue that maintaining TCP connection state information is

an expensive process requiring numerous memory accesses. Virtualizing each connection will double the processing load at the user terminal. We feel that this approach does not favor mobile terminals with stringent processing and power conditions like wearable devices and handhelds that will be prevalent in the 4G era.

2.3.2. Multi-homed TCP

Christian Huitema [19] proposed to remove the fate sharing effect [48] by allowing the set of addresses used by a TCP connection to change over time. He proposed to modify TCP by defining a new type of parameter, PCB-ID, to be used during the initial synchronization. This parameter identifies the "Protocol Control Block (PCB)" associated to the TCP connection. When initiating a connection, the host attaches to the SYN packet the identifier of the local PCB. Hosts identify their local PCB and exchange "Extended TCP" packets, where a 32-bit PCB-ID replaces the 16-bit port number pair at the destination. This way, PCB location becomes independent of the IP addresses. The addresses in use are to be stored in this PCB, and can change in due course of the connection. This approach supports the use of several addresses in parallel for the same connection, which is why the proposal is called "multi-homed TCP (MH-TCP)". Using a 32-bit PCB-ID as a connection identifier can only guarantee uniqueness within the mobile host kernel. The PCB-ID will fail a uniqueness test during an attempt to migrate transport connections from one terminal to another (fine grained mobility). It is in this aspect where we feel the need to introduce a new identifier with no physical and/or resource locality dependencies with the device terminal and kernel respectively.

2.3.3. TCP Migrate

Snoreen Blaksihran proposed TCP Migrate [46], as an extension to TCP to provide mobility. In TCP Migrate, both the MN and CN use a modified form of TCP, which can tolerate a change in IP address during a

connection. This is achieved by the introduction of a new TCP option and inserts an additional field into the TCP Control Block (TCB). The CN uses DNS to learn the current address of the MN, which updates the DNS every time it moves. It requires no modification to the TCP header, packet format, or semantics. Instead, it uses an additional TCP option and then inserts an additional field into the TCB. TCP Migrate does not use an indirection point, thus it can achieve an optimal latency stretch of 1 return time trip (RTT) and is as fault tolerant as IP routing. However, it does not make provision for the support of fine grained mobility as the 16 bit port number of the TCP tuple is assumed to remain constant, for the duration of the transport session. This is not the case if an application is to migrate from between kernels, since it is impossible to guarantee that the port number had not been assigned to another network process, prior to migrating. Furthermore, in TCP Migrate architecture, there are no provisions for a single connection to accept packets from multiple IP addresses (as packets are still routed using the TCP 4-tuple) to cater for aggregate bandwidth applications.

2.3.4. SCTP and M-SCTP

Stream Control Transport Protocol [37] is the 3rd Transport layer protocol in the TCP/IP suite. It is a connection oriented unicast protocol, reliable and full duplex, TCP friendly with Multi-homing and Multi-streaming support. Connection is established via a 4-way handshake with cookie exchange for added security. In SCTP end points may bind to multiple IP addresses, and exchange these addresses when establishing a connection. Afterwards each end point can receive messages from any of the registered addresses. An end point may switch their primary IP address at ease depending on their current network POA, thus catering for seamless handover. However, SCTP does not make any provision for registering a new IP address after connection establishment and it is unfeasible to know

all subnets a mobile user will use prior to connection setup on the go, making this protocol unsuitable for highly mobile users. ADDIP, an extension to SCTP allowing for terminals to add (register) IP addresses after connection setup was defined as M-SCTP. This addresses the deficiency of SCTP for mobile users. However, it does not support mechanisms to dynamically change port numbers within the 4-tuple, hence cannot support fine-grained mobility. SCTP 4 way handshake during connection setup provides much needed security to guard against DoS [38] attacks and is acknowledged in our proposed solution.

2.4. Solutions on other layers

Traditionally TCP/IP protocols are already heavily loaded with functionalities added over the years, the optimization and adding new functionalities to support mobility is very difficult. Therefore, another idea for Internet mobility is to introduce a new layer such as Host Identity Protocol or implement it on a higher layer like Session Initiated Protocol where the Internet mobility may be deployed.

2.4.1. Session Initiated Protocol

The Session Initiation Protocol (SIP) [17] provides an application layer mobility for establishing and tearing down multimedia sessions, both unicast and multicast. SIP uses entities such as user agents, proxy servers and redirect servers for mobility support. A user is addressed using an URL email-like address "user@host", where "user" is a user name or phone number and "host" is a domain name or numerical address [39]. Each new transaction in SIP has a unique call identifier, which identifies the session, this identifier remains constant as the session may be modified (during handover or adding another media). SIP defines a number of methods and messages used by user agents for protocol operations. The SIP server relays these messages, so that it is possible to use a domain name to find a user rather than knowing its IP address. Supporting Mobility for TCP with

SIP [19] spoofs constant TCP endpoints in a similar way to MIP with route optimization. This requires modifying the IP stack of the corresponding host. As an advantage, SIP can work without any mobility support of lower layers and support a variety of mobility types. However, this is subjective to obvious costs. The handoff delay of SIP is extremely high and even with intelligent optimization the delay is not tolerable to real-time applications with high QoS requirements. Also mobility support using SIP suffers from the termination of transport connections, to this effect there are works proposing to use SIP in conjunction with MIP [40, 54]; however this combination is inefficient as there is redundancy in MIP home agent and SIP server on mobile node's home network to keep track of mobiles node current location. This redundancy has been addressed in [55] but the high handover latency still remains an issue.

2.4.2. Host Identity Protocol

The Host Identity Protocol (HIP) [9] is a protocol proposal to secure host mobility and multi-homing using cryptographic-based name space for Internet hosts. HIP supports mobility by decoupling the transport from the network layer, and binding the transport to a host identity. HIP uses cryptographic identifiers termed Host Identity Tags (HIT) at the application layer, which are mapped to IP addresses by a HIP protocol stack layer that interfaces between the transport layer and IP layer. As the HIP namespace is cryptographic in nature and the public keys are used in the connection establishment, heavy computation is required, hence not desirable for wearable miniature devices. In general HIP introduces a new namespace known as the Host Identity (HI) namespace, each host will have one HI, and this is recommended to be stored in DNS, the exception to this is anonymous identities [29]. Hence, with each host identified by a single HI and all transport connections identified with this HI (in a new tuple arrangement that excludes IP addresses); this shifts a transport connection

dependency from an IP address to the HI. This decoupling solves the IP duality role problem, and hence host mobility, however, end-points are more about applications than about hosts, from application mobility point of view. This is what we feel mobility should accomplish in next generation networks.

2.5. Analysis

From the evaluation of existing solutions we can deduce that there is no comprehensive mobility solution that would support continuous network connection sessions when (1) applications move between devices (2) when MN move between subnets and/or (3) simultaneous multi-access with low signaling and still be efficient, scalable and interoperable with the current IP architecture.

In the current architecture IP addresses and Port numbers provide identification for all ongoing transport connection. These are dependent on either the network attachment point or only valid within the host kernel locality. Hence, any scheme that uses these attributes for connection identification will constrain the granularity of mobility. This fine-grained mobility requires each network process and/or transport connection to be defined independent of attributes that are bound physically or logically to the host terminal or kernel resource. Fundamentally, the insistence of using IP addresses and/or Port numbers for connection identification makes it hard to develop handover solutions with low signaling load, with no dependency on foreign entities and that may support this fined grained mobility (e.g. application migration). As application mobility is a subset of host mobility, we feel that attacking mobility from an applications viewpoint will provide more flexibility and support more functionality.

Currently mobility solutions are based on interoperability with TCP semantics. Either mobility is hidden from TCP (network layer solutions) or

TCP itself is enhanced, to include mobility options (transport layer solutions). Generally speaking these solutions may be categorized as patches to the current TCP/IP architecture, synonymous to currently adopted intermediate packet modifying functions like NAT, which have increased network address space at the expense of modifying packets in midstream. We see now the drive for adoption of intelligent middle-boxes (HA, FA) to resolve the mobility 'crisis' as it was with NAT to solve the IPv4 address crisis. This is leading to a 'tug of war' between big businesses including telecom providers and *end-to-end principle* loyalists. On one hand we have people/organizations who want more control of data traversing their access networks to pave way for commercialization, billing, and flexible resource allocations. On the other hand we have the internet community loyalists and open source purists who believe that the internet should be a dummy network and no one should own it.

At the end of the day the driving force behind the adopted mobility scheme will be the killer application; many believe it will be based on wireless with ubiquitously in mind. With the known deficiencies of TCP/IP in a wireless environment, instead of the continual 'patching' of TCP, this research pushes for a total re-engineered solution from the underlying philosophy of TCP/IP. A solution in support of the end-to-end argument noting that the argument isn't just about correctness and performance but also realizes that the closer to the end points you place functionality, the more control users have, which allows more innovation. Importantly this solution should give more control to applications to influence mobility, paving way for new application innovations at the same time being compatible with the current and suitable for the future IP framework.

A total re-engineered mobility schemes will first have to answer the question of *'on what layer should mobility be implemented?'* Wesley [27] discusses this in depth, and suggests the transport layer as the strongest

candidate. In this work we believe mobility is more of a connection problem rather than a routing problem hence should be approached on the transport layer. This is evident with results of current network layered schemes (MIP), despite the advantages from its location in the stack, and although fully functional, the pace of its deployment has been very slow and faces more problems in terms of handover latency and security. A transport layer solution may be deployed optionally to users requiring mobility; as a software upgrade (windows service pack, Linux module). Service providers, will find initiative to update there systems too with steady rise customer demand for mobility and to stay competitive with other service providers. Based on this evaluation, certain desirable design criteria for an IP based continuous connection solution can be identified:

1. Solution to be based on the transport layer so that end to end semantics are maintained;
2. Solution should not depend on attributes that bind to terminals attachment point or kernel memory (IP addresses and Port numbers) so that mobility may be as granular as possible;
3. Solution must be able to support legacy applications, as well as provide application programmers power to make applications influence mobility
4. It must be possible for applications to migrate from one terminal to another;
5. It must not depend on foreign entities for handover, so that the only point of failure is the end point itself;
6. It must not require many signaling for it to achieve handover, thus reducing handover latency and bandwidth in the network – end to end signaling, must be minimized;
7. The solution must be able to support frequent handovers; and

8. The design must not add new security vulnerabilities to harm existing mechanisms

A proposed solution to this apparent deficiency will be presented in chapter 4; we first build on the requirements and underlying philosophy of the solution in chapter 3. The solution is based on Christian Huitema's 1995 Internet proposal [19], and borrows from SCTP connection establishment to guard against Denial of Service attacks. It Uses Diffie Hellman [41] elliptical curve protocol for key establishment to encrypt packets, these are both discussed in detail in chapter 6.

Chapter 3

3 RECONSIDERING SOCKET PAIR FOR CONNECTION IDENTIFICATION

Current mobility schemes view mobility from a host terminal standpoint, treat mobility as a routing problem and approach mobility by either introducing some kind of intelligence in the routing architecture or attempt to patch/modify TCP/IP without changing its underlying philosophy. This chapter motivates the need for a change in philosophy; from treating mobility as a routing problem to a connection identification problem. The rest of this chapter is organized as follows. It begins with a discussion of Transport Layer Associations and Connections, where and why the current transport layer connections fail to preserve ongoing communications during a topological change in Section 3.1. Section 3.2 proposes a mechanism to maintain transport connections during a topological change in address. This mechanism is centered on the idea of introducing a new connection identifier to multiplex transport connections. Mobility elements

are redefined in Section 3.3 to fit the perspective of the proposed mechanism. The chapter concludes in Section 3.4 by discussing attributes and guidelines that should be used in selecting a connection identifier for the proposed scheme.

3.1. Transport layer Association

Historically, transport layer connections identifiers (TCP/UDP) on the Internet have contained topological information. In the case of TCP/IP, for example, the TCP/IP association contains a pair (local and foreign) of Port numbers and IP addresses together known as the TCP/IP tuple. The Port number field is a 16bit integer used to identify a logical connection between an application entity and transport service [65]. The IPv4 address field is a 32bit field where each connected IP interface has a unique value. IPv6 uses effectively the same structure, using a 128 bit identity domain in place of the 32 bit field. In both cases the IP address is a structured identity space where there is a globally significant prefix that is used in the context of routing and forwarding outside to a particular local domain, and a local part that is used to deliver the packet to the correct interface of the associated device within the local network.

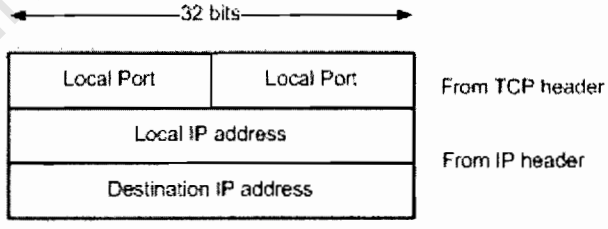


Figure 3-1 TCP 4-tuple

The IP address is unique within the Internet framework and is used to identify an interface. Likewise the Port number is unique within the communicating entity kernel space and used to identify an application. The combined properties of these fields contribute to the uniqueness of the

TCP/IP tuple. Originally the TCP/IP tuple was a good connection identifier, because connections were static and could be identified on account of their topological location (constant IP address) and logical resource space within a kernel (constant port number). However, the introduction of mobility (terminal and application) and multi-homing has changed this situation; we describe three scenarios below, where this property fails to support the promise of next generation networks and application alike.

3.1.1. Terminal Mobility

When a mobile terminal changes its point of attachment during a communication session, it forces all upper layer communication sessions to cease. For the purpose of this dissertation we are concerned with movements that are evident to host on the Internet. Such movements may or may not coincide with the actual physical movement of the mobile terminal (example re-addressing). Internet attachment points may receive new addresses from time to time due to configuration changes in the network. As the Internet address changes, the TCP/IP tuple is invalidated causing all ongoing connections to disconnect.

3.1.2. Fine grained mobility

It is not a far-fetched idea, for applications and/or application subsets in the near future, to move across terminals whilst preserving transport connections. For example a 3G-enabled user engaged in a videoconference session on a mobile phone enters an office equipped with an Internet connected HDTV set [43]. Users will naturally want to maximize their *user* experience and should have a variety of options to do so. Below we describe two such options. It is assumed the video-conferencing application uses two connection streams, one for voice and the other for video.

1. Migrate the both connection streams to the television set (if it has a microphone).
2. Migrate only the video stream of the application to the television set since the 3G network is superior in voice channel.

To support these operations, the transport layer connection should be able to withstand the changes of application space identifiers i.e. port numbers along with a change in IP address. Since port numbers are unique only in the terminals kernel resource space, there is no guarantee that a port number of a particular application on one terminal will be unique if migrated to another terminal; hence port numbers can not remain constant in events of this fine grained mobility. However, a change in the port number field of the TCP/IP tuple will break the TCP connection.

3.1.3. Simultaneous Multi-Access

Due to the availability of a wide variety of wireless access technologies, a mobile host can potentially have subscriptions and access to more than one wireless network at a given time. For example, a user subscribed to both wide-area wireless networks with a data rate of 2Mbs indoors (e.g. 3G systems), and a local-area wireless network with an effective data rate of 5Mbs (e.g. WLAN). Assuming the user is in range of both access networks; as a service of simultaneous multi access - *aggregate bandwidth* addresses the question: '*Can an application that requires reliable and sequenced delivery of data be provided at a data rate of 7Mbs through the use of both interfaces?* In such a case, packets delivered to the multi-homed terminal are to be addressed to different network interfaces. Since the TCP/IP tuple comprises endpoints IP addresses and hence bound to network attachment points, the resulting data streams will not share a common connection identifier. A simple approach to aggregate bandwidth would be to use multiple TCP sockets, one each for every active connection, and use application layer striping and re-sequencing. However,

it turns out that such an approach fails to achieve optimum aggregate bandwidth rate [42]. Using a connection identifier not bound and/or independent to a particular network interface will make it possible for packet streams addressed to different network interfaces to share a common transport connection session.

3.1.4. Summary

The three scenarios above are examples of the lack of the flexibility induced by physically and/or logically binding the mobile terminal resources to the transport connection identifier. As we move to an era where application/service innovations will determine the survival or death of the underlying network architecture and/or business models, there is a need to develop protocols that bundle more flexibility and control to application programmers and hence promote innovativeness. Flexibility in retrospect to a fine grained mobility framework, providing support for individual network processes and/or threads to move across mobile terminals and/or applications, and control in terms of making provision for applications to influence and be aware of handover and mobility in general. It will be difficult to realize these requirements under the current TCP/IP connection identification scheme, hence the need to move to a more accommodating identification scheme.

3.2. A Change in Philosophy

The internet is quickly and constantly evolving, the traditionally modeling the internet today is proving to make it difficult for applications to realize their true potential, and to meet user expectations. There has been global interest by various research institutions to understand and build a future Internet that achieves its potential [60, 61, 64, 68]; by principally approaching its design with a clean slate. We believe that the current connection identification scheme is one of the key elements that have restricted application innovations. This may be demonstrated by the slow

rate of IPv6 deployment in comparison to the advantages it promises to the internet framework. These advantages range from a decrease of routing table size, to the rise of true end to end network communications [59]; all this bundled within a strong IPSec [56, 57] security system. The slow deployment of IPv6 can be explained by the lack of killer applications that will require IPv6. As new exciting applications bring new services which in return means new forms of revenue, the willingness of ISPs to help in the roll out of IPv6 will increase if the promise of new revenue can be realized. The hindrance to such applications is the TCP/IP connection identification philosophy, the design of its protocol control blocks and hence the manner in which it forwards messages through its protocol stack.

