# Micro-Mobility Management and Freeze-TCP Implementation in a Wireless Network

A thesis
submitted in partial fulfilment
of the requirements for the degree
of

Master of Engineering
in
Electrical and Computer Engineering

by

Francis Jing Jyi Chai

Supervisor
Associate Professor Harsha R. Sirisena

Department of Electrical and Computer Engineering
University of Canterbury,
Christchurch, New Zealand


January 2004

# Abstract

With wireless products becoming cheaper, there has been an increased number of mobile users accessing to the Internet wirelessly. With the growing desire to be always on-line, there is a requirement for mobility management of mobile terminals between homogeneous systems, such as access to the wireless LAN in the office or during meeting. Two main issues of research reported in this thesis are related with building a working micro-mobility testbed and an enhancement to the current standard TCP to improve its performance over a micro-mobile wireless network.

A survey of recently proposed micro-mobility protocols is presented, highlighting the protocol features that providing mobility management support to mobile nodes across the Internet. Also a detailed procedure is presented of how a successful micro-mobility testbed can be built, based on the Cellular IP protocol. The results show that the Cellular IP mobile host successfully performs seamless handoffs with no packet loss and maintains connectivity while running a video session over wireless interfaces.

Lastly, the thesis presents an investigation of the current problems faced by standard TCP in the wireless environment and describes recently proposed TCP enhancement schemes for wired-wireless networks. Freeze-TCP has been chosen to be implemented into the micro-mobility testbed to interwork with Cellular IP that replaces standard TCP to enhance TCP performance during handoffs. The results show that Freeze-TCP provides up to 41% improvement in TCP throughput compared to standard TCP.

# Acknowledgement

I would like to express my sincere thanks and gratitude to my supervisor, Associate Professor Harsha Sirisena for his much guidance and support during the time of my research. My thanks to Associate Professor Krzysztof Pawlikowski for his guidance and help in proof-reading my thesis.

Many thanks to Foundation for Research Science & Technology and Allied Telesyn Research Ltd for funding my research. I would like to thank Allied Telesyn Research Ltd and Electrical & Computer Engineering Department for providing the hardwares and other necessary equipment to build my wireless testbed and without them it would not have been possible.

Special thanks to Sean Lin and Dave van Leeuwen for their help and advices in setting up and debugging my testbed. To my colleagues in the Network Research Group, I like to thank you all for the times we spend together in sharing knowledge in all the meetings and conferences that I've attended.

To my fiancée and my parents, thank you for the love, support and prayers throughout this couple of years that encourage me not to give up and do my best to finish my thesis. Lastly, thank you God for Your love and grace being with me all this time (2Cor 13:14).

# Table of Contents

# List of Figures

# List of Tables

# 1.0 Introduction

The number of hosts connected to the Internet is growing at an increasing speed and in the future it is expected that most of these devices will be wireless and hence mobile in nature. Due to the increasing use of notebooks, portable computers and cellular phones, there has been an increasing demand for accessing the Internet wirelessly. There are, however, some problems that have to be solved before mobile access to the Internet can become widespread. One big problem is the way the Internet Protocol (IP) routes packets to their destinations according to their IP addresses. The following sections provide a description of the problem and its solution.

## 1.1 TCP/IP Protocol Suite

Computers communicate with each other under a set of rules, called a protocol. A computer network is a collection of computers with the support of one or a few communication protocols. The Internet is a wide area network of computers that spans the globe. Each computer of this network represents an Internet node. What supports the existing Internet is a collection of protocols based on the TCP/IP protocol suite with its core protocols, Transmission Control Protocol (TCP) and Internet Protocol (IP). TCP/IP is normally considered to be a 4-layer hierarchy, with each layer responsible for a specific task as shown in Fig. 1.1.1.

| | |
|---|---|
| Applications | Telnet, FTP, SMTP, etc |
| Transport | TCP, UDP, SCTP |
| Network | IP, ICMP, RIP, OSPF, BGP |
| Link | Device driver and Interface card |

*Figure 1.1.1. TCP/IP Protocol Suite.*

The application layer handles the details of the particular application. There are many common TCP/IP applications that almost every implementation provides: Telnet (remote login), FTP (File Transfer Protocol), SMTP (Simple Mail Transfer Protocol), SNMP (Simple Network Management Protocol) and so on.

The role of transport layer is to provide a flow of data between two Internet nodes. It accepts data from the application layer above, splits it up into smaller units if necessary, passes these to the network layer, and has it delivered to the other end. In the TCP/IP protocol suite, there are two different transport layer protocols: TCP and UDP (User Datagram Protocol). TCP provides reliable flow of data between two nodes while UDP sends data packets called datagrams, from one node to another without any guarantee that the datagrams reach the other end.

The network layer, sometimes called the IP layer, handles the movement of packets around the network. This layer includes IP, ICMP (Internet Control Message Protocol) and IGMP (Internet Group Management Protocol), with IP as the core protocol, supported by ICMP and IGMP as auxiliary protocols.

The link layer is intended to take a raw data transmission and to provide a service free of transmission errors to the network layer. Normally it includes the device driver in the operating system and the corresponding network card in the computer. An important link layer protocol in the protocol suite is ARP (Address Resolution Protocol). Every node has an address used by the link layer, called the link layer address (or Medium Access Control address for Ethernet link). ARP mutually relates the link layer address and an address used by the network layer, called an IP address.

## 1.2 Mobility Problem in IP Networks

When IP routing was originally defined, mobility of hosts was not considered to be an issue. Routing methods were built for static networks, where the hosts were unlikely to move from one subnet to another. In standard IP, it is assumed that a node’s IP address uniquely identifies the node’s point of attachment to the Internet.

Thus, if a mobile computer, or mobile host, moves from one sub-network to another while keeping its IP address unchanged, its address will not reflect the new point of attachment. Consequently, existing routing protocols will be unable to route datagrams to it correctly. In this situation, the mobile host must be reconfigured with a different IP address, representative for its new location. Not only is this process

cumbersome for operators of mobile hosts but it also presents the problem of informing potential correspondents of their new addresses. Furthermore, changing the IP address will cause already-established transport layer connections (for example, ftp or telnet sessions) to be lost. Another method for a node to change its point of attachment without losing its ability to communicate would be to use Host-specific routing. But this has an obvious disadvantage of severe scaling problems, especially relevant in light of the explosive growth in number of mobile computers.

Under the current Internet Protocol, if the mobile host moves without changing its address, it will lose routing. On the other hand, if it does change its address, it will lose already established connections. So, a new protocol for supporting mobile computers is required.

## 1.3 Overview of Mobile IP

Mobile-IP [1] is an enhancement to IP, which allows a computer to roam freely on the Internet while still maintaining the same IP address. It was produced by the IP Routing for Wireless/Mobile Hosts working group of the Internet Engineering Task Force (IETF), which was formed in June 1992, and it was approved by the Internet Engineering Steering Group (IESG) in June 1996, and published as a proposed standard in November 1996. It is a network layer solution for mobility within the global Internet which is scalable, robust, secure, and which allows nodes to maintain all ongoing communications while changing links. It provides a mechanism for routing of IP packets to mobile nodes, which may be connected to any link while using their permanent IP address. Some special entities, called the Home Agent (HA) and Foreign Agent (FA), are needed to provide mobility services in IPv4. The HA and FA provide certain services, which enable the Mobile Host (MH) to move around without changing its IP address.

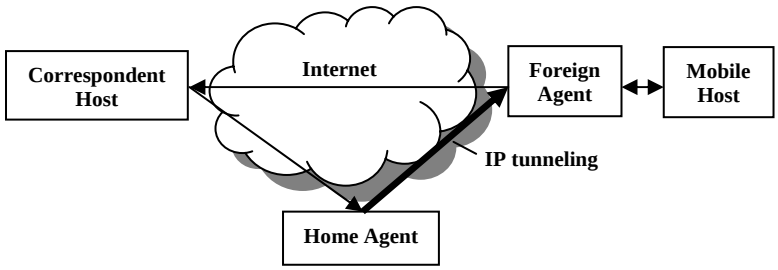

*Figure 1.3.1. Working of Mobile IP in MH, FA and CH.*

Fig. 1.3.1. depicts the working of Mobile IP. HA is a router on a MH's home network which is responsible for tunneling IP datagrams for delivery to the MH while it is away from home, and maintains current location information for the MH. FA is a router on a mobile node's visited network which provides routing services to the mobile node while registered. The FA detunnels and delivers datagrams to the MH that was tunneled by the MH's HA. When MH is away from home, it registers a new care-of address with HA through exchange of a Registration Request and Registration Reply message via a FA. The HA intercepts IP packets meant for the MH and tunnels them to the registered COA. Tunneling refers to the process of enclosing the original datagram, as data, inside another datagram with a new IP header. The destination field in the outer IP header contains the COA – a topologically significant address to which standard IP routing mechanisms can deliver packets. The COA may belong to a specially designated node, a FA, or may be acquired (perhaps temporarily) by the MH, e.g. through DHCP or PPP. At the endpoint of the tunnel, the outer IP header is removed to recover the original IP packet, which is then delivered to the MH. The routing of datagrams between CH to HA to FA is known as the triangle routing.

## 1.4 Mobile IP enhancements

Mobile IP's routing of all incoming packets via the home network may cause additional delays and waste of bandwidth capacity. However if the CH has knowledge of where the MH is, it can send packets directly to the COA of the MH. This is achieved by route optimization [2], which basically provides mobility binding updates to the CH. This solves the problem of triangle routing. Binding updates are send from the HA upon request, or can be sent upon receiving a warning message from a FA if the MH changes location during a communication session.

Outgoing packets from the MH to a CH do not have to be routed via the MH's home network. However, if the sender's home IP address contained in the outgoing packets does not match the network the user is actually in (i.e., a topologically incorrect address), then firewalls may reject to forward such packets. To avoid this and due to some other considerations, reverse tunneling is defined as an extension to Mobile IP

[3]. With reverse tunneling, all packets from a MH go through the HA before reaching the CH. Another optimization proposed is regional registration [4], which suggests registration within a visited domain. In the base Mobile IP, a MH is required to register with its HA each time it changes COA, thus causing signaling delay for the registration if the MH is far away from its HA. Regional registration attempts to decrease the number of registrations by creating a hierarchical structure of FAs. As long as a MH's FA is hierarchically under a so-called gateway foreign agent (GFA), it is unnecessary to relay registration messages to the HA since the HA has already registered the GFA's address as the COA. One drawback of this approach is the reliability issue; failure of a GFA will bring down the whole hierarchy.

The base Mobile IP requires that a MH register with a FA, and subsequently with its HA. To make Mobile IP handoffs (i.e., the registration process) more suitable for real-time and delay-sensitive applications, two methods called the Network Assisted, Mobile and Network Controlled (NAMONC) handoff method and Network Initiated Mobile Terminated (NIMOT) handoff method have been proposed. In NAMONC the MH is informed (assisted) by the network that a Layer two handoff is anticipated. It proposes to use simultaneous bindings (multiple registrations at a time) in order to send multiple copies of the traffic to potential movement locations before actual movement. The other method, NIMOT handoff, proposes extensions to the base Mobile IP so that FAs can utilize information from Layer two. Specifically, foreign agents use Layer two triggers to initiate a pre-registration prior to receiving a formal registration request from the MH. Both methods assume considerable involvement of information from Layer two.

## 1.5 Differences between Macro-Mobility and Micro-Mobility

Macro mobility [5], also known as global mobility, provides mobility over a large area. This includes mobility support and associated address registration procedures that are needed when a mobile host moves between IP domains. Inter-access network handovers typically involve macro-mobility protocols. Mobile IP can be seen as a means to provide macro mobility.

Micro mobility [5], also known as local mobility, provides mobility over a small area. Usually this means mobility within an IP domain with an emphasis on support for active mode using handover, although it may include idle mode procedures as well. Micro-mobility protocols exploit the locality of movement by confining movement related changes and signaling to the access network

Mobile IP suffers from several well known weaknesses that have led to the macro/micro-mobility approach. In the base Mobile IP, the basic mobility management procedure is composed of two parts: (i) the movement detection by the MH and (ii) the registration to the HA. Every time the MH changes its point of attachment, it must perform these two steps to continue to receive packets. However, it is the MH that initiates the process by sending the registration request after it has detected that it has moved from one network to another and has obtained a new COA. This introduces two causes of latency:

- movement in detection latency,
- registration latency.

Movement in detection latency is the time required by the MH to detect changes in its point of attachment. It can be large since the move detection mechanisms in Mobile IP are based on either the expiration of the lifetime indicated in the FA agent advertisements or on the comparison of the address prefix of the two different agent advertisements. Registration latency is the required time to complete the registration with the HA. As the HA can be located anywhere on the Internet, this process can take a long time and sometimes it can be even impossible to complete it.

Those latencies, due to the properties of Mobile IP, introduce a time interval within which the packets destined to the MH cannot be routed to their new location. During this time, the mobile is already connected to its new point of attachment, but the packets that are sent to it are routed to its old point of attachment. These packets are thus lost for the MH. In the case of a quickly moving mobile, the registration process may become inefficient. Moreover, this mechanism produces a large volume of control traffic inside the local domain, and across the Internet as the number of mobile users increases.

The micro-mobility approach tries to reduce the latency of handover management. This approach does not always reduce the control traffic, but it results in a reduction of the number of network stations that process the control packets by restricting the propagation of those packets to a smaller set of stations. In the next section, a number of micro-mobility protocols will be discussed. They are designed for environments where mobile hosts frequently change their points of attachment within the domain to handle real-time multimedia wireless applications with minimal packet delays, packet losses and signal overheads over wireless interfaces.

## 1.6 Goals of the Research

The main goals of research reported in this thesis are as follows:

- to survey recently proposed IP micro-mobility management protocols;
- to develop and test a working prototype of micro-mobility management across a wired-wireless network with notebook PCs as the mobile terminals, together with a micro-mobility protocol implemented in the network and terminals;
- to survey TCP schemes proposed for improving TCP performance in wired-wireless environment;
- to implement an enhancement scheme for TCP proposed for improving performance over micro-mobility management in the working prototype.

## 1.7 Outline of Thesis

Chapter 2 surveys seven recently proposed IP Micro-mobility protocols.

Chapter 3 presents the implementation of Cellular IP protocol for a working prototype of micro-mobility management scheme across a wired-wireless local area network.

Chapter 4 surveys TCP schemes proposed for improving TCP performance over wired-wireless network.

Chapter 5 presents the implementation of Freeze-TCP with Cellular IP protocol for improving TCP performance in the working prototype over wired-wireless network.

Chapter 6 concludes the thesis and suggests further possible enhancements of the prototype testbed.

# 2.0 Survey of IP Micro-Mobility Protocols

Over the past several years a number of Micro-Mobility protocols have been proposed, designed and implemented that complement the base Mobile IP protocol by providing fast, seamless and local handoff control. The development of these protocols has generated considerable interest in industry and academia, and resulted in the ongoing standardization efforts within the IETF Mobile IP [6] and Seamoby [7] Working Groups on low latency handoff [8] and IP paging [9], respectively. This section will take a look at seven recently proposed micro-mobility protocols: Cellular IP, HAWAII, Hierarchical Mobile IP, Proactive Handoff, Fast Handoff, EMA and TeleMIP.

## 2.1 Cellular IP

Cellular IP [10, 11, 12] is a lightweight, robust, host mobility protocol that is optimised to support micro-mobility and frequently migrating hosts but efficiently interworks with Mobile IP to provide wire area mobility. There are four fundamental design principles of the protocol:

1. Location information is stored in distributed databases.
2. Location information referring to a mobile host is created and updated by regular IP datagrams originated from the mobile host
3. Location information is stored as soft state
4. Location management for idle mobile hosts is separated from location management of hosts that are actively transmitting or receiving data.

Cellular IP inherits cellular principles for mobility management such as passive connectivity, paging and fast handoff control, but implements them around the IP paradigm. In summary five key requirements of wireless access networks motivate the design of the Cellular IP protocol:

1. Easy global migration
2. Cheap passive connectivity

3. Flexible handoff support
4. Efficient location management
5. Simple memoryless mobile host behaviour

### 2.1.1 Network Model

The network architectural entities of Cellular IP access networks consists of cellular IP node, gateway, base station and mobile host.

- Cellular IP node – They are responsible for routing IP packets inside the Cellular IP network and communicate with mobile Hosts via a wireless interface. Cellular IP node that has a wireless interface is also called a BS.
- Cellular IP gateway – is a Cellular IP node that is connected to a regular IP network by at least one of its interfaces.
- Cellular IP Base Station – serves as a wireless access point and router of IP packets while performing all mobility-related functions. They are built on regular IP forwarding engine with the exception that IP routing is replaced by cellular IP routing and location management.
- Cellular IP Mobile Host – is a mobile device (i.e., laptop) that implements the Cellular IP protocol.

As shown in Fig. 2.1.1 Cellular IP access networks are connected to the Internet via Cellular IP gateway. Mobility between gateways is managed by Mobile IP while mobility within the access networks is handled solely by Cellular IP. BSs within the cellular IP access network periodically emit beacon signals. Mobile host use the beacon signals to locate the nearest BS and attached to the network using the IP address of the gateway as its Mobile IP care-of address. Every beacon signal contains layer two parameters related to BS, Cellular IP network identifier, gateway IP address and ID of the paging area. Assuming Mobile IP and no route optimization, packets will be first routed to the host's HA and then tunneled to the gateway. The gateway detunnels packets and forwards them toward a base station. Inside a Cellular IP network, mobile hosts are identified by their home address, and data packets are routed without tunneling or address conversion. The Cellular IP routing protocol

ensures that packets are delivered to the host's actual location. Packets transmitted by mobile hosts are first routed toward the gateway and from there on to the internet.

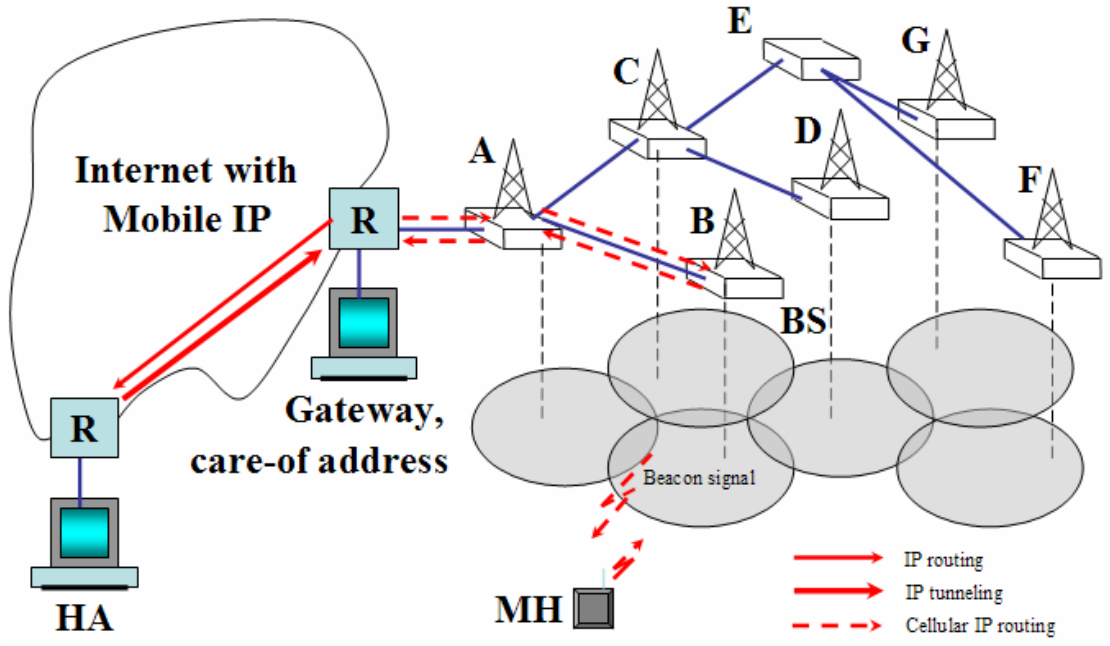


*Figure 2.1.1. Cellular IP access Network*

### 2.1.2 Cellular IP routing

In Cellular IP, location management and handoff support are integrated with routing. To minimise control messaging, regular data packets transmitted by mobile hosts are used to refresh host location information. Uplink packets are routed from a mobile host to the gateway on a hop-by-hop shortest path routing. The path taken by these packets is cached by all intermediate base stations. To route downlink packets addressed to a mobile host, the path used by recently transmitted packets from the mobile host is reversed. When the mobile hosts has no data to transmit, it sends small, special IP packets toward the gateway to maintain its downlink routing state. Following the principle of passive connectivity, mobile hosts that have not received packets for a certain period of time allow their downlink soft-state routes to timeout and be cleared from routing cache. Paging is used to route packets to idle mobile hosts in a cellular IP access network.

Cellular IP uses two parallel cache systems to store the information related to the location of mobile hosts. Nodes maintain one set of mappings, called Paging Caches

(PC), for idle mobile hosts. These mappings have a timeout interval comparable to the migration frequency, possibly in the order of seconds or minutes. Modes maintain another set of mappings, called Routing Caches (RC) for active mobile hosts. These mapping are only maintained for mobile hosts currently receiving or expecting to receive data. For routing cache mappings the timeout can be in the packet time scale. Fig. 2.1.2 shows the state diagram for Cellular IP of sending paging-update when the mobile host is idle and sending routing-update when the mobile host is active.

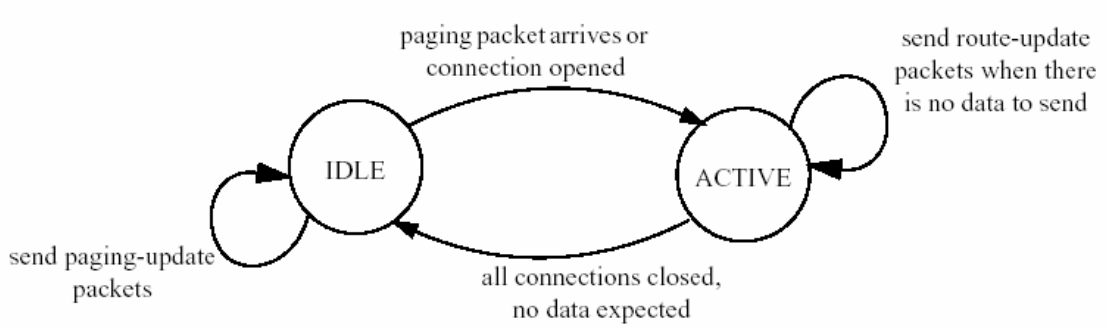


*Figure 2.1.2. State diagram for Cellular IP.*

Using Fig. 2.1.1 as our example, every packet that is transmitted by mobile host is routed to the gateway using shortest path hop-by-hop routing. Each Cellular IP node monitors the passing data packets and uses them to create and update route cache mappings. When a data packet originated by a mobile host enters a base station, the routing cache mapping stores the IP address of the source mobile host and the interface over which the packet entered the node (IP Address, Interface Name). To keep its routing cache mappings valid for a system specific route-timeout, the mobile host transmits route-update packets at regular intervals called route-update time, traverses the same interface coming from the same mobile host. These packets are empty data packets addressed to the gateway. Route-update packets have the same effect on routing cache as normal data packets; however, they do not leave Cellular IP access network.

### 2.1.3 Cellular IP Handoff

Handoff in Cellular IP is always initiated by the mobile host. As MH approaches a new BS, it transmits a route-update packet and redirects its packets from the old to the new BS. The route-update packet will automatically configure a new path of routing

cache mappings for the MH to the new BS. (The paths leading to the old and new BS may overlap). Fig. 2.1.3 illustrates a handoff scenario.

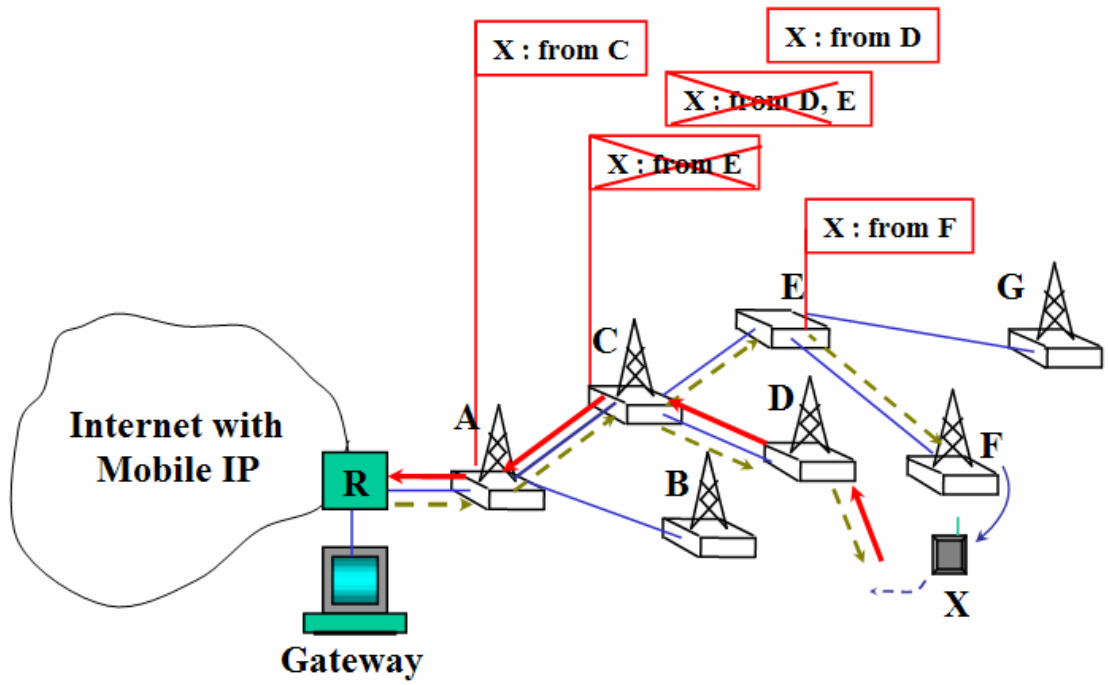


*Figure 2.1.3. A Handoff Scenario*

The mobile host X is moving from cell F to D while it is sending and receiving data packets. Assuming that all nodes having routing caches, before the move the routing cache mappings are as indicated in the flags. After the migration, the mobile-originated data packets, indicated by solid arrows, cause the cache in node C to create a new mapping. For sometime, packets addressed to X are delivered both to D and to F as shown by the dotted arrows. After the route cache timeout, the old mapping to E is cleared, as is the mapping to F in E. From then on, only the up-to-date route is used. The routing cache in A is unchanged during the whole process.

Cellular IP supports two types of handoff scheme. Cellular IP hard handoff is based on a simple approach that trades off some packet loss for minimizing handoff signalling rather than trying to guarantee zero packet loss. Cellular IP semisoft handoff exploits the notion that some mobile hosts can simultaneously receive packets from the new and old BSs during handoff. Semisoft handoff minimises packet loss, providing improved TCP and UDP performance over hard handoff.

### 2.1.4 Hard Handoff

Cellular IP hard handoff algorithm is based on a simple approach to mobility management that supports fast handoff at the price of potentially some packet loss. Handoff is initiated by mobile hosts in a Cellular IP access network. MHs listen to beacons transmitted by BSs and initiate handoff based on Signal strength measurements. To perform a handoff, a mobile host tunes its radio to a new BS and sends a route-update packet. The route-update message creates routing cache mappings en route to the gateway configuring the downlink route cache to point toward the new BS. Handoff latency is the time that elapses between handoff initiation and the arrival of the first packet along the new route. In the case of hard handoff this is equal to the round-trip time between the mobile host and the crossover base station as illustrated in Fig. 2.1.4. The crossover base station is defined as the common branch node between the old and new BSs. In the worst case the crossover point is the gateway. During this interval, downlink packets may be lost. Mappings associated with the old BS are not cleared when handoff is initiated. Rather, mappings between the crossover node and the old BS timeout are removed. No packets are transmitted along the old path once route-update message has created a new mapping at the crossover base station that points toward the new base station.

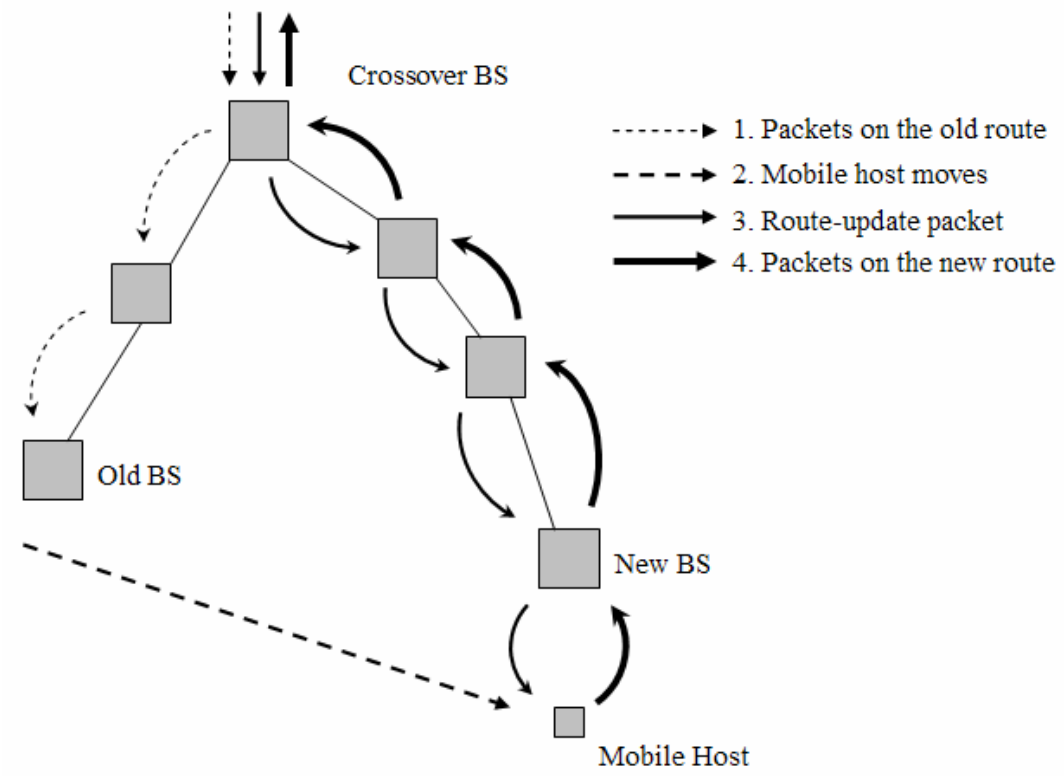


*Figure 2.1.4. Cellular IP handoff*

Although packets may get lost during a hard handoff, the time taken to redirect packets to the new point of attachment is shorter than that in Mobile IP. This is due to the fact that only a local node has to be notified rather than a possibly distant HA in the case of Mobile IP.

### 2.1.5 Semisoft Handoff

Before the mappings timeout, a period exists when both the old and new downlink routes are valid and packets are delivered through both base stations. Semisoft handoff algorithm improves the handoff performance while maintaining the lightweight nature of the base protocol providing probabilistic guarantees instead of fully eliminating packet loss. Semisoft handoff adds one additional state variable to the existing mobile state maintained at mobile hosts and base stations. The semisoft handoff procedure has two components. First, in order to reduce handoff latency, the routing cache mappings associated with the new BS must be created before the actual handoff takes place. When the MH initiates a handoff, it sends a semisoft packet to the new BS and immediately returns to listening to the old BS. While the MH is still in contact with the old BS, the semisoft packet configures routing cache mappings associated with the new BS. After a semisoft delay, the MH performs a regular handoff. The semisoft delay can be an arbitrary value between the mobile-to-gateway round-trip delay. The delay ensures that by the time the MH finally tunes its radio to the new BS, its downlink packets are delivered through both the old and new BSs. Hence the downlink packets consume twice the amount of resources during this period.

While the semisoft packets ensures that MHs continue to receive packets immediately after handoff, it does not, however, ensure smooth handoff between BSs. Depending on the network topology and traffic conditions, the time to transmit packets from the crossover point to the old and new BSs may be different and the packet streams transmitted through the two BSs will typically unsynchronised at the MH. If the new BS “lags behind” the old BS, the MH may receive duplicate packets, which does not disrupt many applications. For example, TCP will not be forced into slow start due to

the arrival of duplicate acknowledgements. If the new BS “gets ahead” then packets will be deemed to be missing from the data stream observed at the receiving MH.

The second component of the semisoft handoff resolves the issue of the new BS getting ahead. The solution to this problem is based on the observation that perfect synchronisation of packet streams is unnecessary. This condition can be eliminated by temporarily introducing a constant delay along the new path between the crossover and new BSs using a simple delay device mechanism. The device needs to provide sufficient delay to compensate, with high probability, for the time difference between the two streams travelling on the old and new paths. Optimally, the device delay should be located at the crossover base station that understands that a semisoft handoff is in progress due to the fact that a semisoft packet has arrived from a MH that has a mapping to another interface. The mapping created by the semisoft packet has a flag to indicate that downlink packets routed by this mapping must pass a “delay device” before being forwarded for transmission along the new path. After handoff is complete, the MH sends a data or route-update packet along the new path which will clear this flag and cause all packets in the delay device to be forwarded to the MH. BSs only need a small pool of delay buffers to resolve this issue. Packets that cannot sustain additional delay can be forwarded without passing through the delay device.

### 2.1.6 Paging

Typically, fixed host connected to the Internet (e.g., desktop computers) remain online for extended periods of time, even though most of the time they do not communicate. Being always connected in this manner results in being reachable around the clock with instant access to Internet resources. Mobile subscribers connected to the wireless Internet will expect similar service. However, in the case of mobile hosts maintaining location information to support being continuously reachable would require frequent location updates which would consume precious bandwidth and battery power.

The definition of an idle mobile device is well understood in the context of cellular systems, which are connection-oriented in nature, its meaning in IP-based mobile

networks is unclear. Cellular IP defines an idle MH as one that has not received data packets for a system specific time active-state-timeout. In this respect, idle MHs allow their respective soft-state routing cache mappings to timeout. These hosts transmit paging-update-time. The paging-update packet is an empty ICMP packet addressed to the gateway that is distinguished from a rout-update packet by its IP type parameter value. Paging-update packets are sent to the BS that offers the best signal quality. Similar to data and route-update packets, paging-update packets are routed on a hop-by-hop basis to the gateway. BSs may optionally maintain paging cache. A paging cache has the same format and operation as a routing cache except for two differences. First, paging cache mappings have a longer timeout period called paging-timeout. Second, paging cache mappings are updated by data and route-update packets sent by the mobile hosts. This results in idle mobile hosts having mappings in paging cache but not in routing cache. In contrast, active MHs will have mappings in both routing and paging caches.

Packets addressed to a MH are normally routed by routing cache mappings. Paging occurs when a packet is addressed to an idle MH and the GW or BSs find no valid routing cache mapping for the destination. If the BS has no paging cache, it will forward the packet to all of its interfaces except the one the packet came through. Paging cache is used to avoid broadcast search procedures found in cellular systems. BSs that have paging cache will only forward the paging packet if the destination has a valid paging cache mapping and only to the mapped interface(s). Without any paging cache the first packet addressed to an idle MH is broadcast in the access network. While the packet does not experience extra delay it does, however, load the access network. Using paging caches, the network operator can restrict the paging load in exchange for memory and processing cost.

Idle MHs that receive a packet move from idle to active state start their active-state-timer and immediately transmit a route-update packet. This ensures that routing cache mappings are established quickly potentially limiting any further flooding of messages to the MH.

## 2.2 Handoff Aware Wireless Access Internet Infrastructure (HAWAII)

HAWAII [13] stands for Handoff Aware Wireless Access Internet Infrastructure is another proposal dealing with the optimisation of Mobile IP. It is also developed with Mobile IPv4 in mind and has many common points with Cellular IP. Intra-domain mobile IP is dealt with HAWAII specialised patch setup schemes that update the forwarding tables with host-based entries in selected routers, whereas inter-domain macro-mobility defaults to conventional Mobile IP.

### 2.2.1 Protocol Overview

A common approach for providing transparent mobility to correspondent hosts is to divide the network into hierarchies. HAWAII uses a similar strategy, segregating the network into a hierarchy of domains, loosely modeled on the autonomous system hierarchy used in the Internet. The network architecture is illustrated in Fig. 2.2.1. The edge router, connecting the access network to the Internet core is called the domain root router. Location management is distributed with many common points with Cellular IP. Each host is assumed to have an IP address and a home domain. Each router has a default route inside the domain pointing towards the domain root router. For every mobile terminal setting up its path, an entry for its IP address is added and associated with the appropriate interface. Packets destined to the mobile host reach the domain root router based on the subnet address of the domain and are then forwarded over special dynamically established paths to the mobile host.

When the mobile host moves into a foreign domain, it is revert to traditional Mobile IP mechanisms. If the foreign domain is also based on HAWAII, then the mobile host is assigned a co-located care-of-address from its foreign domain. Packets are tunneled to the care-of-address by a home agent in its home domain. When moving within the foreign domain, the mobile host retains its care-of-address unchanged and connectivity is maintained using dynamically established paths.

The protocol contains three types of messages for path setup: power-up, update and refresh. A mobile host that first powers up and attaches to a domain sends a path setup power-up message. This has the effect of establishing host specific routes for the mobile host in the domain root router and any intermediate routers on the path towards the mobile host. Thus, the connectivity from that domain root router to the mobile hosts connected through it forms a virtual tree overlay.