3.2.1. TCP/IP Multiplexing scheme

The TCP Protocol Control Block (PCB) contains state information for one endpoint of a given connection. A TCP demultiplexing algorithm (PCB-lookup) must find the PCB corresponding to the connection for each newly arrived TCP packet. The algorithm does this by mapping the packet's source and destination IP addresses and TCP ports (4 – tuple) to the proper PCB. Figure 3-2 briefly describes processes involved during TCP packet demultiplexing. TCP stores the TCP Control Block (TCB) and PCB information of a particular connection in kernel space and the kernel module maintains the association between a socket number and four-tuple (TCP connection identifier). When a network interface receives TCP packets, it calls tcp_input() routine. TCP input retrieves the IP addresses and port numbers from the packets IP and TCP headers respectively. These 4 fields together form the TCP 4-tuple. tcp_input matches this 4-tuple to its corresponding socket number through a process known as PCB lookup [7]. The packet payload is then forwarded to this socket number. Note the socket number addresses the destination application within the kernel. The

process is done in the reverse order at the transmitting end. Refer to Figure 3-2

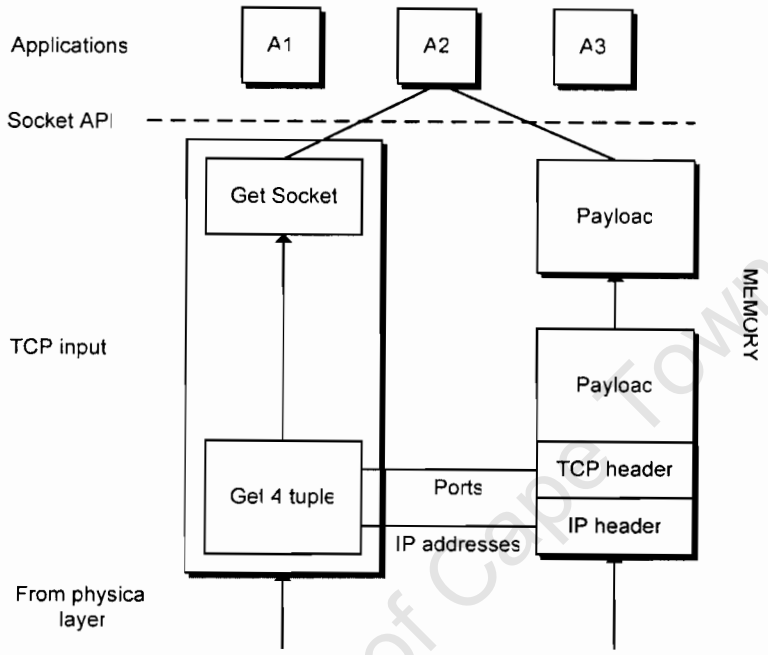


Figure 3-2 TCP/IP packet de-multiplexing process

As shown the kernel maintains a mapping between the socket number and the four-tuple TCP connection identifier; hence the four-tuple has a one-to-one relationship with the socket number. If any parameter in the four-tuple changes, the association gets broken and so does the connection.

3.2.2. A new Multiplexing scheme

If a TCP connection can be independent of its four-tuple identifier, then mobile terminals will be able to change their IP addresses (and hence network interfaces) and/or port numbers at will without disrupting the TCP connection session. We believe that network layer independence from Transport Layer protocols can be achieved through the inclusion of a unique connection identifier (CID) within packet headers.

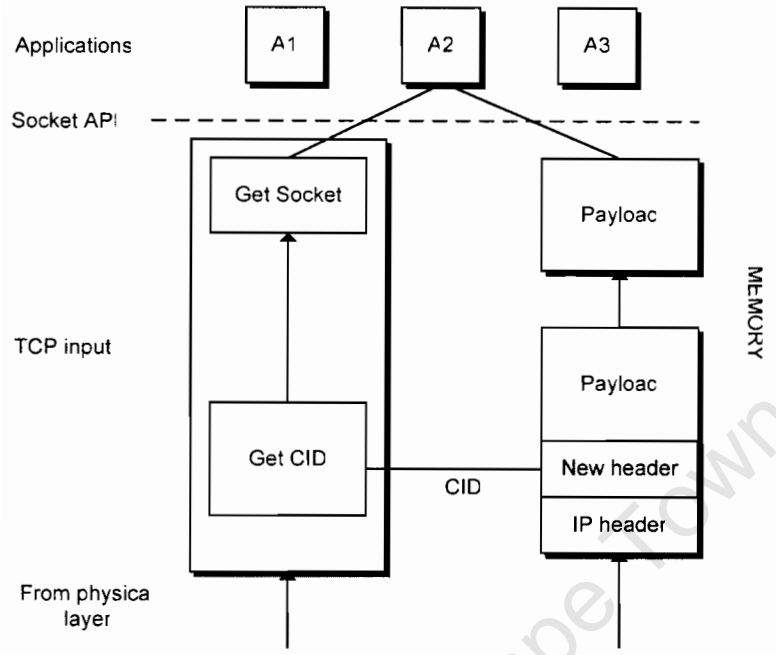


Figure 3-3 Modified de-multiplexing scheme

As Figure 3-3 demonstrates, by introducing a connection identifier the dependency of the IP layer is zeroed out. This way whenever there is a change in four-tuple identifier, it will not affect the connection state. Instead mechanisms can be designed to update corresponding IP addresses when an IP address change occurs.

3.3. Redefining Mobility Elements

Multiplexing packets using identifiers other than the TCP/IP 4-tuple is not an entirely new idea; it was first envisaged by Christian Huitema [19]. This philosophy is based on new ideas, and takes a clean slate approach in viewing the network protocol stack. It also redefines the normal norms of internet terminologies as viewed in the internet today. In this Section we redefine the key mobility elements with retrospect to our proposed philosophy.

3.3.1. End points

Traditional endpoints are defined to be the point of attachment, and are usually identified by the IP addressed assigned to that point of attachment thus physically binding the end point to the network interface card. Defining an end point in this manner reduces the flexibility of the end point in terms of mobility. Especially with the rise of application mobility, where applications are envisaged to move seamlessly from different communication devices, this binding will hinder the development of such applications since their functional operations are physically bound to the network interface.

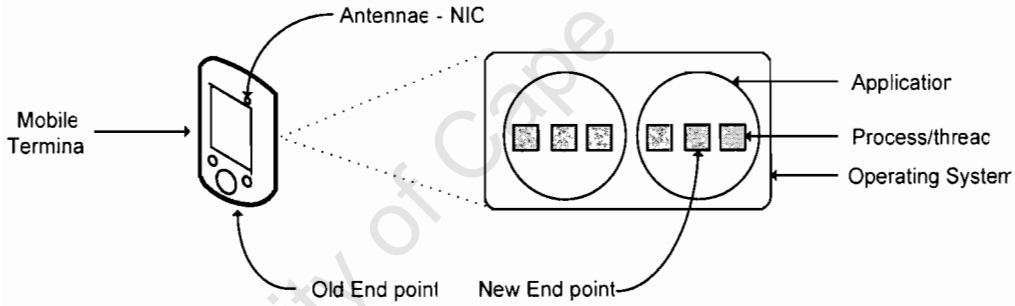


Figure 3-4 End points as defined in current framework

We define end points as the processes which communicate with other processes in their own or remote hosts. These processes (end points) are generally the ultimate source and destination of data in the network [58]. An application might have multiple processes communicating over the network, thus having multiple end points. Defining an end point in this manner makes provision for applications to take advantage of a fine grained mobility. Applications may be designed in such a way that for example, one may wish to migrate the transport connection streaming visual data to a nearby digital television set but maintain the transport connection conversational stream on his phone or earpiece. Solving the

mobility problem on this end point unit as opposed to the entire host device is moving in a direction of vast possibilities and makes provision for future inventions.

3.3.2. Connection Association

A traditional connection represents an end-to-end relationship between two end nodes. Hence local unique identifiers of the respective node pairs, known as sockets, identify the connection; a socket pair forms the TCP/IP tuple, which acts as the connection identifier. The important thing to note here is the manner in which the identifier is defined using entities that are unique to a particular end system (sockets).

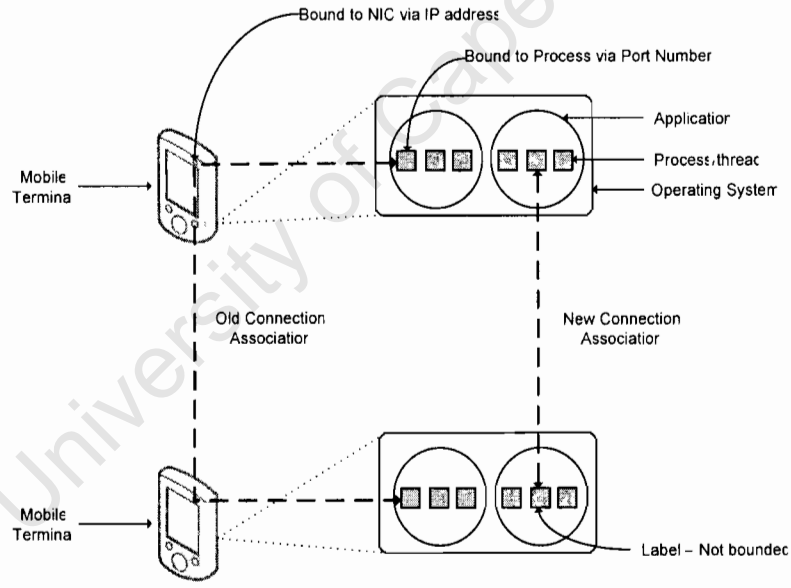


Figure 3-5 A new end point definition

We define a connection association as a label for a flow of data communication between pairs of end points (as defined above). This label is totally self reliant, not bound to any entity within the end points, making it easier to understand the semantics of a connection migrating from one end point to another end point during handover. It is important to note that

when a connection migrates, the actual end points remain stationary and a new endpoint is created at the destination. Only the connection association (label) is exchanged between the old and new end points. With the correct connection identifier and state information (sequence numbers, window size etc), the new endpoint can seamlessly use the transport connection. The corresponding host will note it as a change in IP address and update its PCB accordingly.

3.4. Connection Identification

The failure of TCP/IP tuple to support terminal and application mobility motivates for investigation for a better alternative to the TCP/IP tuple. Before heading down the path of designing a new transport layer connection identity, it is useful to ask what sort of attributes this identifier should have in order to support fine grained mobility functionalities.

3.4.1. Attributes

- **Uniqueness:** It is important to understand that the assertion of one connection identity does not use the same identifier token as another connection. It is imperative that the connection identifier be unique so as not to interrupt communications. The identifier should be globally and internally unique.
- **Persistence:** It would also be good if the identity used when establishing a connection remains the same throughout the lifespan of the transport connection session. Constantly changing identities are, at the very least, difficult to track.
- **Trust:** It would be good if an assertion of a particular connection identity could withstand a challenge as to its validity. Others who would like to use this identity would like to be reassured that they are not being deceived.

- **Robustness:** And lastly, it would be good if the identity realm was capable of withstanding deliberate or unintentional attempts to corrupt it in various ways.

3.4.2. Guidelines

So if these attributes are valuable in any digital identity system, the next step is to combine these attributes and create a new identification scheme that can support application and terminal mobility and hence be used as the driving force for new applications and technologies. Thus the guidelines in creating such an identifier will be

- Make it impossible (highly unlikely) for different terminals to produce a similar identifier for their connection sessions
- Make it impossible for a terminal to produce two unique connection identification numbers
- The creation of the number should not depend on external systems, should all be done internally preferably within kernel space
- The identity should not be topologically bound to the mobile terminal or locally bound to a process within the mobile terminal
- The connection identifier should be self configurable, self manageable and dynamic

These guidelines require the identifier to be locally and globally unique. The proposed philosophy in this dissertation is centered on a mechanism of acquiring such an identifier using the aforementioned guidelines.

Chapter 4

4 SOCKETLESS TCP PHILOSOPHY

One of the main difficulties in supporting mobile end points is preserving on-going communication channels. Currently the association between these channels relies on topological and logical information at both end points. As described in Section 3 these dependencies restrain the granularity and flexibility of current mobility schemes. This chapter builds on the multiplexing scheme described in chapter 3 to introduce a new concept that can be used to support applications on mobile end points. The philosophy is based on a new connection identifier, created using guidelines stipulated in Section 3.4.2. The identifier in this case, is used to demonstrate the concept only, other schemes may be used to create an identifier that abides to the guidelines. The rest of the chapter is organized as follows. In Section 4.1 we describe the goals of the proposed philosophy as a framework to preserve ongoing communications during a topological change in end points. The guidelines, protocol elements, and relationship of Socketless

TCP (SL-TCP) with other protocols are then discussed in Section 4.2. Section 4.3 briefly discusses how the connection identifier relieves the logical and physical dependencies of the transport layer i.e. the basic concept of the SL-TCP model. Using the guidelines stipulated in Section 3.4.2 a suitable connection identifier is constructed in Section 4.4. The chapter closes in Section 4.5 by illustrating higher level protocol operations when mobile node migrates.

4.1. Design Goals

Socketless TCP aims to provide a comprehensive transport layer framework that can support mobile applications in the 4G era. A scalable solution that will encourage innovation of applications as network and access technologies advance; the following are the goals of the proposed solution, to enhance the demultiplexing process of TCP.

4.1.1. End to End Solution

The popularity of the Internet is a result of its freedom in infrastructure; that drew support from many organizations [62]. The design was based on a dummy network that only dealt with routing of packets; all connection handling intelligence was to be implemented at end systems themselves, this was later referred as the end to end argument. However, the end to end argument isn't just about correctness and performance, the closer to the end points you place functionality, the more *control* users have, which promotes innovation. Hence the criteria of an end to end approach to mobility, where only the end hosts are involved in the mobility scheme.

4.1.2. Small signaling overhead

As discussed in Section 1.1.2, signaling overhead contributes the most to handover latency in mobility schemes. By reducing the number of signals used in order to achieve handover, will not only reduce handover latency but also increase bandwidth efficiency in the network links.

4.1.3. Simultaneous Multi-access

Future mobile devices (and already some current) will be equipped with multiple network interfaces. Within a wireless overlay network framework; it may be possible that a mobile device is connected to two or more access points simultaneously. It becomes natural that users and applications alike will want to take advantage of both networks to increase the user experience or lower their operational costs. Unfortunately the current TCP/IP protocol stack does not support simultaneous multiple access; in the form of the transport layer to correctly forward packets received by different interfaces, to a single transport connection. The SL-TCP concept does not focus on how a mobile host can be reached at two different addresses, however, it attempts to provide the transport layer the ability to demultiplex multiple streams of data to a single transport connection end point. Applications can then take advantage of this functionality, by receiving multiple streams of data over different interfaces. This is particular important for aggregating bandwidth for heavy downloads.

4.1.4. Low processing cost

With recent advancements of miniaturization mobile devices have taken up new forms and in the near future one should expect to see wearable network devices. The success of these devices will depend highly on developments on low-power designs. Network protocols have an important part to play to reduce power consumption, especially since reading and writing to memory are among the most expensive operations in a terminal. We observe that the introduction of IPv6 within the Internet framework will increase the amount of processing at end systems; in particular during PCB lookup process, where packet forwarding involves bitwise matching the connection identifier (300 bits) with various PCB's in memory. The importance of the transport layer in the role of reducing these processing costs is thus noted. Reducing the size of the connection identifier reduces

the number of bits that need to be matched and hence, reducing number of bits read and written to memory during packet demultiplexing at the transport layer.

4.2. Design Framework

Presented below are the key design elements of Socketless TCP that allow achieving the goals of transparent host mobility across different subnets.

4.2.1. Guidelines

Based on Cerfs 1983 Department of Defense Internet Architecture paper [63] and documented reports of several other researchers over several years [2, 19, 34, 48 51]. Three important guidelines are identified, that should be followed as hints in designing appropriate network architecture for supporting mobile internet services: *eliminate lower layer dependency from higher layers, eliminate physical or logical terminal dependency from connection identifiers* and *provide support at the end points*. A brief discussion of these guidelines follows:

- ***Eliminate lower-layer dependency:*** The dependency of TCP on network host addresses for part of its connection identifier makes dynamic reconnection difficult, a problem which has plagued network designers since the inception of the ARPANET project in 1968 [63]. The result is that when the underlying network-layer (IP) address of one of the communicating peers changes, the end-to-end transport-layer (TCP) connection is unable to continue because it is bound to the network-layer identifier, tacitly (but wrongly) assuming its permanence for the duration of the connection [51], also known as the mobility problem. This may be solved by providing a mechanism to abstract the host addresses from the connection identifier.
- ***Eliminate logical/physical Terminal Dependency:*** The dependency of TCP on a host's terminal logical resource identifiers (port numbers) for

part of its connection identifier makes it difficult to support finely grained mobility. In particular when attempting to migrate transport connections between terminals (during application mobility). Port numbers are only unique and valid locally within a particular host terminal there. It will be difficult to guarantee the uniqueness of a port number during an attempt to migrate a transport connection. Hence the need of abstracting the port numbers from the connection identifier. We believe that by treating mobility as finely grained as possible, i.e. from a network process perspective; other forms of mobility may be coupled as subsets to it.

- ***Provide support at the end hosts:*** One of the primary complaints for IPv4 and, consequently, one of the principle drivers of IPv6 adoption was IPv4's lack of end-to-end connectivity. The end-to-end argument is discussed in depth by Saltzer [2]; briefly stated, end-to-end connectivity holds that any network elements between a source and a destination be transparent to the endpoints of a packet flow. Future applications, particularly those based on peer to peer communications, can only be correctly executed at the endpoints and any attempt to modify packets by intermediate network elements limits the *functionality* and *innovativeness* of the application from the user and programmers perspective respectively.

4.2.2. Elements

In the design of SL-TCP there are two new elements added to the traditional TCP architectural framework, namely – end points and connection identifier.

- ***End points:*** As defined in Section 3.3.1, end points are the ultimate source and destination of data in the network i.e. the process. Within

kernels these communicating processes are addressed as sockets. Whenever the BSD socket call is made, a new end point is instantiated.

- ***Connection identifier:*** The connection identifier is the field used for matching PCB's in the PCB lookup algorithm. This field is included within SL-TCP header packet.

4.2.3. Host Environment

SL-TCP is to be implemented as a module in an operating system. The user can access SL-TCP through a set of functions. SL-TCP may call on other operating system functions, to help in its implementation. Similar to TCP, SL-TCP does not call on the network device driver directly, but rather calls the internet datagram protocol module which may in turn call on the device driver [7].

4.2.4. Relationship with other protocols

SL-TCP is expected to be able to support higher level protocols efficiently. Emphasis is given on maintaining the current API used by TCP, with additional methods that applications can take advantage of. However, in a mobile aware environment some legacy system calls may return invalid information, example gethostbyname() function call, will be irrelevant in the current world since IP addresses are no longer static but rather ephemeral in nature.