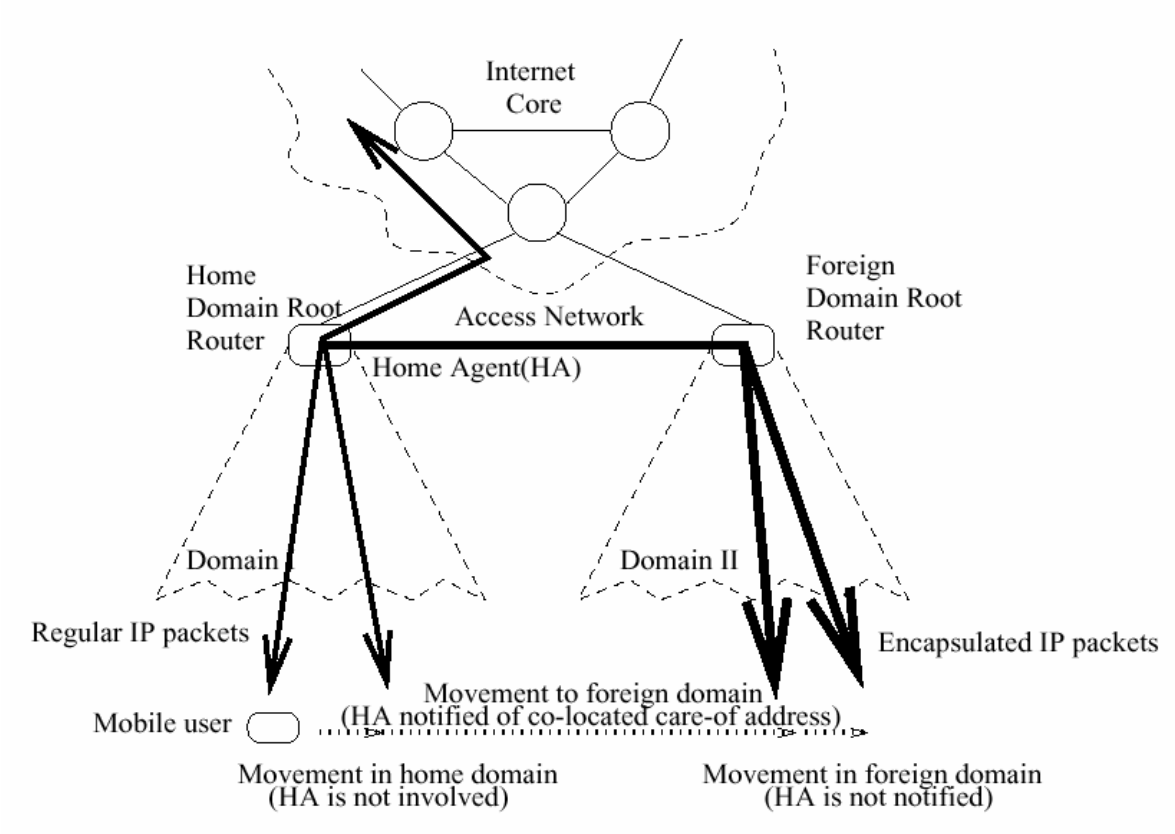


*Figure 2.2.1. HAWAII's Architecture using hierarchy of domains.*

While the mobile host moves within the domain, maintaining end-to-end connectivity to the mobile host requires special techniques for managing user mobility. HAWAII uses path setup update messages to establish and update host-based routing entries for the mobile hosts in selective routers in the domain so that packets arriving at the domain root router can reach the mobile host with limited disruption. The choice of when, how, and which routers are updated constitutes two particular path setup schemes: forwarding and non-forwarding. In forwarding path setup scheme, packets are first forwarded from the old base station to the new base station before they are diverted at the cross-over router. In non-forwarding path setup scheme, as the path setup message travels from the new base station to the old base station, data packets

are diverted at the cross-over router to the new base station, resulting in no forwarding of packets from the old base station.

HAWAII path state maintained in the routers is soft-state. This increases the robustness of the protocol to router and link failures. The mobile host infrequently sends periodic path refresh messages to the base station to which it is attached to maintain the host based entries, failing which they will be removed by the base station. The base station and the intermediate routers, in turn, send periodic aggregate hop-by-hop refresh messages towards the domain root router.

## 2.3 Hierarchical Mobile IP

The Hierarchical Mobile IP [14] is a proposed extension to Mobile IP which employs a hierarchy of foreign agents to locally handle Mobile IP registration as shown in Fig. 2.3.1, in order to support power-constrained operation and to reduce routing state information in the visited domain. In this protocol mobile hosts send Mobile IP registration messages to update their respective location information. Registration messages establish tunnels between neighbouring FAs along the path from the mobile node (MN) to a gateway foreign agent (GFA). Packets addressed to MNs travel in this network of tunnels, which can be viewed as a separate routing network overlay on top of IP. The use of tunnels makes it possible to employ the protocol in an IP network that carries non-mobile traffic as well. Typically one level of hierarchy is considered where all FAs are connected to the GFA. In this case direct tunnels connect the GFA to FAs that are located at access points.

Paging extensions for Hierarchical Mobile IP are presented in [15] allowing the idle MNs to operate in a power saving mode while located within a paging area. The location of MNs is known to HAs and is represented by paging areas. After receiving a packet addressed to a MN located in a foreign network, the HA tunnels that packet to the paging FA, which then pages the MN to re-establishes a path toward the current point of attachment. Paging a MN can take place using a specific communication time-slot in the paging area similar to the paging channel in second generation cellular systems. Paging schemes increases the amount of time a MN can remain in a power

saving mode. In this case, the MN only needs to wakeup at predefined time intervals to check for incoming packet requests.

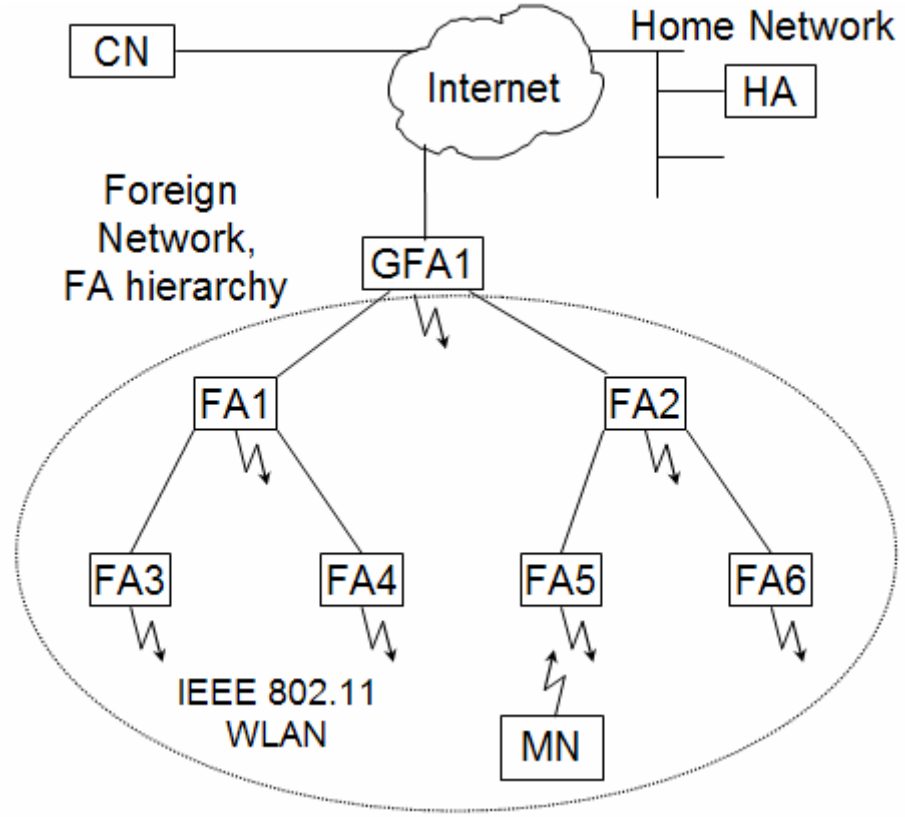


*Figure 2.3.1. Hierarchy of Foreign Agents.*

## 2.4 Proactive Handoff

Proactive Handoff [16], the foreign agent assisted hand-off is assumes to have the same architecture as Hierarchical Mobile IP. This proposal allows one or more FAs to forward packets prior to receiving Mobile IP registration request from a MN. After detecting that a MN is about to perform a handoff to a different location, the MN's serving FA sends a binding updates request to the new FA prior to handoff. This proactive binding update contains the MN's home address, security related information, as well as the serving GFA's address. The proposal assumes that FAs can detect the direction of movement of MNs by taking advantage of link layer and radio specific information. Upon reception of the binding update, the new FA sends a handoff request toward the GFA, which in turn towards packets to all FAs registered by the MN. The proactive protocol completes the layer two handoff and forward data to the MN before the Mobile IP registration proceeds. In essence, proactive handoff delivers IP packets to the MN via the new BS before Mobile IP can handoff.

## 2.5 Fast Handoff

The fast handoff proposal [17] reuses the principle of Hierarchical Mobile IP and addresses two remaining problems. These are mainly the need for a fast handoff management for real-time applications and the presence of triangular routing inside the domain.

This proposal assumes that the serving FA anticipates the movement of MHs by sending multiple copies of the traffic to potential neighbour FAs. “Bicasting” is used to support data forwarding to the old and new FAs while the MH is moving between the old and new access points. Fast handoff predicts the movement of MHs through coupling with layer two functionality that is dependent on the type of access technology used. Bicasting uses simultaneous bindings, where the MHs set the “S” bit in the registration request. Depending on the networking model (i.e., flat or hierarchical model) the receiving agent (HA, GFA) will add a new binding for the MH. As in the case of proactive handoff, the fast handoff proposal also assumes that it can anticipate the movement of MHs in advance of handoff. Fast handoff completes the Mobile IP handoff prior to establishing Layer two connectivity or forwarding data. The total delay for fast handoffs is limited to the time needed to perform a layer two handoff. Fig. 2.5.1 shows the difference between fast handoff and proactive handoff.

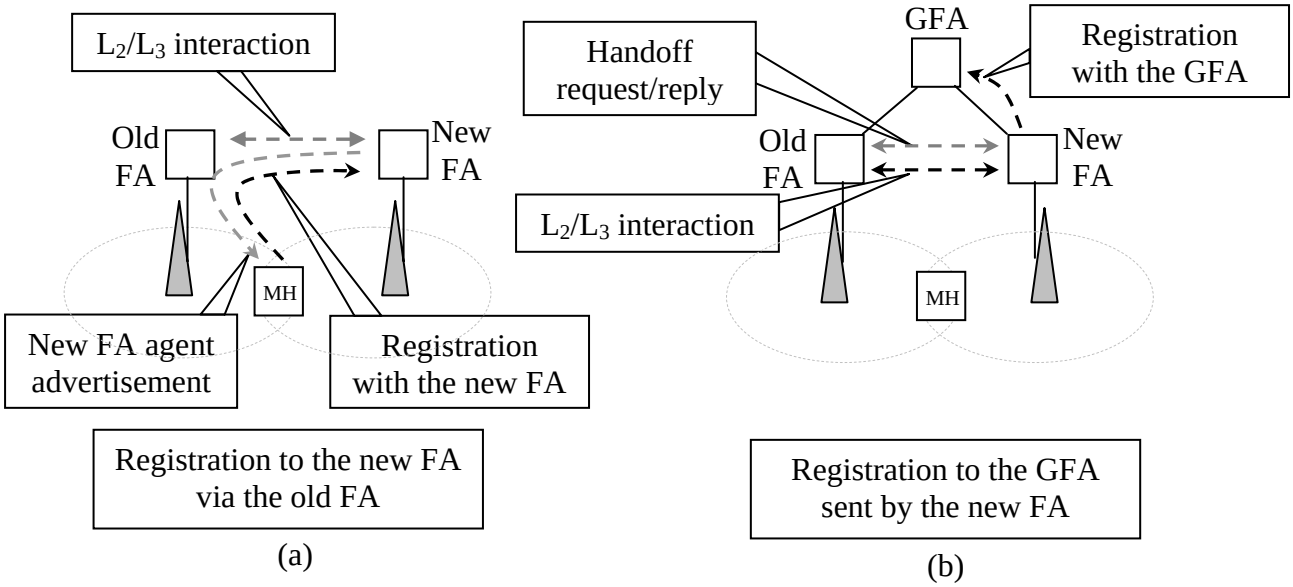


*Figure 2.5.1. Handoff Mechanisms for (a) fast handoff and (b) proactive handoff.*

## 2.6 Edge Mobility Architecture (EMA)

Edge Mobility Architecture [18, 19] is an architecture proposal, focusing on the provision of native IP services in future cellular systems. It is IP based, both in the core and in the access network and builds on the vision of standard IP equipment introduced in the core and access cellular network. While it is based on Mobile IP and standard IP protocols, it proposes changes and addition to them to enhance local mobility.

The evolution path of the telecommunication networks seems to be the Internet protocol architecture both for fixed and mobile networks. 2G cellular networks already provide data services and 3G networks are designed to provide Internet connectivity. Theses systems deal with mobility based on the Layer 2 approach and the introduction of additional entities to support the installed switch infrastructure. IP routing technology is constantly gaining ground towards the network's edge and it may eventually become less cost effective to support the various layer 2 mobility management modules. IP routing has the potential to present a unified solution to the cellular networks' requirements.

Standard Internet routing protocols are designed under the assumption that the location of the IP node is static. They also assume that the allocation of Internet address aims at providing IP address aggregation, so that the address prefixes become routing guidelines. Traditional IP routing protocols need only to deal with infrequent network changes and link failures or permanent modifications to the routing scheme.

Mobile Ad Hoc Networks on the other hand deal with constantly moving nodes having permanent IP address. This approach satisfies the need for the ad hoc topology, but loses the benefit of address aggregation.

The interesting topology is the one where the core network is essentially fixed and the end nodes mobile. It is the classical topology employed in current cellular networks, and handled with the cellular edge mobility technology (GSM, GPRS). The long-term goal of EMA is the shift of this edge mobility functionality up to the network (IP) layer. A combination reaping the benefits of both fixed and ad hoc networks is

proposed to achieve the optimal tradeoff for use in future cellular networks. The primary goal is the ability to move the IP address (interface) in the routing topology when the mobile changes the BSs, so that active IP sessions are maintained. This mobility enhancement is named Mobility Enhanced Routing (MER).

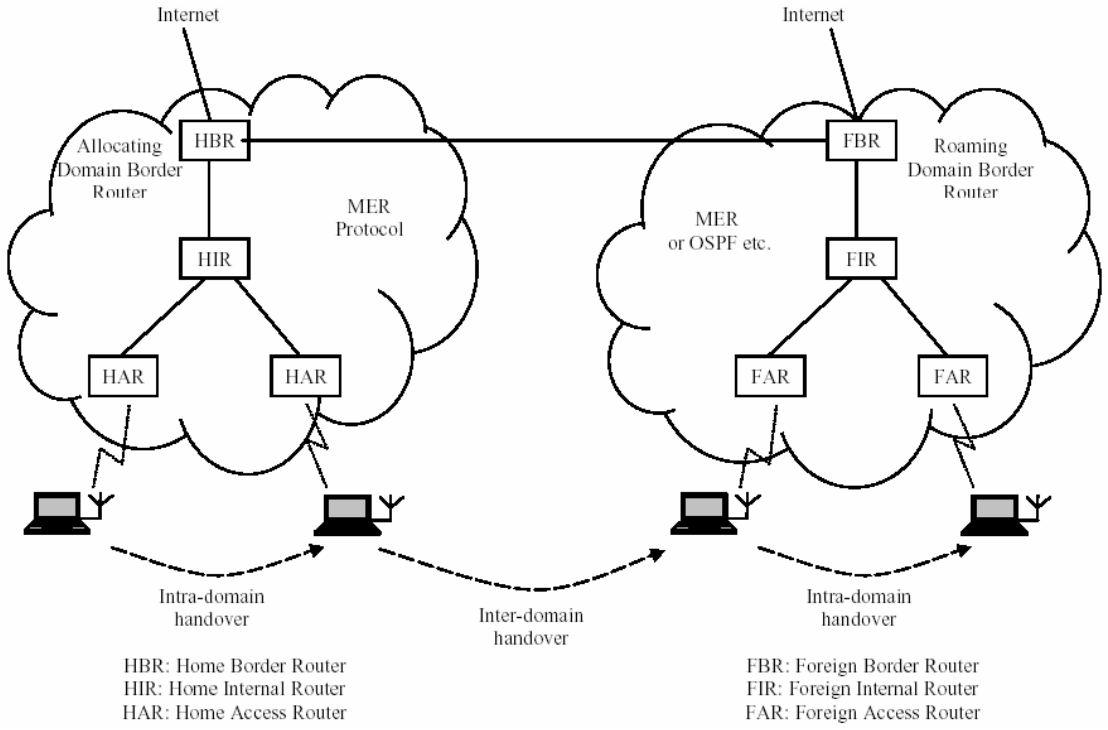


*Figure 2.6.1. EMA Architecture*

### 2.6.1 EMA Architecture

EMA as shown in Fig. 2.6.1 is designed to lay down the routing framework in mobility edge domains, i.e. in domains with their core and/or access network fixed but the terminal nodes mobile. Inside the domain a single routing protocol is proposed to be used, specifically a Mobility Enhanced Routing (MER) Protocol. The routers within the domains are standard IP routers running this MER protocol. Some of the routers are called Access Routers (ARs) and are connected to some wireless link layer that may be TDMA, CDMA, WLAN, etc or any combination of them.

There may exist Border Routers (BRs) and other internal routers. The topology can be mesh-based and not strictly tree-based, in which a hierarchy is implied. For internetworking within and outside the domain, IP based communication is used.

The architecture proposal aims to be generic enough to allow for a wide range of routing protocols to be used as well as diverse types of cellular technology. Some modifications to the actual fixed or ad-hoc routing protocol are necessary in order to comply with EMA. The wireless link technology employed at layer 2 (TDMA, CDMA, WLAN) can possibly offer some or all of the handover models and other capabilities such as break-before-make, make-before-break, power measurement, mobile assisted handovers, paging, security features etc.

The architecture is built upon the following building blocks:

- An intra-domain routing protocol, allowing both prefix routing and host routes. A block of IP addresses is represented by a prefix and is allocated to each Access Router. Host routes are used to support mobile migration away from the allocating Access Router.
- A virtual link for coordination between cooperating Access Routers to locally manage the handover of a mobile terminal.
- A temporary tunnel from the old to the new Access Routers to redirect packets during a handover while routing converges.
- The ability to crate and destroy a host route for a mobile.
- A method to return the allocated IP address to the allocating Access Router when the IP session terminates.
- The use of Mobile IP across provider borders to facilitate seamless roaming in foreign networks.

As it can already be observed, EMA is following a different path in address allocation. Specifically it embraces the model of current dial-up connections and enhances it to support mobility. The concept behind this approach is the use of a semi-temporary IP address. The functionality is described by the operation procedure:

- An IP session starts with the allocation of an IP address out of a pool of available addresses in the Access Router.
- As long as the mobile does not leave the coverage area of that Access Router, routing is performed as for any static-fixed node, based on the prefix of the IP address.

- If the mobile moves over to the coverage area of a different router, a specific host route is injected in the routing topology and packets to the mobile are routed to it based on the longest match.
- On a subsequent move, that host route is destroyed and another created.
- Finally, when the IP session terminated, the IP address is released and returned back to the Access Router, which allocated it.
- If the mobile starts another session, some other IP address will be allocated to it from the current Allocating Access Router, which controls the area including the mobile's location.

The rationale behind this design decision is mostly scalability. The authors discuss the possibility of using the Temporally Ordered Routing Algorithm (TORA) [20, 21] ad-hoc network routing protocol within EMA. A long term specific IP routing state in the system leads to undesired complexity and obscures the mass market introduction of the service.

The IP session is monitored by a session timer procedure, which detects inactivity and signals for the termination of the dynamic session. The time duration needed by the timer is dependent on various aspects such as the specific application, terminal characteristics, or service class.

### 2.6.2 EMA-Mobile IP Convergence

EMA supports a generic framework of handover models and requirements for them, which can be predictive (cellular) or non-predictive (inter-technology, less functionality). The requirements for a full EMA support cannot be fully provided by the current Mobile IP signalling services. Specifically, Mobile IP does not support handover initiated at the old Access Router, and does not provide the necessary information coordination between neighbouring Access Routers to facilitate forward handover. These extensions have already been proposed by individuals and groups in the Mobile IP IETF Working Group, and it is likely that Mobile IP will incorporate them.

The Mobile IP architecture also assumes that a mobile can be from an arbitrary home domain affiliated with an arbitrary Home Agent in that domain. It can also visit an arbitrary foreign domain if it supports Mobile IP and the providers have reached a roaming agreement. No assumptions can be made in EMA, whether the routing protocol used is MER or not. The basic functionality of Mobile IP must be preserved and a mobile with basic Mobile IP knowledge must function. The conclusion therefore is that the operations related to the Home Address as source/destination address, Care-of-Address registration. Home Agent capabilities and route optimisation must be independent to EMA and related only to Mobile IP, EMA deals efficiently with functionality regarding the Care-of-Address, Foreign Agent, Mobile-Access Router signalling and the temporary forwarding on handover features of Mobile IP.

Another important differentiation is that Mobile IP was designed for a different class of Internet connectivity than EMA tries to support. Mobile IP supports roaming in foreign domains from a home domain. The Care-of-Address is usually the Foreign Agent's address and the tunnels' endpoints are in the Foreign Agent. The use of a Co-located Care-of-Address (CCoA), while supported, is essentially discouraged since it is only valid in a specific foreign link. EMA on the other hand is based on this co-location of the Care-of-Address in the mobile, but supports its seamless mobility throughout the domain and not only on a specific link.

The existing UMTS network supports the remote access model from the mobile back through SGSN and GGSN to the external Internet Provider (via GTP tunnels). The Internet Provider issues the Home IP address and upwards Internet connectivity. EMA gives the option to the next UMTS generations to support native IP connectivity using local IP addresses, local IP service infrastructure and direct connectivity between users on that network and the Internet. Remote access (Mobile IP) combined with native services cab be an attractive future direction for cellular networks.

## 2.7 Telecommunication Enhanced Mobile IP (TeleMIP)

TeleMIP stands for Telecommunication Enhanced Mobile IP was introduced in [22] as a two-level hierarchical mobility management architecture that separates intra-domain mobility from global mobility for next-generation cellular networks. It is based on the same principles as Hierarchical Mobile IP. It specifies an Intra-Domain Mobility Management Protocol (IDMP) [23] to manage mobility within the domain and has multiple alternatives for global location management (Mobile IPv4, Mobile IPv6, SIP). TeleMIP is a standardization initiative taken by Telcordia Technologies, Inc.

The Mobile Node gets at least two care-of-addresses. One is for global reachability global care-of-address (GCOA) and one is used to specify the location within the domain local care-of-address (LCOA). The intra-domain mobility is therefore transparent to the Corresponding Node, which sees only the GCOA.

The basic idea is to use two-layered hierarchical mobility to reduce the latency of intra-domain updates without requiring establishment of host-specific routes.

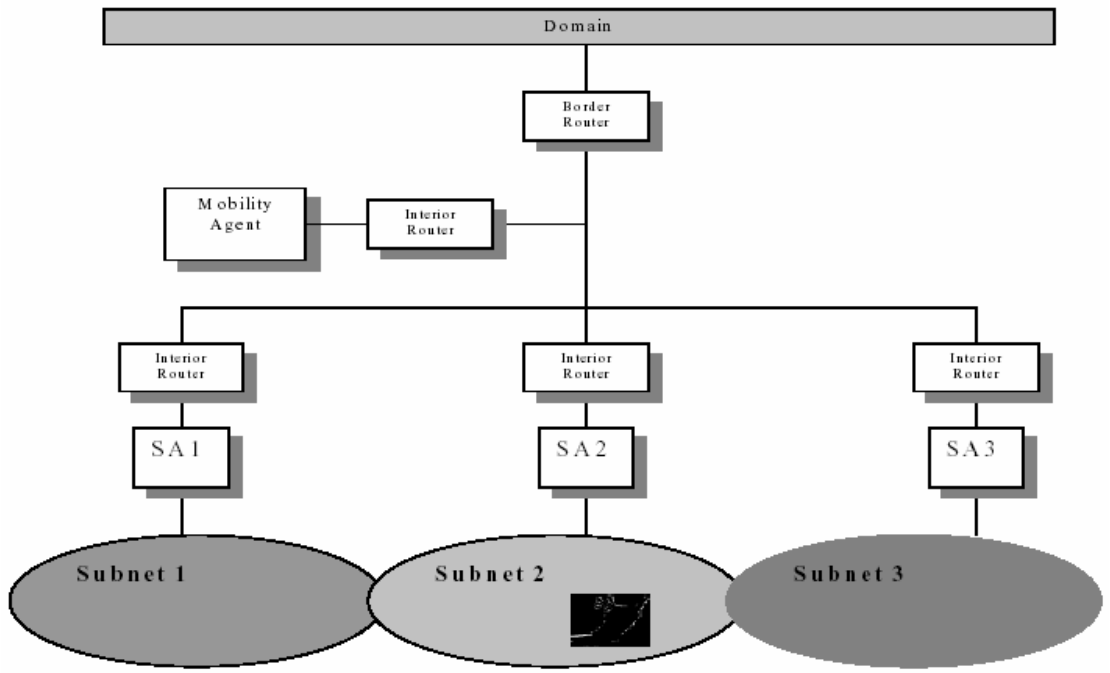


*Figure 2.7.1. IDMP Agent Architecture.*

### 2.7.1 Mobility Management

TeleMIP specifies a two-layer hierarchical mobility management technique to handle intra-domain mobility. It uses Mobile IP for global mobility management and IDMP to manage mobility within the domain as shown in Fig. 2.7.1. IDMP's mobility infrastructure specifies the use of a special nodes, called Mobility Agent (MA), which provide a stable (within the domain) and globally valid care-of-address to a Mobile Host.

A separate LCOA is used in order to perform intra-domain mobility management. The LCOA changes at every subnet, but such updates are localized only up to the MA and this reduces the intra-domain update latency. Intra-domain routing is no longer host-based but uses this LCOA. Since a LCOA can be private (locally scoped), this promotes addressing efficiency since a GCOA is not needed for every for every MH.

Foreign Agents/Dynamic Host Configuration Protocol (DHCP) servers broadcast domain identifiers to indicate intra-domain mobility. The architecture allows the choice of either co-located or Foreign Agent-based local addresses and a MH is dynamically assigned a MA.

The MH updates the MA of its new local binding on every subnet change. The MA then forwards packets to the MH using the Internet routing tables and there is no assumption of tree-based hierarchies inside the domain.

There are two alternative IDMP addressing modes: Mobility Agent-based addressing and global care-of addressing. With MA-based addressing, all MHs associated with the same MA have the same GCOA. Each MH must also have a separate permanent home address for MA demultiplexing that is globally valid. Also with MA-based addressing, tunneling from CH/HA is mandatory.

The global care-of addressing is similar to HAWAII, but it is still uses a separate LCOA. The MA has a pool of public addresses available and each MH gets its own unique GCOA. In this case there is no need for separate permanent address because

the GCOA can be used for global routing and there is also no mandatory need for tunneling from CH/HA to the MA.

The two layer addressing eliminates the need for source routes and separates the intra-domain Authorisation Authentication (AA) from the global AA. Another advantage is that the MA acts as a central point for controlling all mobility-related support and could therefore be used for metering, authorisation and other service specific functions. TeleMIP works with multiple global mobility protocols.

On the other hand, fast handovers and paging functions requires some form of multicasting support, and the MH must explicitly request support from the MA.

The mobility management scheme also means greater signaling complexity since it uses separate and explicit signaling for subnet-level, intra-domain and global mobility.

## 2.8 Summary of IP Micro-Mobility Protocols

The most important problem in IP mobility is handover management. To reduce the risk of packet loss, this handover must be fast and as efficient as possible. Handover management introduces two types of latencies: movement detection latency and IP routing update latency. In Mobile IP, the IP routing update is performed by using registration process. This can take a long time but with micro-mobility approach only one registration with the HA is required when connecting for the first time to a domain.

All of the micro-mobility protocols have shown that movement detection latency can be reduced by using L2 triggered handoff. This method assumes that it is possible to receive a radio handoff trigger efficiently in advance before the actual radio handoff and it also provide new location information of the MH. To reduce routing update latency, the choice of a hierarchical network structure is a good solution. Table 2.8.1 provides a summary of the IP micro-mobility protocols with the types of handoff and other features that are found in the survey.

| **Protocols** | **Handoff Type** | **L2 Triggered Handoff** | **Handoff initiator** | **Nodes Involved** | **Means of Update** | **Paging** | **Tunneling** |
|---|---|---|---|---|---|---|---|
| Cellular IP | Semi-soft handoff & Hard handoff | Yes | MH | All CIP nodes | Data packets | Yes | No |
| HAWAII | Forwarding & Non-forwarding scheme | Yes | MH | All routers | Signalling messages | Yes | No |
| Hierarchical Mobile IP | Within & between hierarchy | No | MH | FAs | Signalling messages | Yes | Yes |
| Proactive Handoff | With anchor & regional registration | Yes | MH or FAs | FAs & GFAs | Signalling messages | No | Yes |
| Fast Handoff | Source trigger, Target trigger & Mobile initiated | Yes | MH or FAs | FAs | Signalling messages | No | Yes |
| EMA | Between AR and BR | Yes | MH | All routers | Data packets | No | Yes |
| TeleMIP | Between FAs & Gateway FAs | No | MH | All routers | Signalling messages | No | Yes |

*Table 2.8.1. Summary of IP micro-mobility protocols.*

# 3.0 Implementation of Micro-Mobility Testbed

## 3.1 Availability of Micro-Mobility Protocol

Currently there are only two micro-mobility protocols implemented in the Internet. They are (i) Cellular IP [24] developed by Columbia University and (ii) Hierarchical Mobile IP [25] developed by Helsinki University, discussed in Sec. 2.1 and Sec. 2.3, respectively. We have selected Cellular IP be implemented in the micro-mobility working prototype for two reasons:

1. Simplicity – Cellular IP software is well-documented, easy to be installed and configured.
2. Cost efficient – Cellular IP is cheaper to build in a testbed, whereas Hierarchical Mobile IP requires more computers to serve as base stations since its design is based on hierarchy of Foreign Agents.

Our goals for implementing this micro-mobility testbed have been as follows:

- to provide mobility management in a wireless LAN while streaming multimedia session, continuous download session, E-mail and surfing across the Internet without losing connectivity.
- to measure TCP performance over mobility management in wireless LAN environment with different application session.
- to measure and compare performance results with those known for real-time testbeds.

## 3.2 Cellular IP Testbed Implementation

### 3.2.1 Hardware

According to COMET group that developed Cellular IP, the minimum requirement for Cellular IP gateways and base stations is to have a CPU clock frequency higher or

equal to 200 MHz and one notebook computer that runs the mobile host module. The following hardware is used in my testbed implementation:

- Cellular IP Gateway and Base Stations are built on Intel Pentium II 233 MHz Computers donated by Allied Telesyn Research.
- Cellular IP mobile node is running on an Intel Celeron 800 MHz Toshiba notebook.
- A 10 Mbps repeater links up the Cellular IP gateway and two Cellular IP base stations onto the same network.
- A dummy Web server is installed in an Intel Pentium 333 MHz Machine to emulate the Internet. This Web server is connected to the Cellular IP gateway with link speed of 100 Mbps.
- The wireless interfaces are Lucent Technologies Orinoco IEEE 802.11b 11Mbps silver cards with 64bit WEP encryption which are used in the Cellular IP base stations and Cellular IP mobile host. Orinoco cards supports the SNR-based handoff in Cellular IP where handoffs are performed based on measurements of control signal strengths from base stations.

### 3.2.2 Software

Cellular IP version 1.1 (linux_cellularip.tar.gz) can be downloaded at [24] and it is required to run on a Linux Operating System (OS). Every Cellular IP nodes in the testbed is running Linux Red Hat 7.3 with kernel 2.4.18-3 OS. Linux Red Hat OS is used instead of Windows OS because it is a free OS and is developed under the GNU General Public License. Linux Red Hat entire OS is freely available for download at [26] on the Internet and can then be burned onto CDs. Therefore there is no need to spend money on the OS for every machine to build the testbed.

C programming language and gcc compilers are used to compile the Cellular IP software suite for gateway, base station and mobile modules. In addition Tcl/TK 8.x is used to run Cellular IP mobile host graphic user interface (GUI). These programming tools are available with the Linux OS itself.

Cellular IP protocol operates transparently with respect to all application level software. Any Internet application is useful for testing the protocol and the testbed configuration, e.g. Netscape.

For a realistic model to test the micro-mobility testbed, a network emulation package NIST net [27] was used. It runs under Linux and is used to generate packet delays, packet losses and bandwidths of links between mobile hosts and web servers. Thus, it emulates the Internet behaviour across the wired-wireless interfaces between web servers and mobile hosts. Ethereal [28], a Network protocol analyzer for Linux, is used to capture data packets and to examine and analyse data passing through the wireless interface. It provides browsing the captured data interactively, summary and detailed information of each captured data packet. Thus it enables statistical data analysis of traffic between mobile host to the web server.

### 3.2.3 Installing IEEE 802.11b Wireless PCMCIA card in Linux OS

Before installing Cellular IP into the Linux Machine, PCMCIA driver and Orinoco wireless card driver must first be installed at the base station and mobile host that has a wireless interface. Linux drivers for the PCMCIA card can be downloaded from [29], and for Orinoco wireless card from [30]. Make sure Linux kernel source has been installed in the OS: check whether /usr/src/linux-2.4.18-3 directory exists or not. If not, Linux kernel source can be installed from Red Hat 7.3 CD by the following command: rpm -ivh kernel-source-2.4.18-3.i386.rpm. If Linux kernel source is installed, do the following steps to install the drivers:

1. Become root user to install the driver by command: su -
2. Copy PCMCIA driver (pcmcia-cs-3.1.34.tar.gz) into /usr/src/ directory and unzip the package with command: tar-zxvf pcmcia-cs-3.1.34.tar.gz
3. Copy Orinoco wireless card driver (wlli616.tgz) into /usr/src/pcmcia-cs-3.1.34/ directory and upzip the package with command: tar -zxvf wlli616.tgz
4. Enter PCMCIA driver directory, cd /usr/src/pcmcia-cs-3.1.34
5. make config
   - Set directory /usr/src/linux to /usr/src/linux-2.4.18-3
   - Set PnP Bios to yes

- Everything else is set to default settings

6. make all
7. make install
8. Reboot Linux OS then the drivers will be installed into the Linux kernel OS.
9. Once Linux is reloaded, edit the following configuration files using any text editor; config.opts, network.opts and wireless.opts
10. gedit config.opts (e.g. Mobile Node Configuration)
    - #options for wavelan/IEEE driver (Access point mode)

      module “wavelan2_cs” opts “irq_mask = 0xdeb8”

      module “wavelan2_cs” opts “network_name = Mobile”

      module “wavelan2_cs” opts “port_type = 1”

      module “wavelan2_cs” opts “channel = 10”

      module “wavelan2_cs” opts “create_ibss = N”

      module “wavelan2_cs” opts “distance_between_aps = 1”

      module “wavelan2_cs” opts “transmit_rate = 3”

      module “wavelan2_cs” opts “medium_reservation = 2347”

      module “wavelan2_cs” opts “card_power_management = N”

      module “wavelan2_cs” opts “microwave_robustness = N”

      module “wavelan2_cs” opts “receive_all_multicasts = Y”

      module “wavelan2_cs” opts “maximum_sleep_duration = 100”

      module “wavelan2_cs” opts “mac_address = 00,02,2D,1A,32,EE”

      module “wavelan2_cs” opts “station_name = Linux”

      module “wavelan2_cs” opts “enable_encryption = N”
11. gedit network.opts (e.g. Mobile Node Configuration)
    - BOOTP = “n”

      DHCP = “n”

      IPADDR = “192.168.1.3”

      NETMASK = ‘255.255.255.0”

      NETWORK = “192.0.0.0’

      BROADCAST = “192.168.1.255’

      DOMAIN = “”

      DNS_1 = “”
12. gedit wireless.opts
    - #Lucent wavelan IEEE

```
#note:wvlan
*,*,*,00:60:1D:*|*,*,*,00:02:2D:*)
INFO = “”
ESSID = “”
MODE = “Ad-Hoc”
RATE = “auto”
KEY = “”
;;
```

13. Restart the PCMCIA interface with command: /etc/rc.d/init.d/pcmcia restart
    - Orinoco wireless card will emit two high beeps which means that the card was identified and configured successfully.
    - If Orinoco wireless card emits one high beep followed by a low beep, this means that the card is identified but could not be configured.
    - If Orinoco wireless card emit one low beep, this means that the card is not identified.

When all the steps above are carried out successfully, execute carctl ident to display the interface information, /sbin/ifconfig to display the network information. The Wireless card must be operating in Ad Hoc mode for the base stations in order for Cellular IP to work.

### 3.2.4 Installing Cellular IP v.1.1 in Linux OS

Firstly make sure that libpcap is installed in the Linux OS before installing Cellular IP. This can be checked by examining whether /usr/include/pcap directory exists or not. If not, the libpcap package can be found in the Linux Red Hat 7.3 CD. To install it just use the rpm command: rpm -ivh libpcap-1.10-8.i386.rpm. If libpcap is installed, do the following steps to install Cellular IP in the Linux OS with Orinoco Wireless Card:

1. Download linux_cellularip.tar.gz and untar the gzip file under any directory: tar –zxvf linux_cellularip.tar.gz. cip-1.1 directory will be created.
2. Change directory to cip-1.1. Before doing make command, open and check Makefile to view if the compiler tag:

- ${CC} ${CFLAG} -c cipmobile.c -DWICACHE -DANCACHE_ -DDEBUG_ -DD_DEBUG_ is used to activate Lucent wireless card.
- ${CC} ${CFLAG} -c cipmobile.c -DANCACHE -DDEBUG_ -DD_DEBUG_ is used to activate Aironet wireless card.
- ${CC} ${CFLAG} -c cipnode.c -DDEBUG_ -DMOBILE_IP is the setting to activated Cellular IP/ Mobile IP internetworking, while adding '_' in –DMOBILE_IP to deactivated Cellular IP/ Mobile IP internetworking which results in ${CC} ${CFLAG} -c cipnode.c -DDEBUG_ -DMOBILE_IP_

3. Compile the software by executing make.
4. Edit cipnode.conf configuration file for gateway and base station. Edit cipmobile.conf configuration file for mobile host.
5. The configuration setup file needs to be modified to reflect the hardware configuration. My Cellular IP testbed is currently configured according to Fig. 3.2.1.

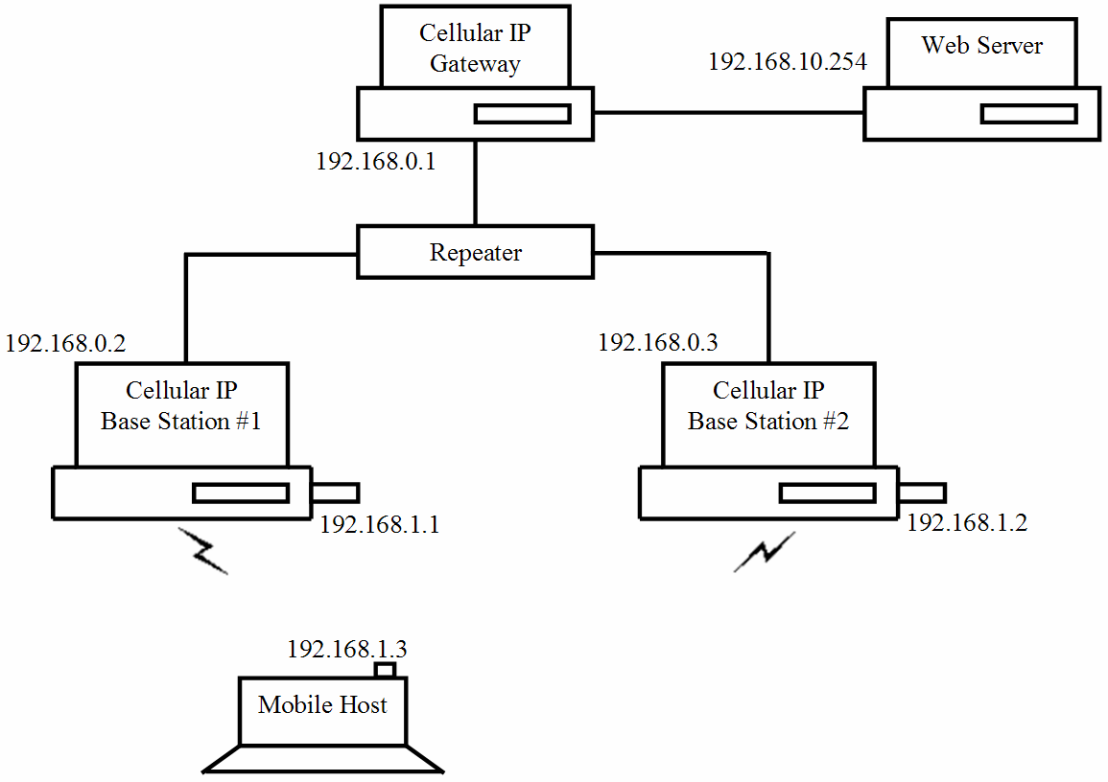


*Figure 3.2.1. Cellular IP Testbed Configuration.*

6. Gateway cipnode.conf configuration:
   - GW: YES

     IF YES, default router's IP address: (wire, eth1, 192.168.10.254)

leaf neighbours(s): (wire, eth0, 192.168.0.2), (wire, eth0, 192.168.0.3)
paging cache: YES
route-timeout: 3000 %in milliseconds
paging-timeout: 30000 %in milliseconds
max number of mobiles in cache: 100
max number of node interfaces: 10
GW IP address: 192.168.0.1

7. Base Station cipnode.conf configuration:
   - GW: NO

   IF YES, default router's IP address: % This is not used in BS
   IF NO, neighbor, uplink direction: (wire, eth0, 192.168.0.1)
   leaf neighbours(s): (wireless, eth1)
   paging cache: YES
   semisoft: NO
   route-timeout: 3000 %in milliseconds
   paging-timeout: 30000 %in milliseconds
   max number of mobiles in cache: 100
   max number of node interfaces: 10
   GW IP address: 192.168.0.1
8. Mobile Host cipmobile.conf configuration:
   - wireless interface: eth0

   mobile's IP address: 192.168.1.3
   air interface name: mobile-air
   route-update-time: 1000 %in milliseconds
   paging-update-time: 10000 %in milliseconds
   acitve-state-timeout: 5000 %in milliseconds
   handoff: 1 %forced (=0) or SNR based (=1)
9. To run Cellular IP Gateway, execute ./cipnode in root user.
10. To run Cellular IP Base Station, execute ./cipnode in root user.(make sure wireless interface is Ad-Hoc mode.
11. To run Cellular IP Mobile Host, execute ./cip in root user.

### 3.2.5 Installing NIST Net Emulator

Before installing NIST net, the Linux kernel requires that enhanced real time clock to be made and built as a module. Do the following step to produce a new kernel for running NIST net emulator:

1. Change to /usr/src/linux-2.4.18-3 directory.
2. Modify Makefile with EXTRAVERSION = “NIST”
3. make mrproper
4. make clean
5. make xconfig
   - Under Character devices, click ‘m’ for Enhanced RTC support to make it a module.
6. make dep
7. nohup make bzImage &
8. tail –f nohup.out
9. make modules
10. make modules_install
11. cp arch/i386/boot/bzImage /boot/bzImage-NIST-2003
12. mkinitrd -v /boot/initrd-2.4.18-NIST.img 2.4.18-NIST (note: 2.4.18-NIST can be seen at /lib/modules directory)
13. gedit /boot/grub/grub.conf (add new line to config file)
    - title Red Hat Linux (NIST version)
      root (hd0,0)
      kernel /bzImage-NIST-2003 ro root=/dev/sda2
      initrd /initrd-2.4.18-NIST.img
14. Restart machine and choose the newly make kernel.
15. Becoming root user: su -
16. choose any directory and untar nistnet: tar –zxvf nistnet.2.0.12.tar.gz
17. gedit Config
    - Comment #BDELAY =-DCONFIG_DELAYSTART
      Uncomment BDELAY = -DCONFIG_DELAYEND
18. Change directory: cd nistnet/monitor
19. xmkmf -a (autogenerate a Makefile related with X server)
20. gedit Imakefile

- change OURXAWLIB = -lxaw3d (read comment for info)

21. ./configure
22. make
23. make install
24. ./Load.Nistnet
25. xnistnet to run NIST net

## 3.3 Cellular IP Testbed Appearance

Cellular IP has been successfully implemented. Fig. 3.3.1 shows the setup of gateway, base station #1 and base station #2. Fig. 3.3.2 shows the machine used for running Apache web server. Fig. 3.3.3 shows the Cellular IP mobile host streaming NIKE Ad from the web server. Cellular IP mobile host has a GUI interface as shown in Fig. 3.3.4. Currently the mobile host is attached to base station #1. Notice that gateway and the base stations are transparent to the mobile host. Fig. 3.3.5 shows the data packets flowing from both the mobile host and the web server while the mobile host is in active mode. Fig. 3.3.6 shows a signal to noise ratio (SNR) based handoff with standard TCP based on the signal strength measurement. This provides mobility from one place to another without losing connectivity across the network. Fig. 3.3.7 shows mobile host browsing the homepage of the web server using Netscape. Using Ethereal as shown in Fig. 3.3.8, the time taken when first loading the web server homepage took about 2.5s as shown from packet sequence number 4 to 18. Later on, reloading the homepage takes only 16ms because the items on the page have already been downloaded to the notebook and so do not require getting any downloads. Fig. 3.3.9 shows the summary of the Ethereal results for the capturing of TCP packets. Fig. 3.3.10 and Fig. 3.3.11 shows mobile host using MpegTV to stream a Nike Advertisement from the web server. Fig. 3.3.12 shows NIST net GUI.

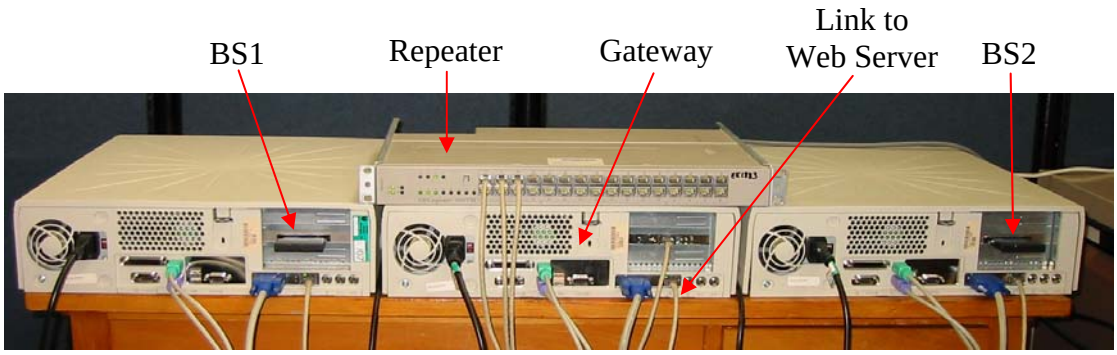


*Figure 3.3.1. Cellular IP gateway and base stations setup.*


*Figure 3.3.2. Apache Web Server.*


*Figure 3.3.3. Cellular IP Mobile Host streaming NIKE Ad from Web Server*

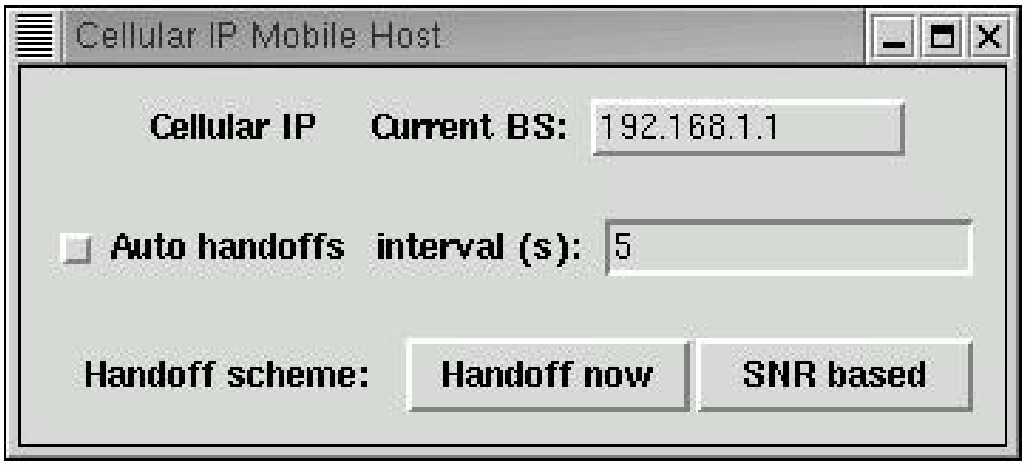


*Figure 3.3.4. Cellular IP Mobile Host GUI.*

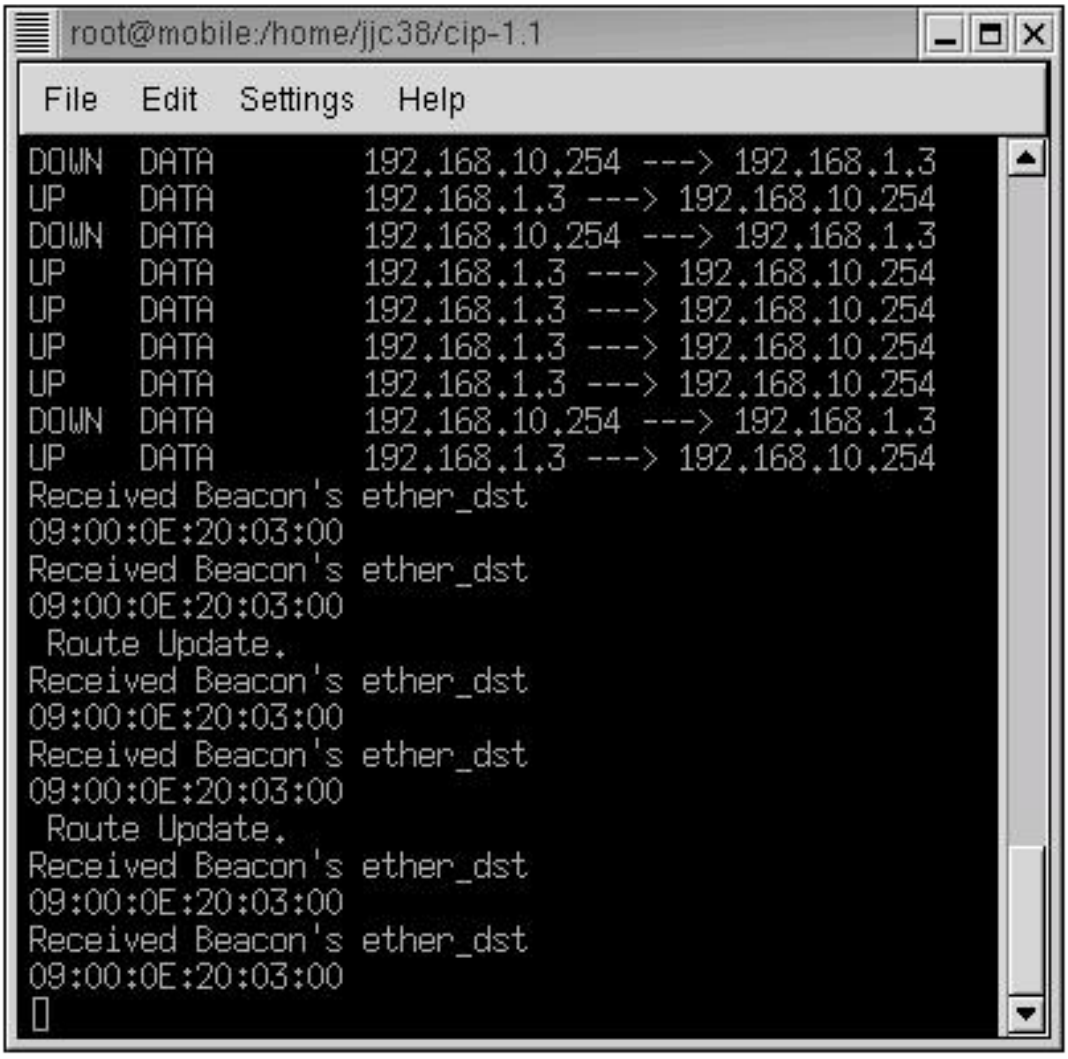


*Figure 3.3.5. Data packets flowing from both side of mobile host and web server while beacon messages is received to update mobile host location.*

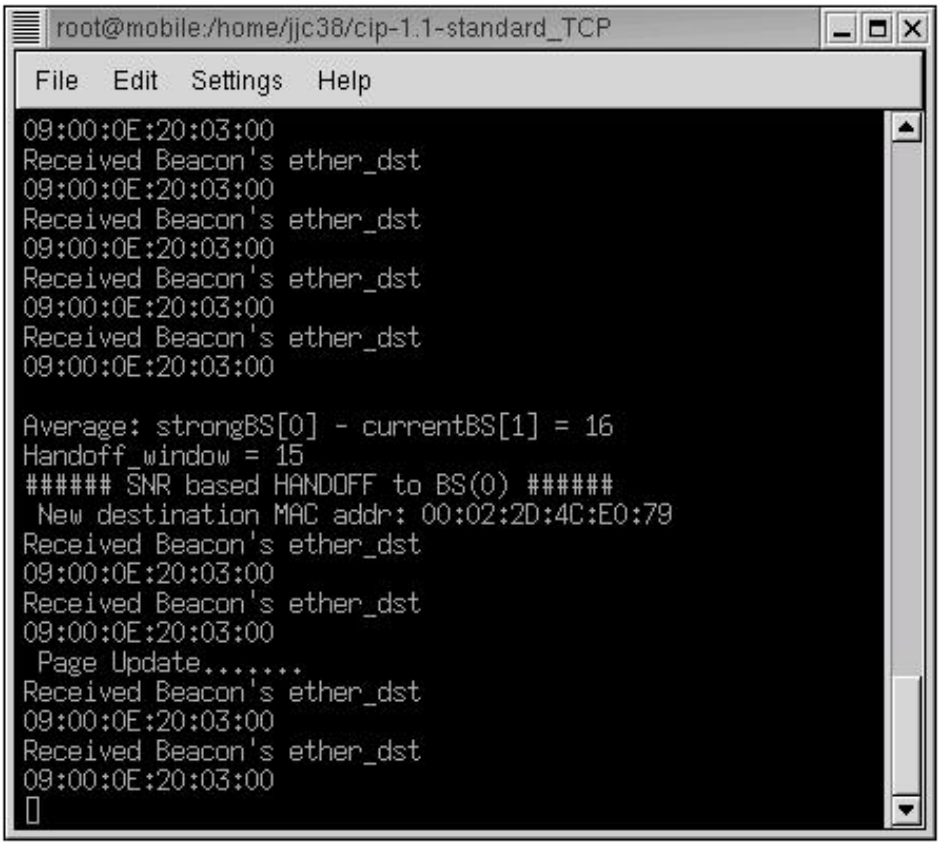


*Figure 3.3.6. Cellular IP SNR based Handoff with standard TCP from BS#2 to BS#1 who has the strongest signal strength.*

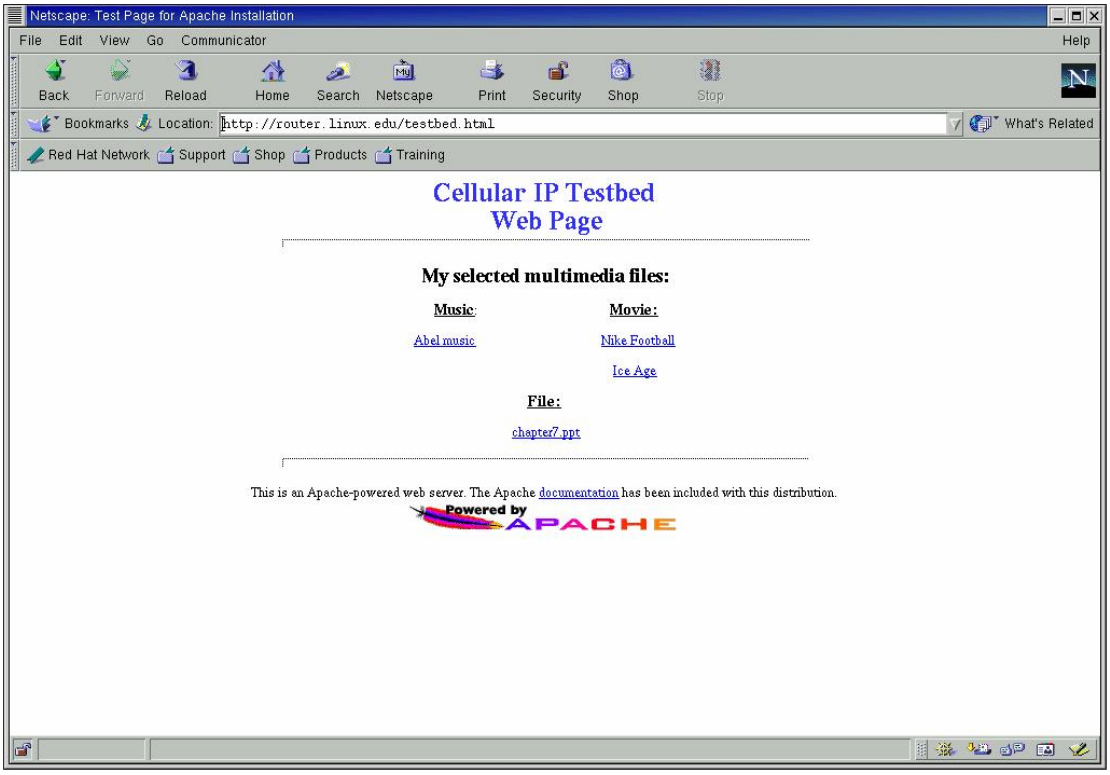


*Figure 3.3.7. Mobile Host browsing the homepage of Web Server.*

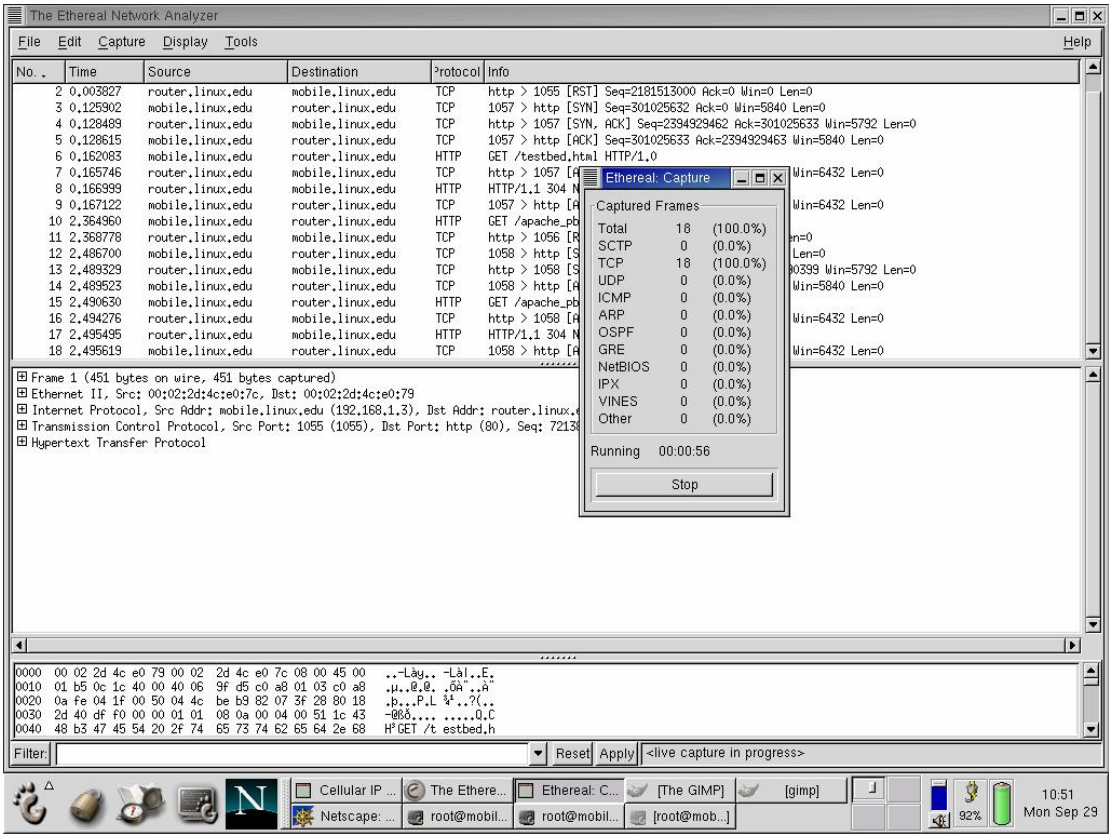


*Figure 3.3.8. Ethereal capturing TCP packets from mobile host browsing web server.*

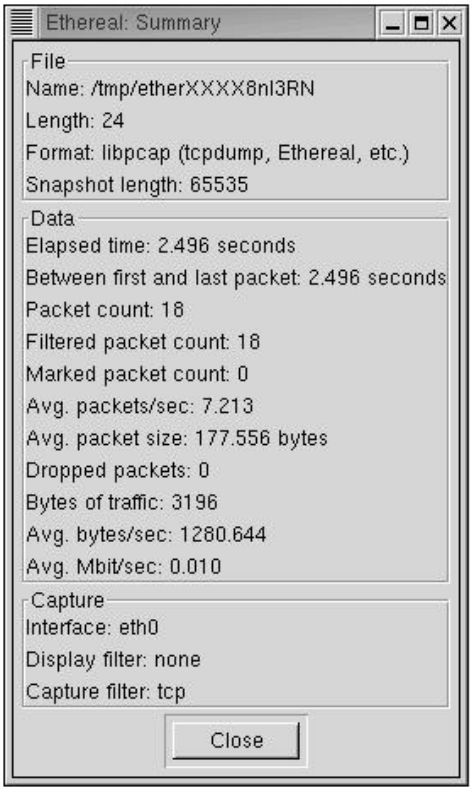


*Figure 3.3.9. Ethereal summary of data packets.*

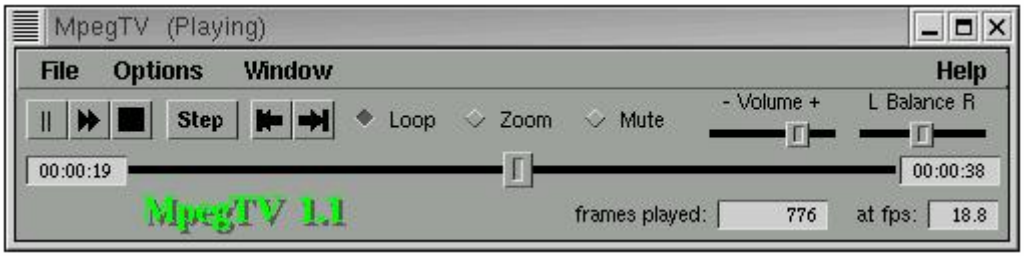


*Figure 3.3.10. MPEGTV video streaming application*

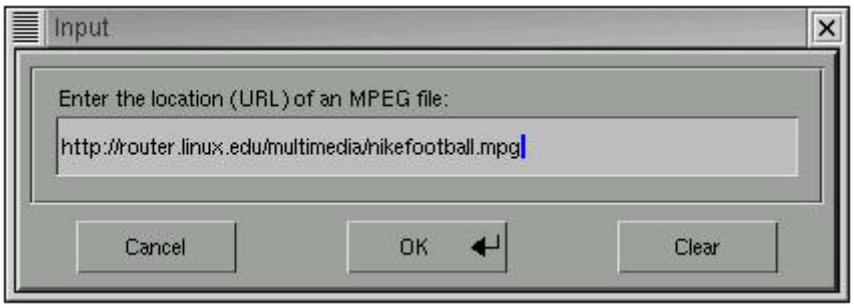


*Figure 3.3.11. Location of Nike Ad for MPEGTV to play.*

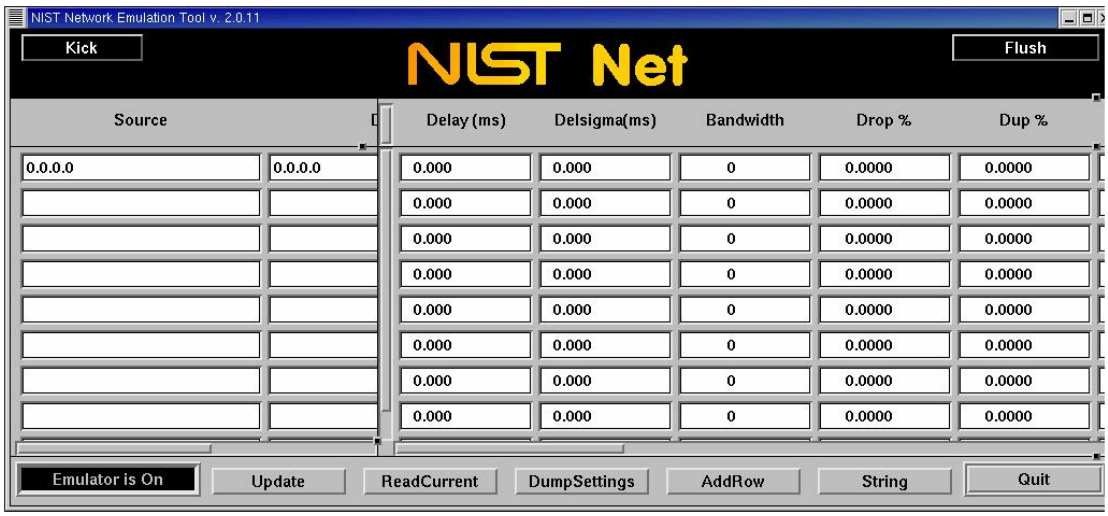


*Figure 3.3.12. NIST net emulator used to varies network parameters to evaluate network performance.*

## 3.4 Performance of Cellular IP testbed

One important question regarding the performance of a wireless LAN is how far the laptop can move, still maintaining high speed connection access to the Internet when running multimedia applications. To test the distance at which the mobile host can perform well with Cellular IP over IEEE 802.11b interface, we considered the TCP performance measured at the mobile host terminal. The experiment was setup while

mobile host remained stationary at various distances away from Base Station while simultaneously running MPEGTV streaming NIKE football advertisement of size 6553604 bytes and FTP downloading a document of size 19,841,024 bytes from the web server. This experiment was carried out in an outdoor environment for distances varying from 0m to 150m away from the base station. The average time to complete MPEGTV and FTP applications simultaneously was around 59s with standard deviation (s.d.) of 210ms.

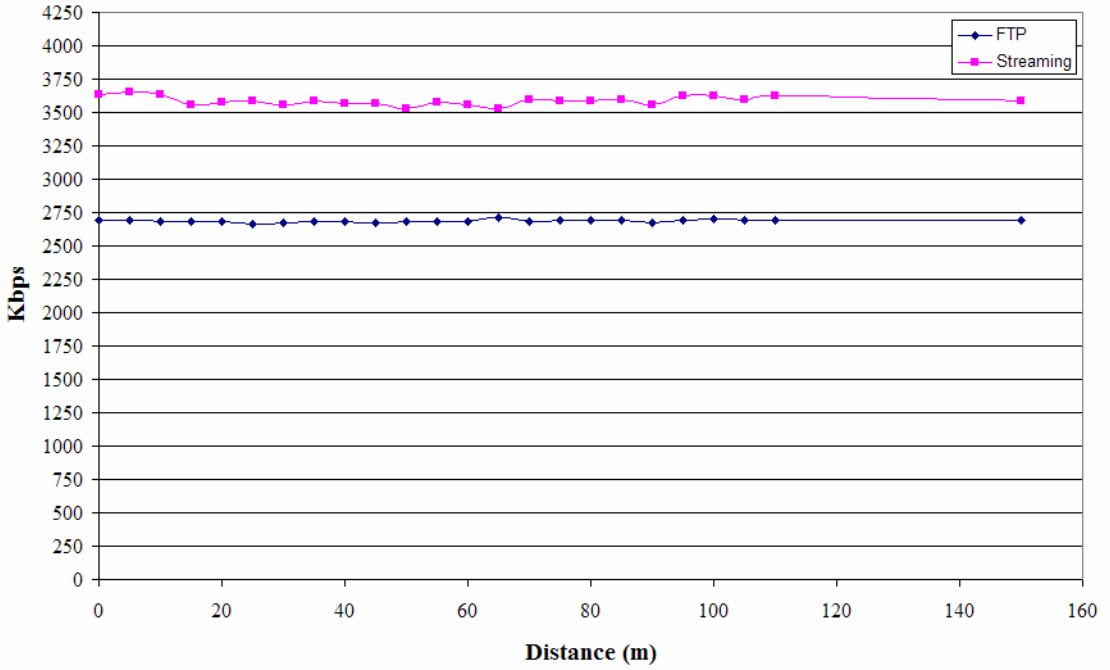


*Figure 3.4.1. TCP Throughput performance of FTP and Streaming session versus Distance away for BS*

Fig. 3.4.1 shows the TCP throughput of running a streaming session using MPEGTV and FTP download simultaneously across the wireless interface. TCP throughput was measured by dividing file size by time taken to complete each session. The figure shows that TCP throughput remains constant throughout a distance from 0 to 150m. For the Nike advertisement, the streaming session had an average TCP throughput around 3600 Kbps with a s.d. of 33.2 Kbps and for FTP download an average throughput of 2700 Kbps with a s.d. of 9.6 Kbps. While simultaneously downloading and streaming, the picture quality shown in MPEGTV started to degrade with glitches and pauses occurring in MPEGTV for distances of 100m and more. This proves that the 802.11b wireless interface has a high bandwidth capability of supporting multiple applications across the Internet with high speed connections and excellent multimedia

quality just as a wired network does. Therefore, the best distance to put each base station apart one another for outdoors is around 200m when building a wireless network for multimedia and download session for high speed connection and excellent multimedia quality for mobile users. The graph shown in Fig. 3.4.1 has the same relationship as the graph shown in Fig. 3.4.2 which report results of [31], by NASA Ames Research Center. The authors of [31] did measurements with Orinoco wireless cards and measured FTP throughput performance over the wireless interface. They also demonstrated constant FTP throughput from 0m to 100m while downloading a file.

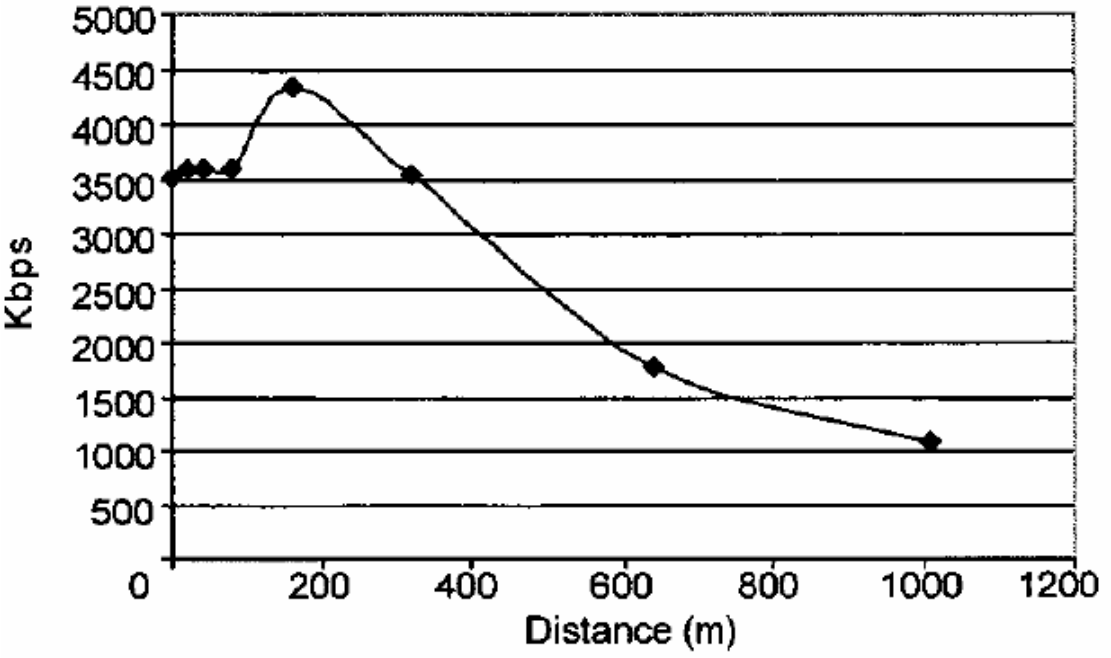


*Figure 3.4.2. Orinoco FTP throughput measured by NASA Ames Research Center [31].*

All the measurements reported above have not taken into account packet delays, therefore NIST net emulator was installed in Cellular IP gateway to help introducing packet delays between the gateway and the web server. This helped to emulate a more realistic model of teletrafic, when a mobile host is accessing the Internet.

Fig. 3.4.3 shows the average bandwidth measured by Ethereal in NIST net and the average bandwidth measured by the internal packet monitor of the NIST net. The packet delay introduced by NIST net is the round trip time measured between the gateway and the web server. The difference in 60 Kbps average bandwidth of both

traces is that Ethereal's average bandwidth consists the sum of Frame bits, Medium Access layer, IP layer, TCP layer and HTTP layer whereas NIST net's average bandwidth consists the sum of IP layer, TCP layer and HTTP layer.

The experiment was carried out at a distance within 10m from the base stations. The hypothesis was that the distance within 10m does not affect the performance of the multimedia session, see in Fig. 3.4.1. Both graphs show that as packet delays are introduced, the bandwidth increases by factor of 2. This is caused by the buffering mechanisms in the MPEGTV application. As we can see, the bandwidth remains constant until packet delays of a 160ms and then starts to drop linearly. It is observed that glitches started to appear from packet delay of 150ms onwards. More glitches were produced and video started to pause for 1s to 2s at 250ms packet delay onwards.

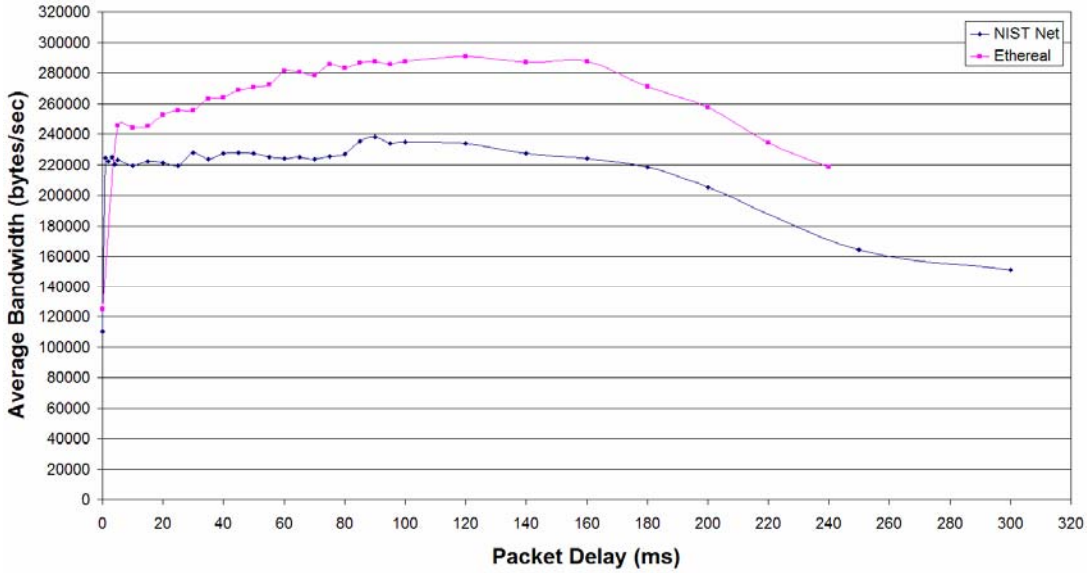


*Figure 3.4.3. Average bandwidth versus packet delay while mobile host is streaming NIKE advertisement from Web Server obtained by using Ethereal and by using the internal packet monitoring tool of NIST net.*

Fig. 3.4.4 shows the average bandwidth measured by NIST net while introducing different percentages of packet loss into the testbed. The experiment set a fixed 20ms packet delay throughout for a more realistic experimental environment as average bandwidth is not affect by this value as seen in Fig. 3.4.3. As seen in Fig. 3.4.4, the average bandwidth remains constant until 20% packet loss and at 50% packet loss, the bandwidth reduces to 40Kbytes/sec which is half of the bandwidth at 20% packet loss.