4.3. Protocol Concept

This Section describes the general concept of operations for a Socketless based mobility scheme. Here, TCP as a connection oriented protocol is used to illustrate the difference and relevance of a Socketless based scheme; however, the concept of SL-TCP can easily be transferred to other transport layer protocols.

4.3.1. Overview

Before considering technical details, the general idea behind Socketless based mobility support can be intuitively described by managing the mapping of connections and hence the demultiplexing of packets from the network layer without using the conventional socket pair that defines a TCP association, hence the name Socketless. The role of conventional sockets is thus redundant, however depending on implementation it may be used during connection establishment phase as network applications require to bind to a local service access point (SAP). To maintain the functionality of defining a connection similar to the socket pair, we adopt a new unique number we call the Connection Identifier (CID). This CID has no binding association with the physical attachment point of the terminal, thus remains unique for the duration of the communication session.

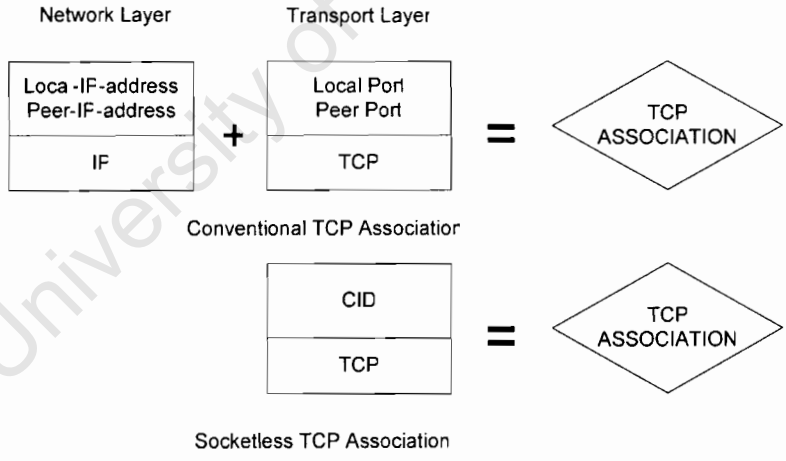


Figure 4-1 Difference between TCP and SL-TCP association

In general, Socketless based endpoint mobility attempts to create an abstraction between the network and transport layer within the TCP/IP model, thus providing the transport layer a degree of independence from the network layers physical attributes (IP address). Figure 4-1 summarizes

this concept showing the difference between a TCP and SL-TCP connection association.

4.3.2. Model

SL-TCP protocol model for mobility aware network applications suggests that any interaction from a MN to a CN over a packet switched network should be based on the connection identifier i.e.

- Connections should be made between end points. An end point defined as the smallest unit of communication i.e. the network process
- Packets at the transport layer should be multiplexed/demultiplexed using a CID. A CID being a unique number acting as a label to a network connection flow.

This provides an elegant method for accommodating the special requirements of mobile hosts and applications in a way that is compatible to existing network applications and also makes provision for innovation of future network applications. All the specialized support that is needed for mobile applications can be built on top of the socket API which will support all current methods and include an additional set of methods that mobile applications can take advantage of.

4.3.3. Model of Operations

End points transmit data by calling on the SL-TCP and passing buffers of data as arguments. The SL-TCP packages the data from these buffers into segments attaches a connection identifier and calls on the IP module to transmit each segment to the destination SL-TCP. The destination SL-TCP is located using the destination IP address which is maintained locally at the end system. The receiving SL-TCP checks the connection identifier and places the data from the segment into the corresponding user's buffer and notifies the receiving user. Socketless TCP like TCP include control

information in the segments in forms of opcodes and synchronization bits to ensure reliable ordered data transmission.

4.4. Connection Identifier

As described in previous Sections, in conventional network architectures including IPv4/IPv6, the network address of a node denotes its identity as well as its location. This is a critical problem for supporting terminal and application mobility, because there is no location independent identity for a mobile end point. To solve this problem, SL-TCP introduces a new entity in the transport layer to support mobility, Connection Identifier.

A connection identifier (CID) is a unique string representing the name of the connection. This name is used to multiplex packets from end points to lower layers and forward packets from the network layer to end points accordingly. In retrospect to the characteristics of a connection identifier as discussed in Section 3.4, we formulate an identification scheme that abides by the aforementioned guidelines.

4.4.1. Guideline 1 – Global Uniqueness

The first guideline states that the CID has to be structured in such a manner that: *'Makes it impossible (highly unlikely) for different terminals to produce a similar identifier for their connection sessions'*. This is achieved by ensuring that every terminal creates an identifier using a string that uniquely identifies the terminal from other terminals in the world, i.e. Global Uniqueness. Global uniqueness may be achieved by using the Medium Access Control (MAC) address. The MAC address is a unique value associated with a network adapter. MAC addresses are also known as hardware addresses or physical addresses. They uniquely identify a card from all others in the world. MAC addresses are 12-digit hexadecimal numbers (48 bits in length). Within SL-TCP philosophy we term this number to be the Global Identity (GID). Similar to MAC addresses the

GID consists of a 48 bit number derived from the MAC address, this property makes the CID globally unique.

4.4.2. Guideline 2 – Local Uniqueness

The second guideline requires the CID to be structured in a manner that: *'Makes it impossible for a terminal to produce two unique connection identification numbers'*. This is achieved by ensuring that every connection has a unique number associated with it within kernel space, i.e. Local Uniqueness. SL-TCP's CID Local uniqueness may be achieved in the same manner as its TCP/IP counterpart 4-tuple, Port Numbers. In TCP/IP the port number is an identifier for a logical connector between an application entity and transport service [7]. SL-TCP distances itself from this definition and function and borrows only the role of TCP/IP port numbers as of uniquely identifying an entity in kernel space locality. Within the SL-TCP model we redefine Port Numbers as the Local Identity (LID). Similar to port numbers the LID consists of a 16 bit number generated and maintained by the host systems kernel.

4.4.3. Guideline 3,4,5

The rest of the guidelines state: *'The creation of the number should not depend on external systems, should all be done internally preferably within kernel space. The identity should not be topologically bound to the mobile terminal or locally bound to a process within the mobile terminal and thus should be self configurable, self manageable and dynamic'*. These guidelines describe the manner in which the CID should be created. Guideline 3 is particularly important for it stipulates that the CID creation should not depend on external systems, stressing the point that foreign entities should not be involved in the creation.

4.4.4. Connection Identifier

Using the above guidelines we ended up with a connection identifier which is made up of 48 bit GID and a 16 bit LID. Both ID's are generated and maintained locally within the kernel, the GID is derived from the MAC address and the LID is maintained by the operating system in the same manner TCP/IP maintains Port Numbers, However it is important to distinguish between the 16 bit LID and TCP/IP port numbers. The distinction lies between the fact that TCP/IP port numbers are used to identify network processes within the kernel i.e. there is a one-to-one relationship between a network process and port number. In SL-TCP case the 16 bit LID is a number concatenated onto the GID to distinguish CID within the local entity.

4.5. A Migration Example

The design of SL-TCP worked around the Internet model whereby a mobile node initiates a connection to a server, the server maybe mobile or stationary. As with normal connection initiation setups, the mobile node first resolves the servers name through its local DNS server, Figure 4-2. Once it acquires the servers IP address, it initiates a SL-TCP connection via a 4-way handshake.

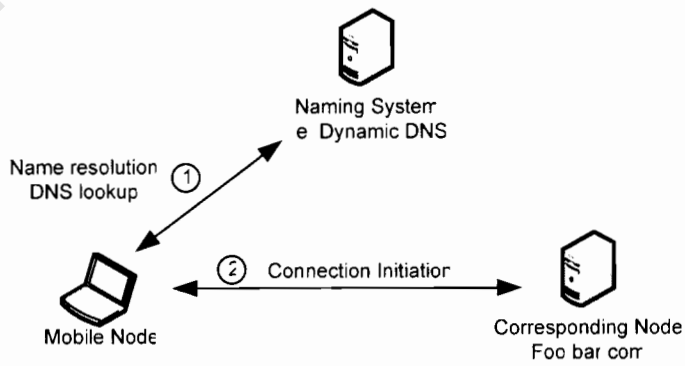


Figure 4-2 Migration example a

Once an SL-TCP connection has been established the peer nodes continue with normal communication; when a node changes its IP address for some reason, (e.g. DHCP lease time, change in POA, readdressing) SL-TCP migration process is initiated, the process activity depends on which node is moving, the server or the client. If it's the server that is migrating, two procedures take place, 1 – sending of a binding update to its corresponding node (the client) 2 – the server performs a DNS update using Dynamic DNS protocol [66], to update its name to IP number resolution on the DNS system Figure 4-3.

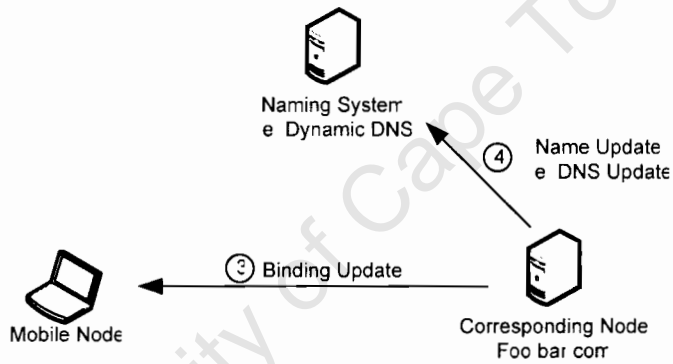


Figure 4-3 Migration example b

For the case where the mobile node is migrating step 4 i.e. name update is not required and is omitted. It is important to note that in SL-TCP a binding update is embedded within the packet communication structure, i.e. no extra signaling is involved, thus reducing overall handover latency. SL-TCP migrating function takes note of address change and updates destination IP address, within the socket mapping.

Chapter 5

5 SOCKETLESS TCP ARCHITECTURE

This chapter presents the core architecture components of the SL-TCP framework; however, we focus only on the key changes made to the TCP/IP protocol stack and leave out protocol specific operations that remain similar in function to that of its TCP/IP counterpart. These include, congestion control specific algorithms and data structures, checksum calculations, sequence number generation, packet retransmission and other implementation specific functionalities. Furthermore, this architecture description is by no means complete and only acts as a guideline to demonstrate and motivate the philosophy behind Socketless TCP framework. The rest of the chapter is organized as follows. Section 5.1 introduces basic architectural components of SL-TCP, detailing SL-TCP header format and briefly discussing how SL-TCP interfaces with the user. Section 5.2 discusses the key changes made to TCP data structures to accommodate the SL-TCP framework. The data path of packets through

the SL-TCP protocol stack when a packet is received and transmitted is discussed in Section 5.3, along with a brief discussion on how a connection is established. The chapter closes in Section 5.4 with an illustration of how the SL-TCP framework supports terminal handover, fine grained mobility and simultaneous multi-access.

5.1. Architecture

The current generation protocol architecture TCP/IP, seem arguably successful at meeting the demands of today's networks. However, there has been a gradual increase in demand for a number of new requirements by casual Internet users, which require necessary innovation in protocol structuring. This Section introduces the key changes to TCP/IP architecture adapted to the SL-TCP framework.

5.1.1. Overview

The SL-TCP architecture is based on the surprisingly simple idea of eliminating the reliance of IP address in identifying a communication session. In current IP networks, it is impossible to keep end-to-end transport connections alive when one or both connection endpoints move because physical network protocol endpoints are used by transport protocol to identify its connection. SL-TCP uses a unique connection identifier to break this tie between the transport protocol and network protocol by substituting the socket pair identification formed by the local and peer IP and port number with a unique connection identifier thus its name Socket-less TCP. Once the transport endpoint identification is made independent of network endpoint identification, the lifetime of a transport connection is no longer limited by changes in network endpoints.

5.1.2. Header Format

Figure 5-1 presents the header format of SL-TCP segments. These segments are sent as internet datagram's. The Internet Protocol header

carries several information fields, including the source and destination host addresses [10]. The SL-TCP header follows the internet header, supplying information specific to the SL-TCP protocol. This division allows for the existence of host level protocols other than SL-TCP.

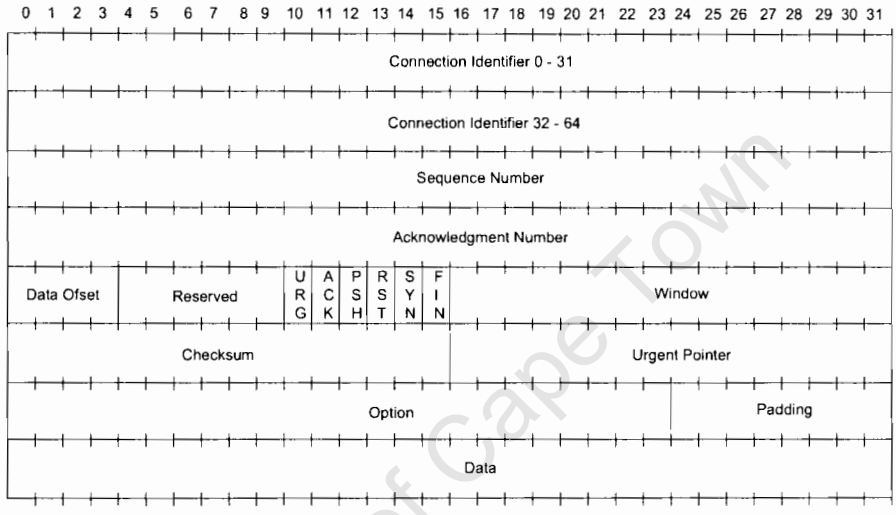


Figure 5-1 SL-TCP header format

The difference between TCP and SL-TCP header is that SL-TCP replaces TCP local and foreign port fields with a 64bit CID. All the other fields remain the same, and have the same functionality as used in TCP.

5.1.3. Basic Architecture

Figure 5-2 provides an overview of the SL-TCP architecture and key data structures. SL-TCP as a transport layer protocol interacts with the application and IP. To be compatible with legacy TCP applications, SL-TCP makes use of the same BSD socket methods in TCP API, but redefines their implementation to support the new architecture and added functionality. SL-TCP provides the same reliable and sequenced delivery semantics as TCP.

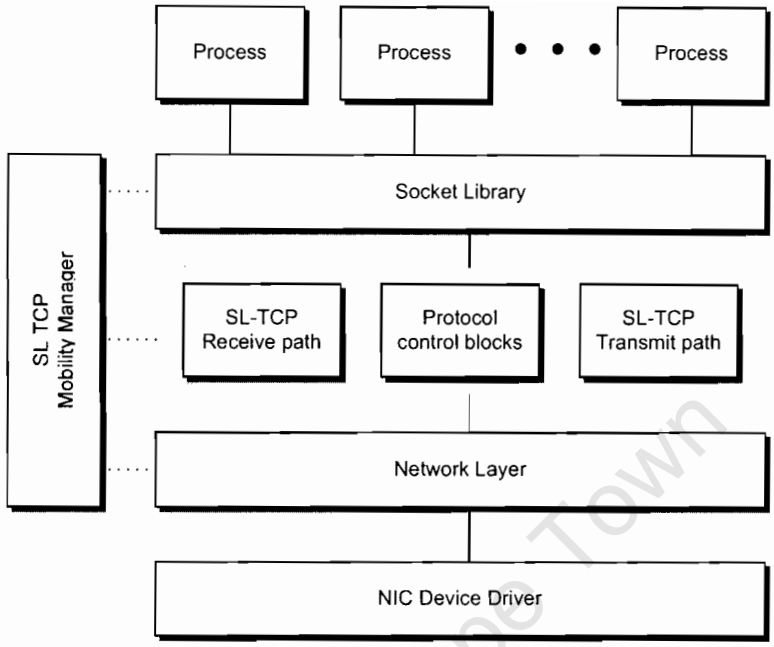


Figure 5-2 Block diagram of SL-TCP architecture

The transport functionalities associated with handover capabilities are built within the SL-TCP protocol packet transmit and receive paths utilizing modified TCP protocol control blocks. SL-TCP mobility manager interacts with other layers in the protocol stack providing provision for a cross layer information flow support system. These components are implemented within the Transport Layer, as a modification to TCP.

5.1.4. SL-TCP Interface

SL-TCP interfaces with the user via normal send() and recv() user calls [69] of the BSD socket API. There are certain flavors of methods that can not be implemented within SL-TCP purely because they are not relevant in a mobility environment; example, methods like gethostbyname() [69] that require IP addresses as an arguments, are no longer relevant since IP addresses no longer reflect the identity of a particular host. Note, this means that legacy applications like FTP [67] that make use of these

methods may not function properly. On top of the current methods new methods can be introduced to SL-TCP architecture, methods that give applications provision to take advantage of mobility, and multi-homing functionalities.

5.2. Data Structures

Here we describe modifications required of TCP to implement SL-TCP. We base this modification from the common reference implementation of TCP/IP from the Computer Systems Research Group at the University of Berkeley. The socket and PCBs are the primary data structures relevant in our discussion.

5.2.1. Mobility Manager

The mobility manager has access to state information specific to other layers within the protocol stack, and uses this capability to gather and provide mobility hints to relevant layers within the protocol stack. This functionality has been introduced, in due acknowledgement to research in cross layer mobility management protocol design [70] which promise extended functionality and/or improved performance, which are hard to gain via a single-layer signaling scheme. The specific structure and functionality of the mobility manager within the SL-TCP framework is not discussed in this dissertation, however, we acknowledge the need for cross layer interaction in the SL-TCP protocol stack. In particular the significant impact to applications during vertical handover, where bandwidth may change abruptly due to physical limitations to the underlying access technology. If the network layer could signal the applications and transport layer of this change, a more graceful adaptation to these abrupt changes may occur.

5.2.2. *Protocol Control Blocks*

Each transport layer protocol maintains state information in its protocol control block. Some protocols like UDP have very little state that they do not have the need to keep a separate control block and use the same control block as IP - Internet Protocol Control Block. TCP uses two; Internet Protocol Control Block and the TCP Control Block. SL-TCP's modifications to TCP data structures and control blocks have no effect to TCP congestion control and/or flow control mechanisms; modifications are rather confined to TCP packet multiplexing and forwarding processes. Hence, the protocol control block of relevance to SL-TCP is the Internet Protocol Control Block (struct inpcb [69]), TCP Control Block (struct tcpcb [69]) is discussed briefly for completeness.

5.2.3. *Internet Protocol Control Block*

The Internet PCB contains information common to all UDP and TCP endpoints: foreign and local IP addresses, foreign and local port numbers, IP header prototype, IP options to use for this endpoint, and a pointer to the routing table entry for the destination of this endpoint. The TCP control block contains all the state information that TCP maintains for each connection including: sequence numbers in both directions, window sizes, retransmission timers and the like. Figure 5-3 summarizes the protocol control blocks showing the difference between a TCP and SL-TCP implementations.

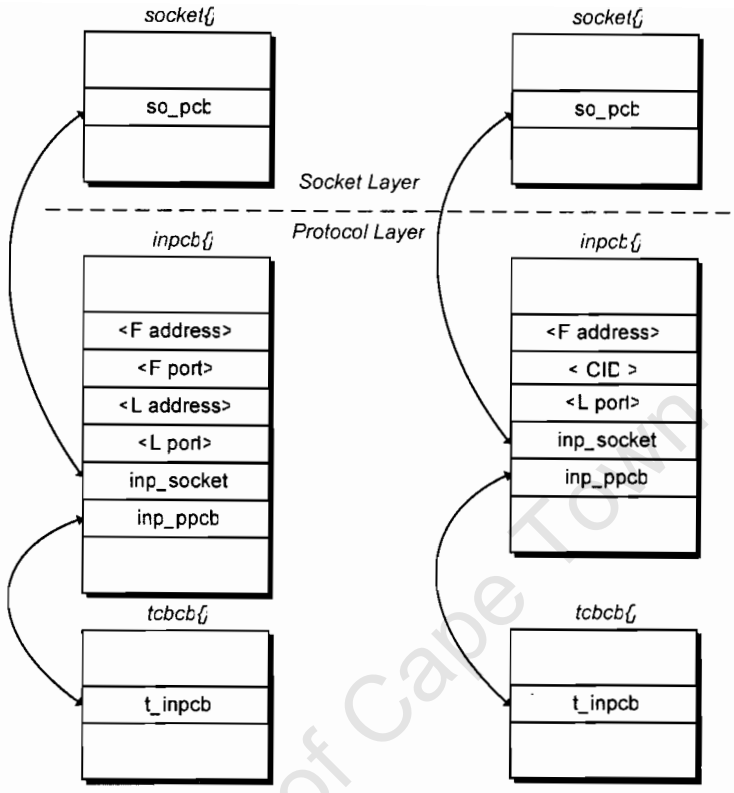


Figure 5-3 Difference TCP and SL-TCP control blocks

The main modification made in SL-TCP to the TCP implementation is replacing the local IP address and foreign port with SL-TCP's connection identifier. For operating systems using IPv4 the introduction of a 64 bit CID will induce (64 – 48) 16 bit overhead, which may be substantial since reading and writing to memory are among the most expensive operations [71]. However, for operating systems using IPv6 the introduction of CID will induce (144 – 64) 80bit saving, which is a substantial cost saving. This modification implies TCB lookup is now based on the CID rather than the TCP 4-tuple. CIDs remain constant through the lifetime of the connection session, and are generated by clients in a manner that will guarantee global uniqueness. This setup implies that in a scenario that the corresponding host changes its IP address, the CID will remain constant, and hence the

TCB containing the corresponding connection state information can be retrieved.

5.3. Protocol Processing

SL-TCP/IP processing can be generally divided into two parts (1) connection management and (2) data path processing. Connection management involves setting up and tearing down the connections and is discussed in Section 5.3.1. The SL-TCP data processing consists of *Transmit* and *Receive* paths (also known as paths); both of which can be quite different in their requirements as shown in Figure 5-5 and Figure 5-6. We focus only on mechanism relating to SL-TCP, IP functions and other TCP procedures remain the same; hence the main focus is on packet multiplexing and forwarding process.

5.3.1. Connection Setup

The initialization process consists of the steps stipulated in Figure 5-4, assuming that SL-TCP endpoint 'A' tries to set up an association with SL-TCP end point 'Z' and 'Z' accepts the new association.

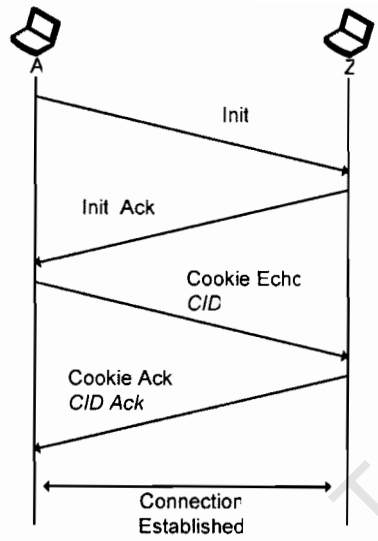


Figure 5-4 SL-TCP connection establishment

SL-TCP adopts a four way handshake connection establishment scheme similar to that of SCTP [37]; it incorporates the use of cookies to guard against DoS [75] attacks. A complete CID is generated at 'A' and received by 'Z' which creates a TCB using SL-TCP header information and maps this TCB with the CID generated by 'A'.

5.3.2. Transmit Path

The operations required in the transmit path are illustrated in Figure 5-5. When the application needs to transfer data, it signals the CPU and passes the socket number of the applications end point. The socket number is used as a key to read the CID, destination IP address and a pointer to the application buffer among other things from the TCB. The stack then copies data from the applications buffer to the socket buffer. Writing the SL-TCP header and IP header follows this. The TCB is then updated and procedure is passed on to the datalink layer.

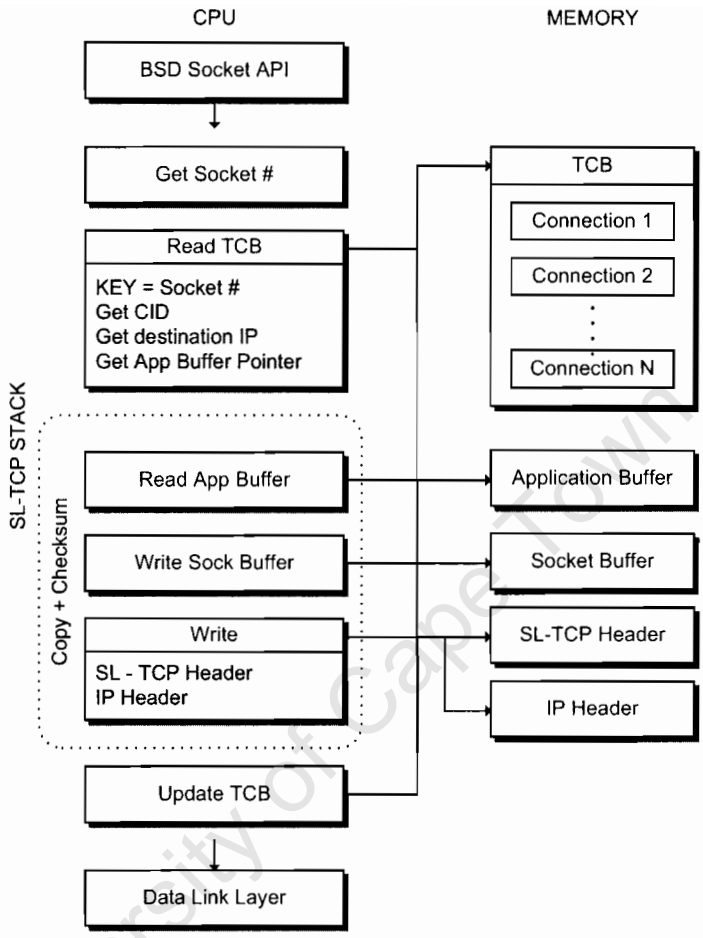


Figure 5-5 SL-TCP transmit path

5.3.3. Receive Path

In case of receive, the NIC receives data, places data into the memory and interrupts the processor. This receive path is illustrated in Figure 5-6. The respective NIC device driver receives the interrupt and passes the buffer containing the data to the SL-TCP stack and reads the TCP and IP header information. The stack then reads the TCB using the CID retrieved from the SL-TCP header as a key. The stack retrieves the Socket number and current destination IP address associated with the CID among other things (window size, timer info etc) from the TCB. Based on the receive SL-TCP

window size, the stack determines when to send the acknowledgment of the received data. The data is either copied into the socket receive buffer associated with that connection or placed directly into the application buffer if one is available. Depending on the implementations the stack can either *immediately* overwrite the destination IP address of the corresponding TCB with the Source IP address of the received packet or compare the two IP addresses and overwrite the corresponding TCB *only* if the two IP addresses are different (handover occurred). This is to ensure that the last received IP address is used for acknowledging the received packet.

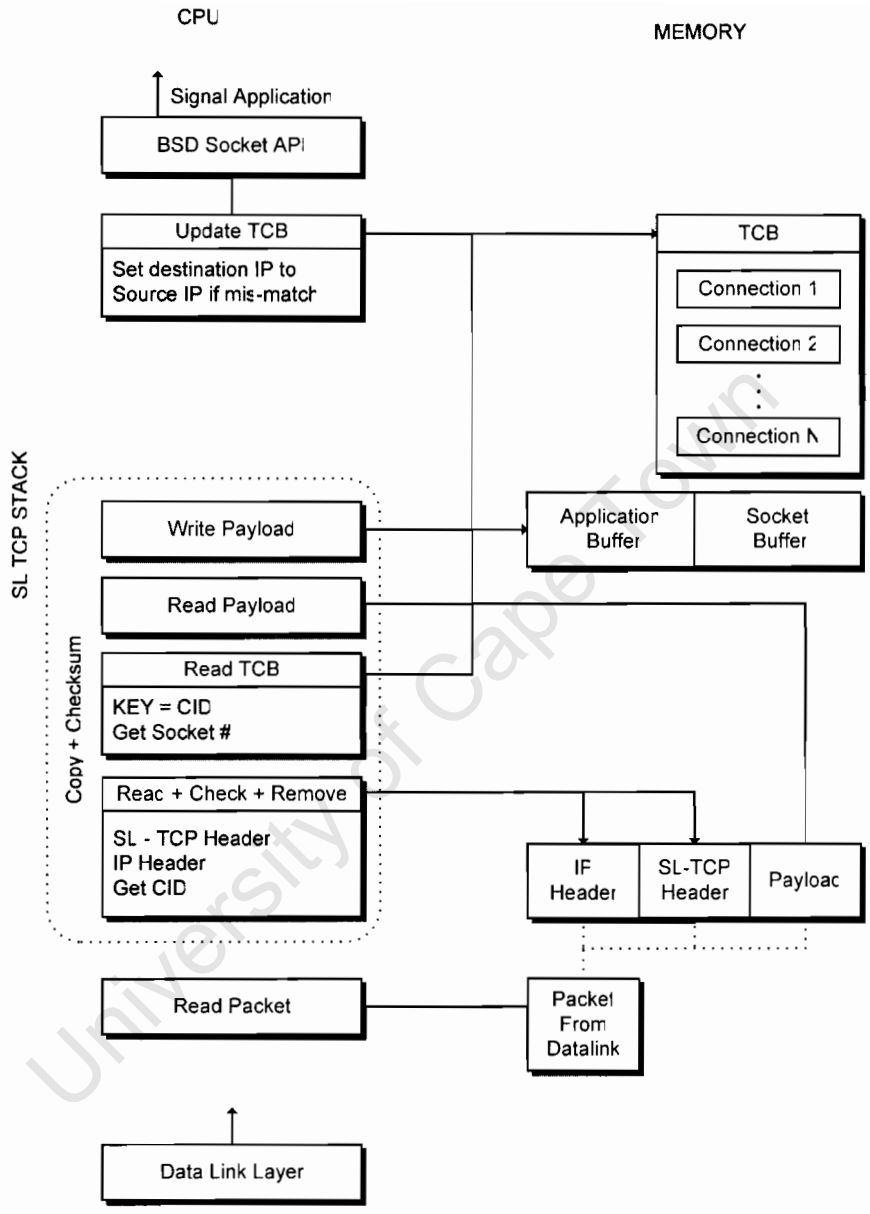


Figure 5-6 SL-TCP receive path

5.3.4. Checksum Calculations

The risk of *errors* being introduced into a transport layer segment during its travel across the Internet is an important critical impediment to the safe transmission of data. Corrupted data may be received at end points, if the

corruption is not detected. TCP includes a 16 bit *checksum* field in its header to provide basic protection against errors in transmission. SL-TCP also adopts this mechanism, with similar checksum calculation the only difference is that SL-TCP has a different header structure as described in Section 5.1.2.

5.4. Protocol Illustration

In this Section we illustrate how SL-TCP protocol handles different mobility functionalities.

5.4.1. Terminal Mobility

SL-TCP is designed to support seamless handover during network disconnections. Handover requires the communicating end points maintain correct information of the destination end point within its local PCB and provide a mechanism of identifying pertaining connections regardless of an IP address change.

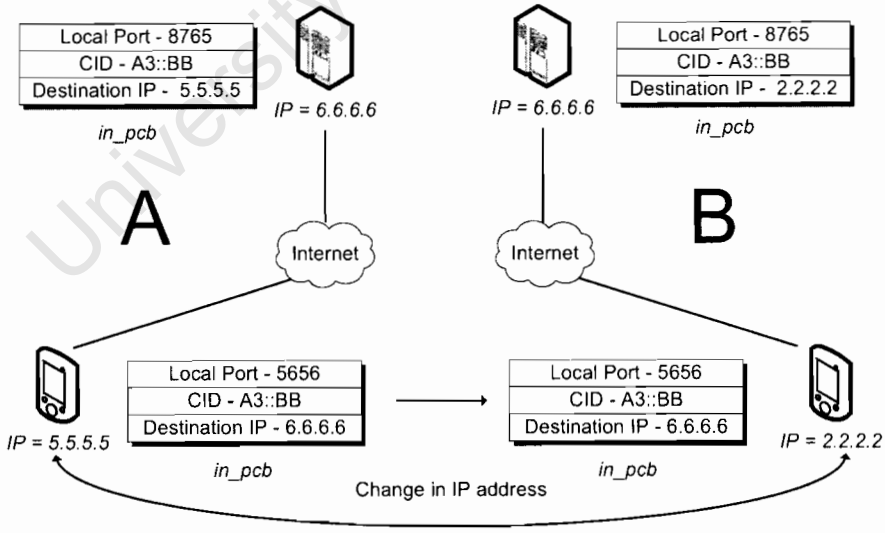


Figure 5-7 SL-TCP illustration - terminal mobility

Figure 5-7 illustrates a classical example of terminal mobility where a terminal moves from network A to B and acquires a new IP address. SL-TCP at the mobile node terminal need not be concerned with this change, as IP will attach its header reflecting this change in IP address. However, SL-TCP at the stationary node will notice a change in IP address and update the destination IP address field in its *in_pcb* for the corresponding CID, to reflect this change; hence, subsequent packets from the stationary node will be addressed to this new IP address, finalizing the handover process.

5.4.2. Fine grained mobility

SL-TCP provides support for fine grained mobility. This involves tracking the set of active connections and devices for a user then routing transport connections to the optimal device. Devices register themselves for a particular service whenever they detect an authorized device/service is nearby (e.g. devices have Bluetooth transceivers which may act as smart cards for identification). Devices register by setting a trigger with an identifier representing the user device. Given multiple simultaneous registered devices, the devices follow an agreement protocol to decide which one will handle a particular transport connection stream (e.g. voice packets are routed to a telephony device whilst video traffic are routed to a larger area LCD display device). Levering the SL-TCP infrastructure for application mobility removes the cost of deploying infrastructure specific to any particular application mobility scheme. Figure 5-8 and 5-9 illustrates by example how a video streaming connection is migrated from a PDA to a High Digital Television. The migration occurs in two steps (1) Registration and (2) Takeover.

Registration: This involves the communicating device (PDA) notifying possible devices that can and/or support the current application, then selects the optimal device (HDTV). The HDTV sets up a connection with identical PCB (refer to Figure 5-8).

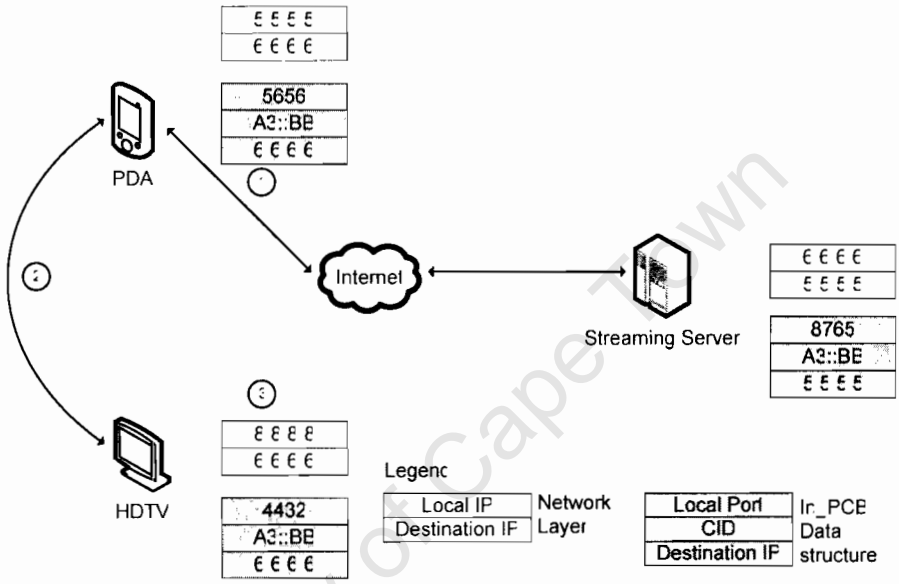


Figure 5-8 SL-TCP illustration - fine grained mobility registration

Takeover: The HDTV then receives the details of the last packet received by the PDA and sends an ACK to the streaming server acknowledging the packet on behalf of the PDA. Upon receiving this acknowledgment, the streaming server updates its destination IP address and continues the connection using this new IP address (refer to Figure 5-9).