Most packet loss is either affected by congestion in the network or by multipath distortion and signal level fading in the indoor environment.

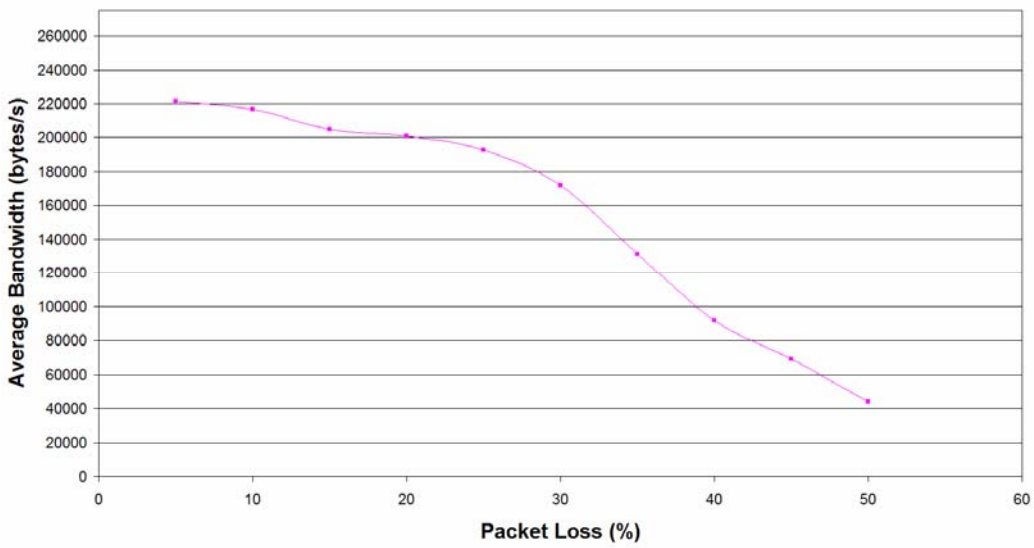


*Figure 3.4.4. Streaming NIKE Ad, Varying packet loss in NIST Net at BS and MH, fixed 20ms packet delay at Gateway.*

## 3.5 Summary of Cellular IP testbed

Implementation of the reported micro-mobility testbed required to design a number of of procedures. The testbed was successfully implemented with a minimal cost, using P2 machines with Linux OS donated by Allied Telesyn Research as its platform.

The results have shown an approximately constant TCP throughput of 3600 Kbps with IEEE 802.11b wireless interface while the MH performs a multimedia session at distances up to 100m outdoors. The maximum distance that MH performs perfectly with streaming session of NIKE advertisement is about 100m with line of sight. The best performance of MH while running a multimedia session at a distance of 10m without glitches and pauses occurs when packet delays are less than 50ms and packet loss are below 10%.

# 4.0 Survey of TCP Enhancement Schemes over Wired-Wireless Networks

## 4.1 Wired-Wireless Internet

During the past few years the Internet has become more heterogeneous with more powerful PCs and workstations coexisting with WebCast [32], wireless phones, and Personal Digital Assistants (PDAs) [33]. Currently it is an open question whether it is possible to continue to do so while providing the same kind of network services to all hosts in the wireless world.

Diversification in end-host capabilities limits to some extent the applications that would be running on each category of devices. The differences in computing power, memory size, input and output devices, in addition to mobility and power consumption issues will determine the potential uses of each end-host category. At the same time, end hosts use a far larger spectrum of networking technologies to connect to the Internet. This includes traditional wires and optical fibers, as well as wireless (802.11, Bluetooth) and infrared media.

Users need a reliable transmission medium for web browsing, email, file transfers, etc., and TCP is the dominant reliable transport protocol on top of which all of these services run. However, TCP was designed with certain assumptions for transmission in wired medium [34], and later modified as needed (e.g. [35]). For example, segment losses are assumed to be due to congestion, because in a wired medium transmission errors are relatively rare. This is not true for the wireless medium where, due to fading channels and user mobility, transmission losses are more frequent. Since certain assumptions behind TCP's highly tuned algorithms do not hold for such transmission-heterogeneous media, current TCP implementations do not perform well in wireless environments [36][37][38][39]. This chapter provides a thorough survey of the issues that current TCP faces in the wireless environment and several TCP implementations that have been studied over recent years to enhance TCP performance over the wireless medium.

## 4.2 TCP issues in a wireless environment

As stated earlier, TCP is not designed to perform well in wireless networks. This section discusses some of the issues that TCP suffers in the wireless environment. For a detailed description on the standard TCP, refer to several excellent books, such as [48][49].

### 4.2.1 Limited Bandwidth

Wireless networks offer limited raw bit rates. For example, CDPD networks [47] offer a low raw bit rate of 19.2 Kbps, which can be shared amongst up to 30 users. On the other hand, the current generation of wireless LAN standards offers much more bandwidth; for example IEEE 802.11b standard offers raw bit rates of up to 11 Mbps, while 802.11a and HIPERLAN/2 offer up to 54 Mbps. Table 4.2.1 shows a general overview of various wireless network technologies offered by the bandwidth.

| wireless network technology | bit rates |
|---|---|
| CDPD (AMPS) | 19.2 Kbps |
| IS-95 CDMA | 9.6 – 14.4 Kbps |
| GSM/GSM Phase 2+ | 9.6 – 14.4 Kbps |
| High-Speed Circuit Switched Data (HSCSD) | up to 64 Kbps |
| General Packet Radio Service (GPRS) | up to 128 Kbps |
| CDMA 2000 1xEV | up to 384 Kbps |
| Enhanced Data Rates for GSM Evolution (EDGE) | up to 384 Kbps |
| Universal Mobile Telecommunications System (UMTS) | up to 2 Mbps |
| IEEE 802.11b | up to 11 Mbps |
| HIPERLAN/1 | up to 25 Mbps |
| HIPERLAN/2 | up to 54 Mbps |
| IEEE 802.11a/g | up to 54 Mbps |

*Table 4.2.1. Overview of bit rates offered by wireless network technologies. [40][41][42][43][44][45][46][47]*

### 4.2.2 Long Round Trip Times

In general, wireless media exhibit longer latency delays than wired ones [46][50][51][52]. This affects TCP throughput and increases the interactive delays perceived by users. In addition, it has been shown that under certain scenarios TCP-Tahoe is "unfair" for connections with longer RTTs, Lakshman et al. [53]. Since the congestion window increases proportionally to the rate of incoming acknowledgments (ACKs), two competing connections will experience different growth rates in their congestion window and therefore will achieve different throughput levels. In particular, the connection with the longer RTTs will see a much more moderate growth in the congestion window, and consequently will experience a smaller throughput. One can see that connections that include a wireless path are not favoured by TCP when compared to other flows with the same number of hops.

### 4.2.3 Random losses

Wireless media are more prone to transmission losses and disconnections, for example, due to fading channels, shadowing and handoffs. TCP was designed with a wired medium (copper cables) in mind, which has bit error rates (BER) in the order of $10^{-6}$ to $10^{-8}$. This translates into a segment loss rate of about 1.2% for 1500-byte segments. BER are much higher in the wireless domain, usually in the order of $10^{-3}$, and sometimes as high as $10^{-1}$ [50]. With BER in the order of $10^{-3}$, the packet loss rate is an order of magnitude more in a wireless environment (about 12%). However, using Forward Error Correction (FEC) schemes at the link layer can effectively bring the False Alarm Error rate down to the order of $10^{-6}$ [50].

In the presence of high BER, intermittent connectivity and handoffs' characteristics of wireless environments, TCP reacts to packet losses as it would in the wired environment: it drops its transmission window size before retransmitting packets, initiates congestion control or avoidance mechanisms [35] (e.g., slow start [58]) and resets its retransmission timer (Karn's Algorithm [59]). These measures result in an unnecessary reduction in the link's bandwidth utilization, thereby causing a significant degradation in performance in the form of poor throughput and very high interactive delays [37] [39].

### 4.2.4 User Mobility

Wireless networks enable the user to move freely. Perkins [52] makes a clear distinction between portability and mobility. In the first case, the user may use, for example, a laptop that can plug into the network at several different locations with access points. However, this implies that there is some time (practically, the time spent between each transition) during which the user does not enjoy network services. The term mobility means that the user can roam freely and at the same time seamlessly enjoy network services without interruptions. A number of issues related to user mobility led to the creation of the Mobile IP working group by the Internet Engineering Task Force (IETF) [6].

Wireless networks are usually designed in a cellular fashion, where each cell covers a particular area. Users inside this area are serviced from a single base station (BS) or access point (AP), a host that is connected to the wired network and offers wireless connectivity to wireless hosts. During a handoff, all necessary information must be transferred between the two base stations so that the mobile host can continue to be connected. Cáceres and Iftode were amongst the first to study the implications of mobility on TCP performance [37]. There is now a consents within the research community that it is desirable for reliable transport protocols to be able to differentiate between congestion-related, transmission (or random), and motion-related losses.

### 4.2.5 Short Flows

Web browsing and email account for the majority of today's Internet teletraffic. These services usually include the transmission of rather small amounts of data. This means that when the application layer protocol opens a TCP connection for data transfer there is a very large possibility that the whole transfer is completed while the TCP sender is in the slow start phase. Therefore, the TCP connection never manages to fully utilize the available bandwidth. Savage et al. [55], for example, claim that a 10 KB download under perfect conditions and with infinite bandwidth will not proceed with a throughput faster than 300 Kbps. Indeed, if the client is 35 ms "away" from the server, the 10 KB transfer will need a 70 ms RTT for the connection establishment, and at least another three RTTs for the actual transfer. This adds up to 280 ms,

making the perceived throughput 286 Kbps. Of course, evolution in application level protocols that are aware of TCP workings allow for better performance in some cases. For instance, the first version of HTTP (Hypertext Transfer Protocol) would open a new TCP connection for every single object in a web page: a page with three images could mean four TCP connections. HTTP/1.1 [56] solves this problem by introducing the concept of persistent connections: in the above scenario, the first TCP connection that is used to fetch the HTML document will be used to fetch the three images as well.

### 4.2.6 Power Consumption

For battery-operated devices like laptops, PDAs and wireless phones, power consumption is a very important aspect. Typically, communicating over a wireless medium consumes more battery power than CPU processing [51]. Of course, TCP was not designed with energy expenditure in mind. However, it has been shown in several studies, including [55], TCP is very efficient in terms of retransmitted segments. TCP manages to have a very low overhead, i.e. it does not waste bandwidth. For example, TCP is shown to achieve almost perfect goodput (defined therein as the ratio of the actually transmitted bytes divided by the number of bytes to be transferred) [57]. On the other hand, energy efficiency does not only depend on the amount of avoidable extra data, but also on the total duration of the connection. Prolonged communication times lead to an excessive power consumption too.

## 4.3 TCP Enhancements Schemes for Wireless Environment

During the past few years many protocols have been proposed to improve the performance of TCP in wireless environments. Fig. 4.3.1 shows the protocols classified into three categories: link layer protocols, end-to-end protocols and split connection protocols. The link layer approach tries to increase the quality of the lossy wireless link. Thus, it hides the characteristics of the wireless link from the transport layer and tries to solve the problem at the link layer. In the end-to-end approach, the TCP sender attempts to handle the losses in a way that improves the performance over regular TCP. Therefore this category maintains the end-to-end semantics of TCP. In the split connection approach, the main idea is to isolate mobility and wireless related

problems from the existing network protocols. This is done by splitting the TCP connection between the mobile host and the fixed host into two separate connections: a wired connection between the fixed host and the base station and a wireless connection between the base station and the mobile host. In this way a wired connection does not need to have any changes in the existing software on the fixed hosts and the wireless connection can use a mobile protocol specialized to provide better performance.

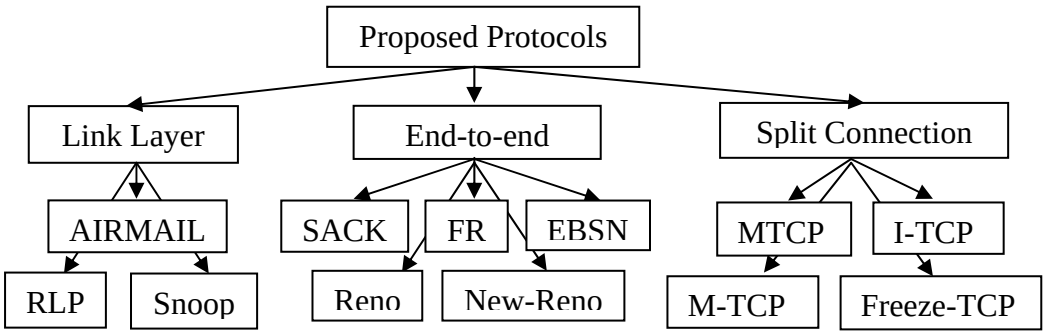


*Figure 4.3.1. Proposed protocols to improve the TCP performance over wireless networks.*

### 4.3.1 AIRMAIL

Ayanoglu et al. [62] proposed a protocol named AIRMAIL (Asymmetric Reliable Mobile Access In Link-layer). The protocol is asymmetric in the sense that the base station is the side responsible for making decisions, whether it is transmitting or receiving. This is because the mobile host has limited battery power and smaller processing capability. Thus the asymmetry places the bulk of the intelligence at the base station with the goal of reducing the processing load at the mobile. Reliability is established by using a combination of automatic repeat request (ARQ) and forward error correction (FEC). The protocol requires the base station to send periodic status messages, while allowing the mobile to combine several acknowledgments into a single one to conserve power. The mobile acknowledgment, unlike the base station, is event driven to reduce the processing load on the mobile. There are three possible levels of FEC: bit-level which is achieved in hardware at the physical layer, byte-level which is done by a per-packet cyclic redundancy check (CRC) and packet-level which is done by allocating some packets for correction which are used for recovery of lost packets without retransmission. Ayanoglu et al. showed that a different level of FEC is needed depending on the characteristics of the mobile channel. Therefore they designed an algorithm that adaptively uses the three levels of channel coding. Thus

the bandwidth expansion due to FEC is minimised. The authors also handle handoff by window management and state transfer.

### 4.3.2 Radio Link Protocol (RLP)

Nanda et al. [63] proposed a point-to-point automatic repeat request (ARQ) for radio channels. Their protocol exploits in order delivery of link-layer packets over the radio link. In their basic protocol, a packet is retransmitted only if the transmitter is sure that it was not received. This makes the protocol very efficient in the sense that the receiver gets no more than one copy of any packet. Feedback packets from the receiver together with sequence number of packets and a send sequence number at the transmitter are used to determine whether the packet was received or not. In the basic protocol, the channel may be forced to be idle during periods when all retransmissions have been completed. An enhanced version of their protocol preemptively retransmits unacknowledged packets during this time. This enhancement results in higher throughput and lower delays. Also, the basic protocol requires frequent full receiver state feedback, which is inefficient if user data is to be carried in the reverse direction. So, the enhancement protocol piggybacks partial receiver state in the reverse channel on user data packets.

### 4.3.3 SNOOP

One proposed solution by Balakrishnan et al. [64] for packet losses caused by high BER is the Berkeley Snoop Module. It is also known as a TCP-aware link layer protocol. The snoop module resides at an intermediate host near the mobile user (i.e. base station). It monitors every TCP data packet that passes through the connection in both directions and maintains a cache of TCP data packets that have not yet been acknowledged by the mobile host. A packet loss is detected either by the arrival of duplicate acknowledgments or by a local timeout. To implement the local timeout, the module has its own retransmission timer. Using the information, the snoop module detects lost packets and performs local retransmissions to the mobile. Thus, the base station hides the packet loss from the fixed host, hence avoiding its invocation of an unnecessary congestion control mechanism. Later, the authors added selective

retransmissions from the base station to the mobile host to improve the performance of the Snoop module [65].

### 4.3.4 Reno

Reno TCP is like a regular (Tahoe) TCP, except it includes fast recovery [58][69]. The TCP sender enters fast recovery if it times out or receives three duplicate acknowledgements. The sender then retransmits the lost packet and reduces ssthresh by half. Unlike TCP Tahoe, the sender then does not enter slow start. It reduces the value of the congestion window (cwnd) by half, then increments it by one for each duplicate acknowledgement received. When a "new ACK is received, the sender exits fast recovery, sets cwnd to ssthresh and enters the congestion avoidance phase when it increases the window linearly. A "new" ACK is an ACK with a value higher than the highest seen so far – a non-duplicate ACK.

### 4.3.5 New-Reno

This implementation was proposed by Hoe [70]. It is a modification of Reno TCP for improving the performance of Reno when multiple packets are lost within the same window. To achieve that, it modifies the action taken upon receiving a "new" ACK. Unlike Reno, New-Reno does not exit fast recovery unless all data transmitted at the start of the fast retransmission phase have been acknowledged. Thus, it only exits fast recovery when it receives an ACK for the highest sequence number sent. Thus, a "new partial" ACK (an ACK that represents some, but not all, of the outstanding packets) is used as indication that the packet immediately following the acknowledged packet in the sequence space has been lost, and should be retransmitted. When multiple packets are lost in the same window, Reno recovers by waiting for serial retransmission timeout. This is because in Reno, the reception of a new partial ACK causes the sender to exit fast recovery by "deflating" the window.

### 4.3.6 SACK

Selective Acknowledgments was defined in RFC 2018 by Mathis et al. [66] and later extended in RFC 2883 by Floyd et al. [67]. SACK TCP further improves TCP performance by allowing the sender to retransmit packets based on the selective

ACKs provided by the receiver. The implementation constitutes a SACK field that contains a number of SACK blocks. The first block reports the most recently received packets. The additional blocks repeat the most recently reported SACK blocks.

TCP SACK uses the basic congestion control algorithms and uses retransmit timeouts as a last option for recovery. The main difference is the way it handles the loss of multiple packets from the same window. In fast recovery, SACK enters into fast recovery upon receiving duplicate ACKs. It then retransmits a packet and cuts its congestion window in half. In addition, SACK has a new variable called the pipe, and a data structure called the scoreboard. The pipe is incremented when the sender sends a new or a retransmitted packet. It is decremented when the receiver receives a new packet. This is indicated when the sender receives a duplicate ACK with a SACK option. The scoreboard stores ACKs from previous SACK options, allowing the sender to retransmit packets that are implied to be missing at the receiver, i.e. the senders exits fast recovery when all the outstanding data are acknowledged.

### 4.3.7 Fast Retransmission

Caceres and Iftode were among the first to address the impact of mobility and wireless networks on the performance of TCP, [37]. They used the fast retransmission scheme employed by current implementation of TCP to provide smooth handoffs on networks that lost packets during handoffs. They made minimal changes to the Mobile IP software on the mobile host to signal the completion of the handoff. Consequently the TCP software on the mobile host forwards the signal to the fixed host. The TCP on the fixed host then invokes the retransmission scheme. This forces the fixed host to reduce the congestion window to half and retransmit one segment immediately. Experiments showed that this scheme causes connection to resume communication after 50 ms as opposed to 650 ms using regular TCP. In this scheme, triplicate acknowledgements are used to invoke the fast retransmission procedure, then no changes in the TCP software is required. Otherwise, minimal changes to TCP are needed if a specially marked TCP acknowledgement packet containing the sequence number of the last data packet successfully received by the mobile host is used.

Caceres and Iftode also recommended the use of small cells with little or no overlap between them. They relied on the fact that little or no overlap between small cells provides high aggregate bandwidth, support low-powered mobile transceivers, and provide accurate location information.

### 4.3.8 Explicit Bad State Notification (EBSN)

The effect of local error recovery and explicit feedback by the base station was studied by Bakshi et al. [68]. Their idea is that although local recovery by the base station using link layer protocols is found to improve performance, timeouts can still occur at the fixed host, causing redundant packet retransmissions. Their experiments showed that explicit feedback from the base station to the fixed host can completely eliminate the possibility of timeouts occurring at the fixed host, while the wireless link is in a bad state. When a wireless link is in a bad state, little data is able to reach the mobile, causing packets from the fixed host to be queued at the base station. It would appear that using an existing feedback mechanism, like Explicit Congestion Notification (ECN) would solve the problem. Sending an ECN from the base station to the fixed host would decrease the flow of new packets to the base station, and thus prevent unnecessary timeouts. But it does not prevent timeouts of packets already on the network. The solution to this problem is to send an Explicit Bad State Notification (EBSN) that would reset the timeout at the fixed host during local recovery. The EBSN will totally eliminate timeouts even for packets that had already been put on the network before the wireless link entered the bad state, and at the same time not cause the fixed host to decrease its congestion window. EBSN was found to provide up to 100% performance improvement over regular TCP in wide-area networks, and up to 50% in local area networks, which is very close to the theoretical maximum.

Bakshi et al. also investigated the effect of packet size variation. They showed that choosing an optimal packet size could improve performance by up to 30%. They found that the optimal packet size depends on the error conditions of the wireless link. A fixed table could be maintained at the base station that maps a particular wireless link error characteristic to its optimal packet size.

### 4.3.9 Indirect-TCP & MTCP

Indirect-TCP protocol is a solution that uses a split connection approach to solve packet losses caused by high BER [60]. I-TCP splits connection between a fixed host and a mobile host in two parts: one between fixed host and Mobile Support Station (MSS) and the other between the MSS and the Mobile Host (MH) as shown in Fig. 4.3.2. TCP in the FH-to-MSS link and a modified version of TCP (includes mobility features) is used in the MSS-to-MH link. In I-TCP the MSS immediately acknowledges information sent from the TCP sender even though this has not yet been received by the TCP receiver. This breaks the end-to-end TCP semantics since the end-to-end congestion avoidance standard procedures are not maintained as usual and the sender always considers information sent has been received; this could be a problem for some applications. Furthermore, the TCP sender may flood the MSS with data while the MSS is unable to reach the mobile host.

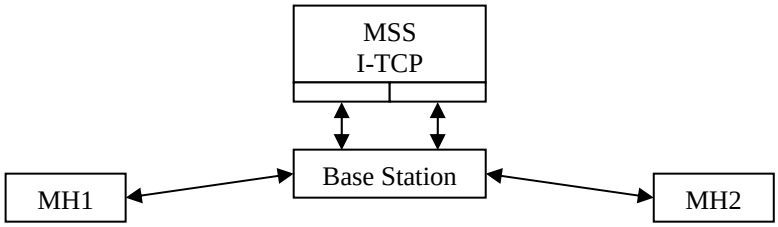


*Figure 4.3.2. Indirect TCP approach on the MH to MH configuration.*

The MTCP protocol of [61] is very similar to I-TCP. It also splits a TCP connection in two – one from the MH to MSS and another from the MSS to the FH (i.e., the Fixed Host or sender). The MH to MSS connection passes through mobile host protocol, a session layer protocol. Two implementations for mobile host protocol are proposed. In the first, mobile host protocol uses TCP over the wireless link to the MSS. In the second implementation, a selective repeat protocol, SRP, is used over the wireless link. SRP lies below the socket layer but above TCP. SRP is designed to recover quickly from high and bursty packet losses. Then, the sender in turn retransmits the missing packet. SRP also can recover more than one packet in one round trip time.

Both I-TCP and MTCP achieve better throughputs than standard TCP for the following reason: the node where the connection is split is, hopefully, one or two hops away from the MH's cell and can adapt more quickly to the dynamic mobile

environment because the round trip time (RTT) is very small. This means that even if the MSS sender (the node where the connection is split) reduces its congestion window to one because of lost acknowledgments from the mobile, the window will quickly build up to the maximum again when new acknowledgments arrive from the mobile because of the short RTT. MTCP/SRP also increases throughput by using selective repeat which reduces duplicate sends.

### 4.3.10 M-TCP

Brown and Singh [71] propose a hierarchical based split connection approach as shown in Fig. 4.3.3 for M-TCP where their goals are to support dynamically bandwidth availability, frequent long periods of disconnection dues to handoffs and power saving features.

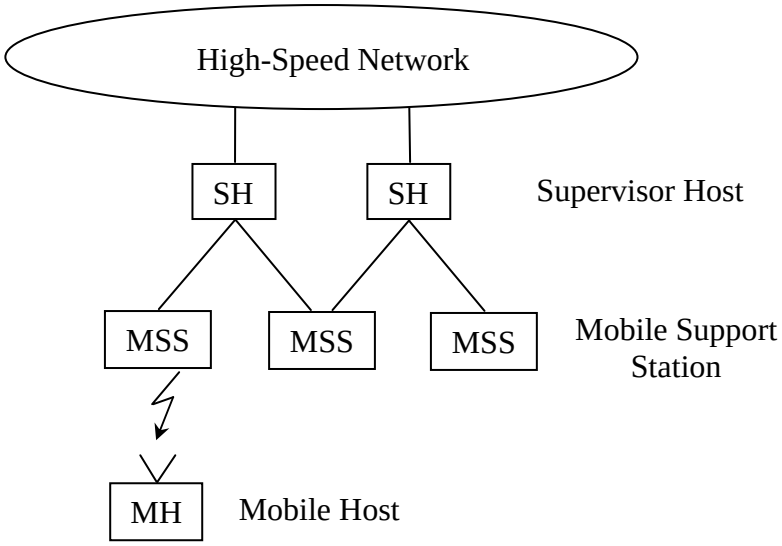


*Figure 4.3.3. M-TCP Proposed Architecture*

This protocol is designed to dynamically assign a fixed amount of bandwidth to each mobile node based on their changing needs. It performs local error recovery to solve problems resulting from lossy wireless link. It reduces the power consumption at the mobile node by ensuring that the numbers of duplicate packets are kept small. The protocol also ensures efficient handoff and deals with the problems caused by long or frequent disconnections.

The TCP connection between the fixed host and the mobile host is split at the supervisor host as shown in Fig. 4.3.4.

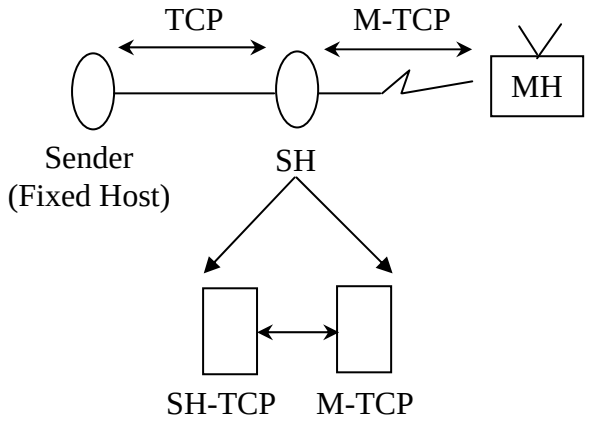


*Figure 4.3.4. TCP connection is split between fixed host and mobile host via supervisor host.*

Regular TCP is used on the fixed host (between the fixed host and supervisor hosts), while a special version of TCP is used over the wireless link. The TCP client at the supervisor host is called SH-TCP while that on the mobile host is called M-TCP. When the SH-TCP receives data from the sender, it passes it to the M-TCP client, which replies by an acknowledgement. The SH-TCP passes this acknowledgement to the sender. To ensure that the sender does not go into congestion control when the ACK does not arrive because the mobile host temporarily got disconnected, the SH-TCP does not forward the ACK of the last byte to the sender until it knows that the mobile host has disconnected. This forces TCP to go into persist mode by setting the window size to zero. Therefore, TCP will not suffer from retransmit timeouts, nor will it close its congestion window. When the mobile host reconnects, the TCP sender is ready to transmit at full speed. If the mobile host did not disconnect but has little available bandwidth, the SH-TCP shrinks the sender's window before it exponentially backs off its retransmission timer. At the M-TCP client, when the mobile host disconnects, it freezes all its timers to ensure that the congestion control is not invoked. When it reconnects, it unfreezes all the timers and resumes normal operation, as if it did not lose any data during disconnection.

### 4.3.11 Freeze-TCP

A mechanism called “Freeze-TCP” that allows the mobile host to inform the sender for a possible disconnection has been proposed by Goff et al. [72]. The goal of this approach is to enhance TCP performance over the wireless link with minimal packet loss and to avoid congestion control during the period of disconnection. This scheme requires only the mobile host’s TCP code to be modified without any modification made to the intermediary i.e. to the base station or the sender.

During a handoff or a possible disconnection, the mobile host will sent out a few zero window advertisement (ZWA) with zero window size. When the sender receives a zero window advertisement, the sender will freeze all re-transmit timers, stop transmitting data and enter into persist mode. During persist mode, the sender will send probes called zero window probes (ZWPs) periodically to find out whether the window size of the receiver has been increased. When the window size of the receiver has been increased, the sender will send data at full speed as shown in Fig. 4.3.5 and thus improve TCP throughput.

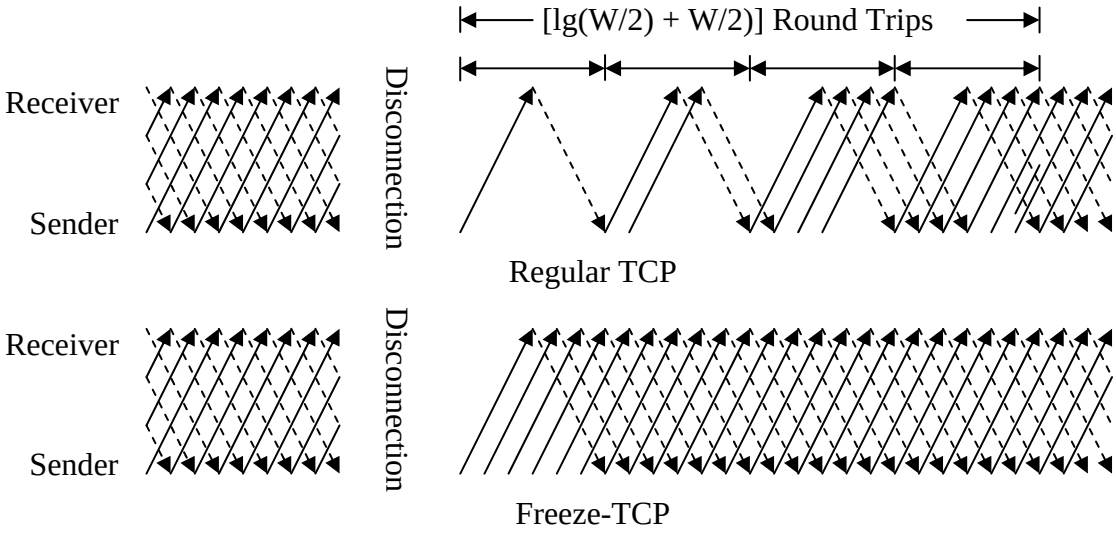


*Figure 4.3.5. Illustration of increased throughput due to Freeze-TCP.*

Since ZWPs are exponentially backed off, there is the possibility of substantial idle time after the reconnection. This could happen if the disconnection period is too long and reconnection happened immediately after losing a ZWP from the sender. To avoid this idle time, as soon as a connection is re-established, the receiver will leave persist mode and send three copies of ACK for the last data segment it received prior to the

disconnection. This scheme is known as TR-ACKs (Triplicate Reconnection ACKs), [37]. In standard TCP, packet retransmissions are exponentially backed off therefore post reconnection idle time can occur as well. The author suggests for a fair comparison, the standard TCP on the sender side be also modified to optionally send TR-ACKs. This way, the effect of only Freeze-TCP mechanism (i.e., forcing the sender into ZWP mode prior to a disconnection) can be isolated.

For M-TCP, there is no advantage in holding back the ACK to the last byte because even when the mobile host was disconnected, the fixed host could still signal the sender on behalf of the client. In the case of Freeze-TCP, since changes are restricted to the client end, holding back the ACK for the last byte does not help. Freeze-TCP will avoid any re-packetization penalty at the sender end (which M-TCP might incur because it holds back the ACK to the last byte).

## 4.4 Summary of proposed protocols

| | Airmail | RLP | SNOOP | FR | EBSN | MTCP | I-TCP | M-TCP | Freeze-TCP |
|---|---|---|---|---|---|---|---|---|---|
| Handle High BER | Yes | Yes | Yes | | Yes | Yes | Yes | | |
| Handle Bursty error | Yes | | Yes | | Yes | Yes | | | |
| Handle handoff | Yes | | Yes | Yes | | | Yes | Yes | Yes |
| Long Disconnection | | | | | | | Yes | Yes | Yes |
| Bandwidth | Yes | Yes | | | | | | Yes | Yes |
| Cell Size | | | | Yes | | | | | Yes |
| Power scarcity | Yes | | | | Yes | | | Yes | Yes |
| Serial timeouts | | | | | Yes | Yes | Yes | Yes | Yes |
| Packet size variation | | | | | Yes | Yes | | | |
| End-to-end TCP semantics | Yes | Yes | Yes | Yes | Yes | | | Yes | Yes |
| Handle encrypted traffic | | | | | | | | | Yes |
| Requires Intermediaries | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | |

*Table 4.4.1. Summary of proposed protocols for TCP enhancements in wireless environment.*

Table 4.4.1 summarizes the problems handled by the protocols discussed earlier. Reno, New-Reno and SACK are not included in the table since they were basically designed for wired networks, hence they only satisfy the end-to-end TCP semantics.

The main advantages of link layer protocols are to maintain the end-to-end semantics of TCP. They are good at reducing bit error rate by one or two orders of magnitude. Therefore link layer protocols perform very well in environments with high bit error rates. However link layer protocols suffer in the case of long or frequent disconnections.

End-to-end protocols suffer from the major drawback that they require modification to TCP at the fixed host code, unlike Freeze-TCP that only requires modification to the mobile host TCP code. This saves cost and doesn't required any recompilation and relinking of the existing applications on the fixed host. This is one of the major advantages of split-connection protocols. They shield the mobility and wireless problems from the fixed host. In addition, they can handle disconnections efficiently.

In conclusion, let us list the features that need to be satisfied in a standard mobile TCP. It should:

- avoid erroneously triggering congestion control mechanisms on the fixed host;
- avoid serial timeout problem on the fixed host;
- handle high bit error rate by solving problems arises in the wireless link;
- handle frequent handoff effectively;
- handle frequent and long disconnections of the mobile host;
- take limited bandwidth and power scarcity of mobile hosts into consideration;
- use dynamic packet size for mobile hosts with dynamic bandwidth availability;
- provide end-to-end TCP semantics; and
- have ability to handle encrypted traffic effectively.

# 5.0 Implementation of Freeze-TCP with Cellular IP

Until now TCP performance has not been studied taking into account handoffs within the networks. The handoffs of Cellular IP with standard TCP in the testbed have been enhanced with Freeze-TCP [72] which only requires the modification of the TCP code of the mobile host.

As discussed in Section 4.3.11, Freeze TCP is a new technique used to get over problems with TCP behavior in mobile networks. It incorporates splitting the end to end connection into two, (wired and wireless) connections, pushing the sender into "persist mode" during handover by sending zero window probes to freeze all re-transmit timers and handling long and frequent disconnections.

First, the modification is done on the Linux kernel TCP code to implement Freeze-TCP in the mobile host. Then, Cellular IP source code is modified to activate Freeze-TCP during the periods of handoffs. For details on the modification of the Linux Red Hat 7.3 kernel source code with Freeze-TCP and the modification of Cellular IP with Freeze-TCP source code, please refer to Appendix A and B. The advantage of Freeze-TCP is that it avoids slow start congestion during handoff, whereas with standard TCP goes into slow start when it detects a disconnection during the transfer as shown previously in Fig. 4.3.5. With this enhancement, TCP throughput performance with handoffs should greatly improve.

## 5.1 Performance Evaluation of Cellular IP with Freeze-TCP

Cellular IP protocol currently supports two handoff schemes: forced handoff (manually handover to other base station) and Signal-to-Noise ratio (SNR) based handoff. TCP performance measurements are taken by mobile host telnetting to the web server and

using the ping command to ping mobile host from web server as shown in Fig. 5.1.1 and Fig. 5.1.2. Using Ethereal, we observed a stream of TCP packets transmitted following the ping command when they are subjected on number of forced or SNR based handoffs within 60 seconds. The number of forced or SNR based handoffs varied from two versions of TCP: standard TCP and Freeze-TCP were considered.