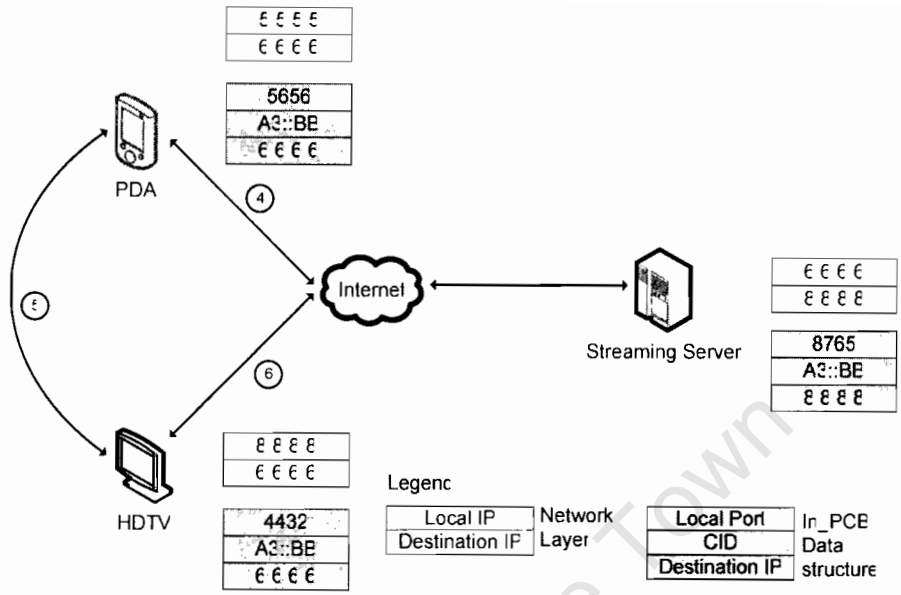


Figure 5-9 SL-TCP illustration - fine grained mobility takeover

Note that all signaling between the two terminals are controlled by the mobility manager, as it has access to state information specific to the application and transport layer.

5.4.3. Simultaneous Multi-access

SL-TCP provides support to application wishing to take advantage of the multiple network attachment the device is currently connected to. Multi-homing requires for applications to be able to access different network interface card(s) (NIC) simultaneously. The transport layer needs to make provision for notifying the applications asynchronously via a standard API that a network interface is active and ready to use. Also the transport layer must be able to route datagram's destined to the application regardless of its source address.

SL-TCP caters for this functionality by its property of routing packets based only on a CID. The application may take two approaches accomplishing this:

1. The application dynamically creates a new connection with a different CID and connects to its peer host via the new NIC.
2. The application uses the current CID
3. It is the applications duty to notify SL-TCP of the network interface each outgoing packet will use.

5.5. Security Enhancements

This Section describes an approach that can be integrated within SL-TCP architecture to guard it against common Internet attacks. The focus is to ensure that the SL-TCP does not make the Internet more vulnerable than it is already.

5.5.1. Security Threats

There are two basic security problems within the current internet framework: address stealing and address flooding. Address flooding come in various flavors, Denial of Service being its most common form. A denial of service attack is where resources are tied up on the server side so that it is impossible to respond to legitimate connections. The attacker issues vast amounts of SYN requests (a message requesting set-up of a connection) to the server and when it receives the SYN, ACK (refer to Figure 5-10 a) it simply discards it, not bothering to respond with an ACK. This causes the server to retain the partial state that was allocated after the SYN request, and if carried out repetitively will lead to a denial of service. Address stealing, is usually associated with connection hijacking, and occurs when a malicious node acts as if it was one of the end points and manages to read and send messages to the corresponding node. In its simplest form connection hijacking is illustrated in Figure 5-10 b.

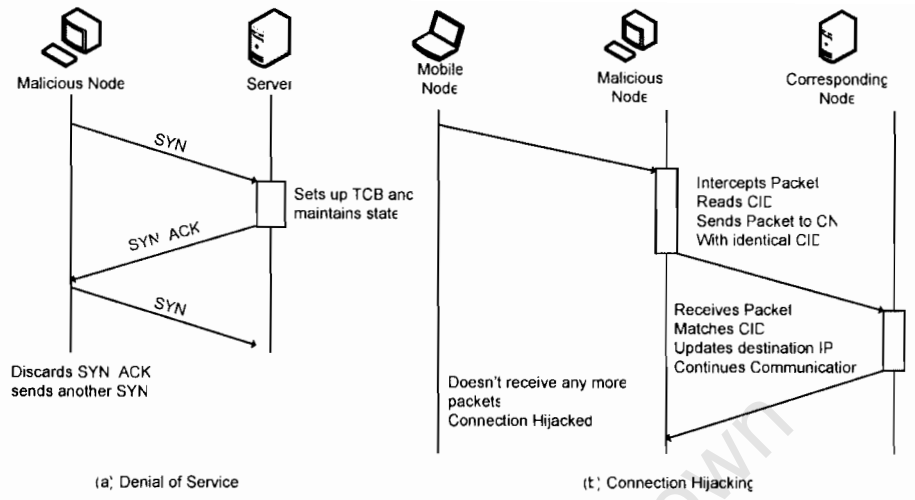


Figure 5-10 Common Internet security threats

5.5.2. *Solution*

These types of security threat characterize our solution. DoS attacks are launched during connection setup, and are due to a server and/or receiving node initializing TCB and maintaining its state, before making sure that it has a legitimate corresponding node. SCTP 4-way handshake with cookie exchange [38] is adopted to address this issue. Connection hijacking is generally stopped by end points not accepting an IP address update without being completely sure that the sending endpoint is not a malicious node. This can be achieved by encrypting the whole SL-TCP packet (using a pre-negotiated secret key) prior to transmitting the message, rendering the packet (SL-TCP header + payload) unreadable to any intermediate node. Furthermore, since the recipient nodes are required to decrypt each incoming packet, all forged packets will be discarded, as the recipient node will fail to regenerate the correct expected SL-TCP packet structure. Checksum calculation based on the SL-TCP header, original source and destination IP address will guard against scenarios where the malicious node intent is to divert all packets to itself, regardless of its inability to

decrypt and read the packet. Such an attack may be achieved by simply intercepting the packet, changing its source IP address, then forwarding it to the destination node.

5.5.3. Protocol Enhancements

To accommodate such changes to the protocol stack, certain data structures and processing procedures need to be modified. We discuss three key changes to the SL-TCP architecture to enforce the SL-TCP security approach. These are: a new field in PCB to store the secret key, a new connection establishment procedure to accommodate a secure key exchange and a modified header format.

1. A new field in PCB data structure

Every connection needs to be encrypted and decrypted independently using secret keys established prior to the connection. Since keys are assigned per connection basis, each connection has to maintain its own key. Thus we introduce a new field within the PCB to contain a secret key used to encrypt and decrypt packets within the protocol stack.

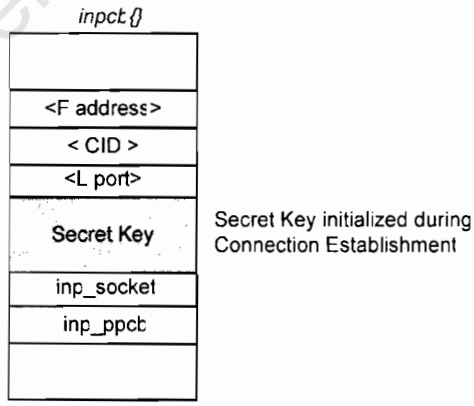


Figure 5-11 Modified PCB data structure with secret key field

Depending on the encryption code chosen this key can be from 64 to 128 bit long. A long key will increase the processing overhead and hence power consumption however will be more secure. A shorter key will be less secure but more power and processing efficient. Thus it is a compromise between security and efficiency.

2. Key Establishment

Keys are established during connection setup. Key establishment follows Diffie Hellman [41] elliptical curve algorithm. **Diffie-Hellman key exchange** is a cryptographic protocol which allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher [72]. This algorithm is suitable for this application since it was designed for the secure exchange of keys in an insecure network and/or communication channel. As a design criteria of SL-TCP we discourage the use of foreign entities, including keying authorities making SL-TCP self sufficient and self configurable. The key exchange is coupled with the 4-way handshake during connection establishment phase as illustrated to Figure 5-12. Where p is prime and g is a primitive mod p.

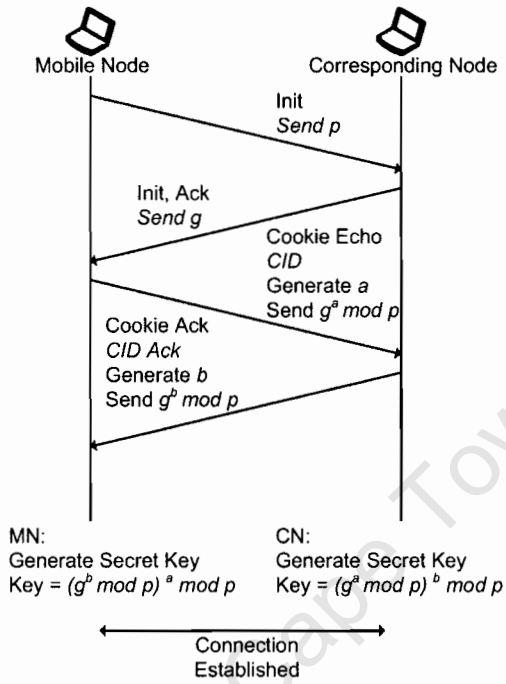


Figure 5-12 Modified connection establishment with key exchange

3. Modified Header format

Due to the encryption and decryption mechanisms performed on message segments during transmitting and receiving packets a new header format is required. The whole TCP segment is encrypted except the CID (refer to Figure 5-13).

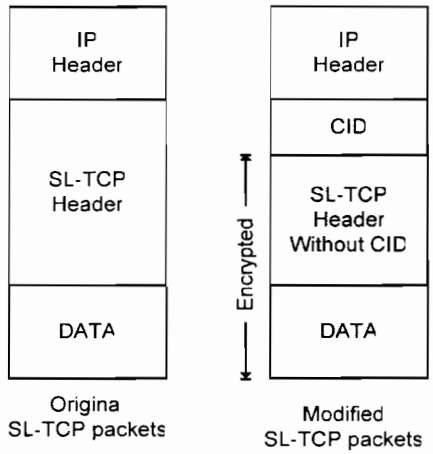


Figure 5-13 Original and Modified SL-TCP header packets

Thus the CID can still be used for locating the correct PCB. SL-TCP protocol stack determines which end point may know what is in the packet only if (1) it has the correct CID and (2) the end point connection matches the Sequence/Acknowledgment numbers otherwise the packet is dropped.

5.5.4. Protocol Illustration

Prior to transmitting the TCP packet needs to be encrypted, the key used to be encrypted is stored within the PCB. The whole data packet except for the CID is encrypted. Figure 5-14 summarizes the stack operations when receiving and transmitting a packet.

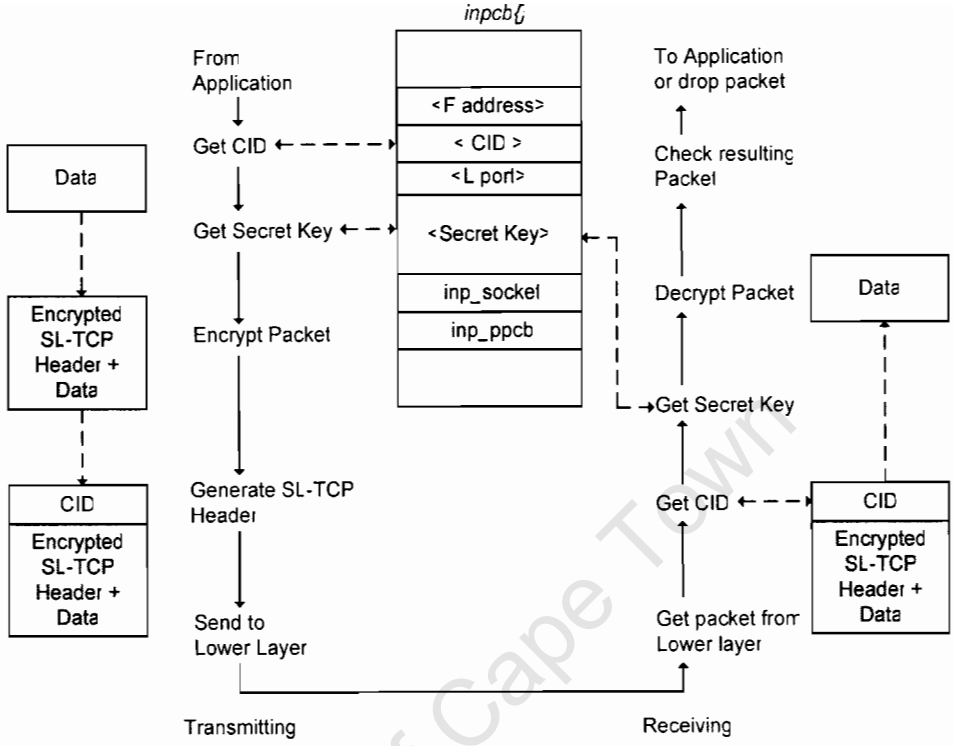


Figure 5-14 SL-TCP packet processing - security

Chapter 6

6 EVALUATION PLATFORM

The evaluation platform used to test the SL-TCP scheme is presented in this chapter. A stripped down protocol, focusing only on the packet multiplexing and demultiplexing aspect of the SL-TCP architecture has been implemented and is introduced in Section 6.1. The main support infrastructure for managing incoming and outgoing packets is implemented in XDP core described in Section 6.2, which interacts with the XDP client and server (Section 6.3) to manage CID based packet multiplexing. Section 6.4 describes the handover mechanism. The chapter concludes with an outline of known shortcomings of the evaluation platform.

6.1. Introduction

We introduce XDP – extended datagram protocol, which is used to evaluate various handover scenarios and related costs in respect to the SL-TCP philosophy. XDP is implemented in two parts: application module acting as clients and/or servers at the end hosts and an application XDP

core mimicking transport layer connection multiplexing functionalities. All these functionalities are implemented in user space and are based on the following design criteria

- It should be based on SL-TCP connection management techniques, particularly those discussed in Chapters 4 and 5.
- Its performance in respect to handover latency should be on par with other handover schemes evaluated under similar conditions. This will not only verify the feasibility of SL-TCP, but would also provide a good foundation for further improvement. Furthermore, if the performance of the platform is way below that of other systems, then the credibility of any subsequent improvements may be questioned.
- The software created should be well-commented and modularized so that it can easily be reused and modified. Also, design decisions and protocol algorithms should be made to limit memory usage as read and write operations to memory are expensive, hence may induce unnecessary delay.

6.1.1. Objectives

The aim is to demonstrate the applicability of SL-TCP in preserving ongoing communications and document its performance metrics to be used in comparison with other mobility schemes. For these purposes a striped protocol termed extended datagram protocol XDP, which acts as a transport layer protocol implemented on top of UDP has been implemented. The main objectives of the design are:

- Demonstrate SL-TCP philosophy
- Show that multiplexing packets using a CID can preserve communications during a change in IP address
- Calculate vertical and horizontal handover latencies

6.1.2. *Framework*

The evaluation platform is built up using Linux boxes installed with Debian 2.6.9-1.667 kernel release, it comprises of a transport layer XDP and application layer XDP client and server as different modules (refer Figure 6-1 below) implemented on top of the user datagram protocol (UDP).

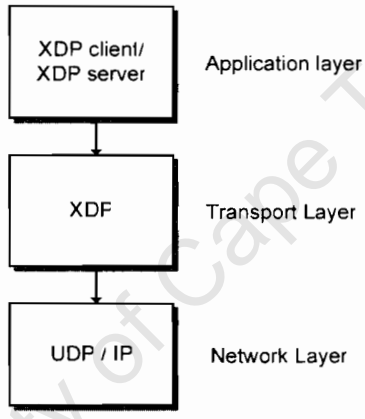


Figure 6-1 XDP block diagram

The evaluation platform ideally should be set up on top of the network layer, and communicated directly with IP; however, due to time limitations and programming expertise we opted to use UDP as an IP delivery agent. UDP relieves the need to perform various routine but cumbersome tasks of the transport layer; few are outlined below:

- The need to perform checksum calculations for each incoming packet
- The need to perform intrinsic system calls for transferring data from application layer to IP and vice versa.
- Abstracts the IP layer via the BSD socket API [69], thus providing a user-friendlier interface to IP for novice programmers.

- Buffer and memory management within the kernels paging structures

6.1.3. Data Structures

Key changes to the TCP/IP framework were confined to modifying PCB and TCP packet header; to accommodate these changes three data structures were implemented, using a struct object [73] namely: *map, packet and tcb. Struct tcb* (Figure 6-2 b) synonymous to TCB (TCP Transmission Control Block) [69] is used to store the current local and remote end points *sockaddr_in* [69] *struct*. The *map struct* (Figure 6-2 c) maintains a one to one relationship between connection identifiers and *tcbs* of all ongoing connections.

```
struct packet {

   int      CID                    /** Connection Identifier **/
   int      blockno                /** Block number **/
   int      type                   /** type – ACK or DATA **/
   int      is_handover            /** 1 – handover occurred  0  – no handover **/
   string   status                 /** either handover or normal **/
   string   conn_name              /** name of the connection **/
   struct   timeval tv             /** timestamp **/
}
        (a)

struct tcb {

   struct   sockaddr_in   d_addr   /** stores current destination IP**/
   struct   sockaddr_in   _addr    /** stores local port number **/
};
        (b)

map < int CID  struct tcb >        /** maps CID to struct tcb
        (c)
```

Figure 6-2 XDP data structures

The packet struct (Figure 6-2 a) is what is sent between the mobile end points, the connection identifier (CID) remains constant throughout the whole communication session. The block number (*blockno*) for packets of type DATA is incremented each time a packet of type ACK is received,

whenever a change in IP address is noticed the integer *is_handover* is set to 1 and status is set to 'handover' otherwise *is_handover* is set to 0 and status set to 'normal'. Every connection is given a meaningful name to distinguish between connections. Each packet is time stamped and its value is kept in the struct *timeval tv*, as a packet reaches an end point, all attributes are logged into a file with its corresponding timestamp.

6.1.4. Testbed Network Topology

Figure 6-3 shows the network topology of the testbed used in this evaluation platform. The testbed consists of a mobile node and corresponding node both running XDP core with either XDP server or client.

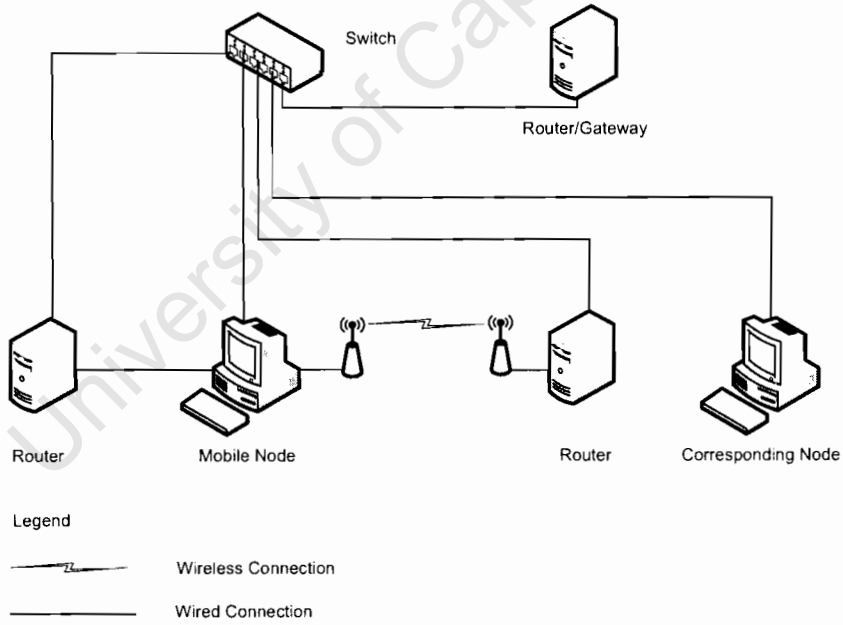


Figure 6-3 Testbed Network topology

The mobile node is equipped with 3 network interfaces, two wired and one wireless to enable it with vertical and horizontal handover capabilities.

6.2. XDP core

XDP core is responsible for managing all open connections on an end host. Normally, the mapping of CID to protocol control blocks is run per connection request from the user application. Because the objective of this framework is to evaluate performance, and not for connection setup evaluations, the connection setup procedure has been omitted from the client and server application. In order to create the CID – protocol control block mapping, the user has to specify these parameters when XDP is starting up. Mappings can be added while XDP is running to cater for more applications; with a constraint of ensuring a unique CID is chosen per new connection mapping. The CID is used to differentiate connection streams per application basis. Hence, each application can receive the correct connection stream. Both local and remote end host XDP mappings must be consistent with each other, so a user must take care to ensure both end host XDP mappings involved in any transport connection session are configured to the correct values.

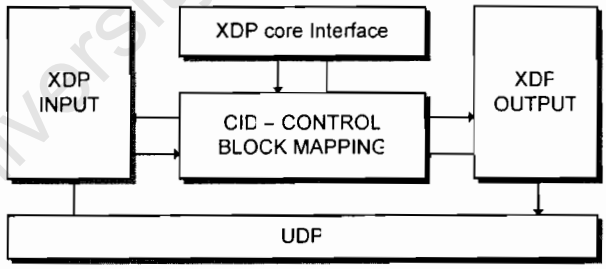


Figure 6-4 XDP core block diagram

We assume the XDP core is started up on both end hosts and configured with consistent mappings and runs continuously. If either hosts system or XDP core crashes, all active connection sessions are lost. Our current implementation does not attempt to recover any transport connection sessions or continuations lost due to a system or XDP core crash. XDP

core is implemented on top of UDP and consists of three key components i.e. the MAP struct, XDP input and XDP output. Figure 6-4 shows how these components interact with one another within the application.

6.2.1. Connection state management

During XDP core start up, the user is required to input relative connection state parameters, via the command line as follows: *./xdp <CID> <local address> <local port> <remote address> <remote port>*. A *tcb* data structure (refer to Figure 6-2 b) is formed using the local and remote port and IP addresses. XDP interface then registers this data structure to the *MAP*, with the input CID as a primary key. The *MAP's* corresponding data structure in BSD TCP/IP implementation is *in_pcb* [69]; since neither flow control nor reliability is implemented in our platform, we only include state information required to forward packets to the correct application i.e. *sockaddr_in* [69] included in the *tcb* data structure.

6.2.2. Preserving Connections – Handover

A trimmed version of SL-TCP protocol processing functionality as discussed in Section 5.4 is implemented in *XDP* core by *x_input* and *x_output* threads. These threads mimic the functionality of *tcp_input* and *tcp_output* in BSD TCP/IP [69] implementation respectively. Figure 6-5 and 6-6 show the typical path of a packet through *xdp core*. *x_input* and *x_output* run simultaneously and are bound to listen to port *X_INPUT* and *X_OUTPUT* respectively as defined in *packet.h*. All outgoing packets forward packets to port *X_OUTPUT* likewise all incoming packets are addressed to port *X_INPUT*. This setup ensures all *xdp packets* are processed by *XDP core* before forwarded to their respective destinations.

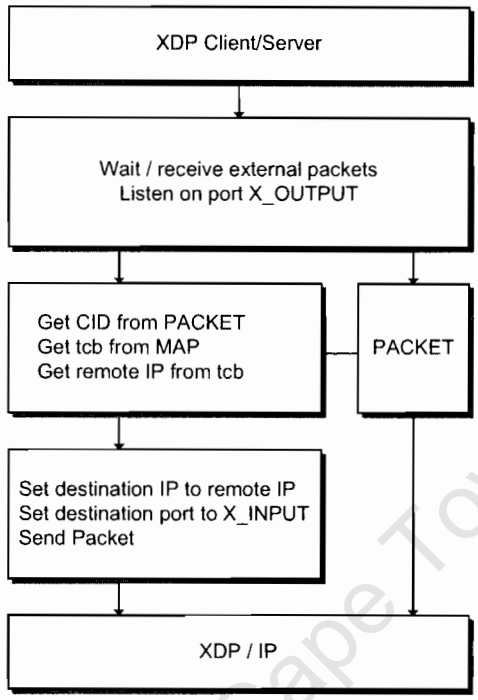


Figure 6-5 XDP transmit path x_output

x_output receives packets from applications (*xdp client/server*) via port *X_OUTPUT*, it then checks the packet CID using the *map* locates the correct *TCB* and extracts the current IP address. It then uses this IP address and a predefined port *X_INPUT* to initialize a *sockaddr_in* struct [69] to be sent to the destination using *udp* as a delivery service, (refer to Figure 6-5).

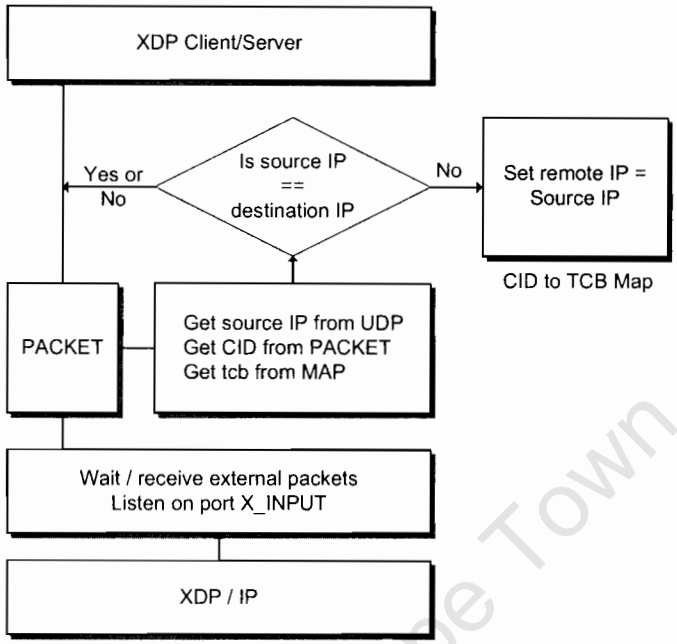


Figure 6-6 XDP receive path x_input

Figure 6-6 shows a block diagram of packet processing path on the receiving end. *x_input* listens to all incoming packets on port X_INPUT, once it receives a packet, it extracts its CID and source IP address. Using the CID it locates the corresponding *tcb* from the *map* and compares the destination IP from the *tcb* with the source IP from the received packet. If the IP addresses are the same *x_input* proceeds directly to step 4, otherwise if they are different, meaning a handover took place it does the following:

1. sets int *is_handover* to 1
2. sets string *status* to handover
3. updates the destination IP address in the TCB for the corresponding CID – TCB mapping
4. Delivers the packet to the corresponding application via port number from the corresponding TCB in the CID –TCB map.

6.3. XDP server and client

The client sends DATA packets and receives ACK packets, the roles are reversed with the server. The client needs to specify the port number of the destination server, a connection identifier and connection name; for demonstration purposes the connection identifier is any unique integer. Likewise the server requires a port number specified to bind to. When a packet is received at the server or client, all attributes within the packets header are logged onto a file. The client initiates the transfer and performs necessary latency calculations, is installed at the mobile node and carries out handover procedures.

6.4. Handover Mechanism

The handovers we are concerned with are those that are evident to the network layer that is a change in IP address. Handovers are performed at the server; and are enforced by either changing the default route in the routing table or by enabling and disabling the network interfaces. Since each NIC is assigned an IP address, by changing the default route and/or disabling a specific route, a new IP address is used as the source, for all subsequent outgoing packets. Note that when enforcing handovers by enabling and disabling the network interface, the NIC initialization procedure introduces additional latencies. Handovers enforced by changing the default route are performed using a script. This script uses IP tables [74] to (1) delete the current route (2) add a new route and finally (3) set the default route.

6.5. Platform Limitations

This Section outlines known shortcomings of the described evaluation platform as a result of time and resource constraints:

- A send and wait technique is used for the transmission, inducing very low throughput; hence any experiments and/or results based on throughput will be very poor.
- Reliability in terms of packet retransmission is not implemented; hence lost packets are not accounted for. Since a send and wait technique is used, if packets are lost during transmission, the packet stream ceases, hence only short lived experiments (approx 300 sec) are performed; longer experiments are advised to be done during the night when Internet usage is low and network load is at minimum.
- The platform does not implement any type of flow control, coupled with the lack of flow control within UDP, making it possible for clients to overrun the server; in such cases communication ceases.
- Connection establishment is not implemented within the platform; hence overhead induced during connection establishment phase can not be measured.
- Timestamps used are dependent on the operating system running on the hosts, and are not synchronized between the two end points; hence measurements may only be collected from one terminal.
- All measurements are reliant on the accuracy of the operating systems timer module, and are limited to micro-seconds.
- Since timestamp information is collected in user space, the measured delay actually contains additional overhead introduced by the operating system

Chapter 7

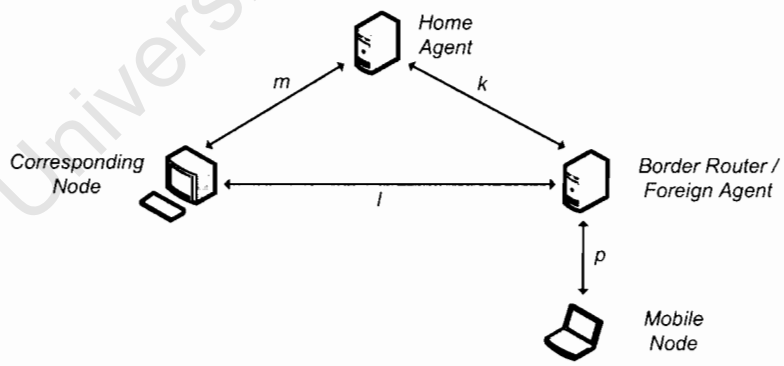
7 EVALUATING SL-TCP

This chapter evaluates SL-TCP in three different ways. First, we generate an analytical model and quantify the cost of SL-TCP and other mobility schemes in terms of delay associated with connection migration (handover) in Section 7.1. We find that with SL-TCP the handover cost is tied directly to the round-trip time (RTT) between the communicating peers – the total delay is almost exactly one RTT thus catering for real time services. Second, we demonstrate the applicability of SL-TCP using the evaluation prototype described in chapter 6. We show that handover latencies are indeed proportional to the RTT. Finally, we consider the impact of SL-TCP on packet processing overhead in Section 7.3. This overhead is constrained to the impact of the connection identifiers in matching PCBs in the PCB lookup algorithm. Our results show that SL-TCP can reduce end hosts processing overhead considerably compared to MIPv4 and MIPv6. This motivates the suitability for mobile devices with stringent power

resources. SL-TCP can reduce processing cost by 2/3rd of MIPv4 and 7/9th of MIPv6.

7.1. Analysis of Handover delay

This Section introduces the model used for the study and presents the parameters and assumptions chosen in the analysis. We derive equations for the handover delay for 4 mobility management protocols, two based on the network layer (MIP) and two on the transport layer including SL-TCP. The resulting equations are a function of 'distances' between the different elements involved in most mobility management protocols. The generic elements involved in most mobility management schemes and the distances between these elements are shown in Figure 7-1. We assume the reader is familiar with handover procedures in MIPv4, MIPv6 and Migratory TCP. Here we only briefly describe the handover procedure. For a detailed description of their architecture and implementation the reader is referred to [3], [4] and [46] respectively. This model is based on ideas extracted from Galli and Morera [76].



m	Number of hops on shortest path between HA and CN
k	Number of hops on shortest path between BR and HA
l	Number of hops on shortest path between BR and CN
p	Number of hops on shortest path between BR and MN

Figure 7-1 Distance between generic mobility nodes

There are two aspects in handover which contribute to the delay. These are a) mobility detection and b) registration /rebinding delay. Of the two components of Handover Delay (HOD), mobility detection can be heavily dependant on lower layer mechanisms and particular implementations. For the purpose of comparing exclusively on higher layer mechanisms, i.e. the mobility protocol, we assume the same mobility detection procedure for all protocols and do not include this parameter inside this analysis. Thus, we express the HOD only as a function of the registration (REGD) and binding update (BU) delay when the MN is the receiver of the communication. When the MN is the transmitter, the HOD is a function of packet delivery mechanism in specific protocols, e.g. via triangular routing or route optimization.

HOD is a function of the processing delay per-packet (C_p) at each network entity and transmission delay per-packet (C_{tx}) between two neighboring nodes one hop apart. We assume all hops have similar delays (e.g., an all wireless network), so sending a packet over i hops has a delay of iC_{tx}. In heavily loaded networks, C_{tx} and C_p are a function of *network load, mobility agent and routers' capabilities, Scheduling scheme mechanism, etc;* however, as an approximation, the variables C_{tx} and C_p can be considered constant in lightly loaded networks.

7.1.1. MIPv4 with Triangular Routing

In the mobile IPv4 framework, the handover delay (HOD_{MIPv4}) after the move is proportional to the time it takes the MN to send a registration packet to its Home Agent (HA) and get the acknowledgement back. If we assume that a FA acts as an intermediary that is one hop away from the MN and distance k from the HA, then:

$$HOD_{MIPv4}(k) = \begin{cases} 2C_{p,FA} + C_{p,HA} + 2(k+1)C_{tx}, \\ (\text{when MN is roaming}) \\ C_{p,HA} + 2C_{tx}, \\ (\text{when MN is in its home network}) \end{cases} \quad (1)$$

where $C_{p,FA}$ is the processing cost at the FA and $C_{p,HA}$ is the processing cost at the HA. In MIPv4 with FA, the HOD is the same whether the MN is transmitting or receiving, as the BR/FA will not forward any packets unless the HA registration is complete.

7.1.2. MIPv6 with route optimization

In MIPv6 with route optimization, the MN has to send a HA registration and execute the Return Routability (RR) procedure prior sending the BU to the CN. Thus, registration delay (REGDmipv6) has three components: registration delay, Return Routability Administration (RRA) procedure delay and Binding Updates (BU) delay

$$REG_{HA}(k) = C_{p,HA} + 2(k+1)C_{tx} \quad (2)$$

$$RRD_{RR}(k,l,m) = 2\{C_{p,CN} + C_{p,HA} + (k+m+l+2)C_{tx}\} \quad (3)$$

$$BUD(l) = C_{p,CN} + 2(l+1)C_{tx} \quad (4)$$

7.1.3. TCP Migrate

In Migratory-TCP (M-TCP) handover is initialized by the MN. The MN sends a SYN packet with a Migratory option set, the packet includes the secret token shared by the nodes created during connection setup. The CN responds with a SYN/ACK packet back to the MN the MN then finalizes the handover by acknowledging this packet.

$$HOD_{M-TCP} = C_{p,CN} + 3(l+1)C_{tx} \quad (5)$$

7.1.4. *Socketless TCP*

In SL-TCP the MN just has to send the next acknowledgement (if receiver) or next data packet which is yet to be acknowledged to the CN, for the case whereby the MN is a server, it will have to send a Secure DNS update message to the DNS, this need not be included in the handover process since this process may be performed after handover. Thus the handover delay is proportional to the added processing time at the corresponding node.

$$BUD(l) = C_{p,CN} + 2(l+1)C_{tx} \tag{6}$$

7.1.5. *Analysis*

If we carefully look into equations 1 to 6 we clearly note two things. First, the HOD of SL-TCP and TCP migrate are proportional to the number of hops between the border router and corresponding node. Secondly HOD in MIPv4 is always smaller than the HOD in MIPv6 with route optimization, when l > k; the reason being that, HA registration, RR and BU with CN have to happen sequentially in order to prevent the network from security attacks. Interestingly for an "idealized" version of route optimization, i.e. where the BU can be sent directly to the CN without waiting for the HA registration and RR procedure to be complete; the HOD becomes equivalent to that of SL-TCP i.e. $HOD_{MIPv6} = C_{p,CN} + 2(l+1)C_{tx}$.