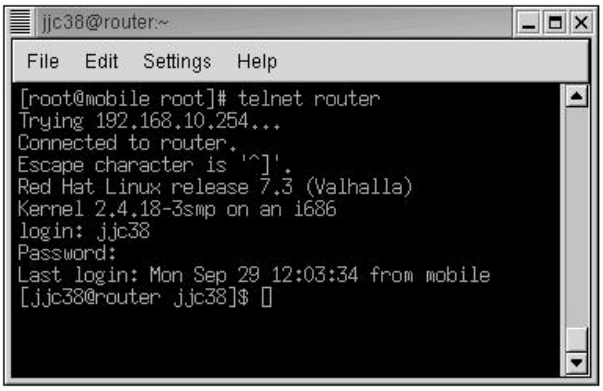


*Figure 5.1.1. Mobile host telnet to web server.*

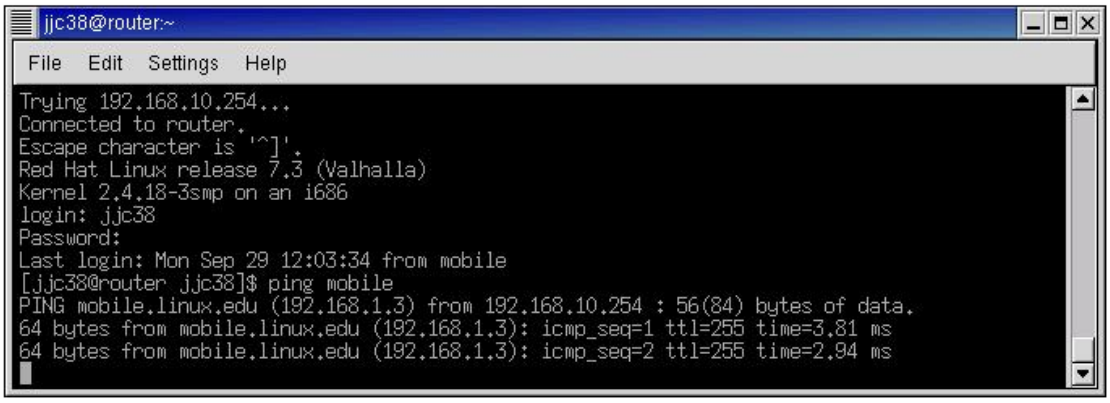


*Figure 5.1.2. From web server, mobile host ping back itself through telnet.*

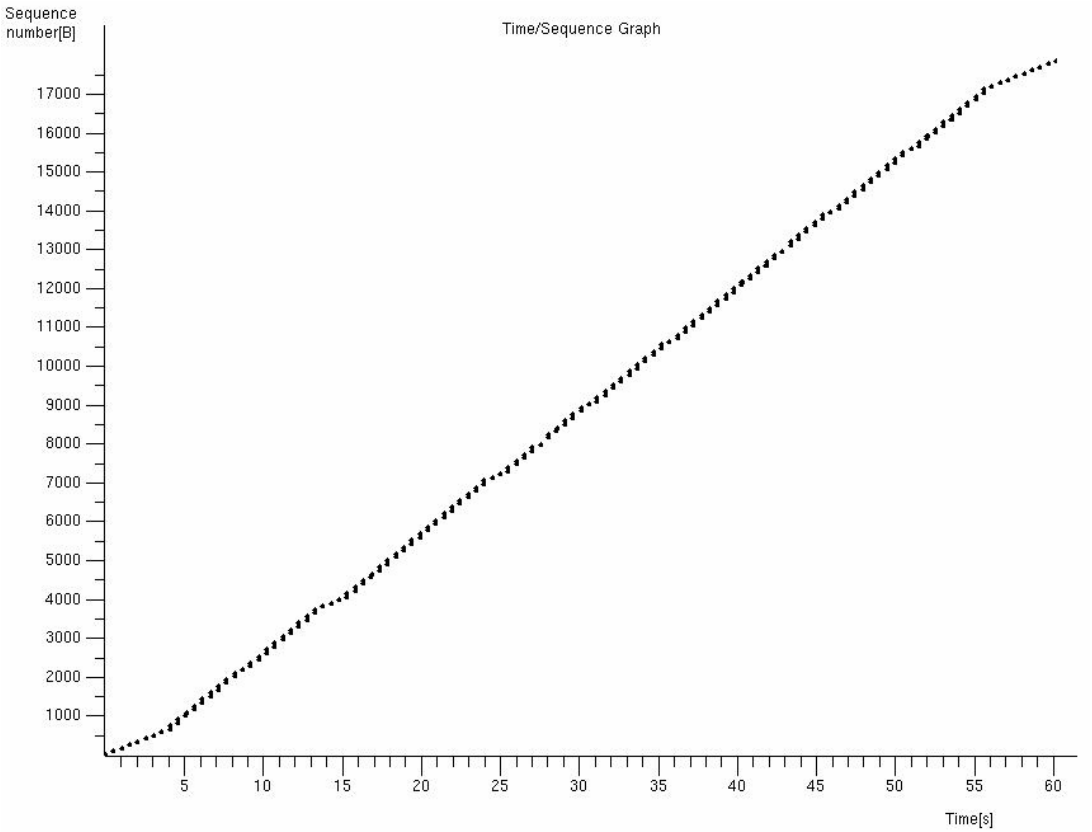


*Figure 5.1.3. Sequence graph of 10 forced handoffs with standard TCP in 60s.*

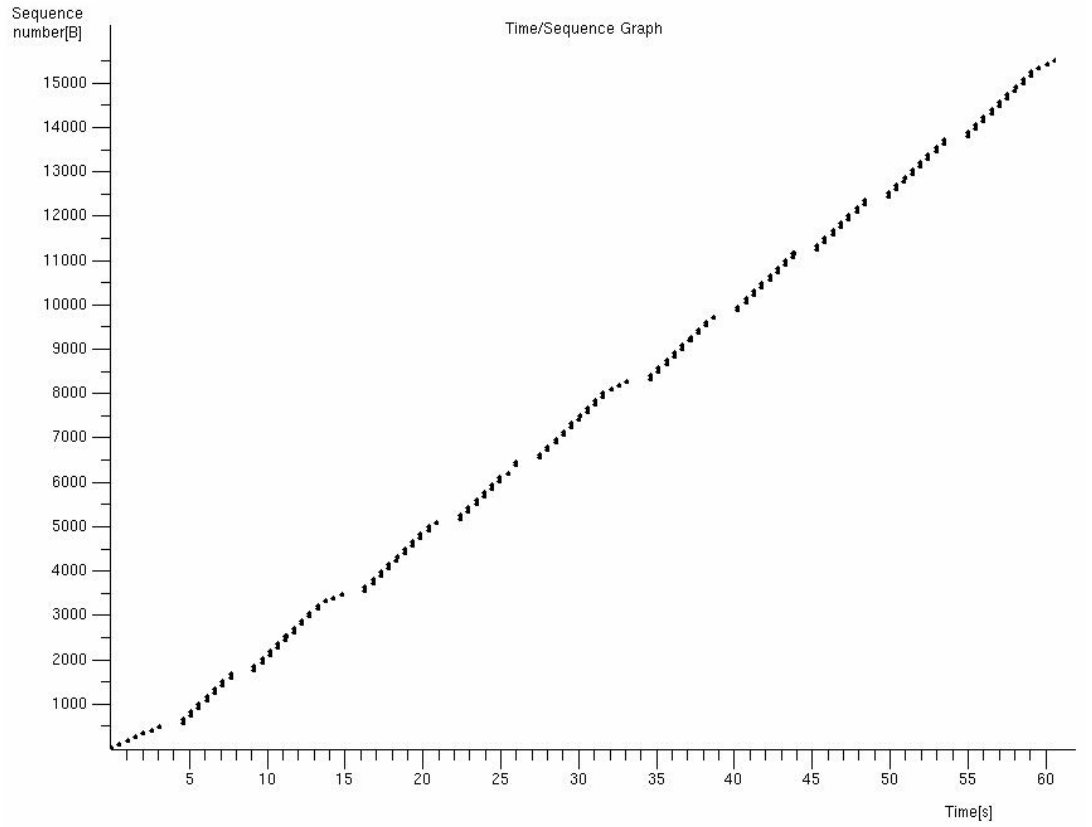


*Figure 5.1.4. Sequence graph of 10 forced handoffs with Freeze-TCP in 60s.*

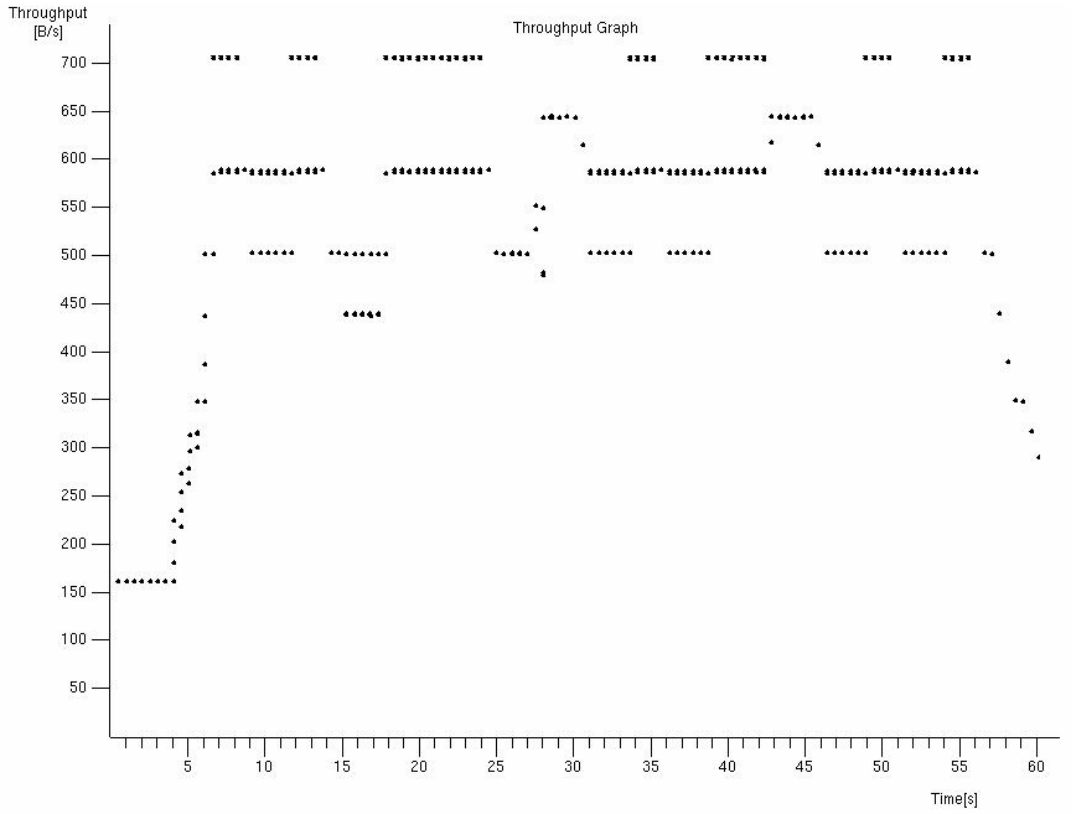


*Figure 5.1.5. TCP throughput graph of 10 forced handoffs with standard TCP in 60s.*

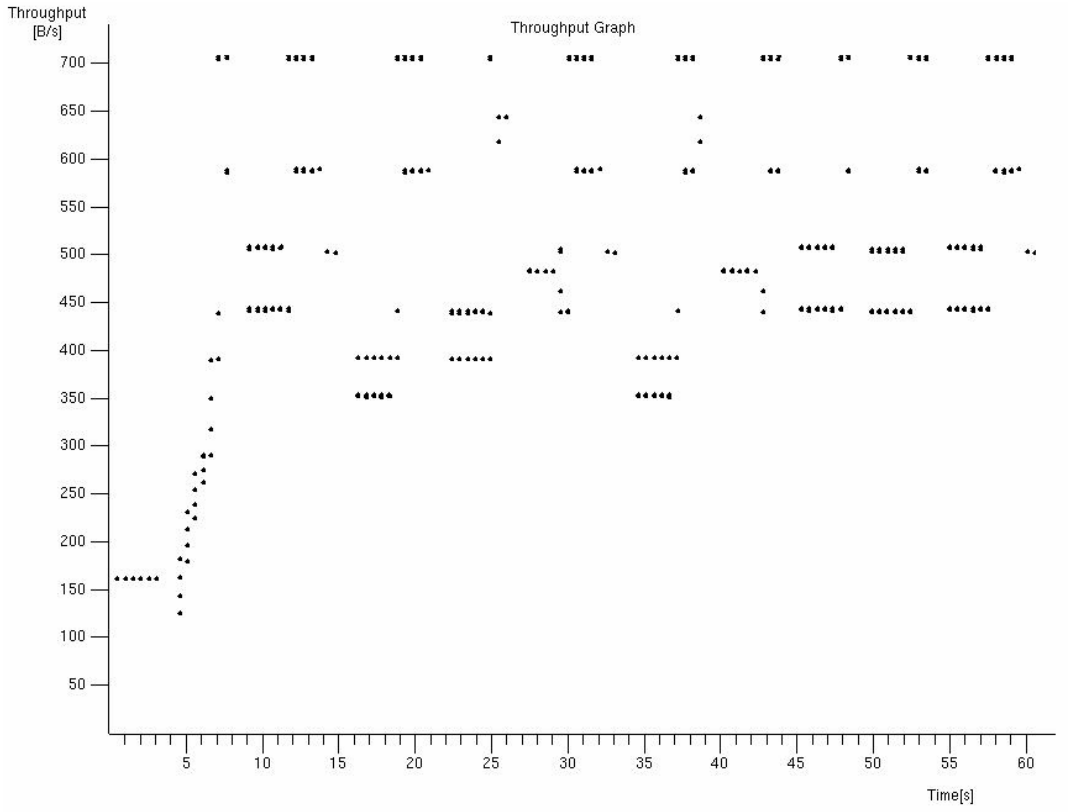


*Figure 5.1.6. TCP throughput graph of 10 forced handoffs with Freeze-TCP in 60s.*

Figs. 5.1.3 and 5.1.4 show the graphs of sequence number of TCP packets generated by ping command with 10 forced handoffs with standard TCP and 10 forced handoffs with Freeze-TCP within 60s respectively. Fig. 5.1.4 shows a bit clearer of the number of forced handoffs that Freeze-TCP performed, where the sequence number of TCP packets halted for at least 1 second in each handoff. As we observed, 10 forced handoffs with standard TCP produces more duplicated packets than 10 forced handoffs with Freeze-TCP within 60s. The results show that as more duplicate acknowledgements are produced, this results in a higher TCP throughput during the period of handoffs from the old base station to the new base station. This will caused significant network congestion when more than one mobile host perform regular handoffs on the same subnet. TCP throughput graphs shown in Fig. 5.1.5 and Fig. 5.1.6 reveal that forced handoffs with standard TCP produce a slightly higher throughput than forced handoffs with Freeze-TCP during the handoff period. Here, forced handoffs with Freeze-TCP introduce approximately a 1s delay each time between turning on and off Freeze-TCP which generates approximately a 10% reduction of TCP throughput. However, the 1s delay introduced during the turning on and off of Freeze-TCP have been observed to have no effect on the quality of the NIKE advertisement during the streaming session. Thus, the results show that Cellular IP handoffs with Freeze-TCP reduce network congestion during regular handoffs.

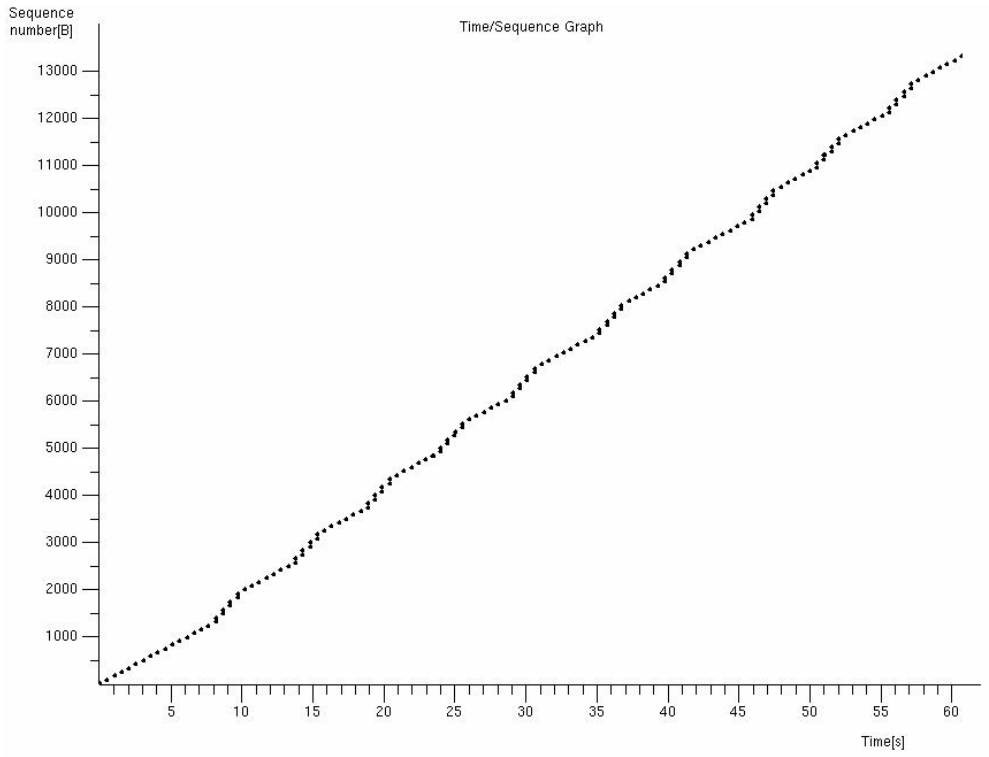


*Figure 5.1.7. Sequence graph of 10 SNR handoffs with standard TCP in 60s.*

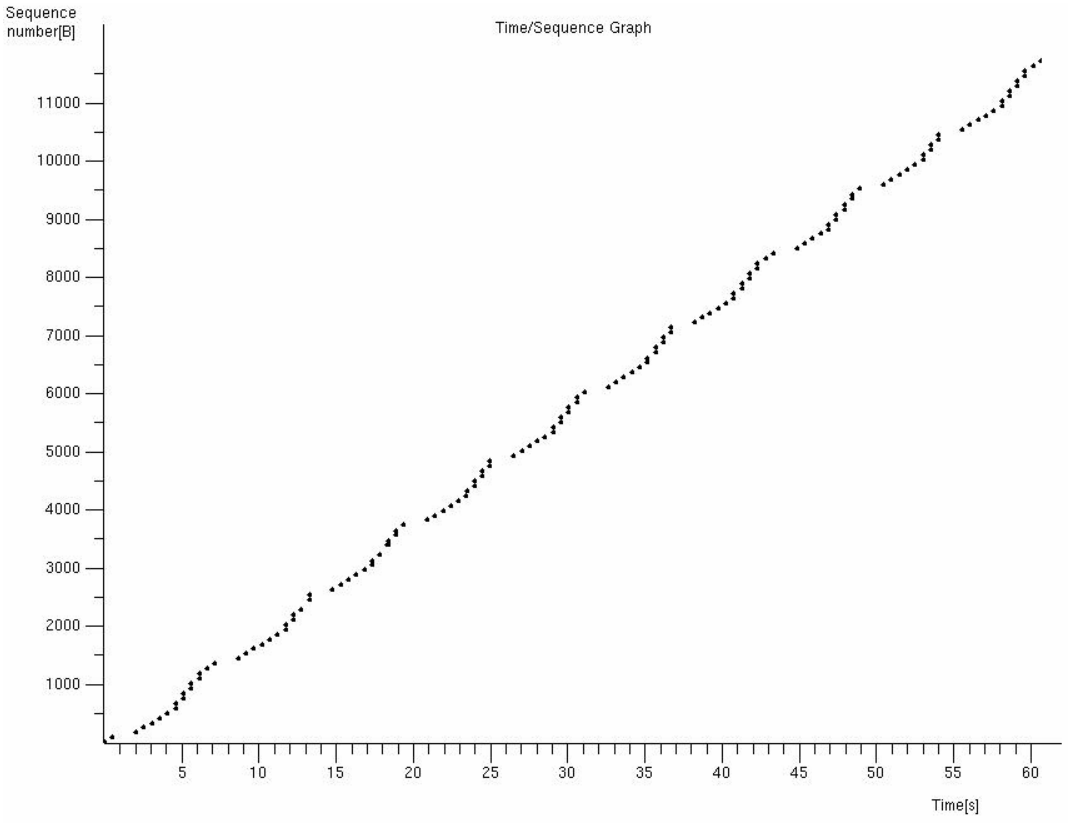


*Figure 5.1.8. Sequence graph of 10 SNR handoffs with Freeze-TCP in 60s.*

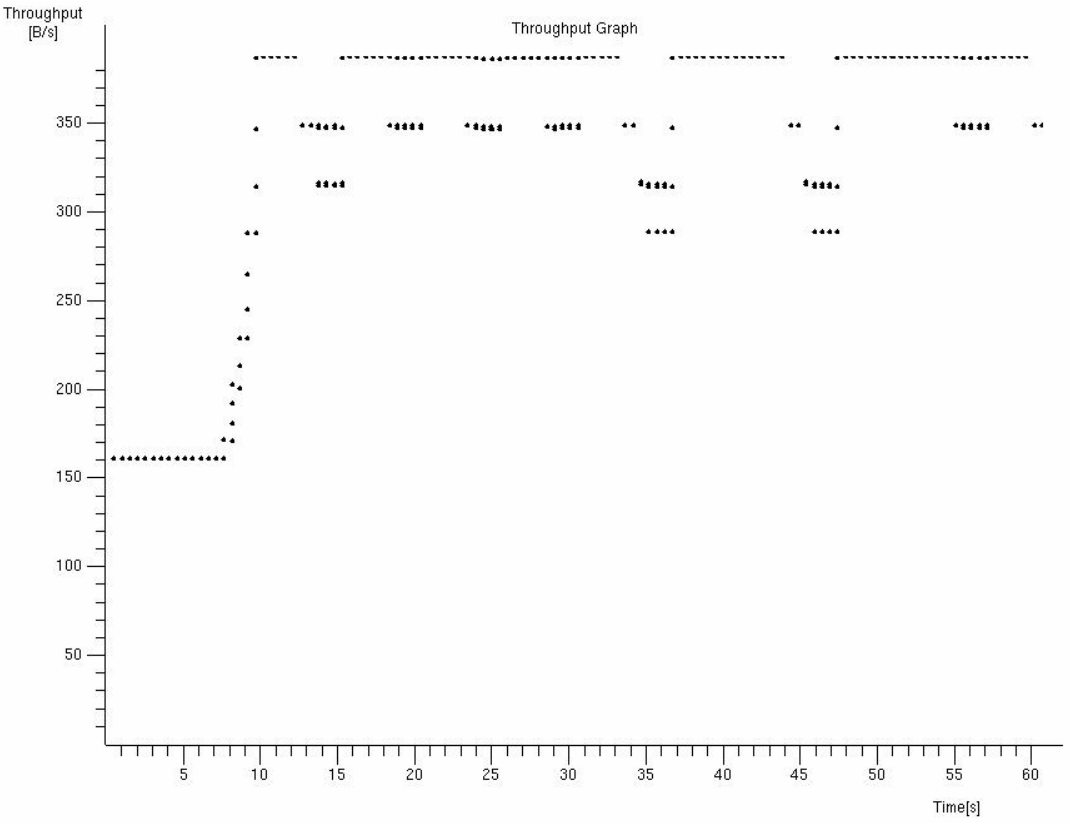


*Figure 5.1.9. TCP throughput graph of 10 SNR handoffs with standard TCP in 60s.*

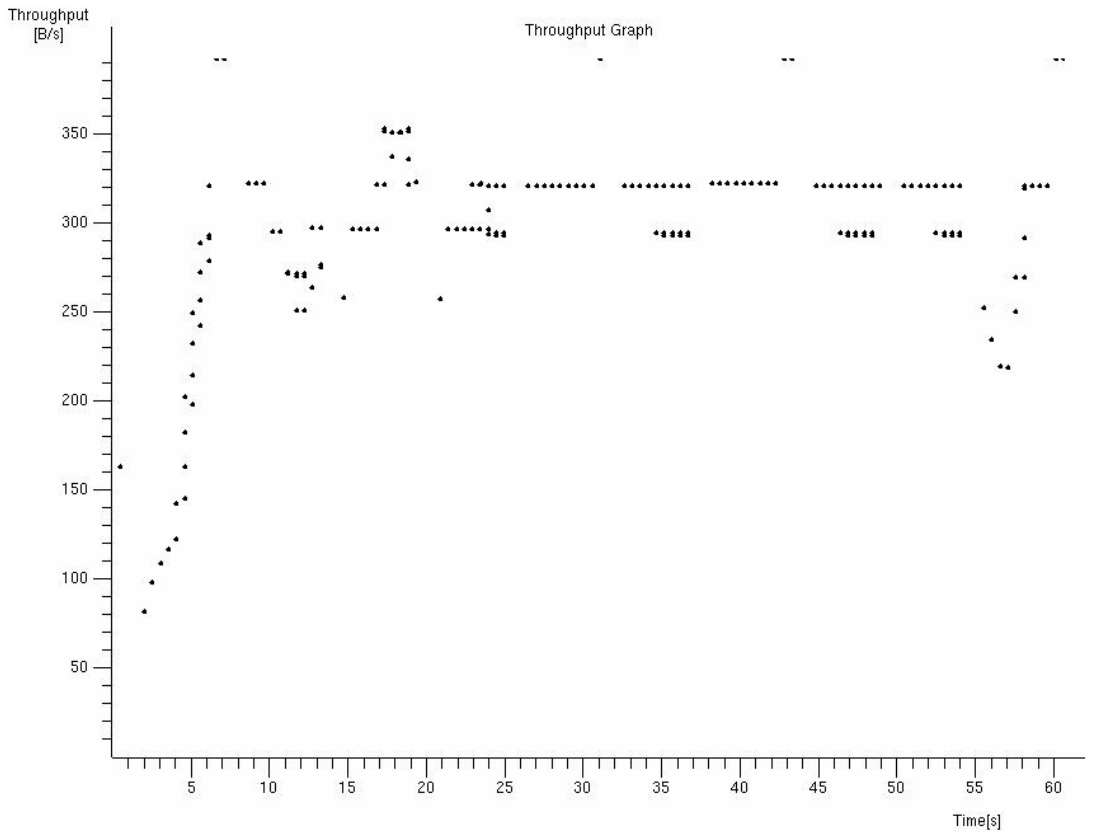


*Figure 5.1.10. TCP throughput graph of 10 SNR handoffs with Freeze-TCP in 60s.*

Comparing SNR handoffs with standard TCP and SNR handoffs with Freeze-TCP as shown in Fig. 5.1.9 and Fig. 5.1.10, the TCP throughput is reduced with Freeze-TCP compared with standard TCP by 6%. As shown in table 5.1.1, SNR handoffs with Freeze-TCP attained 35% better in TCP throughput than forced handoffs with Freeze-TCP and 41% better in TCP throughput than forced handoffs with standard TCP. Therefore, the experiment concludes that SNR handoffs with Freeze-TCP perform the best of the handoff schemes. All other results, for scenarios varying from one handoff to ten handoffs with the four different handoff schemes, are summarised in Appendix C.

| 10 Handoffs in 60s | Forced – TCP | Forced – FTCP | SNR – TCP | SNR – FTCP |
|---|---|---|---|---|
| Avg. bytes/sec | 1452.468 | 1307.208 | 907.319 | 853.588 |

*Table 5.1.1. Summary of TCP throughput with different handoff schemes.*

# Chapter 6.0 Conclusions and Future Work

## 6.1 Conclusions

Chapter 2 presented produced a survey on the micro-mobility protocols that are currently proposed by researchers. Only Cellular IP and Hierarchical Mobile IP have been found to be popular and actively pursued by researchers. Chapter 3 described the creation of a working micro-mobility testbed implementing the Cellular IP protocol. The choice of Cellular IP protocol was made because that the software is free to download and it is cheap and easy to install, build and configure. The testbed is built with a minimum of two base stations, one gateway, one web server and one mobile host. The results show that the Cellular IP mobile host successfully performed seamless handoffs with no packet loss when streaming a Nike advertisement with MpegTV software. The best quality of the video is maintained up to 100m outdoors.

Chapter 4 presented an investigation of the issues regarding TCP in the wireless environment. Various TCP schemes proposed over the years for improving TCP performance in wireless networks were also discussed. Whenever a new proposed TCP scheme is introduced, the designer has to take into account all of the drawbacks that standard TCP faces in the wireless environment.

Freeze-TCP scheme was chosen to enhance the testbed because other TCP scheme requires TCP code to be modified at the intermediaries e.g. the base stations and any other nodes that are associated with it. However Freeze-TCP requires only the modification of TCP code in the mobile host. Chapter 5 presents a successful implementation of the Cellular IP protocol with Freeze-TCP to improve the TCP performance during the handoffs. The results have shown that Forced handoff with standard TCP produces the worst case but SNR handoff with Freeze-TCP provides a 41% improvement in TCP throughput during the handoff period. SNR handoff with Freeze-TCP proved to be the best handoff scheme in the experiment in respect of the TCP performance during the handoff period.

## 6.1 Future Work

Implementing another micro-mobility protocol like HAWAII or Hierarchical Mobile IP into the testbed would round of the study and give a more thorough comparison of the micro-mobility protocols. Another thing that can be done is that Mobile IP, a macro-mobility protocol, can be implemented to interwork with Cellular IP. This enhancement could provide a greater coverage area for the mobile host by being able to roam into networks using possibly different wireless technologies while maintaining connection to the Internet.

Currently Cellular IP software only supports IPv4. This can be extended to support IPv6 as 3G cellular services like SIP use 128-bit IP addresses for signalling and messaging their gateway to establish a connection.

Different other wireless-enhanced TCP schemes like Snoop, I-TCP and M-TCP can also be implemented to the testbed but would require time in hacking the kernel source code of the sender and the receiver.

Lastly, to improve the accuracy of the performance measurements, the computer parts of every node within the network especially the CPU and memory of the system require an upgrade to at least the CPU of 800 MHz clock and 512 MB memory for base stations, gateway and mobile host. Then many applications would run simultaneously to capture data results to analyse TCP performance for the wireless network.

# References


[1] C. E. Perkins (Ed.), “IP Mobility Support for IPv4”, RFC 3344, IETF, Aug. 2002.

[2] C.E. Perkins and D. B. Johnson, "Route Optimization in Mobile IP", Internet Draft, draft-ietf-mobileip-optim-11.txt, Sep. 2001. Work in Progress.

[3] G. Montenegro, “Reverse Tunneling for Mobile IP, revised”, RFC 3024, IETF, Jan. 2001.

[4] E. Gustafsson, A. Jonsson, and C. E. Perkins, “Mobile IPv4 Regional Registration” Internet Draft, draft-ietf-mobileip-reg-tunnel-06.txt, March 2002. Work in Progress.

[5] J. Manner and M. Kojo, “Mobility Related Terminology”, Internet Draft, draft-ietf-seamoby-mobility-terminology-05.txt, Nov. 2003, Work in Progress.

[6] IETF Mobile IP Working Group, http://www.ietf.org/html.charters/mobileip-charter.html.

[7] IETF Seamoby Working Group, http://www.ietf.org/html.charters/seamoby-charter.html.

[8] K. El Malki (Ed.), “Low Latency Handoffs in Mobile IPv4”, Internet Draft, draft-ietf-mobileip-lowlatency-handoffs-v4-05.txt, Dec. 2002. Work in Progress.

[9] J. Kempf, “Dormant Mode Host Alerting (‘IP Paging’) Problem Statement”, RFC 3132, IETF, June 2001.

[10] A.T. Campbell et al., “Cellular IP”, Internet Draft, draft-ietf-mobileip-celllularip-00.txt, June 2000. Work in Progress.

[11] A.T. Campbell, J. Gomez, S. Kim, A. G. Valko, and Chieh-Yih Wan, “Design Implementation and Evaluation of Cellular IP”, IEEE Wireless Communication Magazine, vol. 7, no. 4, Aug. 2000, pp. 42-49.

[12] A.G. Valko, “Cellular IP: A New Approach to Internet Host Mobility”, Comp. Commun. Review, vol. 29, no. 1, Jan. 1999, pp. 50-65.

[13] R.Ramjee et al., “IP Micro-mobility support through HAWAII”, Internet Draft, draft-ietf-mobileip-hawaii-01.txt, July 2000. Work in Progress.

[14] E. Gustafsson, A. Jonsson, C. E. Perkins, “Mobile IPv4 Regional Registration”, Internet Draft, draft-ietf-mobileip-reg-tunnel-06.txt, March 2002. Work in Progress.

[15] H. Haverinen, J. Malinen, “Mobile IP Regional Paging”, Internet Draft, draft-haverinen-mobileip-reg-paging-00.txt, June 2000. Work in Progress.

[16] P. Calhoun et al., “Foreign Agent Assisted Hand-off”, Internet Draft, draft-calhoun-mobileip-proactive-fa-03.txt, Nov. 2000. Work in Progress.

[17] K. El-Malki and H. Soliman, “Fast handoffs in Mobile IPv4”, Internet Draft, draft-elmalki-mobileip-fast-handoffs-03.txt, Sept. 2001. Work in Progress.

[18] A. O'Neill, G. Tsirtsis, and S. Corson, “Edge Mobility Architecture”, Internet Draft, draft-oneill-ema-01.txt, March 2000. Work in Progress.

[19] A. O'Neill and S. Corson, “An Approach to Fixed/Mobile Converged Routing”, Technical Report TR-2000-5, University of Maryland, Institute for Systems Research, March 2000.

[20] V. D. Parks and M. S. Corson, “Temporally-Ordered Routing Algorithm (TORA) version 1: functional specification, Internet draft, draft-ietf-monet-tora-spec-02.txt, Oct. 1999. Work in Progress.

[21] V. D. Parks and M. S. Corson, “A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks”, IEEE Proc. INFOCOM, April 1997.

[22] S. Das, A. Misra, P. Agrawal and S.K. Das, “TeleMIP: Telecommunication Enhanced Mobile IP Architecture for Fast Intra-Domain Mobility”, IEEE PCS Magazine, vol. 7, no. 4, Aug. 2000, pp. 50-58.

[23] A. Misra, S.Das, A.McAuley, A. Dutta and S.K. Das, “IDMP: An Intra-Domain Mobility Management Protocol using Mobility Agents”, Internet draft, draft-mobileip-misra-idmp-00.txt, July 2000. Work in Progress.

[24] Cellular IP, http://www.comet.columbia.edu/cellularip (Sited Jan. 2004)

[25] Dynamics Mobile IP, http://dynamics.sourceforge.net/ (Sited Jan. 2004)

[26] Red Hat, http://www.redhat.com/download/mirror.html (Sited Jan. 2004)

[27] Nist Net, http://snad.ncsl.nist.gov/itg/nistnet (Sited Jan. 2004)

[28] Ethereal Network Analyser, http://www.ethereal.com (Sited Jan. 2004)

[29] Linux PCMCIA, http://pcmcia-cs.sourceforge.net (Sited Jan. 2004)

[30] Proxim, http://www.orinocowireless.com (Sited Jan. 2004)

[31] Richard Alena, Darin Evenson, Victor Rundquist, “Analysis and Testing of Mobile Wireless Networks”, NASA Ames Research Center, IEEEAC, 2002.

[32] Microsoft WebTV Networks Inc. http://www.webtv.com (Sited Jan. 2004)

[33] Wireless week magazine, Wireless Industry Terms, http://www.wirelessweek.com/industry/terms.htm (Sited Jan. 2004)

[34] J.B. Postel, “Transmission Control Protocol”, RFC 793, Sept 1981.

[35] M. Allman, V.Paxson, and W. Richard Stevens, “TCP Congestion Control”, RFC 2581, Apr 1999.

[36] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, “A Comparison of Mechanisms for Improving TCP Performance over Wireless Links”, IEEE/ACM Transactions on Networking, vol.5, pp. 756-769, 1997.

[37] R. Cáceres, L. Iftode, “Improving the performance of Reliable Transport Protocols in Mobile Computing Environments”, IEEE Journal on Selected Areas in Communications, Vol. 13, No. 5, June 1995.

[38] A. DeSimone, M. C. Chuah, O. C. Yue, “Throughput Performance of Transport Layer Protocols over Wireless LANs”, Proceedings of Globecom 93, Dec. 1993.

[39] V. Tsaoussidis, H. Badr, K. Pentikousis, X. Ge, “Energy/Throughput Tradeoffs of TCP Error Control Strategies”, Proceedings of IEEE Symposium on Computer and Communications, France, July 2000.

[40] S. Tabbane, “Handbook of Mobile Radio Networks”, Artech House Mobile Communications Library, Norwood, MA, 2000.

[41] European Telecommunications Standard Institute, Broadband Radio Access Networks, http://www.etsi.org/bran.

[42] B. H. Walke, “Mobile Radio Networks – Networking, Protocols and Traffic Performance”, John Wiley & Sons, 2nd Edition, 2002.

[43] IEEE P802.11, The Working Group for Wireless Local Area Networks, http://grouper.ieee.org/groups/802/11

[44] CDMA Development Group, http://www.cdg.org

[45] R. Bekkers, J. Smits, “Mobile Telecommunications: Standards, Regulation and Applications”, Artech House Publishers, 1999.

[46] J. Tisal, “GSM Cellular Radio Telephony”, John Wiley & Sons, West Sussex, England, 1998.

[47] J. Agosta, T. Russel, “CDPD: Cellular Digital Packet Data Standards and Technology”, McGraw-Hill, New York, 1996.

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

[49] G. R. Wright, W. R. Stevens, “TCP/IP Illustrated, Volume 2: The Implementation”, Addison- Wesley, 1995.

[50] W. C. Y. Lee, “Mobile Communications Design Fundamentals”, 2nd edition, John Wiley & Sons, 1993.

[51] S. Mann, “Programming Applications with the Wireless Application Protocol: The Complete Developer’s Guide”, John Wiley & Sons, 1999.

[52] P. Sinha, N. Venkitaraman, R. Sivakumar, V. Bharghavan, “WTCP: a reliable transport protocol for wireless wide-area networks”, Proceedings of the 5th annual ACM/IEEE International Conference on Mobile Computing and Networking, Aug. 1999.

[53] T. V. Lakshman, U. Madhow, “The performance of TCP/IP for networks with high delay×bandwidth products and random loss”, IEEE/ACM Transactions on Networking, vol. 5, no. 3, June 1997.

[54] C. Perkins, “Mobile IP Design Principles and Practices”, Addison-Wesley, 1998.

[55] S. Savage, N. Cardwell, T. Anderson, “The Case for Informed Transport Protocols”, Proceedings of the Seventh Workshop on Hot Topics in Operating Systems”, March 1999.

[56] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, “Hypertext Transfer Protocol – HTTP 1.1”, RFC 2616, June 1999.

[57] H. Balakrishnan, V. Padmanabhan, S. Seshan, M. Stemm, R. H. Katz, “A comparison of mechanisms for improving TCP performance over wireless links”, IEEE/ACM Transactions on Networking, Dec. 1997.

[58] V. Jacobson, “Congestion avoidance and control” in Proc. SIGCOMM 88, Aug. 1988.

[59] P. Karn, C. Partridge, “Improving Round-Trip Time Estimates in Reliable Transport Protocols”, ACM Transaction on Computer Systems, Vol. 9, No. 4, 1991.