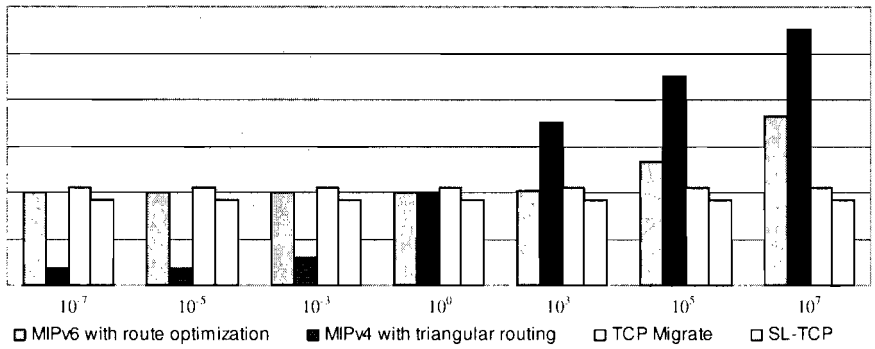


Figure 7-2 Log plot of average (HOD/Cost) over d

Figure 7-2 shows a log plot of the average $\frac{HOD}{Cost}$ over 0 to 1000 hops (l) for several values of the ratio $\frac{k}{l}$. We assumed that $C_{p.CN} = C_{HA} = C_{tx}$ and $\frac{k}{l} = d$. This implies that when d is less than 1 (one), the number of hops between the border router and corresponding node *(l)* is greater than the number of hops between the border router and home agent *(k)*. Likewise, when d is greater than 1 (one), $k > l$. Figure 7-2 shows that network layer mobility schemes perform poorly as the ratio d increases; this is because packets have to be tunneled to the HA during handover procedure. We also observe that the HOD cost remains constant for different values of d in transport layer schemes, as the HOD is only dependent on the distance between MN and CN. When the number of hops between MN and CN equals that of between MN and HA i.e. when d = 1, the HOD of all mobility schemes are approximately equal; this is because both paths are optimal. We observe that for d = 0.001, MIPv4 documents the lowest HOD. This is attributed to the fact that when d falls below 1, this means that $k < l$ i.e. the number of hops between MN and HA is lower than that between MN and CN, hence optimal path for packets is via HA.

7.2. Handover delay – Experimental results

A quantitative measurement of SL-TCP handover latency was performed using our implemented prototype XDP. We measured handovers that occur on the transport layer i.e. the handover latencies in respect to what the application perceives. This is the total of data-link, physical layer handoffs and IP layer handoff in addition to transport layer handoffs. Handover latency is thus characterized as the average time it takes for a mobile node to receive the next packet after changing its IP address.

The XDP testbed introduced in chapter 6 was created to evaluate the performance of the SL-TCP concept under different handover scenarios. The testbed consists of a mobile host (server) and corresponding node (client) with both machines running the Linux operating system. The mobile host is a Pentium III 500 Mhz based desktop running Debian 2.6.9-1.667 kernel, equipped with 802.11b wireless interface and two 100Mbs wired connection. The CN is of a similar make except it is only equipped with one 100Mbs wired connection. The testbed uses XDP to realize connection management in accordance to SL-TCP concept described in Chapter 4 at end hosts. *iptables* [74] and the system calls ifup() and ifdown() [81] were used on the mobile host to manually enforce handover.

7.2.1. Horizontal Handover

Horizontal handover refers to the ability to handover a transport connection session from one access point to another within the same technology [76]. Figure 7-3 shows the evaluation platform setup used to measure these handoffs.

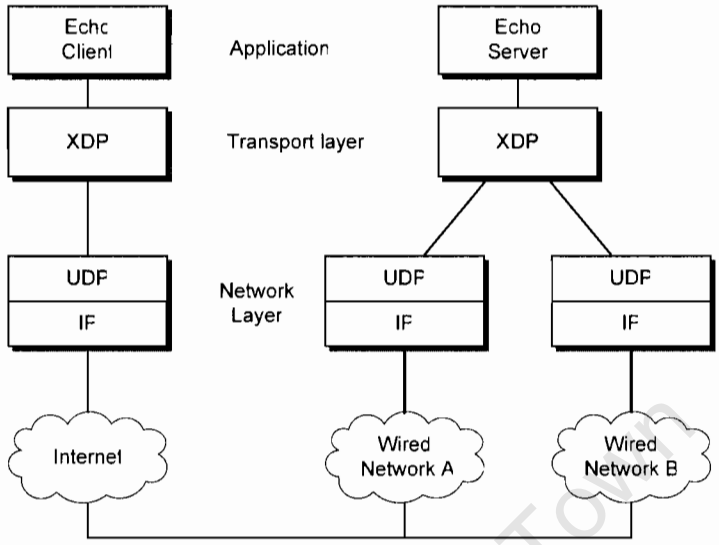


Figure 7-3 Horizontal handoff testbed setup

Both end hosts run XDP, setting the correct values in the MAP. An XDP client is then activated from the corresponding host (client) and sends packets to the mobile host (server) which echo's the packets back. The experiment runs for approximately 4 minutes and approximately after every 10 seconds into the experiment handover is triggered at the mobile host. Handovers were enforced using two techniques 1) changing the default route and hence network interface enforcing a change in IP address and 2) by disabling and enabling network interfaces using *ifup* and *ifdown* system calls, thus forcing packets to leave via another network card. In both scenarios packets where time stamped prior to transmitting and immediately after receiving a reply. The handover latency was then calculated by subtracting the difference of these two and logged into a file. The corresponding handover delays logged using technique 1 are shown in table 7-1.

Table 7-1 Horizontal Handoffs results set 1

	Max (s)	Min (s)	Mean (s)	Std dev
Return time trip	36.99	0.594	0.8016	0.4263
Handover latency	1.172	0.808	0.9709	0.1029

Note that, since we collect the timestamp information in user space, the measured delay actually contains additional overhead introduced by the operating system. Moreover, these results show that the handover latency is approximately equal to the return time trip of the packet, as suggested in our analytical model. In practice however, a number of issues make these results un-attainable. Since handovers were enforced by dynamically changing configuration files within the routing table, the cost of initializing network interfaces and dynamically acquiring IP addresses are not quantified. Another experiment was performed under similar settings but enforcing handovers using technique 2 (enabling and disabling network interfaces after receiving 10 packets). We investigated these latencies under two settings 1) with dynamically acquiring an IP address and 2) a static IP address was issued. The corresponding results are shown in table 7-2.

Table 7-2 Horizontal handoffs result set 2

	Max (s)	Min (s)	Mean (s)	Std dev
Return time trip	0.7155	0.0007	1.2879	0.01738
Handover latency to dynamic IP	6.0141	2.1877	4.0075	1.4751
Handover latency to static IP	2.9843	2.9698	2.9791	0.003

We observe an increase in handover latency. This is attributed to additional operating system and network interface specific tasks during NIC initialization. These tasks are highly dependent on the kernel and equipment used. These costs are very difficult to assess, since datalink and

network information are hidden from us. However, these costs vary slightly during handover procedure, and are not dependent on distance between communicating peers. Hence, the costs may be considered negligible as the distance between end points increases. To quantify the cost of initializing network interfaces during handover, we subtracted the time stamps before and after a network interface is initialized. The corresponding results are shown in table 7-3.

Table 7-3 Interface initialization delays

	Max (s)	Min (s)	Mean (s)	Std dev
Interface Initialization	2.816	2.784	2.8034	0.007

These results suggest that excluding network interface initialization time, attainable handover latencies approach 0.1757sec and 1.2041sec for handing over to static and dynamic IP respectively.

7.2.2. Vertical Handover

Theoretically, the SL-TCP concept does not distinguish between vertical and horizontal handoffs procedures as handovers are represented by a change in IP address only. Moreover, vertical handover costs are subjected more to link layer characteristics which are hidden from the transport layer and hence difficult to assess. We thus focus on assessing how the SL-TCP concept reacts when subjected to vertical handover. The experimental setup of the testbed is shown in Figure 7-4; a similar setting to the horizontal case (refer to Section 7.2.1) but using a wireless and wired interface instead of two wired interfaces.

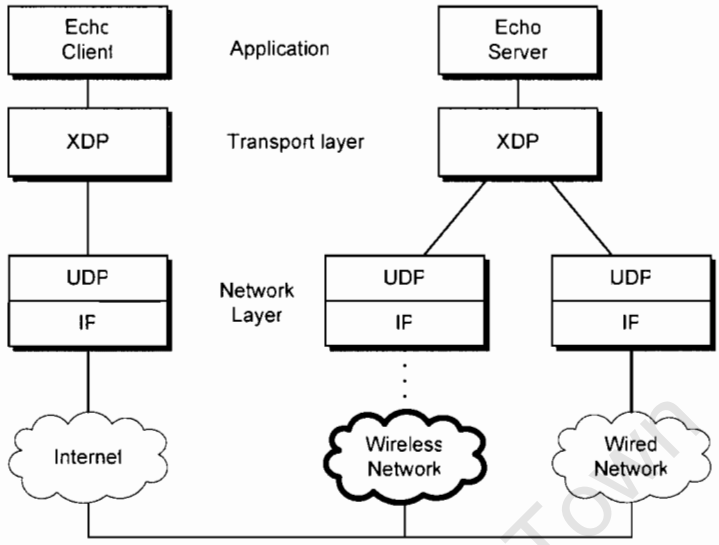


Figure 7-4 Vertical handoffs testbed setup

The experiment tested the handover delay from 802.11b WLAN to wired Ethernet connection and vice versa. This handover is termed 'vertical' due to the need to migrate the existing connection to a different network technology. Here we measured handover latencies for the scenario when a connection is migrating from a wired to wireless access network and vice versa. This was investigated under conditions where the network interface had to acquire an IP address dynamically using DHCP or was manually configured. The experimental procedures and environment are similar to the horizontal handover case in Section 7.2.1. The corresponding results are shown in table 7-4 and 7-5.

Table 7-4 Vertical handoff delays: WLAN - LAN

	Max (s)	Min (s)	Mean (s)	Std dev
Return time trip	0.297	0.1048	0.2068	0.0442
Handover latency to static IP	0.1788	0.1764	0.1769	0.0005
Handover latency to dynamic IP	2.9843	2.9698	2.9791	0.003

Table 7-5 Vertical handoff delays: LAN - WLAN

	Max (s)	Min (s)	Mean (s)	Std dev
Return time trip	0.8931	0.1676	0.3344	0.2109
Handover latency to static IP	0.1883	0.1668	0.1722	0.0005
Handover latency to dynamic IP	23.431	1.4109	4.3819	4.7414

According to the results in Table 7-4, the average handover delay from static IP to dynamic IP address is about 0.1769secs, and the average handover delay from dynamic IP to static IP address is about 2.9791secs. The difference is due to the fact that while switching from dynamic IP address, the MN needs to search the network and negotiate with the DHCP server for a new IP address; whereas in the switch from dynamic IP to static IP address, no new address must be found and the handover delay is much lower. Note that, since we collect the timestamp information in user space, the measured delay actually contains additional overhead introduced by the operating system.

The results also show that the handover latency is approximately equal to that of horizontal handover presented in Section 7.2.1. We also observe that handover latency from LAN to WLAN when the WLAN has to dynamically attain an IP address is higher than its LAN counterpart. This is attributed to the unpredictability of the wireless link, when IP is being requested via DHCP packets some packets get lost and need to be retransmitted.

7.3. Processing costs

Processing cost is difficult to analyze comprehensively and in detail. Processing cost differs from implementation to implementation, and within a given implementation, processing cost is influenced by many factors. There are two basic elements of processing cost; processing speed and

hardware complexity. These two elements can be traded off – reductions in processing speed generally require more complex hardware. In spite of the difficulty of precise and comprehensive analysis of processing cost, we can make a useful generalization. We can make it that processing cost is directly related (as a first approximation) to the number of memory accesses (read and writes) required, by a traditional CPU/memory hardware architecture, to process a packet. This analysis primarily focuses on access to the packet header in particular the connection identifier.

Most of the protocol processing is dealt within the transport layer (assuming no cross layer interactions). Hence, we direct our attention to quantifying power consumed during the process of packet multiplexing and forwarding within the transport layer. We analytically quantify and compare overhead induced using our proposed architecture SL-TCP with MIPv4 and MIPv6. Based on the above observations, we settle here for using a simple count of memory accesses to approximate processing cost for the 3 mobility protocols. We assume a processor with a native word length of 64 bits. Thus, up to 64 bits can be read in a single read. However, we consider reading or writing any single field to be a single read or write, even though multiple such fields might be otherwise accessible with a single 64 bit read. Hence, it will take two memory accesses to compare two words of 64 bits, since only 32 bits can be compared against each other at a time. Table 7-6 shows the corresponding memory access when matching connection identifiers from 3 mobility schemes

Table 7-6 Memory accesses during PCB lookup

	Size of CID	Number of memory accesses
MIPv4	96	3
MIPv6	288	9
SL-TCP	64	2

7.3.1. *Model*

A Transmission Control Protocol (TCP) protocol control block (PCB) contains state information for one endpoint in a given connection. A TCP demultiplexing (a.k.a. PCB-lookup) algorithm must find the PCB corresponding to the connection for each newly arrived TCP packet. The algorithm does this by mapping the packets *Connection Identifier* to the proper PCB. Hence the size of the connection identifier directly influences the *processing cost* during pcb lookup. This cost can be substantial as network activity increases within the terminal.

7.3.2. *Analysis*

Our analysis is based on a simple PCB management scheme, which utilizes an approach of linear linked lists of PCB's. The transport layer protocol searches a simple linear list of PCBs, with a single entry cache containing the PCB last found. Figure 7-5 shows a schematic of this list just after the arrival of a packet for the connection corresponding to PCB 'B'.

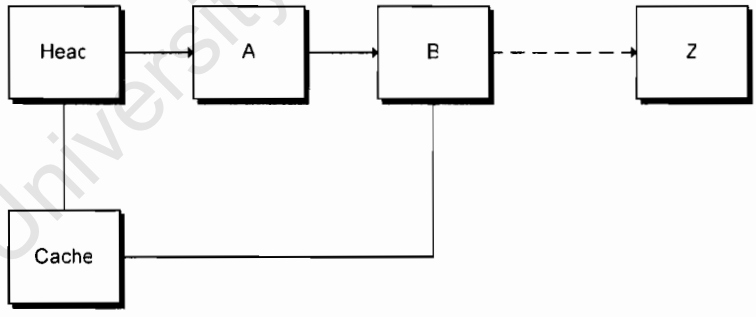


Figure 7-5 PCB management using linear linked lists

The hit rate for the PCB cache is $1/N$ where N is the number of PCBs at a particular point of time. The average number of PCBs that must be examined is just one if we hit the cache and an additional $(N+1)/2$ if we

miss. The probability of a hit is 1/N, hence the probability of a miss is $(N-1)/N$. Thus:

$$Cost = \left(1 + \frac{2N-1}{2N}\right) * Number\ of\ memory\ accesses$$

This equation is used to predict the improvement factor of packet multiplexing and forwarding based on SL-TCP connection identifier over identifiers of other transport layer protocols. Figure 7-6 shows a plot of corresponding processing costs for different mobility protocol schemes over a range of 0 to 2000 concurrent transport connections

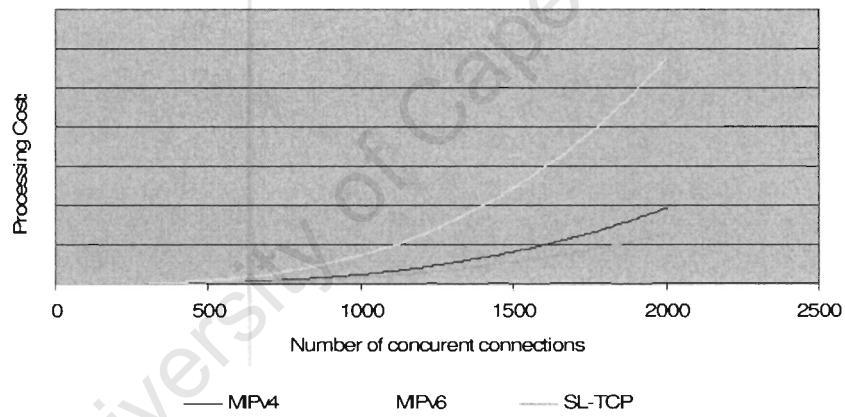


Figure 7-6 Protocol processing costs

As shown in Figure 7-6, SL-TCP induces the least processing overhead as compared to both MIPv4 and MIPv6. As the numbers of concurrent connections increases beyond 1000, we notice in MIPv6 a substantial increase in processing cost. This is attributed to the large size of its connection identifier, in particular the two 128 bit IPv6 addresses. From table 7-6 we observe that in theory SL-TCP reduces processing overhead by $2/3^{rds}$ and $7/9^{th}$ of IPv4 and IPv6 PCB lookup based protocols respectively. With the advent of mobile, battery-operated devices, power

consumption of CPUs become a major determinant of battery life. As mobile devices are getting smaller and smaller, and IPv6 considered the addressing platform of choice, such processing overhead is of major concern.

7.4. Summary

This chapter discussed the evaluation technique, procedure and results of the SL-TCP concept proposed in this study. The analytical model analyzed here verified that this concept indeed produces reduced handover latencies, to values that approximate the return time trip of a packet. This shows that such a concept may accommodate real time applications during an IP address change. These analytical models were backed up by experimental results that show that the handover latencies are approximately equal to the return time trip of the packet. Furthermore, it was shown using a simple analytical model that the proposed concept reduces processing overhead by $2/3^{rds}$ and $7/9^{th}$ of IPv4 and IPv6 PCB lookup based protocols respectively. This shows that SL-TCP concept is theoretically better equipped (compared to current proposals) to support real time applications for miniature devices in mobile environments. This satisfies the two important properties driving and/or defining future applications – *small and mobile*.

Chapter 8

8 SUMMARY AND CONCLUSIONS

This dissertation addresses the problem of having a logical and/or physical dependent transport connection identifier within the current Internet environment. This dependency is highlighted by the complexity and lack of flexibility to provide mobile hosts a continuous transport connection. As we move to a network generation of ubiquitous connectivity, users of mobile devices have come to expect seamless operation on their local applications despite changes in network attachment point (handover). To maximize user experience, they expect their network devices to be intelligent enough to accept packets from all network interfaces in coverage (multi-homing). In addition they expect to have the ability (control) to route traffic of a particular transport connection session from one network terminal to another. This fine grained mobility model is not well supported for network applications by today's operating systems. Furthermore, traditional network connections are unable to survive changes

in attachment points, or use multiple network interfaces. This forces applications to manage all aspects of nomadic computing themselves. Unfortunately for today's user, many popular network applications do not include this functionality.

We have presented *Socketless TCP (SL-TCP)*, a concept that can support fine-grained Internet mobility. Applications can rely on SL-TCP to transparently manage any changes in attachment points, use more than one attachment point at a time or migrate a transport connection to another network application on another device, thus providing mobile users with seamless and continuous connectivity they expect from laptops and PDAs. In addition SL-TCP can be integrated with new set functions that mobile aware applications can use to easily adapt to changes in network conditions. SL-TCP supports a fine-grained mobility model that can significantly maximize user experience for example by migrating a *video stream* transport connection from a PDA to a compatible digital television. Approaching mobility from a fine grained point of view provides more flexibility to the programmer as other forms of mobility are subsets to this viewpoint.

This chapter begins with a summary of our conclusions in Section 8.1. Section 8.2 provides a brief discussion of possible future directions inline with work done on this dissertation. A number of open and unresolved issues not covered in this dissertation are discussed in Section 8.3. Finally, the dissertation closes with some concluding remarks.

8.1. Conclusions

In an attempt to design an efficient, flexible, and easy-to-use architecture to support handover, multi-homing and fine grained mobility, this dissertation draws the following conclusions based on our experiences and evaluation process:

- The reliance of TCP/IP 4-tuple for connection identification burdens the handover mechanisms of mobility schemes. Furthermore, the 4-tuple restricts these schemes to support a granular form of mobility, where transport connections can migrate from one terminal to another. We presented a connection identification scheme that eliminates physical and logical dependencies from the connection identifier, based on a 64 bit CID. This scheme provides support of a more granular form of mobility.
- The use of a 64 bit connection identifier for multiplexing and forwarding datagram's within the kernel has an overhead cost saving on $2/3^{rd}$ and $7/9^{th}$ from IPv4 and IPv6 connection identification schemes. This cost saving is attributed to the reduced number of bits that need to be matched during PCB lookup algorithm. We showed that these cost savings become substantial as the number of concurrent connections increase beyond 1000.
- Handover latencies are directly proportional to the number of signals used in the handover procedure and the time it takes for these signals to be replied. The SL-TCP concept achieve handover with no additional signaling, hence can attain the lowest achievable latencies. We demonstrated this using our prototype implementation.
- We conducted a study of relative handover cost for different mobility schemes based on an analytical model to investigate whether a mobility scheme based on the SL-TCP philosophy could reduce total handover costs. The study showed that while handover costs of network based techniques varied with the number of hops to HA and CN, handover cost of transport layer techniques remain constant. Interestingly, SL-TCP achieved similar handover delay to an ideal model of MIPv6 with route optimization without Return Routability process.
- We evaluated the efficiency of SL-TCP based on a user space implementation prototype XDP (extended datagram protocol). Our

results show that the time taken for the corresponding node to be aware of a change of IP address is approximately equal to the return time trip of the packet. Hence, latencies induced by handover procedures are reduced to normal packet delay. Additional delays are attributed to the process of configuring a new IP address and initializing the network interface. These latencies may be assumed negligible for terminals that do not require network card initialization and use IPv6 auto-configuration to configure IP address.

8.2. Directions for future work

As a result of the scope, limitations, findings and conclusions of this study, the following recommendations for future work involving Socketless TCP are made:

- In this study, handover latencies were only evaluated with an internal network of computers within a lab. As such, further research should be conducted into measuring these latencies in a larger network, with higher packets return time trips. This would provide further insight into the hypothesis/claim that handover latency is approximately equal to the packet return time trip.
- The 64 bit connection identifier has been shown to reduce processing overhead during packet multiplexing and forwarding within the kernel. However, the analysis was restricted to analytical methods based on a very simple BSD lookup algorithm. It is recommended that a more thorough investigation be conducted into other cost savings or earnings that may be attributed to using a 64 bit connection identifier.
- We proposed a security enhancement to SL-TCP architecture based on encryption techniques. However, we neither analyzed implementation techniques nor quantified processing costs induced by these enhancements. Furthermore, the applicability of the proposed scheme

has not been investigated. It is thus recommended that further investigation focusing on the suitability of such a security scheme.

- Our evaluation platform XDP is implemented in user space on top of UDP. The platform experiences numerous known deficiencies as described in Section 6.5. These deficiencies constrained the number of scenario that could be evaluated on the platform. It is recommended that further work should be done to update the evaluation platform and hence pave way for further experimental setups.
- XDP being implemented in user space does not provide a realistic evaluation of SL-TCP. A true evaluation of SL-TCP would include creating a new transport layer protocol within the kernel space on top of IP. This will pave the way for more thorough testing using renowned benchmark tools like netperf [49]

8.3. Open Issues

Despite our best intentions, this dissertation has left several mobility issues unresolved. Within this context it has raised more questions than it addresses. We discuss several of the most interesting and pressing ones below:

- Reachability: The two main issues of mobility are locating the mobile host, and preserving ongoing communications. This dissertation focused on the latter; however we acknowledge that for any complete mobility scheme, issues of reachability must be resolved. Our concept can be easily integrated with schemes that ensure terminal reachability. The most promising ones are MIP home agents [3] and Dynamic DNS [78] with secure binding updates.
- Double Jump Problem: A known problem in SL-TCP and other end to end mobility schemes alike, is that when both end points move simultaneously there is no mechanism for either of them to receive a

binding update from its counterpart. This is because there is no intermediary holding point where packets can be kept and directed. This issue can only be resolved by introducing some form of intelligent routing within the IP framework. We feel that SIP [17] may be used to address this issue.

- NAT: There are NAT [28] implementations that support port number translation [79]. Since port numbers have been excluded from SL-TCP headers, such NAT configurations will not be compatible within the SL-TCP framework. This issue is of great concern due to the widely deployment and dependency of NAT systems in the current Internet framework.
- Stream-less roaming: There is no mechanism to provide support for stream-less roaming within the SL-TCP framework. Consider a portion of data which is on-fly in the network. In case a node migrated from one network to another there is no way for it to receive that on-fly data without intermediate forwarding agent.

8.4. Concluding Remarks

In future networks, the Internet model will be more business oriented. The underlying network and protocols will be oriented to suit this need. Business models will be driven by application innovation. Hence there is a need for protocols to give the programmer and user more control and flexibility over the application functionality. We believe as a first step, this can be realized by enforcing a clear separation between layers of the protocol stack, in particular by abstracting TCP dependency on IP addresses and port numbers – the conceptual core of SL-TCP philosophy.

One desirable consequence of enhancing mobility functionality at endpoints is ease of deployment. Enquiring modifications in the network in order to deploy mobility is still typically more difficult than modifying end

nodes. The counterargument – that many end nodes are now essentially closed boxes which are not updatable and that most users don't want to update them anyway – does not apply to all nodes and users. Even for closed boxes and uninvolved users, downloadable code can provide fast mobility modifications to end systems. Requiring someone to convince ISPs or corporate administrators to modify their network for mobility purpose is much more difficult than simply putting up a Web page with some downloadable software implementing an end-to-end mobility protocol.

In conclusion, we would like to point out that true end-to-end techniques will become indispensable when IPSec [56, 57] or other security mechanisms are employed to encrypt the IP payloads [80]. Drastic changes will be required in the future since the internet was not conceived to support mobility, security, quality of service (QoS), etc. The success of mobility schemes will be measured by mobility applications supported by that particular scheme constrained by its power efficiency. A fine grained mobility scheme with low processing overhead is of utmost importance and Socketless TCP techniques satisfy these criteria.

BIBLIOGRAPHY

[1] P. Bhagwat, C. Perkins, and S. Tripathi. "Network layer mobility: an architecture and survey". *Personal Communications, IEEE*, 3:54–64, June 1996.

[2] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-End Arguments in System Design" MIT laboratory for Computer Science, 1984.

[3] Charles E. Perkins, "RFC320: IP Mobility support for IPv4," Jan. 2002.

[4] D. B. Johnson, C. E. Perkins, J. Arkko, "Mobility Support in IPv6," draft-ietf-mobileipipv6-17.txt, IETF, May 2002

[5] G. Su and J. Nieh, "Mobile communications with Virtual Network Address Translation", Technical Report CUCS-003-02, Dept of Computer Science, Columbia University, February 2002

[6] Gang Wu and Alper Yegin, "Link Layer Triggers for Fast Handovers", IEEE P802.20 submission, Oct 20, 2003

[7] Jon Postel, "Transmission Control Protocol: DARP Internet Program Protocol specification", RFC 793, September 1981

[8] Paresh Jain, Rakesh Kelkar, "Mobile IP: Enabling Mobility for the 3G Wireless", TATA Consultancy services, April 2003

[9] "Host Identity Protocol," http://homebase.htt-consult.com/hip.html.

[10] Jon Postel, "Internet Protocol, DARPA Internet Program Protocol Specifications", RFC 791, September 1981

[11] C. E. Perkins, "Mobile Networking Through Mobile IP," *IEEE Internet Computing*, vol. 2, no. 1, pp. 58–69, January/February 1998.

[12] Jarkko Sevanto, Mika Liljeberg, and Kimmo E. E. Raatikainen, "Introducing quality-of-service and traffic classes into wireless mobile networks," *Proceedings of the 1st ACM international workshop on Wireless mobile multimedia*, Dallas, Texas, pp. 21–

29, 1998.

[13] K. El Malki, "Low latency handoffs in Mobile IPv4." IETF DRAFT, <draft-ietf-mobileip-lowlatency-handoffs-v4-07.txt>, Ericsson, October 2003.

[14] A. Festag, "Optimisation of Handover Performance by Link Layer Triggers in IP-Based Networks: Parameters, Protocol Extensions and APIs for Implementation", Version 1.0, August 2002

[15] S. Aust, D. Proetel, A. Fikouras, C. Gorg, "Policy Based Mobile IP Handoff Decision (POLIMAND) using Generic Link Layer Information", ComNets, Bremen, Germany

[16] G. Huston "A Summary of Proposals to Support Multi-Homing for IPv6", draft-huston-multi6-proposals-00.txt, June 2004

[17] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "Sip: Session initiation protocol," RFC2543, IETF, March 1999.

[18] Daniel Soohong Park, Eric Njedjou, Nicolas Montavont, "L2 Triggers Optimized Mobile IPv6 Vertical Handover", draft-daniel-mip6-optimized-vertical-handover-00.txt, February 2004

[19] Christian Huitema, "Multi-homed TCP" Internet draft, May 1995

[20] Jayanth Mysore and Vaduvur Bharghavan, "A New Multicasting-Based Architecture for Internet Host Mobility," in *Proceedings of ACM/IEEE MobiCom*, 1997.

[21] J. Mysore and B. Vaduvur, "Performance of Transport Protocols Over a Multicasting-based Architecture for Internet Host Mobility," in *IEEE ICC*, 1998.

[22] S. Deering and D. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM TOCS*, May 1990.

[23] Ahmed Helmy, "A Multicast-based Protocol for IP Mobility Support," in *Proceedings of NGC*, 2000.

[24] F. Teraoka et. al., LIN6: A Solution to Mobility and Multi Homing in IPv6, work in progress, Internet Draft draft-ter-aokaipng-lin6-01.txt, 16 August 2001.

[25] M. Crawford et. al., *Separating Identifiers and Locato rs in*

Addresses: An Analysis of the GSE Proposal for IPv6, w ork in progress, Internet Draft (expired), draft-ietf-ipngwg-esd-analysis-05.txt, October 1999, http://www.ie tf. org/proceedings/99nov/I-D/draft-ietf-ipngwgesd-analysis-05.txt

[26] J. D. Wells, 'A Network Mobility Survey and Comparison with a Mobile IP Multiple Home Address Extension' Thesis Virginia Polytechnic Institute and State University July 17, 2003

[27] Wesley M. Eddy, 'At what layer Does Mobility Belong' NASA GRC/Verizon FNS

[28] K. Egevang, P. Francis, "The IP Network Address Translator (NAT)", RFC 1631, May 1994

[29] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet Indirect Infrastructure" In ACM SIGCOMM'02, 2002

[30] Jin Hyeock Choi , "Fast Router Discovery with AP Notification", draft-jinchoi-l2trigger-fastrd-01.txt, June 2002

[31] Kamel Baba et al. "Fast Handoff L2 Trigger API", draft-singh-l2trigger-api-00.txt, September 2002

[32] S. Deering, R. Hinden "Internet Protocol, Verstion 6 (IPv6) Specifications", RFC 2460, December 1998

[33] SHOCH, J. "Inter-network naming, addressing, and routing", In Proceedings of IEEE Computer Conference, COMPCON (Washington, D.C., Fall). IEEE, New York, 1978, pp. 72-79.

[34] SALTZER, J. On the naming and binding of network destinations. In Proceedings of IFIP/WG6.4 Symposium on Local Computer Networks (Florence, Italy, Apr.). North-Holland, Amsterdam, 1982, pp. 311-317.

[35] P. Francis and R. Gammadi. "IPNL: A NAT-extended Internet Architecture", *Proc. ACM SIGCOMM 2001*, pp 69-80, 2001.

[36] Shelley Zhuang, Kevin Lai, Ion Stoica, Randy Katz, and Scott Shenker, "Host Mobility Using an Internet Indirection Infrastructure", 2002.

[37] SCTP RFC: http://www.ietf.org/rfc/rfc2960.txt

[38] Randall Stewart and Qiaobing Xie, "Stream Control Transmission Protocol(SCTP): A Reference", Publisher: Addison-Wesley

[39] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: Session initiation protocol," Internet Draft, Internet Engineering Task Force, Nov. 2000. Work in progress.

[40] Ying-Hong Wang, Chien-Wen Wang and Hui-Min Huang, "A Novel micro-mobility management method in wireless communication network", in proceedings ICPADS'05 pp. 133-139

[41] W. Diffie and M. E. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, 22(6); pg 644-654, November 1976

[42] Y. Hsieh,R. Sivakumar. "A Transport Layer Approach for Achieving Aggregate Bandwidths on Multi-homed Mobile Hosts", Georgia Institute of Technology.

[43] http://www.washington.edu/hdtv/ 'internet hdtv'

[44] Gong Su and Jason Nieh "Mobile Communication with Virtual Network Address Translation", Columbia University,

[45] Snoeren, A. C. "*A Session-Based Architecture for Internet Mobility*". PhD thesis, Massachusetts Institute of Technology, Dec. 2002.

[46] A.C. Snoeren and H. Balakrishnan. "TCP Connection Migration" http://nms.lcs.mit.edu/papers/draftsnoeren-tcp-migrate-00.txt, November 2000. Internet Draft, work in progress, expired.

[47] P. Francis, "Addressing in Internetwork Protocols", PHD Thesis, University College London, 1994

[48] B. Carpenter "Architectural Principles of the Internet", Network Working Group, RFC 1958, June 1996.

[49] R. Jones, "*Netperf: a Network Performance Benchmark*", Information Networks Division, Hewlett-Packard Company, February 1996. http://www.netperf.org/netperf/NetperfPage.html

[50] Network Working Group "Real Specific IP for End to End IPSec" RFC 3104.

[51] Alex C. Snoeren, Hari Balakrishnan, and M. Frans Kaashoek. "Reconsidering Internet Mobility", Proc. of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII), May 2001

[52] Network Working Group "Realm Specific IP: Protocol Specifications" RFC 3103

[53] Borko Furth, Mohammad Ilyas, "Wireless Internet Handbook", CRC PRESS, 2004.

[54] A. Dutta, R. Jain, K. D. Wong, J. Burns, K. Young and H. Schulzrinne, "Multilayered Mobility Management for Survivable Network", to appear, *Proceedings of Milcom*, October 2001.

[55] Q. Wang, M. A. Abu-Rgheff, "Integrated Mobile IP and SIP approach for Advanced Location Management", London Communications Symposium, 2002.

[56] Kent, S., and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[57] Kent, S., and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[58] Cerf, V.G., and Kahn, R.E. "A protocol for packet network intercommunication" IEEE Trans. Comm. COMM- 22, 5 (May 1974), 637-648

[59] Lawton George. "Is IPv6 Finally Gaining Ground?" *Computer Innovative Technology Computer Professionals.* IEEE Computer Society. August 2001.

[60] A. Perrig, D. Clark, and S. Bellovin, Editors. "Secure Next Generation Internet," NSF Workshop Report, July 2005 (pending).

[61] Dave Clark, et. al. "Making the world (of communication) a different place," Report of the End-to-end Research Group. January 2005.

[62] Cerf, V. (1999) "The Internet is for Everyone", *On the Internet*, July- August: 8-9.

[63] Cerf, V. G., and Cain, E. "The DoD Internet Architecture model",

Computer Networks 7 (Oct. 1983), 307-318

[64] Global Environment for Network Investigations, http://www.geni.net/

[65] Geoff Houston, 'a survey of internet identities', The ISP Column, The Internet Society, June 2003

[66] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[67] J. Postel, J. Reynolds, "File Transfer Protocol (FTP)" RFC 959, October 1985

[68] L. Peterson. "A Strategy for Continually Reinventing the Internet," Office of Science and Technology Policy. May 2005.

[69] W.R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, 1994.

[70] Qi Wang and Mosa Ali Abu-Rgheff, "A Multi-Layer Mobility Management Architecture Using Cross-Layer Signaling Interactions" Mobile Communications Research Group, University of Plymouth

[71] Joseph T. Dixon and Kenneth L. Calvert, "Increasing Demultiplexing Efficiency in TCP/IP Network Servers", Georgia Institute of Technology

[72] Wikipedia "Diffie Hellman key exchange" http://en.wikipedia.org/wiki/Diffie-Hellman Nov 2005

[73] N. M. Josuttis. "*The C++ Standard Library: A Tutorial and Reference"* Addison-Wesley, 1999.

[74] Rusty Russell, "Linux iptables HOWTO", URL: http://netfilter.samba.org

[75] L. Garber. "Denial-of-Service Attacks Rip the Internet", IEEE Computer, vol. 33(4):pp. 12-17, April 2000.

[76] S. Galli, R. Morera. "An Analytical Approach to the Performance Evaluation of Mobility Protocols: The handoff Delay Case" Telcordia Technologies, 2004

[77] Zaga Novakovic "Roaming and WLAN" http://www.mobilein.com/perspectives/Authors/ZagaWLANRoaming.htm, May 2004

[78] Eastlake, 3rd, D. E. "Secure domain name system dynamic update", RFC 2267, IETF, Jan 1998

[79] Y. Takeda. "Symmetric NAT Traversal using STUN". Internet draft, June 2003

[80] Tom Goff, James Moronski "Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments"

[81] R. W. Smith. "Advanced Linux Networking", Addison Wesley, June 2002

APPENDIX

A. ACCOMPANYING CD

- This manuscript in pdf format
- Cited references in pdf format
- XDP Source Code
- Experimental results