[60] A. Bakre and B. R. Badrinath, “I-TCP: Indirect TCP for Mobile Hosts”, In Proc. 15th International Conf. on Distributed Computing Systems (ICDCS), May 1995.

[61] R. Yavatkar and N. Bhagawat, “Improving end-to-end performance of TCP over mobile internetworks”, Proc. of the workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, pg. 146-152, 1995.

[62] E. Ayanoglu, S. Paul, T. F. LaPorta, K. Sabnani and R. Gitlin, “AIRMAIL: A Link-layer protocol for wireless networks”, Wireless Networks, Vol. 1, pp. 47-60, 1995.

[63] S. Nanda, R. Ejzak and B. Doshi, “A retransmission scheme for circuit-mode data on wireless links”, IEEE Journal on Selected Areas in Communications, Vol. 12, pp. 1338-1352, 1994.

[64] H. Balakrishnan, S. Seshan and R. H. Katz, “Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks”. ACM Wireless Networks, Vol. 1, 1995.

[65] H. Balakrishnan, S. Seshan and R. H. Katz, “A Comparison of Mechanisms for Improving TCP Performance over Wireless Links”, ACM SIGCOMM’96, pp. 256-269, Aug. 1996.

[66] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, “TCP selective acknowledgement options” IETF, RFC 2018, Jan. 1996.

[67] S. Floyd, S. Mahdavi, M. Mathis and M. Podolsky, “An extension to the selective acknowledgement (SACK) option for TCP”, IETF, RFC 2883, 2000.

[68] B. Bakshi, P. Krishna, N. Vaidya and D. K. Pradhan, “Improving Performance of TCP over wireless networks”, 17th International Conference on Distributed Computing Systems, pp. 365-373, 1997.

[69] W. R. Stevens, “TCP Slow start, congestion avoidance, fast retransmission and fast recovery algorithms”, IETF RFC 2001, Jan. 1997.

[70] J. C. Hoe, “Improving the start-up behavior of a congestion control scheme for TCP”, ACM SIGCOMM, pp. 270-280, 1996.

[71] K. Brown and S. Singh, “M-TCP: TCP for mobile cellular networks”, Computer Communications Review, pp. 19-43, July 1997.

[72] T. Goff, J. Moronski and D. S. Phatak, “Freeze-TCP: A True End-to-End TCP Enhancement Mechanism for Mobile Environments”, In Proceedings of the IEEE INFOCOM, Vol. 3: 1537-1545, 2000.

# Appendix A: Modification of kernel source code to implement Freeze-TCP

The entire kernel source code of Linux Red Hat 7.3 is located at the directory "/usr/src/linux-2.4.18-3". To implement Freeze-TCP reported in this thesis, I have modified the kernel source codes, recompiled and made a new kernel for the Linux operating system. The following source codes that are modified are as follows:

1. /usr/src/linux-2.4.18-3/arch/i386/kernel/entry.S
   Line 648 #ifdef _FTCP_
   Line 649 .long SYMBOL_NAME(sys_ftcp)
   Line 650 #endif /* _FTCP_ */

2. /usr/src/linux-2.4.18-3/arch/i386/kernel/Makefile
   Line 10 ifdef CONFIG_FTCP
   Line 11 FTCP = -D_FTCP_
   Line 12 endif

   Line 15 $(CC) $(FTCP) $(AFLAGS) -traditional -c $< -o $*.o

3. /usr/src/linux-2.4.18-3/include/asm-i386/unistd.h
   Line 247 #ifdef _FTCP_
   Line 248 #define __NR_ftcp 239
   Line 249 #endif /* _FTCP_ */

4. /usr/src/linux-2.4.18-3/include/linux/netdevice.h
   Line 430 #ifdef _FTCP_
   Line 431 unsigned char ftcp;
   Line 432 #endif /* _FTCP_ */

5. /usr/src/linux-2.4.18-3/include/net/ftcp.h (new file)

```
#ifndef _FTCP_H
#define _FTCP_H

#define FTCP_FREEZE (0x01)
#define FTCP_IFOFF  (0x02)

typedef struct {
  char *ifname;
  int val;
} ftcp_t;

enum {
  FTCP_FREEZE_CHECK,
  FTCP_FREEZE_OFF,
  FTCP_FREEZE_ON,
  FTCP_IF_CHECK,
  FTCP_IF_DISABLE,
  FTCP_IF_ENABLE,
  FTCP_DUP_ACKS1,
  FTCP_DUP_ACKS3
};

/* function prototypes */
int sys_ftcp(int func, ftcp_t *fp);
void ftcp_freeze_check(ftcp_t *fp);
void ftcp_freeze_on(ftcp_t *fp);
void ftcp_freeze_off(ftcp_t *fp);
void ftcp_if_check(ftcp_t *fp);
void ftcp_if_disable(ftcp_t *fp);
void ftcp_if_enable(ftcp_t *fp);
void ftcp_send_dup_acks(ftcp_t *fp, int num);

#endif /* _FTCP_H */
```

6. /usr/src/linux-2.4.18-3/net/core/dev.c

```
Line 110   #ifdef _FTCP_
Line 111   #include <net/ftcp.h>
Line 112   #endif /* _FTCP_ */

Line 741   #ifdef _FTCP_
Line 742       /* Initialize freeze TCP stuff */
Line 743           dev->ftcp = 0;
Line 744   #endif /* _FTCP_ */

Line 1015  #ifdef _FTCP_
Line 1016  /* Discard sk_buff if interface is off */
Line 1017      if (dev->ftcp & FTCP_IFOFF)
Line 1018      {
Line 1019              kfree_skb(skb);
Line 1020              return -ENOMEM;
Line 1021      }
Line 1022
Line 1023  /* Set advertised window size to zero if freeze TCP is on */
Line 1024  if ((skb->sk != NULL) && (skb->sk->protocol == IPPROTO_TCP) &&
Line 1025              (dev->ftcp & FTCP_FREEZE))
Line 1026     skb->h.th->window = 0;
Line 1027  #endif /* _FTCP_ */

Line 1250  #ifdef _FTCP_
Line 1251  /* Discard sk_buff if interface is off */
Line 1252      if (skb->dev->ftcp & FTCP_IFOFF)
Line 1253      {
Line 1254              kfree_skb(skb);
Line 1255              return softnet_data[this_cpu].cng_level;
Line 1256      }
Line 1257  #endif /* _FTCP_ */
```

7. /usr/src/linux-2.4.18-3/net/ipv4/Config.in

   Line 4 bool ' TCP: Freeze TCP support' CONFIG_FTCP

8. /usr/src/linux-2.4.18-3/net/ipv4/ftcp.c (new file)

```
#include <net/ftcp.h>
#include <net/tcp.h>
#include <linux/sched.h>
#include <linux/errno.h>

/* ftcp system call */
asmlinkage int sys_ftcp(int func, ftcp_t *fp)
{
 if (!capable(CAP_NET_ADMIN))
  return -EACCES;

 switch (func) {
 case FTCP_FREEZE_CHECK:
  ftcp_freeze_check(fp);
  break;
 case FTCP_FREEZE_OFF:
  ftcp_freeze_off(fp);
  break;
 case FTCP_FREEZE_ON:
  ftcp_freeze_on(fp);
  break;
 case FTCP_IF_CHECK:
  ftcp_if_check(fp);
  break;
 case FTCP_IF_DISABLE:
  ftcp_if_disable(fp);
  break;
 case FTCP_IF_ENABLE:
  ftcp_if_enable(fp);
  break;
```

```
  case FTCP_DUP_ACKS1:
    ftcp_send_dup_acks(fp, 1);
    break;
  case FTCP_DUP_ACKS3:
    ftcp_send_dup_acks(fp, 3);
    break;
  default:
    return -EINVAL;
  }
  return 0;
}

/* Check if freeze TCP is on for an interface */
void ftcp_freeze_check(ftcp_t *fp)
{
  /* Structure name change from device to net_device */
  struct net_device *d;
  /*
     dev_get_by_name return a structure but
     dev_get change to return an integer
  */
  if ((d = dev_get_by_name(fp->ifname)) == NULL)
  {
    fp->val = -ENODEV;
    return;
  }

  fp->val = (d->ftcp & FTCP_FREEZE) ? 1 : 0;
  return;
}

/* Turn freeze TCP on for an interface */
void ftcp_freeze_on(ftcp_t *fp)
{
```

```
  struct net_device *d;

  if ((d = dev_get_by_name(fp->ifname)) == NULL)
  {
    fp->val = -ENODEV;
    return;
  }

  fp->val = (d->ftcp & FTCP_FREEZE) ? 1 : 0;
  d->ftcp |= FTCP_FREEZE;
  return;
}

/* Turn freeze TCP off for an interface */
void ftcp_freeze_off(ftcp_t *fp)
{
  struct net_device *d;

  if ((d = dev_get_by_name(fp->ifname)) == NULL)
  {
    fp->val = -ENODEV;
    return;
  }

  fp->val = (d->ftcp & FTCP_FREEZE) ? 1 : 0;
  d->ftcp &= ~FTCP_FREEZE;
  return;
}

/* Check if an interface is disabled */
void ftcp_if_check(ftcp_t *fp)
{
  struct net_device *d;
```

```
  if ((d = dev_get_by_name(fp->ifname)) == NULL)
  {
    fp->val = -ENODEV;
    return;
  }

  fp->val = (d->ftcp & FTCP_IFOFF) ? 1 : 0;
  return;
}

/* Disable an interface */
void ftcp_if_disable(ftcp_t *fp)
{
  struct net_device *d;

  if ((d = dev_get_by_name(fp->ifname)) == NULL)
  {
    fp->val = -ENODEV;
    return;
  }

  fp->val = (d->ftcp & FTCP_IFOFF) ? 1 : 0;
  d->ftcp |= FTCP_IFOFF;
  return;
}

/* Endable an interface */
void ftcp_if_enable(ftcp_t *fp)
{
  struct net_device *d;

  if ((d = dev_get_by_name(fp->ifname)) == NULL)
  {
    fp->val = -ENODEV;
```

```
    return;
  }

  fp->val = (d->ftcp & FTCP_IFOFF) ? 1 : 0;
  d->ftcp &= ~FTCP_IFOFF;
  return;
}

/* Send duplicate ACKs out on all established TCP connections */
void ftcp_send_dup_acks(ftcp_t *fp, int num)
{
  /*extern struct sock *tcp_established_hash[TCP_HTABLE_SIZE];*/
  int i;

  /*
   *  Find established tcp connections that are not in TIME_WAIT state
   *    (1st half of hash table)
   */
  /* for(i = 0; i < TCP_HTABLE_SIZE/2; i++) */
  for (i = 0; i < tcp_ehash_size; i++)
  {
    /*struct sock *sk = tcp_established_hash[i];
      for (sk = tcp_established_hash[i]; sk != NULL; sk = sk->next) */
    struct sock *sk;
    struct tcp_ehash_bucket *head;
    head = &tcp_ehash[i];
    for (sk = head->chain; sk != NULL; sk = sk->next)
    {
      if (!(sk->reuse))
      {
      int j;

      /* FIXME: Check if this connection uses the given device */
```

```
      /* Send num duplicate ACKs */
      for(j = 0; j < num; j++)
        tcp_send_ack(sk);
      fp->val++;
      }
    }
  }
  return;
}
```

9. /usr/src/linux-2.4.18-3/net/ipv4/Makefile

   Line 21 obj-$(CONFIG_FTCP) += ftcp.o

10. /usr/src/linux-2.4.18-3/net/ipv4/tcp_input.c

```
Line 71   #ifdef _FTCP_
Line 72   #include <net/ftcp.h>
Line 73   #endif /* _FTCP_ */

Line 1875 #ifdef _FTCP_
Line 1876   /* Sender side fix for implemenatation of Freeze TCP */
Line 1877   if( nwin == 0 )
Line 1878     tp->snd_wnd = 0;
Line 1879   /* End of sender side fix */
Line 1880 #endif /* _FTCP_ */

Line 2589 #ifdef _FTCP_
Line 2590       if(!(((tp->send_head != NULL) &&
Line 2591         (tp->send_head->dev->ftcp & FTCP_FREEZE)) &&
Line 2592       (((TCP_SKB_CB(skb)->end_seq)-(tp->rcv_nxt)) <= 1)))
Line 2593 #endif /* _FTCP_ */

Line 3032 #ifdef _FTCP_
Line 3033 /* Out receive window == 0 (Zero Window Probe) JM --fix for ZWP */
Line 3034 (tcp_receive_window(tp) == 0) ||
```

```
Line 3035  #endif /* _FTCP_ */

Line 3339  #ifdef _FTCP_
Line 3340      /* Fix to zero window probe */
Line 3341           if ((tcp_receive_window(tp) != 0) &&
Line 3342             (!(((tp->send_head != NULL) &&
Line 3343             (tp->send_head->dev->ftcp & FTCP_FREEZE))
Line 3344             && ((TCP_SKB_CB(skb)->end_seq-tp->rcv_nxt) <= 1))))
Line 3345  #endif /* _FTCP_ */
```

11. /usr/src/linux-2.4.18-3/Makefile

```
Line 4      EXTRAVERSION = -160703

Line 98     ifdef CONFIG_FTCP
Line 99     CFLAGS += -D_FTCP_
Line 100    endif
```

## Recompile and build a new kernel

After all of the source code is modified, the kernel of Linux is ready to compile to activate Freeze-TCP for the operating system. It is wise to build a new kernel that is separated from the old kernel in case the new one fails. Performed the following steps with “root” to build a new kernel.

1. make mrproper
2. make clean
3. make xconfig
   - Select Load Configuration from file (/usr/src/linux-2.4.18-3/configs/kernel-2.4.18-i386.config)
   - Select “yes” for TCP: Freeze-TCP support under Networking options
   - Save and Exit
4. make dep
5. nohup make bzImage &

6. tail –f nohup.out
7. make modules
8. make modules_install
9. cp arch/i386/boot/bzImage  /boot/bzImage-FTCP-16Jul2003
10. mkinitrd -v /boot/initrd-2.4.18-160703.img  2.4.18-160703
    - library modules is found at “/lib/modules” directory
11. gedit or vi /boot/grub/grub.conf
    - title Red Hat Linux (Freeze-TCP Implemented)

      root (hd0,0)

      kernel /bzImage-FTCP-16Jul2003 ro root=/dev/sda2

      initrd /initrd-2.4.18-160703.img
12. After the new kernel with Freeze-TCP is made, PCMCIA interface requires to be recompiled again for the new kernel to work with the wireless card.

# Appendix B: Modification of Cellular IP source code to work with Freeze-TCP

All of Cellular IP source code is under the directory “/cip-1.1” [24]. I have modified and recompile the following source codes to get Freeze-TCP interwork with Cellular IP protocol:

1. /cip-1.1/cipmobile.h

```
Line 31     #include "ftcp.h"

Line 34     #define IFNAME "eth0"          /* Wireless Network interface to use with
            Freeze TCP*/

Line 171    void error(char *);
Line 172    void check_FTCP();
Line 173    void enable_FTCP();
Line 174    void disable_FTCP();
Line 175    void send_3_dup_acks();
```

2. /cip-1.1/cipmobile.c

```
Line 421    /*-------------------------------------------------------------------*/
Line 422    /* currently signal measurement of aironet card is not working      */
Line 423    /* properly.                                                        */
Line 424    /* only explicit forced handoff is available in arionet card.       */
Line 425    /* SNR based Auto Handoff with Freeze TCP                           */
Line 426    /*-------------------------------------------------------------------*/
Line 427    void handle_beacon(char * beacon_pkt)
Line 428    {
Line 429      struct in_addr        ia;
Line 430      int                   targetBS_index;
Line 431      u_char                *targetBS_ip;
```

```
Line 432        struct ip               *ip_hdr;
Line 433
Line 434      #ifdef WICACHE
Line 435        //beacon signal packet has been received.
Line 436        //measure SNR per beacon and compute average
Line 437        wi_readcache(cfg.name, beacon_pkt);
Line 438
Line 439        //SNR based auto handoff with Freeze TCP
Line 440        if(cfg.handoff) {
Line 441            // check if handoff is needed
Line 442              targetBS_index = need_handoff(currentBS);
Line 443
Line 444            if((targetBS_index != -1) && (targetBS_index != currentBS)) {
Line 445              printf("###### SNR based HANDOFF to BS(%d) ######\n",
                      targetBS_index);
Line 446
Line 447                {  /* handoff process */
Line 448              // Check Freeze TCP status
Line 449              check_FTCP();
Line 450              // Enabling Freeze TCP on Wireless Interface
Line 451              enable_FTCP();
Line 452              //update gateway ip address of control packet
Line 453              if(baseInfo[currentBS].gwId != baseInfo[targetBS_index].gwId) {
Line 454                modify_gw_ip(full_pkt, baseInfo[targetBS_index].gwId);
Line 455              }
Line 456
Line 457              // change currently active BS
Line 458              currentBS = targetBS_index;
Line 459
Line 460              ip_hdr = (struct ip*)(full_pkt + sizeof(struct ether_header));
Line 461
Line 462              // change the packet type
Line 463              if(STATE == 1)
Line 464                ip_hdr->ip_p = IPPROTO_CIPRU;          //route-update
```

```
Line 465                else
Line 466                  ip_hdr->ip_p = IPPROTO_CIPPU;          //page-update
Line 467
Line 468                // change the destination ethernet address of outgoing packet
Line 469                modify_dest_mac(full_pkt, baseInfo[currentBS].bs_ether);
Line 470                // Disabling Freeze TCP on Wireless Interface
Line 471                disable_FTCP();
Line 472                // Send 3 duplicate acknowledgements
Line 473                send_3_dup_acks();
Line 474                // show ip address of active base station in tcl/tk
Line 475                ia.s_addr = baseInfo[targetBS_index].bsId;
Line 476                targetBS_ip = inet_ntoa(ia);
Line 477                write(sd_bs, targetBS_ip, strlen(targetBS_ip));
Line 478                   } /* handoff process */
Line 479             }
Line 480          }
Line 481       #else
Line 482
Line 483       # ifdef ANCACHE
Line 484          an_readcache(cfg.name, beacon_pkt);
Line 485       # endif
Line 486
Line 487       #endif
Line 488
Line 489       }


Line 1643      /*-----------------------------------------------------------------*/
Line 1644      /* Error messages for FREEZE-TCP during SNR BASED AUTO
               HANDOFF      */
Line 1645      /*-----------------------------------------------------------------*/
Line 1646      void error(char *msg)
Line 1647      {
Line 1648        fprintf(stderr, "ftcp: ERROR");
```

```
Line 1649       if (msg != NULL)
Line 1650         fprintf(stderr, " %s", msg);
Line 1651       if (errno != 0)
Line 1652         fprintf(stderr, " [%s]", strerror(errno));
Line 1653       fprintf(stderr, ".\n");
Line 1654
Line 1655       exit(1);
Line 1656     }


Line 1658     /*-----------------------------------------------------------------*/
Line 1659     /* Check Freeze TCP Status for Wireless Interface IFNAME           */
Line 1660     /*-----------------------------------------------------------------*/
Line 1661     void check_FTCP()
Line 1662     {
Line 1663       ftcp_t fp;
Line 1664
Line 1665       fp.ifname = IFNAME;
Line 1666       if (ftcp(FTCP_FREEZE_CHECK, &fp) == -1 || fp.val == -ENODEV)
Line 1667         error("FTCP_FREEZE_CHECK failed");
Line 1668
Line 1669       printf("Freeze TCP is %s for interface " IFNAME ".\n",
Line 1670             (fp.val == 1) ? "enabled" : "disabled");
Line 1671     }


Line 1673     /*-----------------------------------------------------------------*/
Line 1674     /* Enabling Freeze TCP for Wireless interface IFNAME               */
Line 1675     /*-----------------------------------------------------------------*/
Line 1676     void enable_FTCP()
Line 1677     {
Line 1678       ftcp_t fp;
Line 1679
Line 1680       fp.ifname = IFNAME;
```

```
Line 1681        /* Enable Freeze TCP for Wireless interface IFNAME */
Line 1682        printf("Enabling Freeze TCP for interface " IFNAME ".\n");
Line 1683        if (ftcp(FTCP_FREEZE_ON, &fp) == -1 || fp.val == -ENODEV)
Line 1684          error("FTCP_FREEZE_ON failed");
Line 1685
Line 1686        /* Wait a short while so packet(s) with zero window advertisement
Line 1687         get sent. */
Line 1688         usleep(100000);
Line 1689
Line 1690        /* Drop all packets going through IFNAME */
Line 1691        printf("Dropping all packets on interface " IFNAME ".\n");
Line 1692        if (ftcp(FTCP_IF_DISABLE, &fp) == -1 || fp.val == -ENODEV)
Line 1693          error("FTCP_IF_DISABLE failed");
Line 1694
Line 1695      }



Line 1697      *-------------------------------------------------------------------*/
Line 1698      /* Disabling Freeze TCP for Wireless Interface IFNAME              */
Line 1699      /*-------------------------------------------------------------------*/
Line 1700      void disable_FTCP()
Line 1701      {
Line 1702        ftcp_t fp;
Line 1703
Line 1704        fp.ifname = IFNAME;
Line 1705        /* Disable freeze TCP for network interface IFNAME */
Line 1706        printf("Disabling Freeze TCP for interface " IFNAME ".\n");
Line 1707        if (ftcp(FTCP_FREEZE_OFF, &fp) == -1 || fp.val == -ENODEV)
Line 1708          error("FTCP_FREEZE_OFF failed");
Line 1709
Line 1710        /* Allow packets back through IFNAME */
Line 1711       printf("Enabling network interface " IFNAME ".\n");
Line 1712       if (ftcp(FTCP_IF_ENABLE, &fp) == -1 || fp.val == -ENODEV)
Line 1713         error("FTCP_IF_ENABLE failed");
```

```
Line 1714       }
Line 1716       /*---------------------------------------------------------------*/
Line 1717       /* Send 3 Duplicate Acknowledgements                             */
Line 1718       /*---------------------------------------------------------------*/
Line 1719       void send_3_dup_acks()
Line 1720       {
Line 1721         ftcp_t fp;
Line 1722         /* Send duplicate ACKs */
Line 1723         printf("Sending 3 duplicate ACKs on all established TCP connections.\n");
Line 1724         if (ftcp(FTCP_DUP_ACKS3, &fp) == -1 || fp.val == -ENODEV)
Line 1725           error("FTCP_DUP_ACKS3 failed");
Line 1726       }
```

# Appendix C: Handoff Schemes data results

The following tables and figures summarise all the results obtained from Ethereal while the mobile node performs either forced or SNR handoffs with either standard TCP or Freeze-TCP. These data results were obtained while mobile node firstly established a telnet connection to the web server and then ping-ed IP packets back to mobile node for 60 seconds. See Chapter 5 for discussion of results presented in this appendix.

| Forced-TCP | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Elapsed time (sec) | 60.689 | 60.179 | 60.179 | 60.689 | 60.179 |
| Packet count | 294 | 341 | 406 | 455 | 520 |
| Avg. packets/sec | 4.844 | 5.666 | 6.747 | 7.497 | 8.641 |
| Avg. packet size (bytes) | 107.663 | 109.572 | 109.621 | 110.097 | 110.435 |
| Bytes of traffic | 31653 | 37364 | 44506 | 50094 | 57426 |
| Avg. bytes/sec | 521.560 | 620.881 | 739.564 | 825.426 | 954.260 |
| Avg. Mbit/sec | 0.004 | 0.005 | 0.006 | 0.007 | 0.008 |
| Forced-TCP | 6 | 7 | 8 | 9 | 10 |
| Elapsed time (sec) | 60.688 | 60.179 | 61.198 | 60.689 | 60.179 |
| Packet count | 570 | 622 | 685 | 725 | 781 |
| Avg. packets/sec | 9.392 | 10.336 | 11.193 | 11.946 | 12.978 |
| Avg. packet size (bytes) | 110.667 | 110.910 | 111.165 | 111.302 | 111.918 |
| Bytes of traffic | 63080 | 68986 | 76148 | 80694 | 87408 |
| Avg. bytes/sec | 1039.406 | 1146.341 | 1244.284 | 1329.638 | 1452.468 |
| Avg. Mbit/sec | 0.008 | 0.009 | 0.010 | 0.011 | 0.012 |

*Table C-1. Forced Handoff with standard TCP results.*

| Forced-FTCP | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Elapsed time (sec) | 60.719 | 60.688 | 60.179 | 60.179 | 60.664 |
| Packet count | 284 | 345 | 396 | 453 | 505 |
| Avg. packets/sec | 4.677 | 5.685 | 6.580 | 7.404 | 8.325 |
| Avg. packet size (bytes) | 108.363 | 108.388 | 108.763 | 109.024 | 109.168 |
| Bytes of traffic | 30775 | 37394 | 43070 | 49388 | 55130 |
| Avg. bytes/sec | 506.844 | 616.168 | 715.695 | 807.269 | 908.777 |
| Avg. Mbit/sec | 0.004 | 0.005 | 0.006 | 0.006 | 0.007 |
| Forced-FTCP | 6 | 7 | 8 | 9 | 10 |
| Elapsed time (sec) | 60.633 | 60.649 | 60.608 | 60.649 | 60.589 |
| Packet count | 563 | 610 | 663 | 688 | 719 |
| Avg. packets/sec | 9.285 | 10.058 | 10.939 | 11.344 | 11.867 |
| Avg. packet size (bytes) | 109.474 | 109.580 | 109.704 | 109.924 | 110.156 |
| Bytes of traffic | 61634 | 66844 | 72734 | 75628 | 79202 |
| Avg. bytes/sec | 1016.505 | 1102.147 | 1200.070 | 1246.978 | 1307.208 |
| Avg. Mbit/sec | 0.008 | 0.009 | 0.010 | 0.010 | 0.010 |

*Table C-2. Forced Handoff with Freeze-TCP results.*

| SNR-TCP | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Elapsed time (sec) | 60.689 | 60.179 | 60.178 | 60.689 | 60.689 |
| Packet count | 265 | 301 | 316 | 343 | 369 |
| Avg. packets/sec | 4.367 | 5.002 | 5.251 | 5.652 | 6.080 |
| Avg. packet size (bytes) | 108.275 | 108.478 | 108.557 | 109.032 | 109.333 |
| Bytes of traffic | 28693 | 32652 | 34304 | 37398 | 40344 |
| Avg. bytes/sec | 472.790 | 542.583 | 570.039 | 616.223 | 664.768 |
| Avg. Mbit/sec | 0.004 | 0.004 | 0.005 | 0.005 | 0.005 |
| SNR-TCP | 6 | 7 | 8 | 9 | 10 |
| Elapsed time (sec) | 60.179 | 60.178 | 60.179 | 60.689 | 60.7 |
| Packet count | 393 | 419 | 446 | 474 | 499 |
| Avg. packets/sec | 6.531 | 6.963 | 7.411 | 7.794 | 8.221 |
| Avg. packet size (bytes) | 109.608 | 109.838 | 109.942 | 110.207 | 110.369 |
| Bytes of traffic | 43076 | 46022 | 49034 | 52128 | 55074 |
| Avg. bytes/sec | 715.801 | 764.764 | 814.802 | 858.940 | 907.319 |
| Avg. Mbit/sec | 0.006 | 0.006 | 0.007 | 0.007 | 0.007 |

*Table C-3. SNR Handoff with standard TCP results.*

| SNR-FTCP | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Elapsed time (sec) | 60.179 | 60.649 | 60.118 | 60.139 | 59.649 |
| Packet count | 262 | 289 | 309 | 338 | 360 |
| Avg. packets/sec | 4.354 | 4.765 | 5.140 | 5.620 | 6.035 |
| Avg. packet size (bytes) | 107.313 | 107.426 | 107.890 | 107.485 | 107.683 |
| Bytes of traffic | 28116 | 31046 | 33338 | 36330 | 38766 |
| Avg. bytes/sec | 467.210 | 511.898 | 554.538 | 604.098 | 649.903 |
| Avg. Mbit/sec | 0.004 | 0.004 | 0.004 | 0.005 | 0.005 |
| SNR-FTCP | 6 | 7 | 8 | 9 | 10 |
| Elapsed time (sec) | 60.648 | 60.118 | 60.639 | 60.143 | 60.649 |
| Packet count | 389 | 405 | 435 | 455 | 475 |
| Avg. packets/sec | 6.414 | 6.737 | 7.174 | 7.565 | 7.832 |
| Avg. packet size (bytes) | 107.738 | 108.899 | 108.386 | 108.958 | 108.987 |
| Bytes of traffic | 41910 | 44104 | 47148 | 49576 | 51769 |
| Avg. bytes/sec | 691.031 | 733.622 | 777.523 | 824.434 | 853.588 |
| Avg. Mbit/sec | 0.006 | 0.006 | 0.006 | 0.007 | 0.007 |

*Table C-4. SNR Handoff with Freeze-TCP results.*

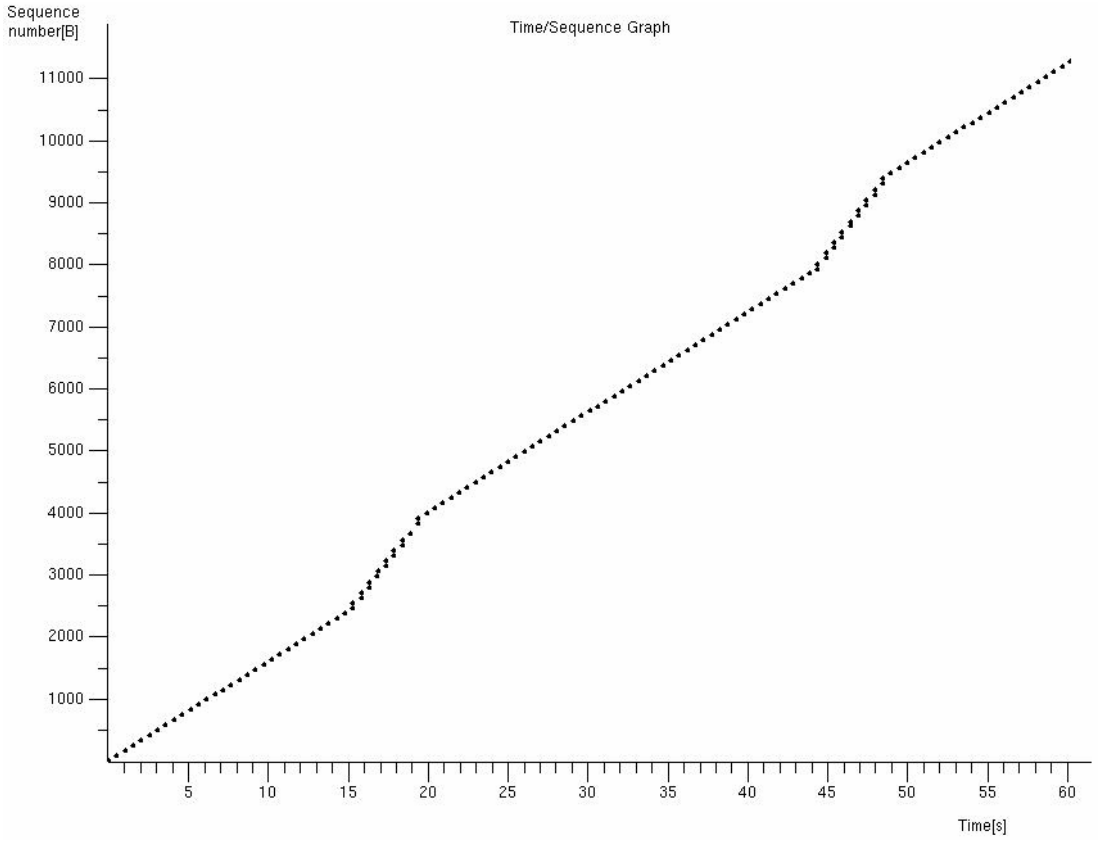


*Figure C-1. Sequence Graph of 2 forced handoffs with standard TCP in 60s.*

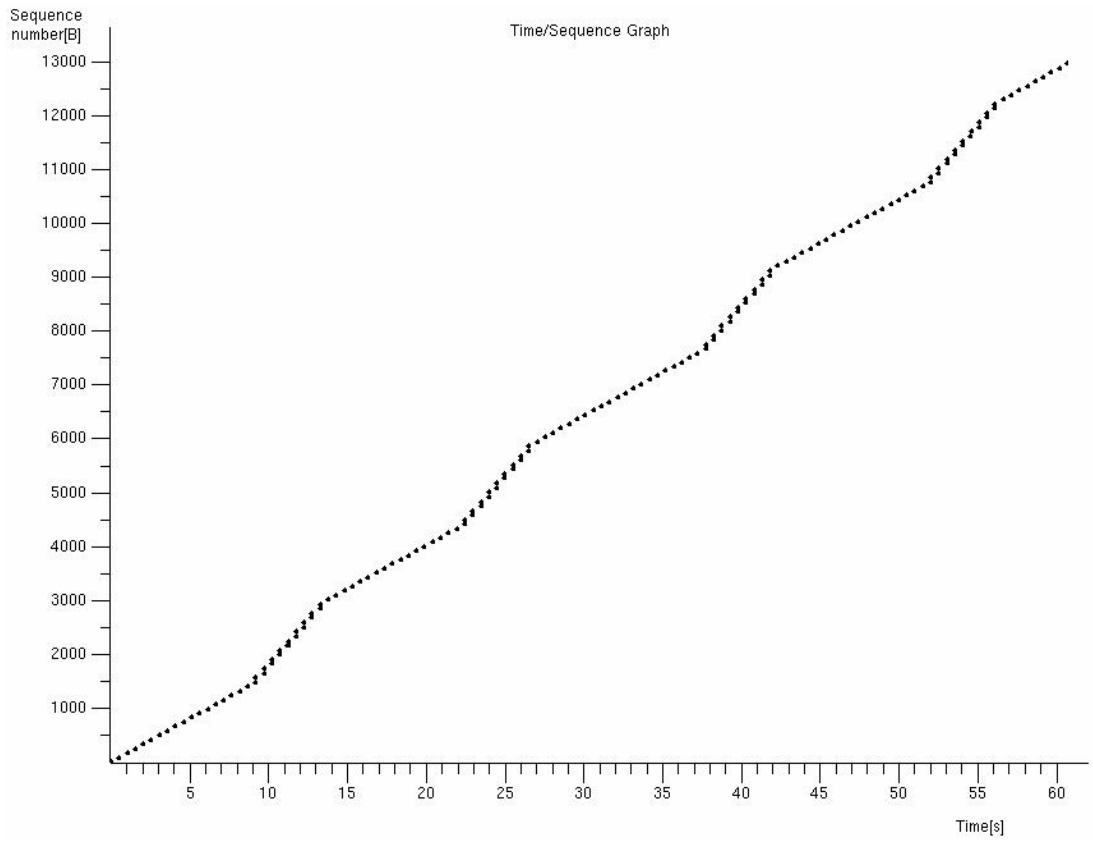


*Figure C-2. Sequence Graph of 4 forced handoffs with standard TCP in 60s.*

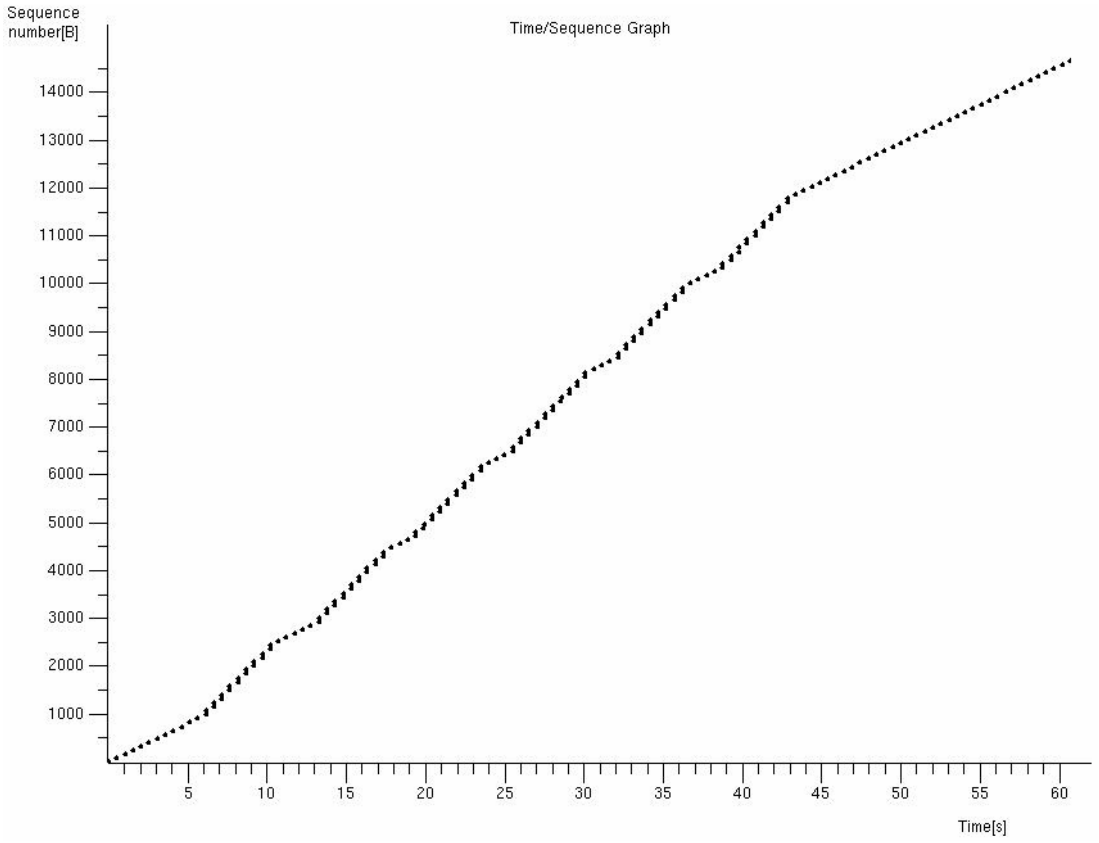


*Figure C-3. Sequence Graph of 6 forced handoffs with standard TCP in 60s.*

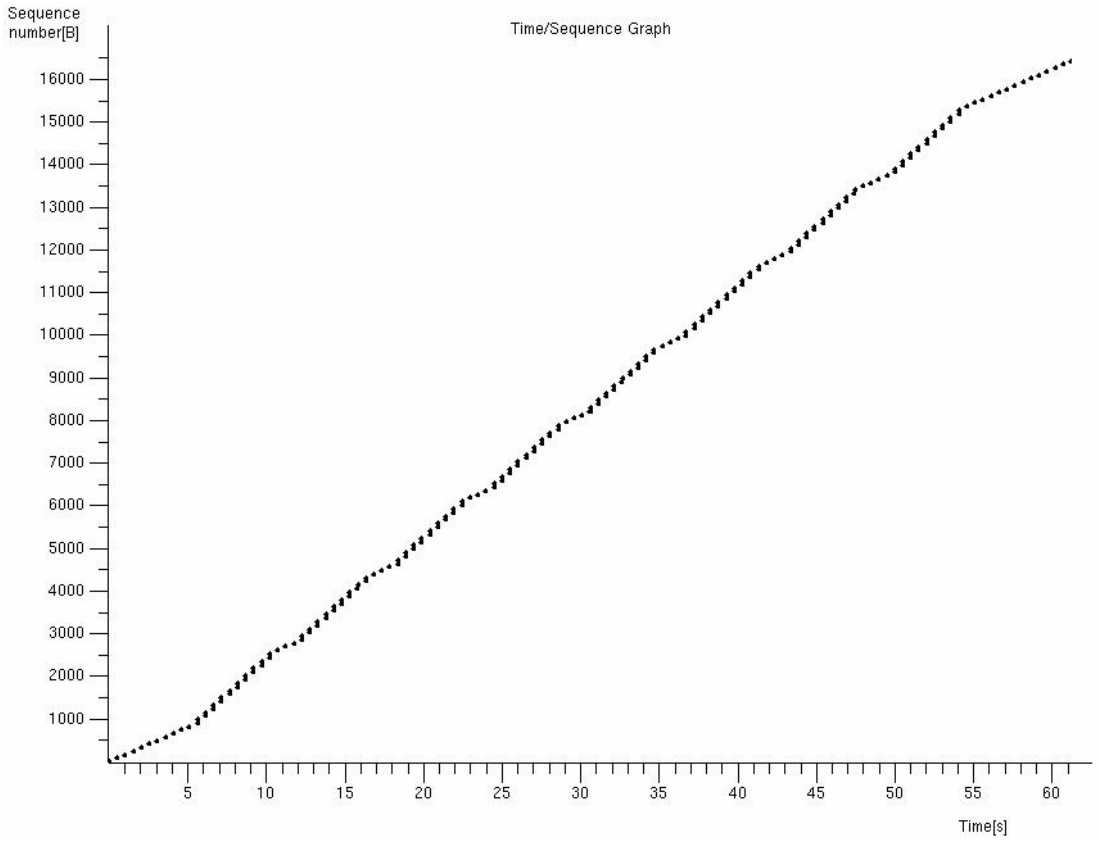


*Figure C-4. Sequence Graph of 8 forced handoffs with standard TCP in 60s.*

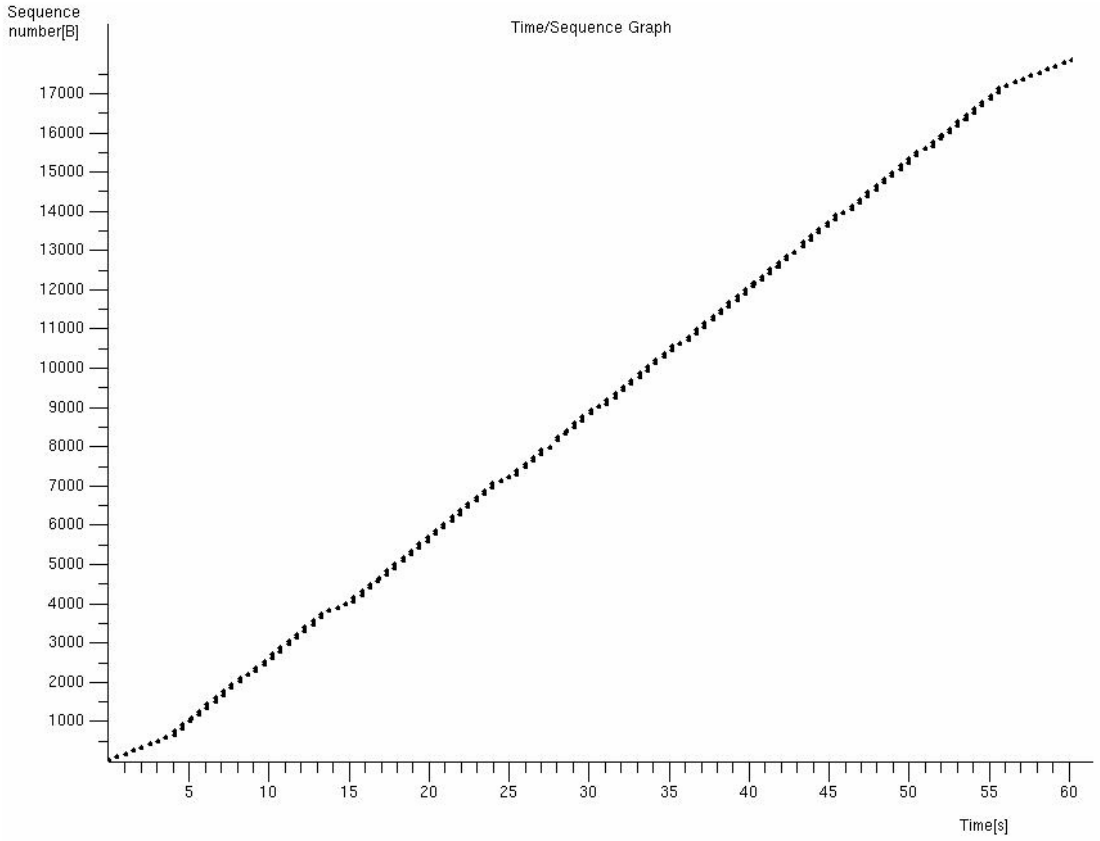


*Figure C-5. Sequence Graph of 10 forced handoffs with standard TCP in 60s.*

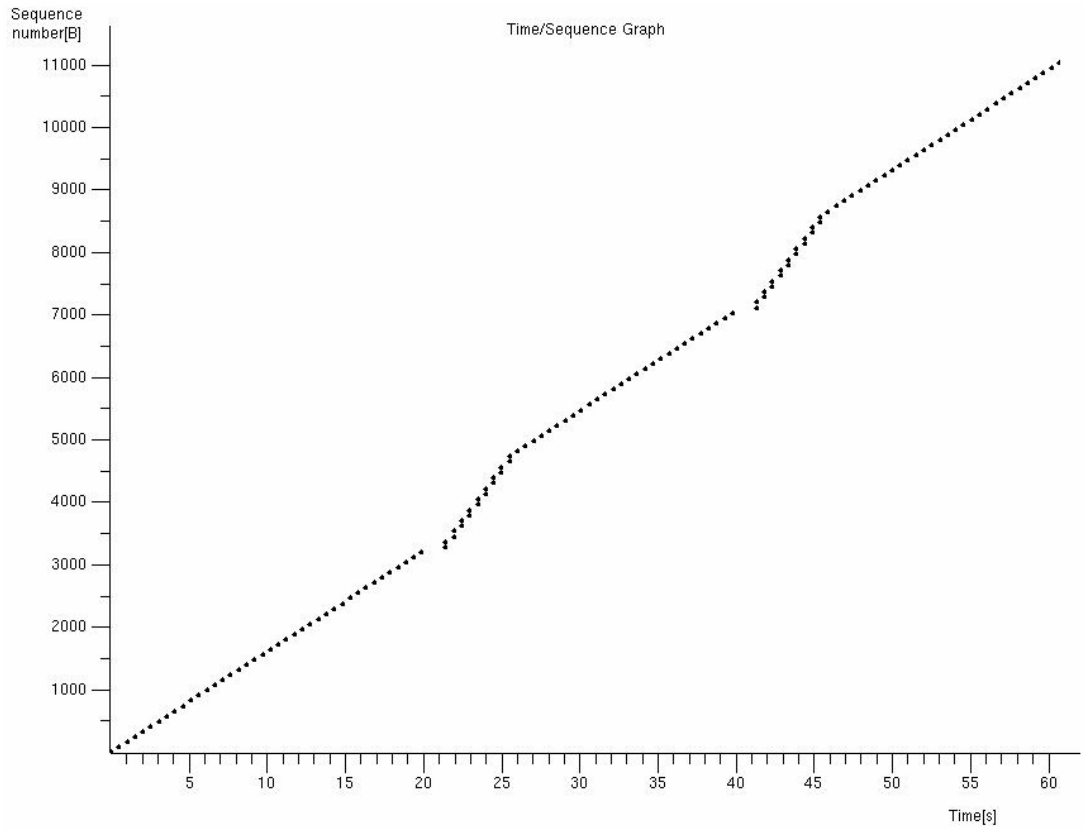


*Figure C-6. Sequence Graph of 2 forced handoffs with Freeze-TCP in 60s.*

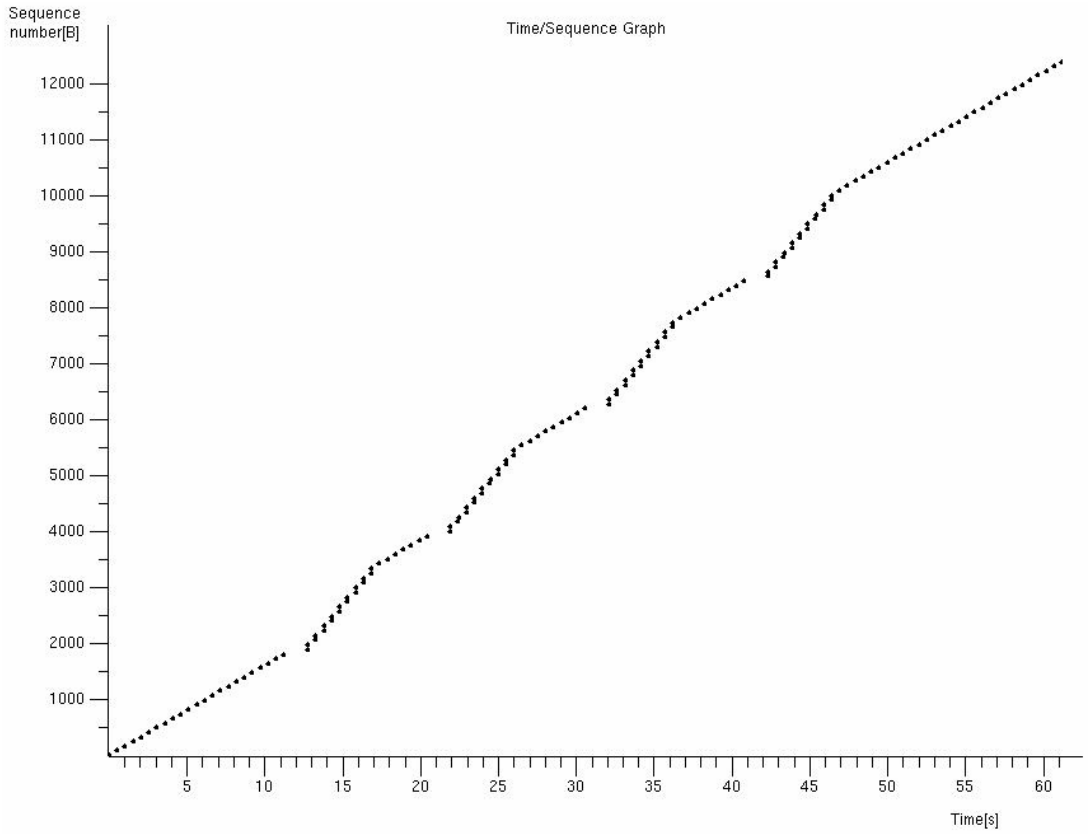


*Figure C-7. Sequence Graph of 4 forced handoffs with Freeze-TCP in 60s.*

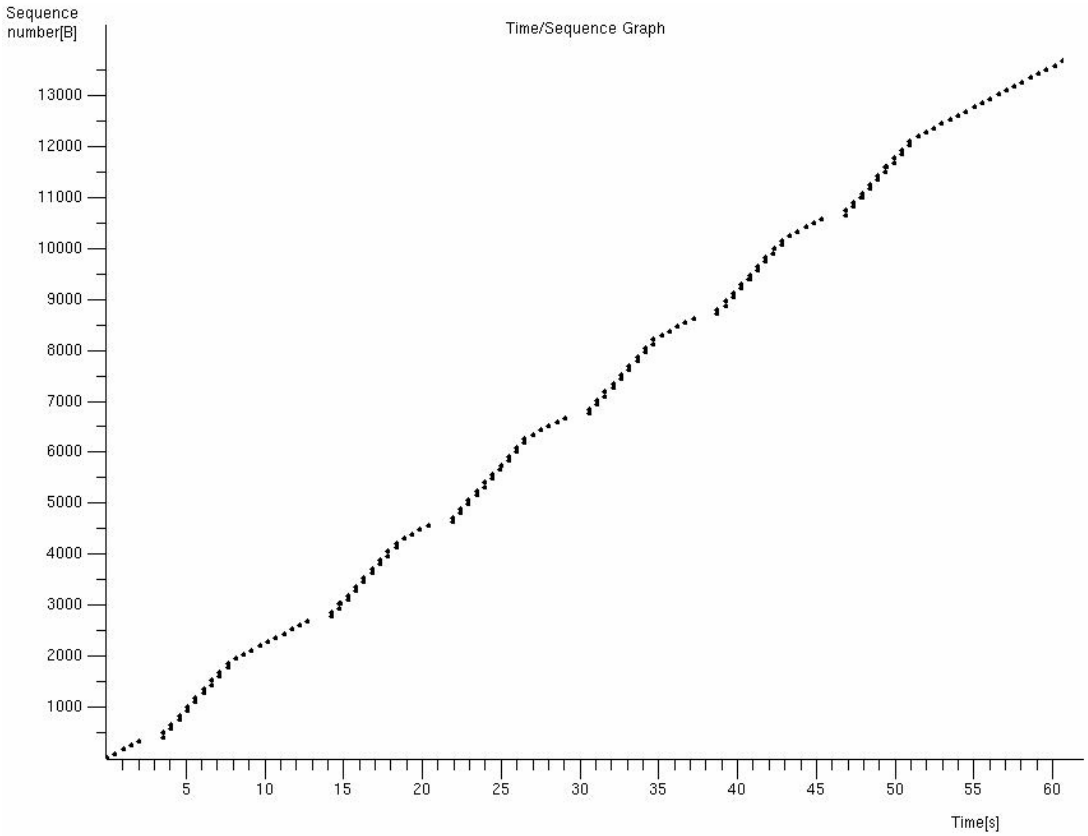


*Figure C-8. Sequence Graph of 6 forced handoffs with Freeze-TCP in 60s.*

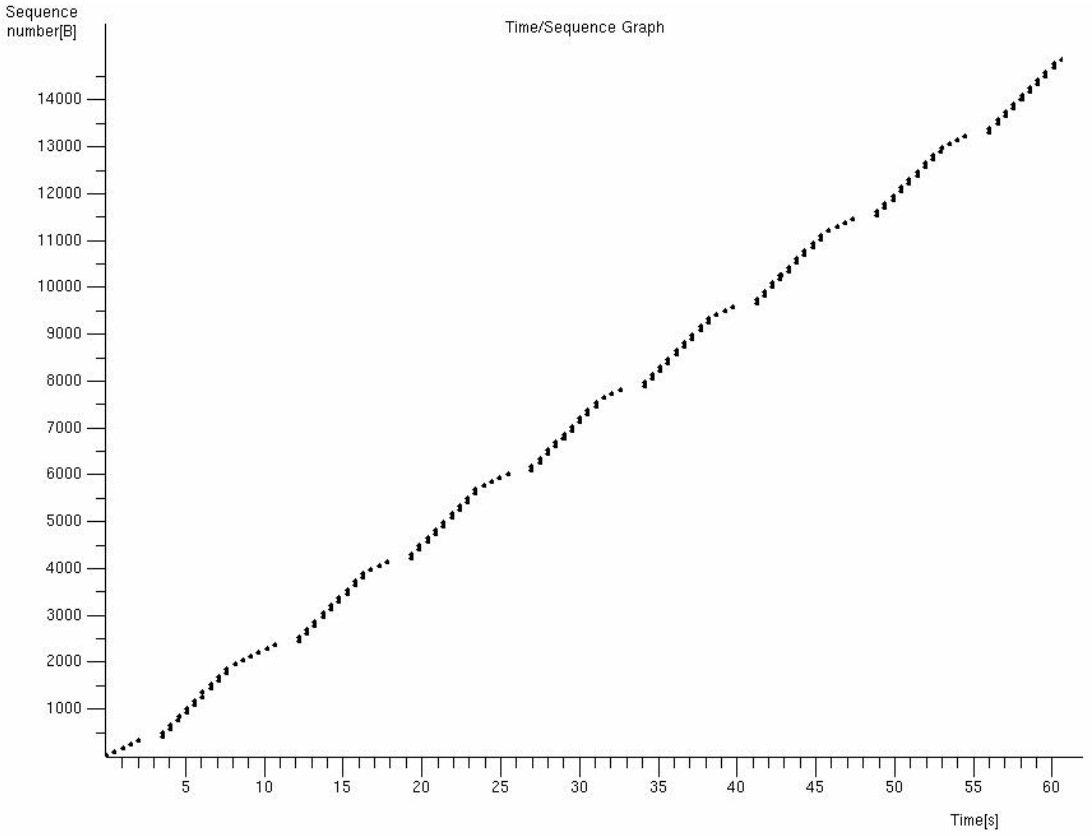


*Figure C-9. Sequence Graph of 8 forced handoffs with Freeze-TCP in 60s.*

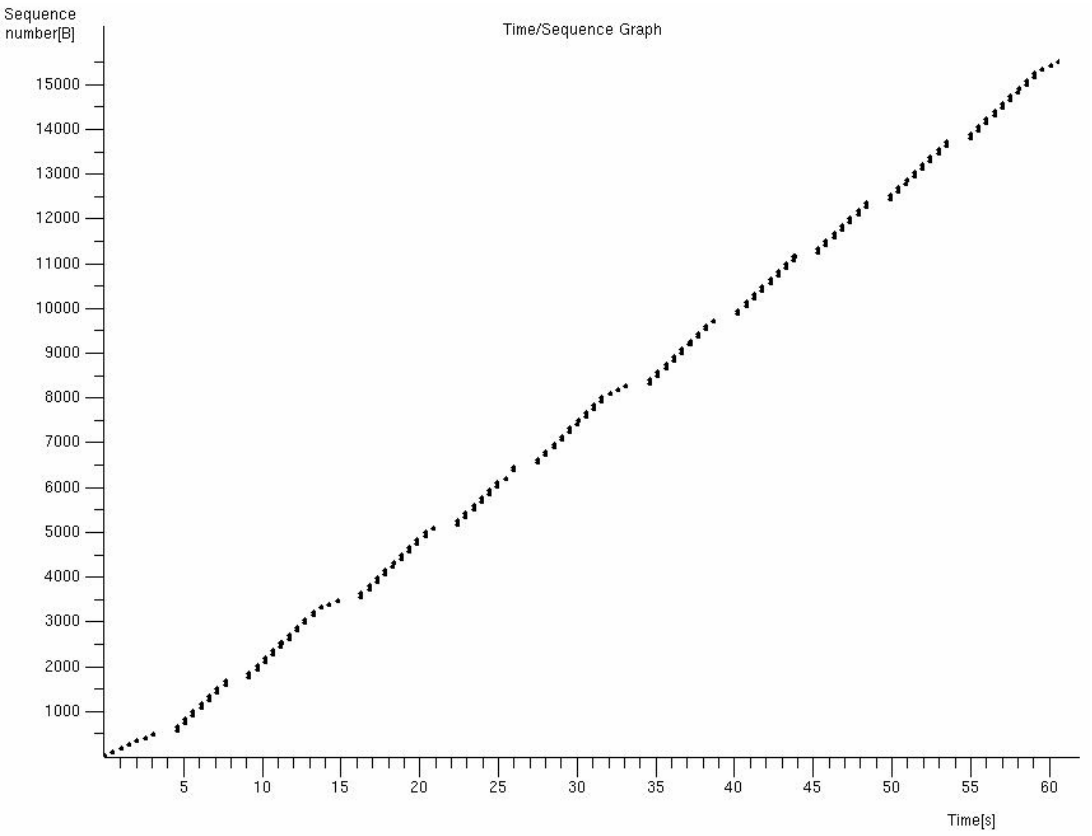


*Figure C-10. Sequence Graph of 10 forced handoffs with Freeze-TCP in 60s.*

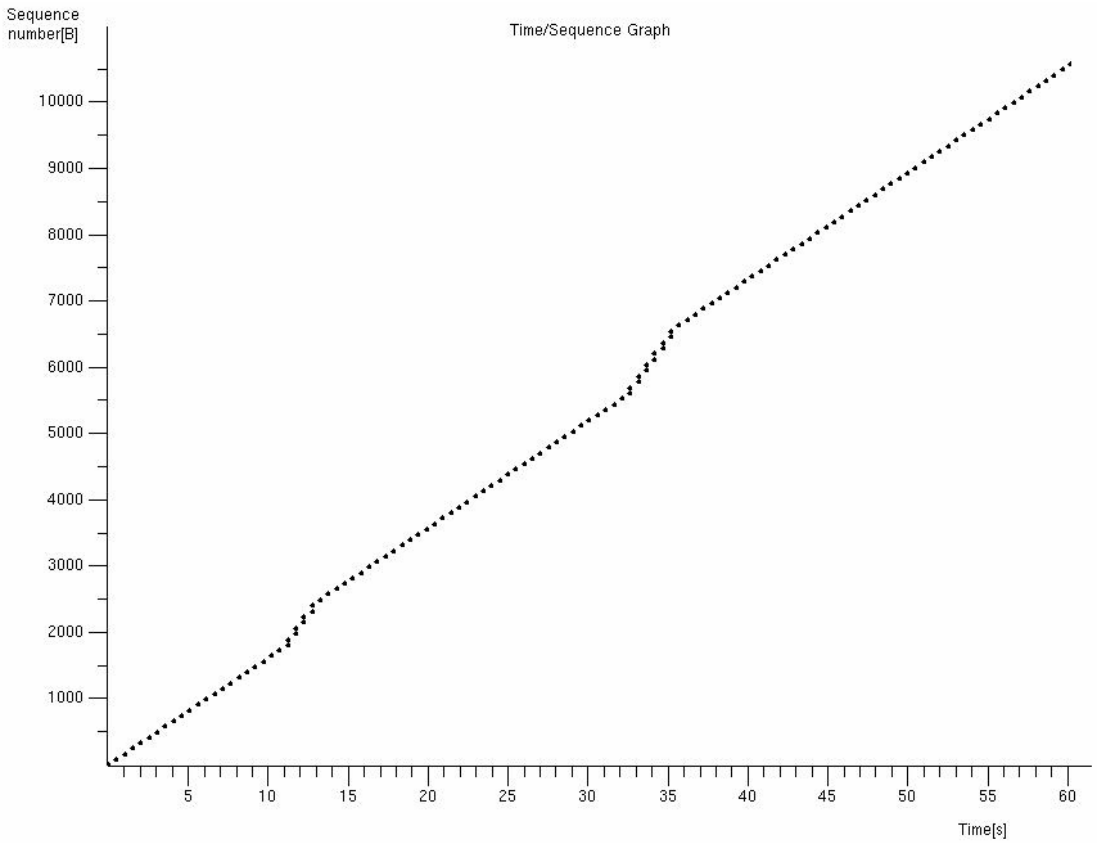


*Figure C-11. Sequence Graph of 2 SNR handoffs with standard TCP in 60s.*

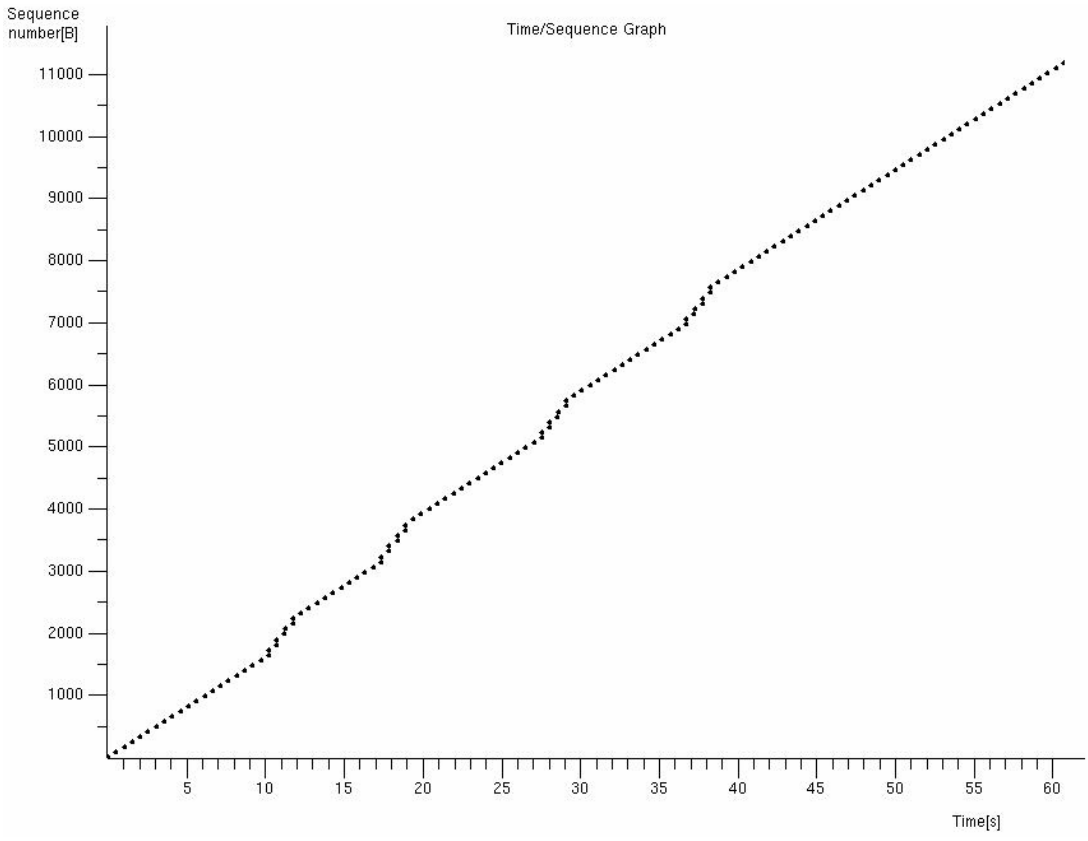


*Figure C-12. Sequence Graph of 4 SNR handoffs with standard TCP in 60s.*

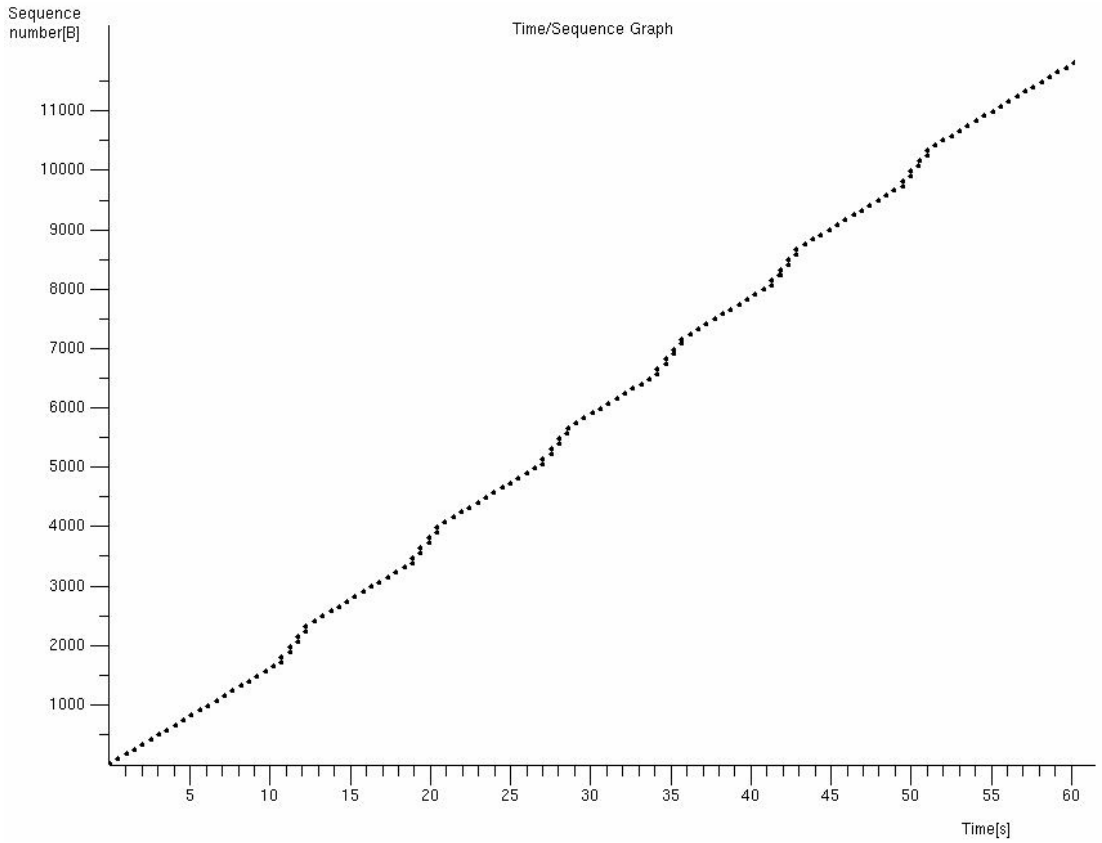


*Figure C-13. Sequence Graph of 6 SNR handoffs with standard TCP in 60s.*

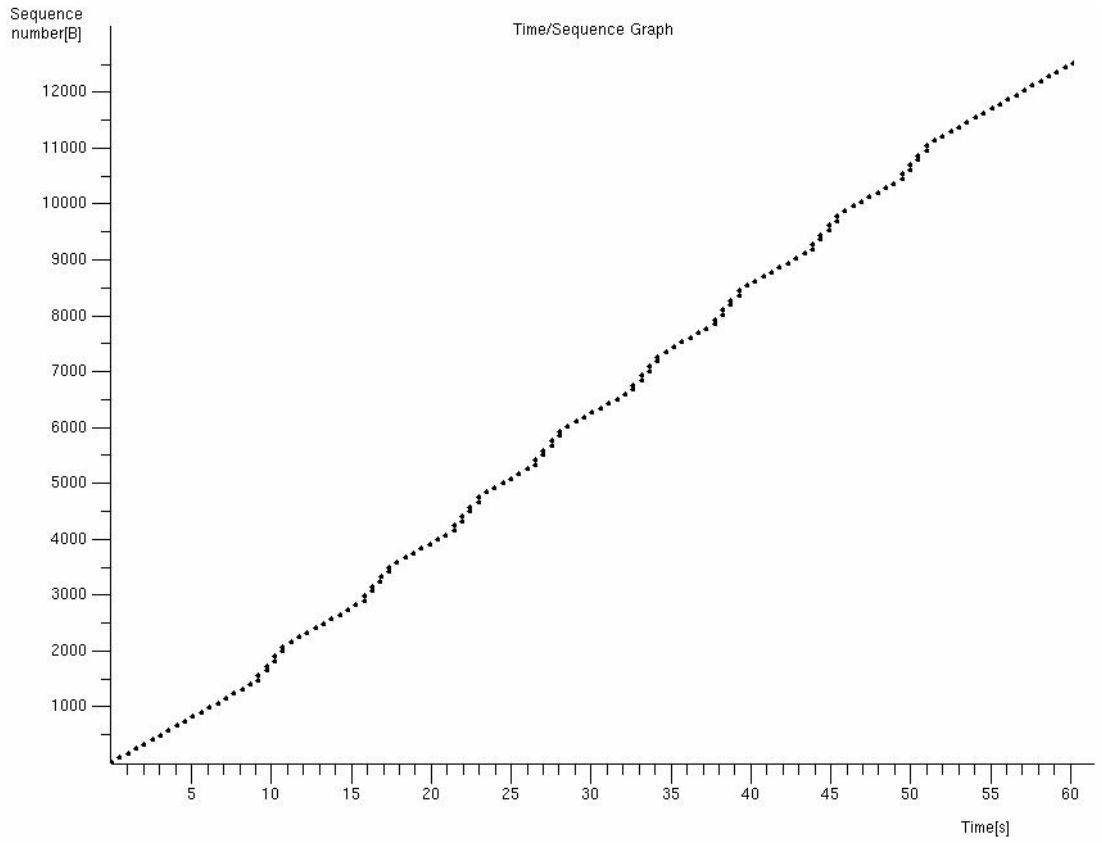


*Figure C-14. Sequence Graph of 8 SNR handoffs with standard TCP in 60s.*

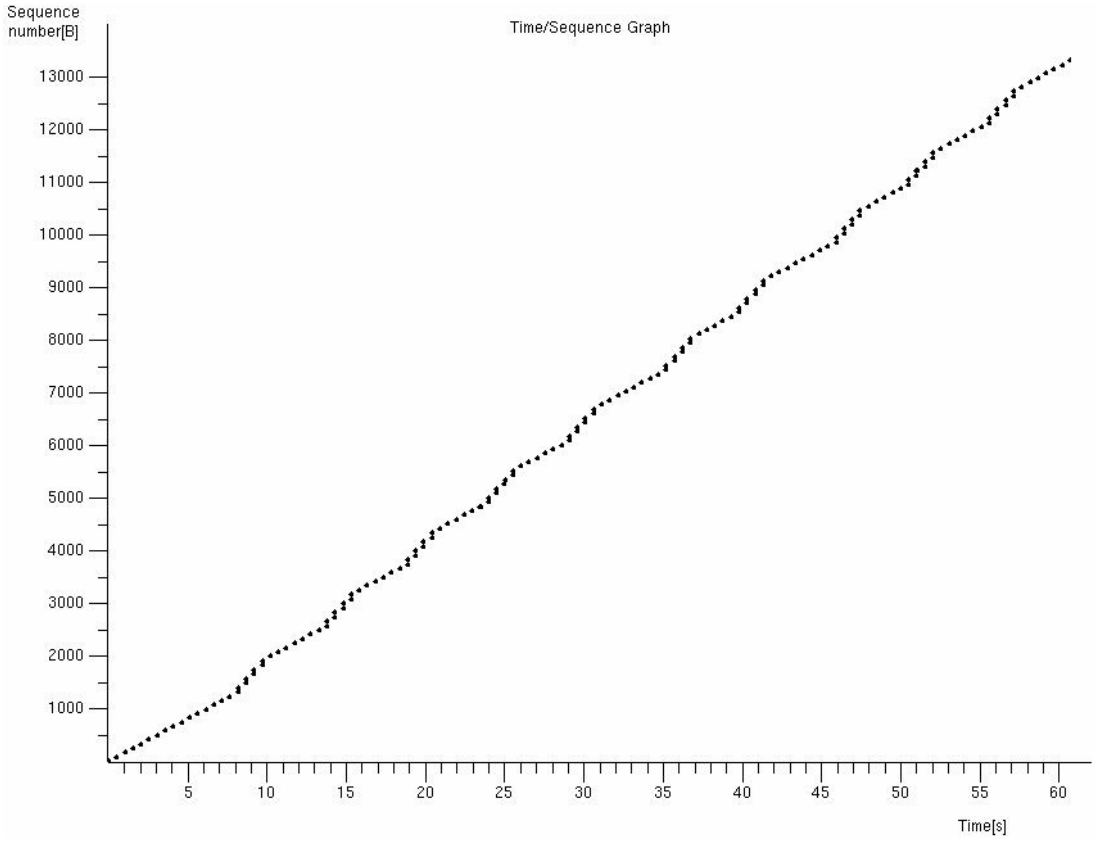


*Figure C-15. Sequence Graph of 10 SNR handoffs with standard TCP in 60s.*

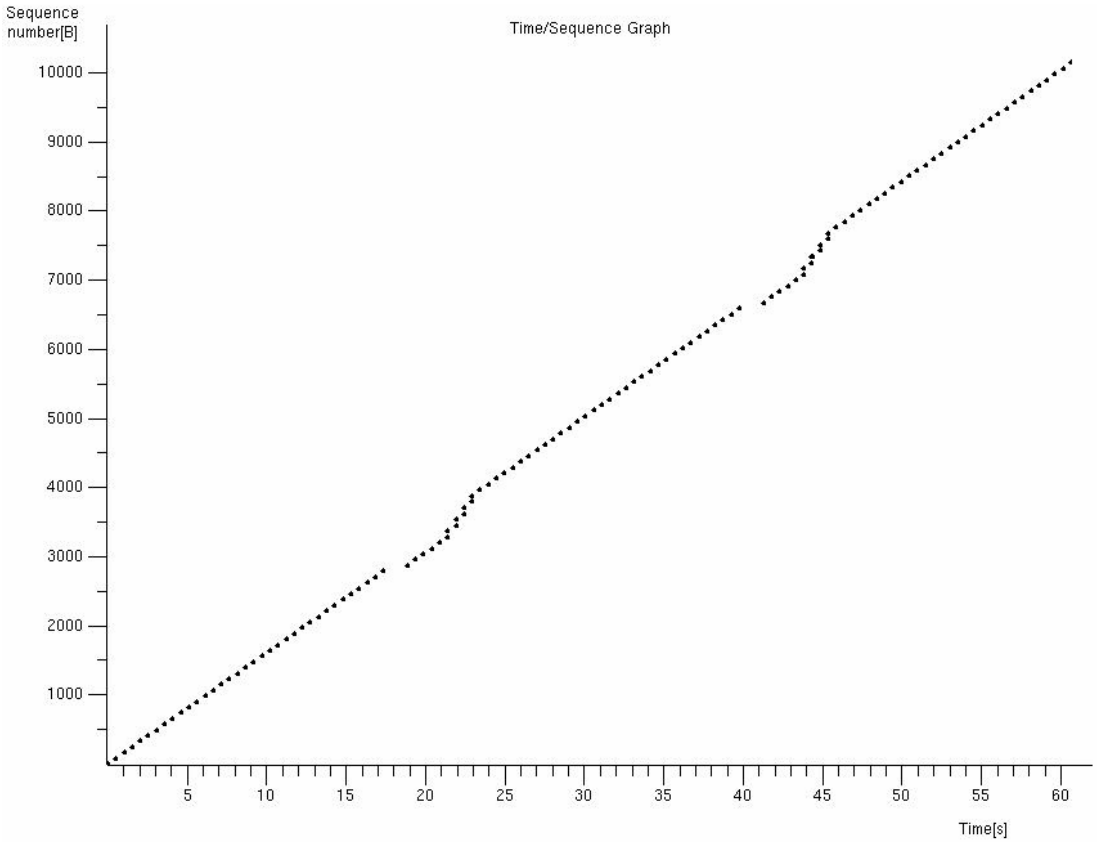


*Figure C-16. Sequence Graph of 2 SNR handoffs with Freeze-TCP in 60s.*

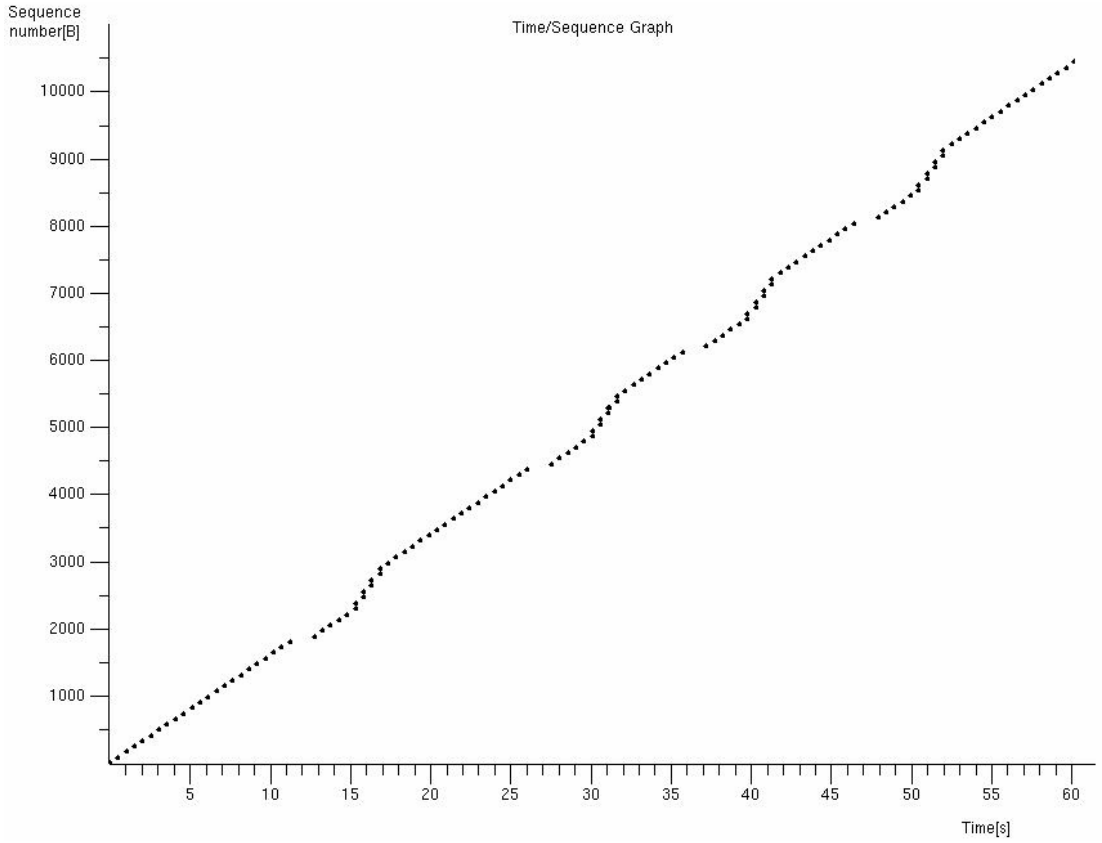


*Figure C-17. Sequence Graph of 4 SNR handoffs with Freeze-TCP in 60s.*

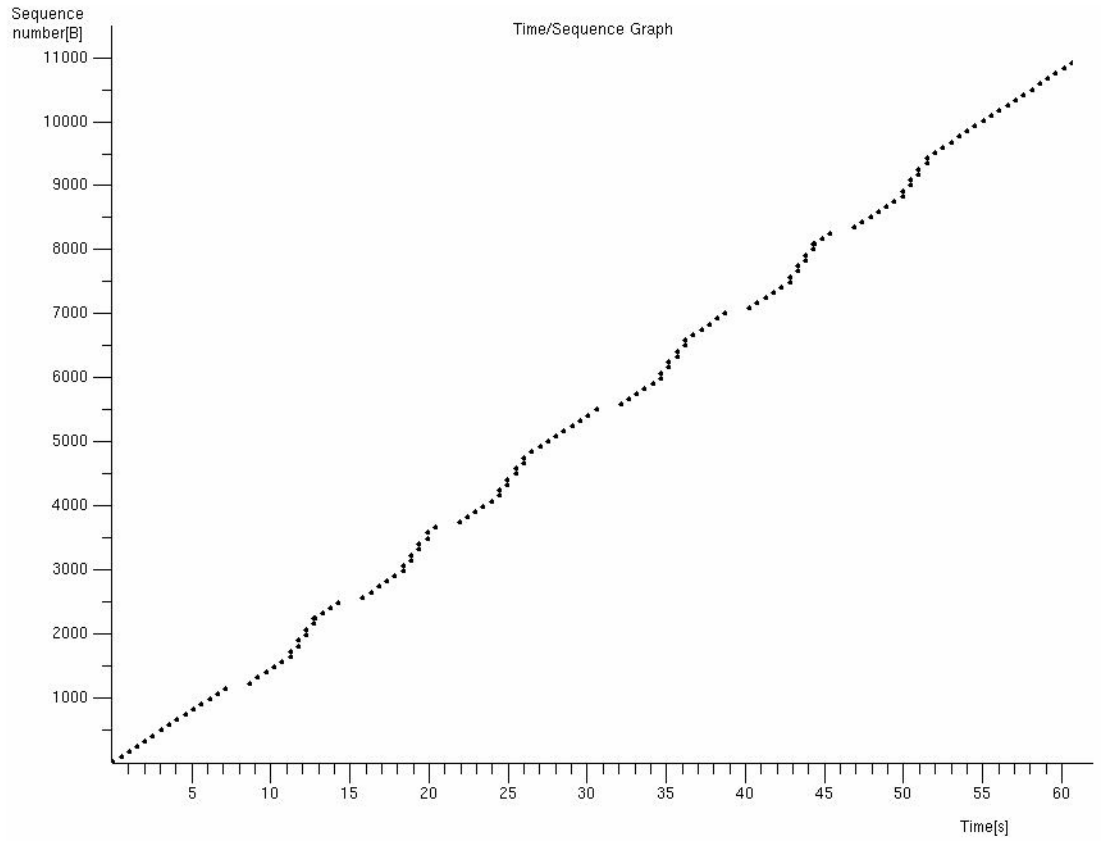


*Figure C-18. Sequence Graph of 6 SNR handoffs with Freeze-TCP in 60s.*

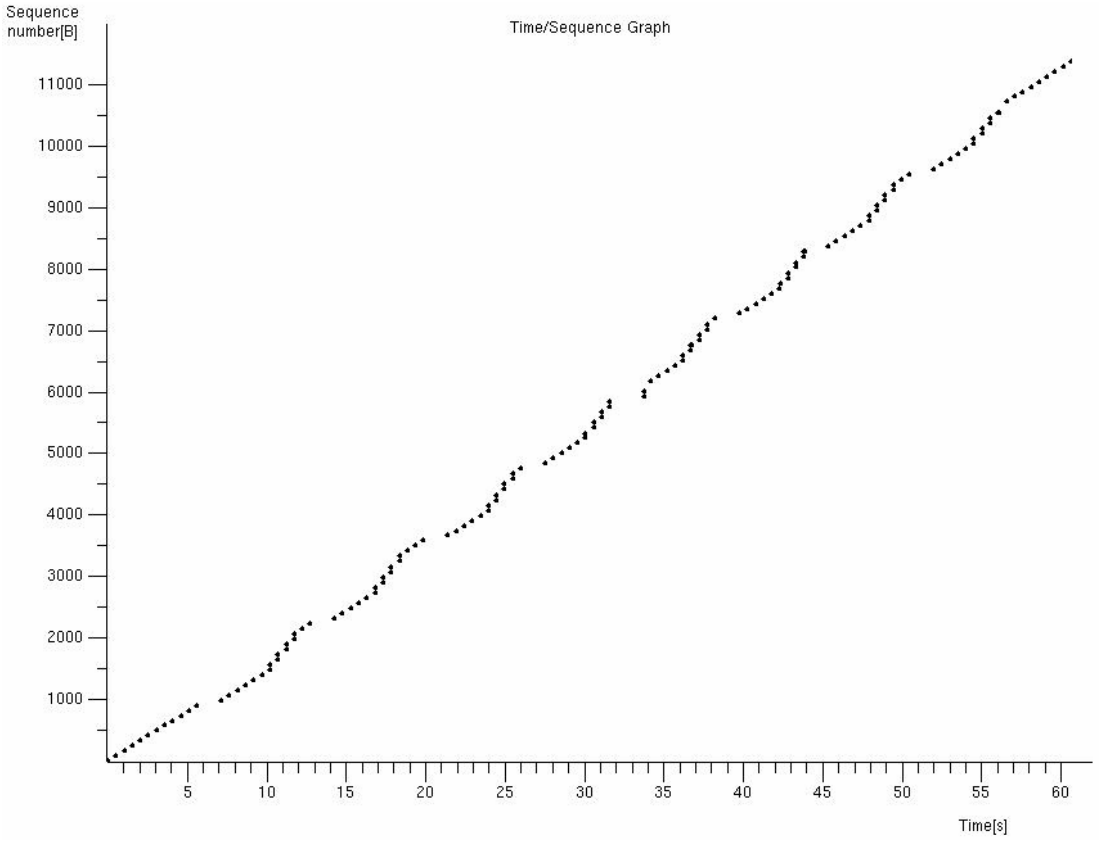


*Figure C-19. Sequence Graph of 8 SNR handoffs with Freeze-TCP in 60s.*

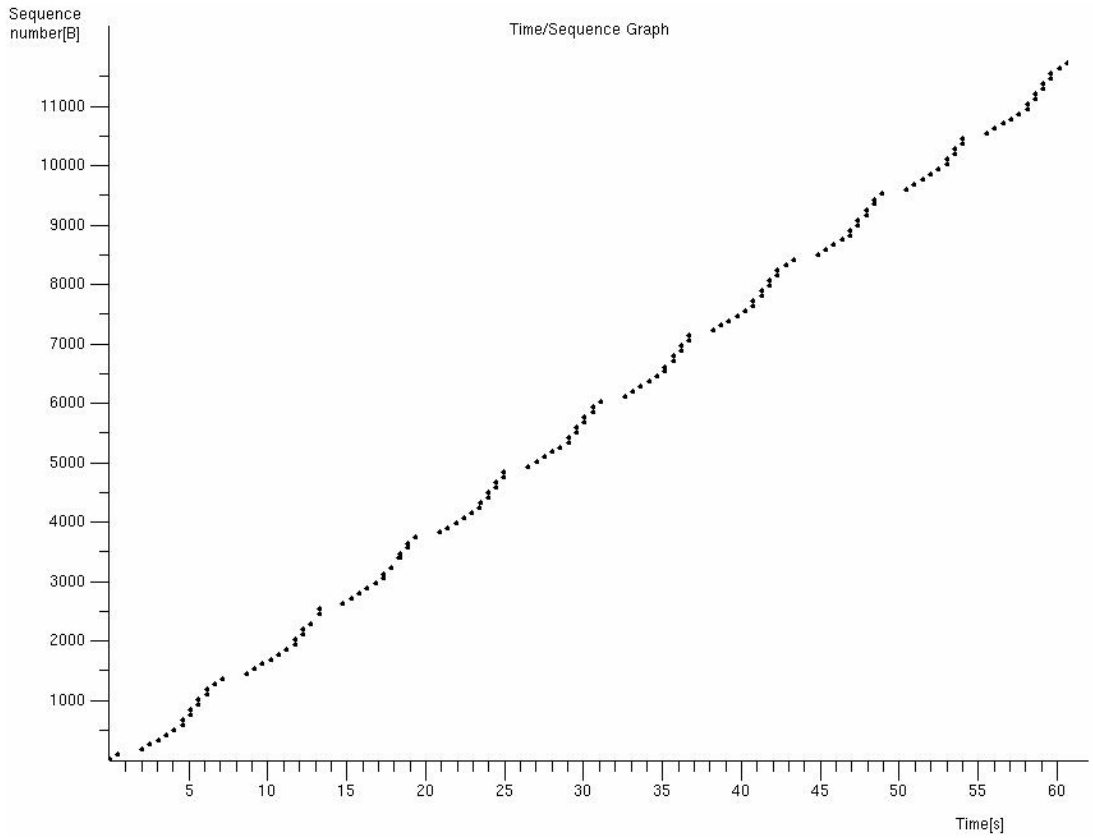


*Figure C-20. Sequence Graph of 10 SNR handoffs with Freeze-TCP in 60s.*

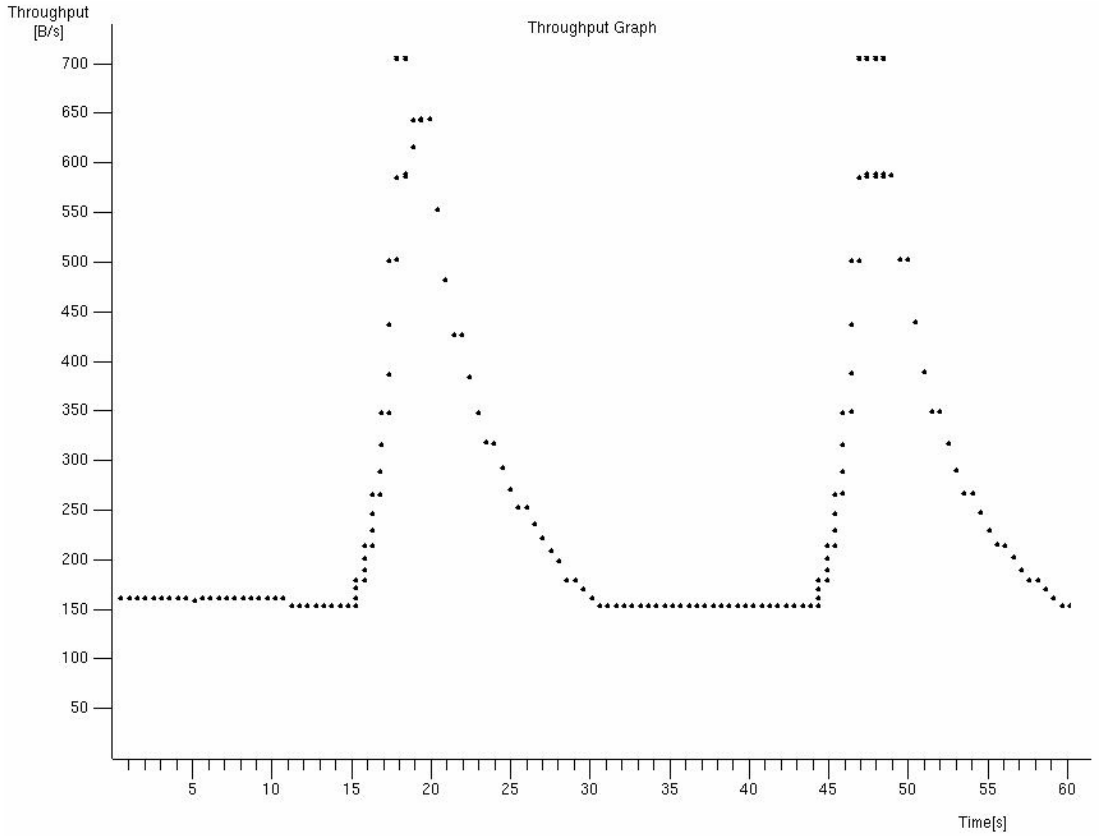


*Figure C-21. TCP throughput graph of 2 forced handoffs with standard TCP in 60s.*

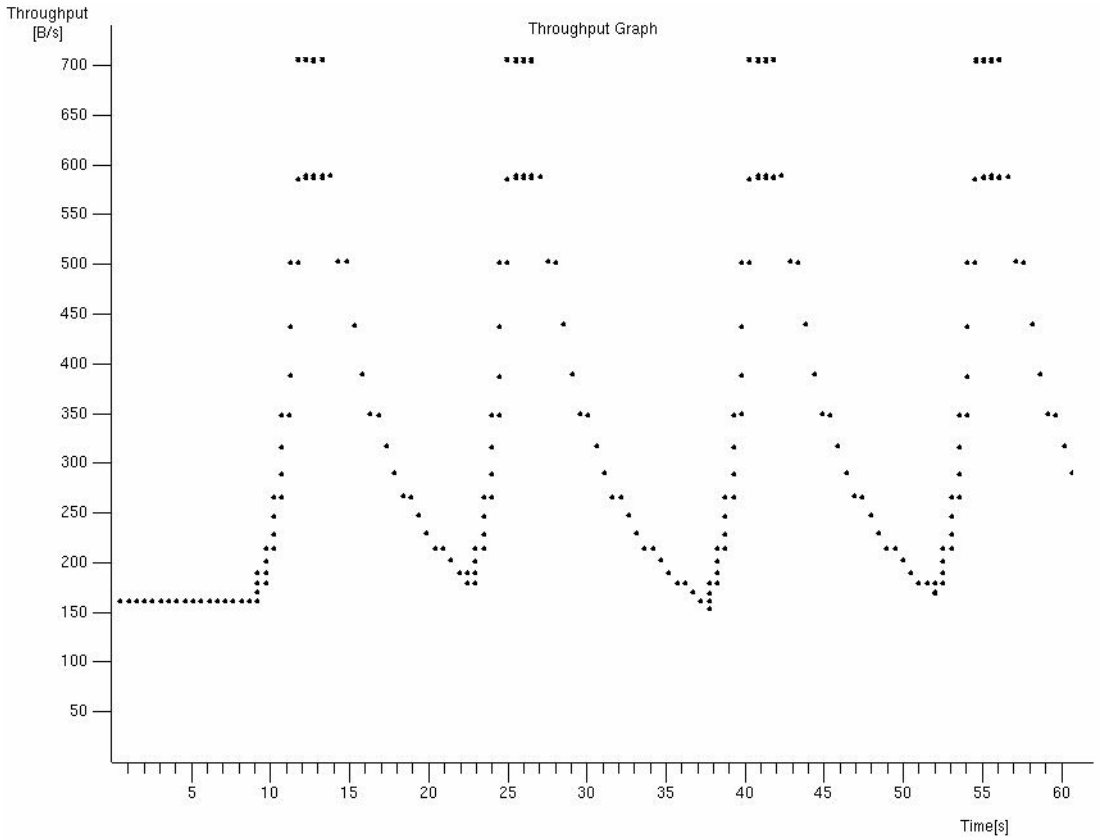


*Figure C-22. TCP throughput graph of 4 forced handoffs with standard TCP in 60s.*

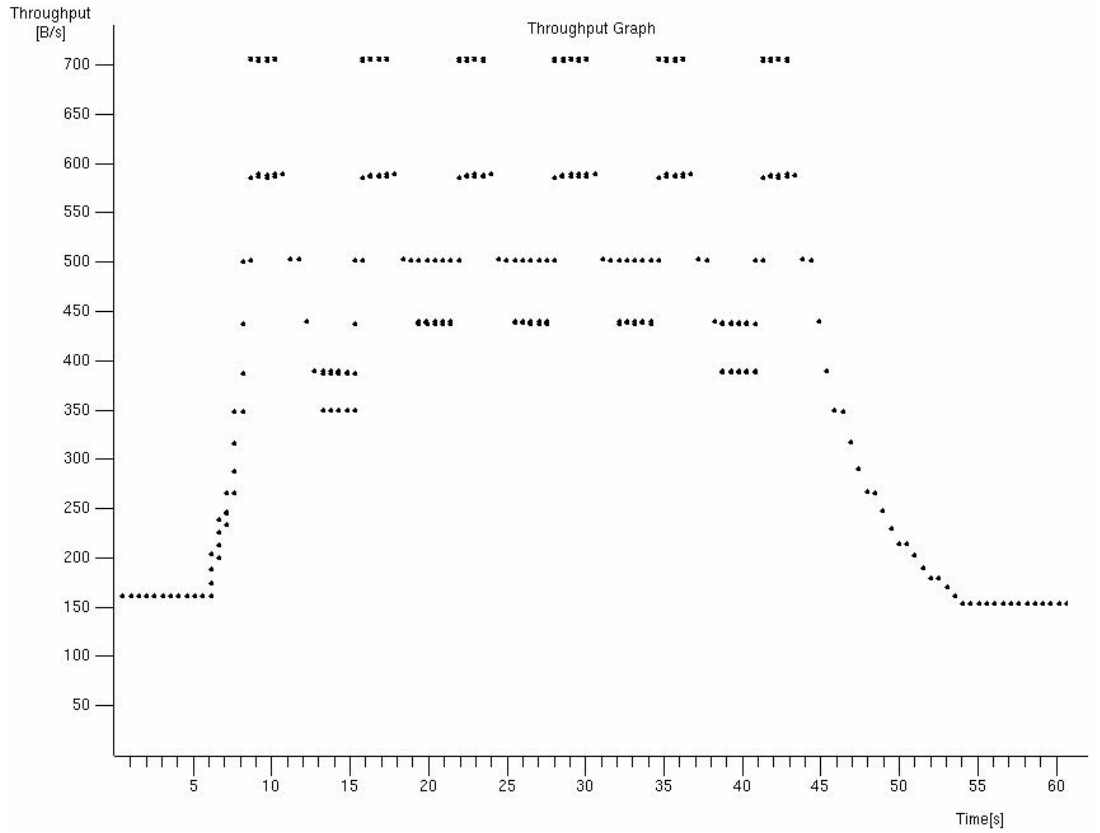


*Figure C-23. TCP throughput graph of 6 forced handoffs with standard TCP in 60s.*

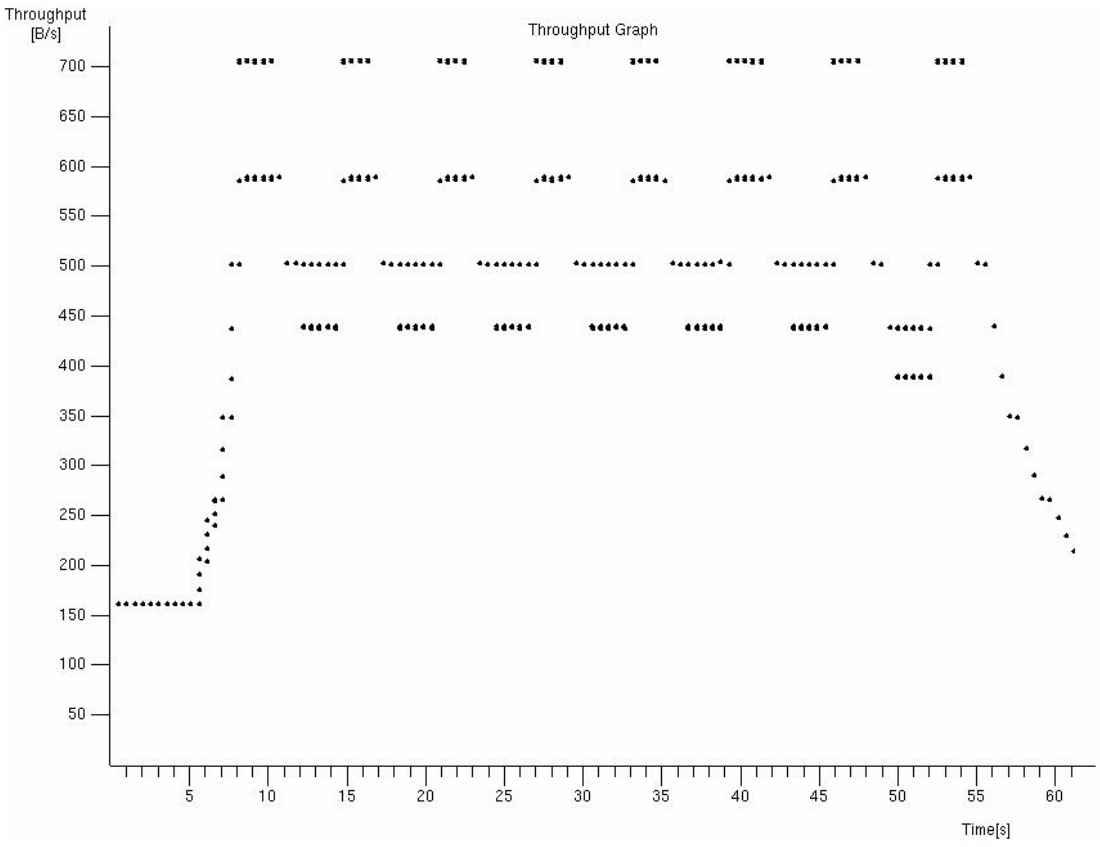


*Figure C-24. TCP throughput graph of 8 forced handoffs with standard TCP in 60s.*

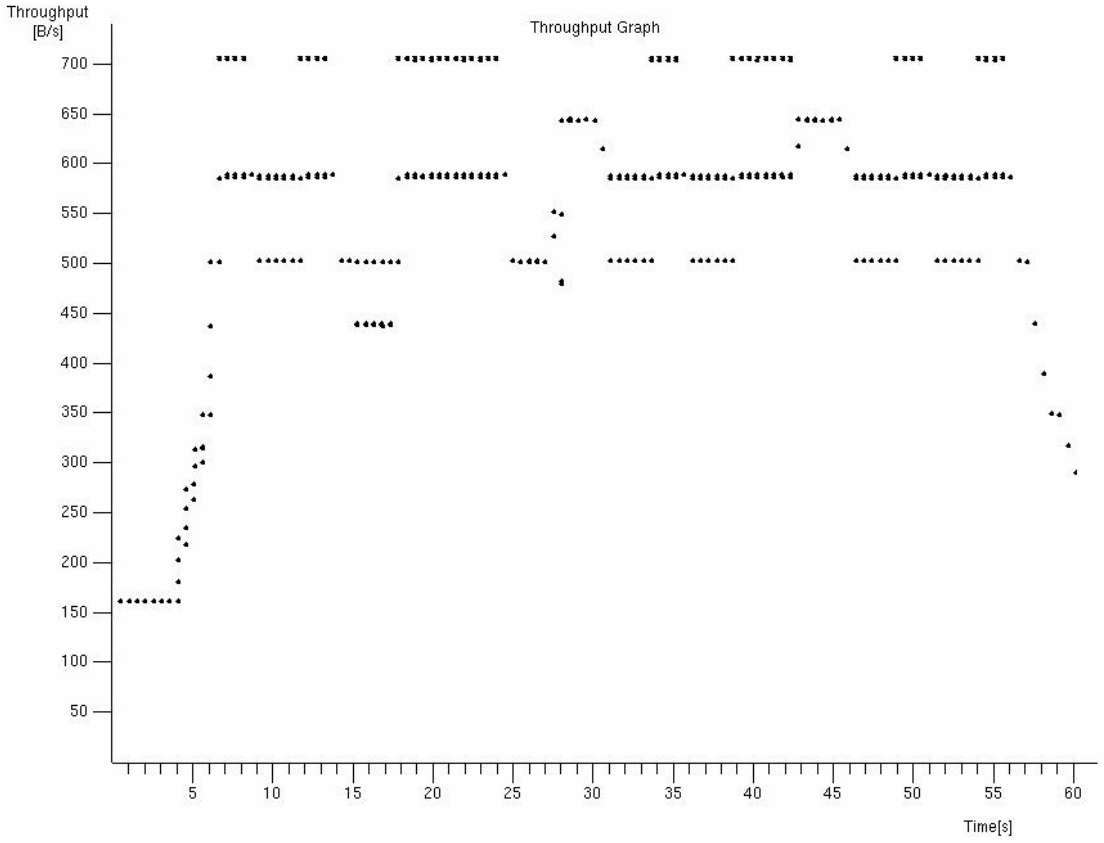


*Figure C-25. TCP throughput graph of 10 forced handoffs with standard TCP in 60s.*

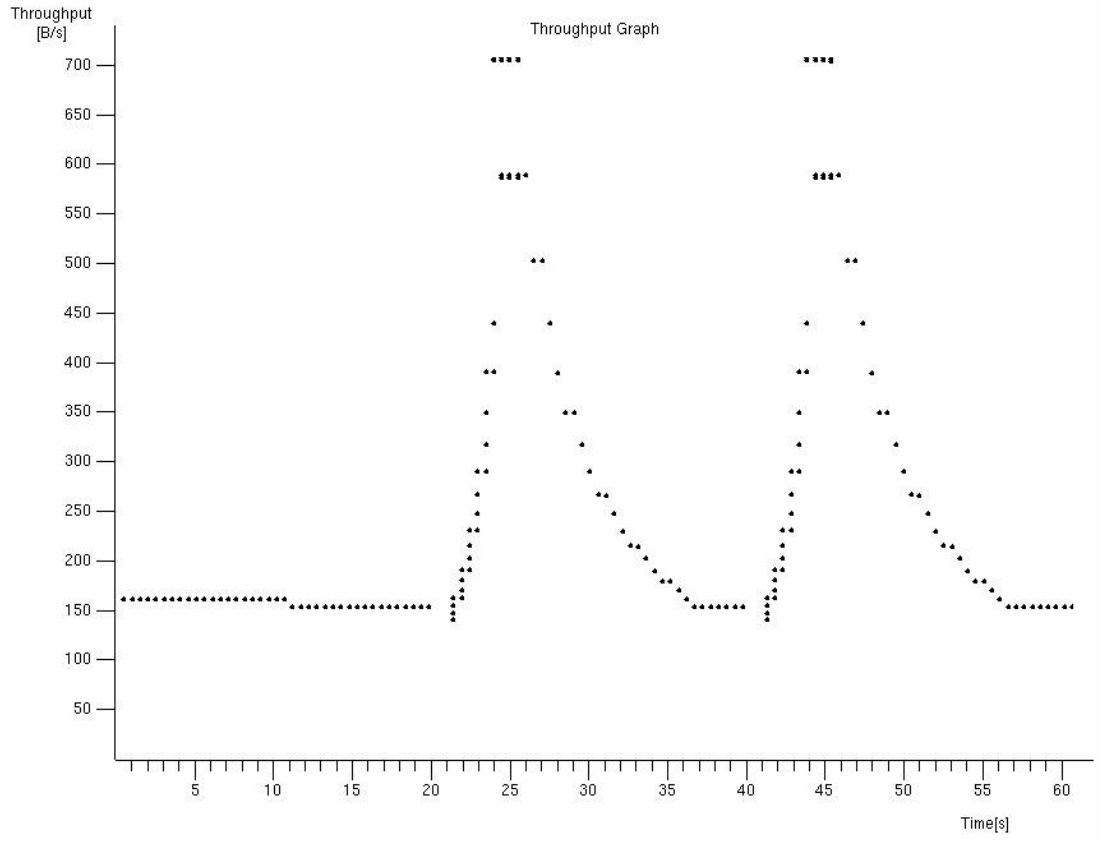


*Figure C-26. TCP throughput graph of 2 forced handoffs with Freeze-TCP in 60s.*

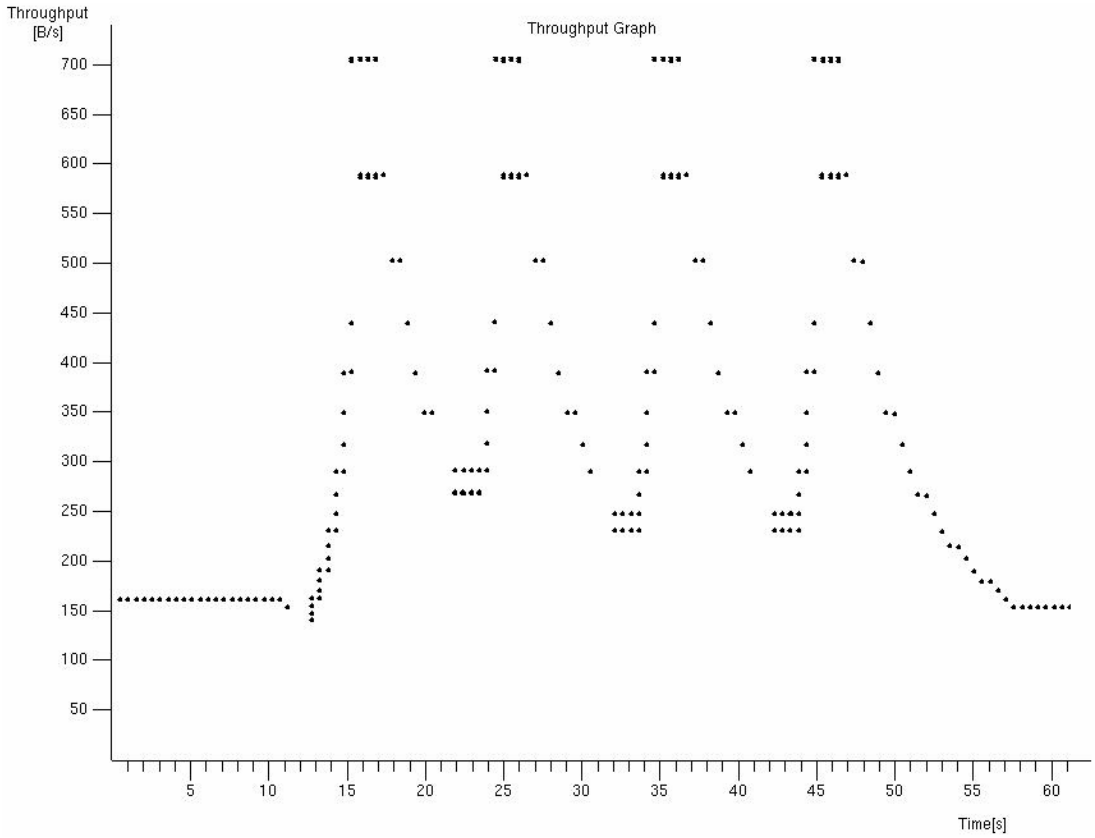


*Figure C-27. TCP throughput graph of 4 forced handoffs with Freeze-TCP in 60s.*

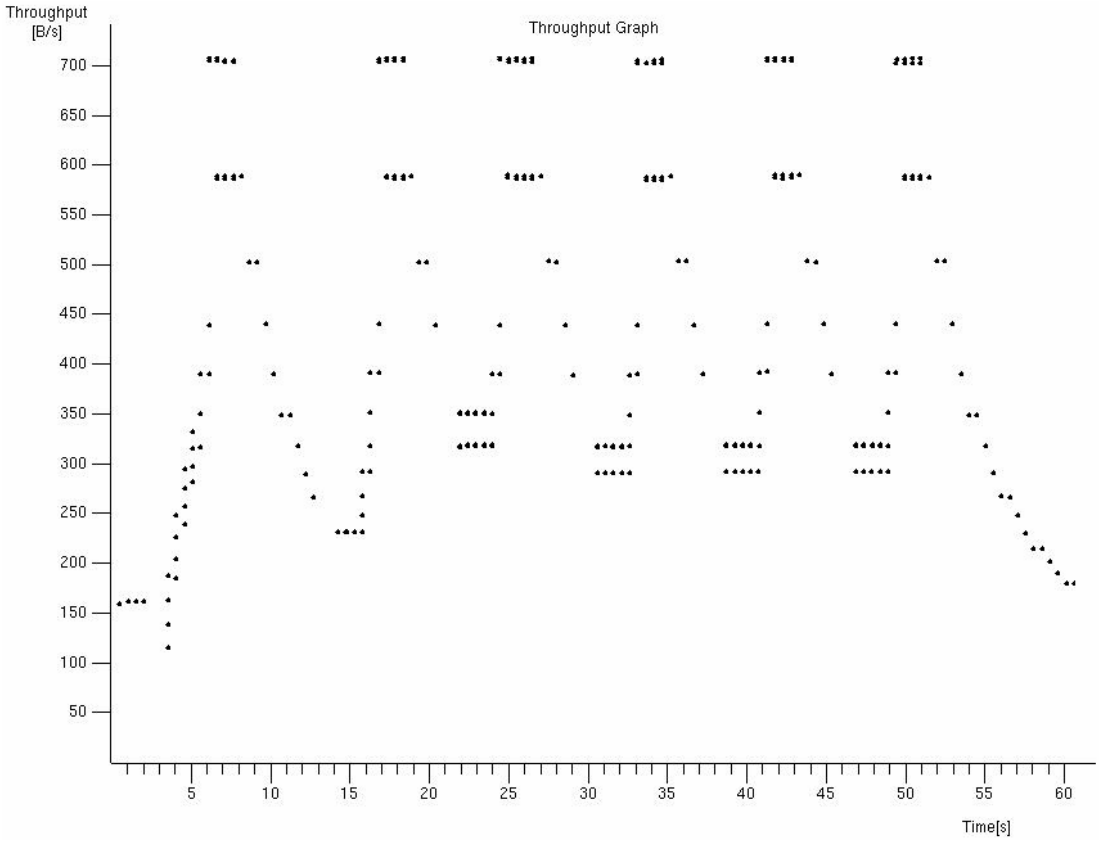


*Figure C-28. TCP throughput graph of 6 forced handoffs with Freeze-TCP in 60s.*

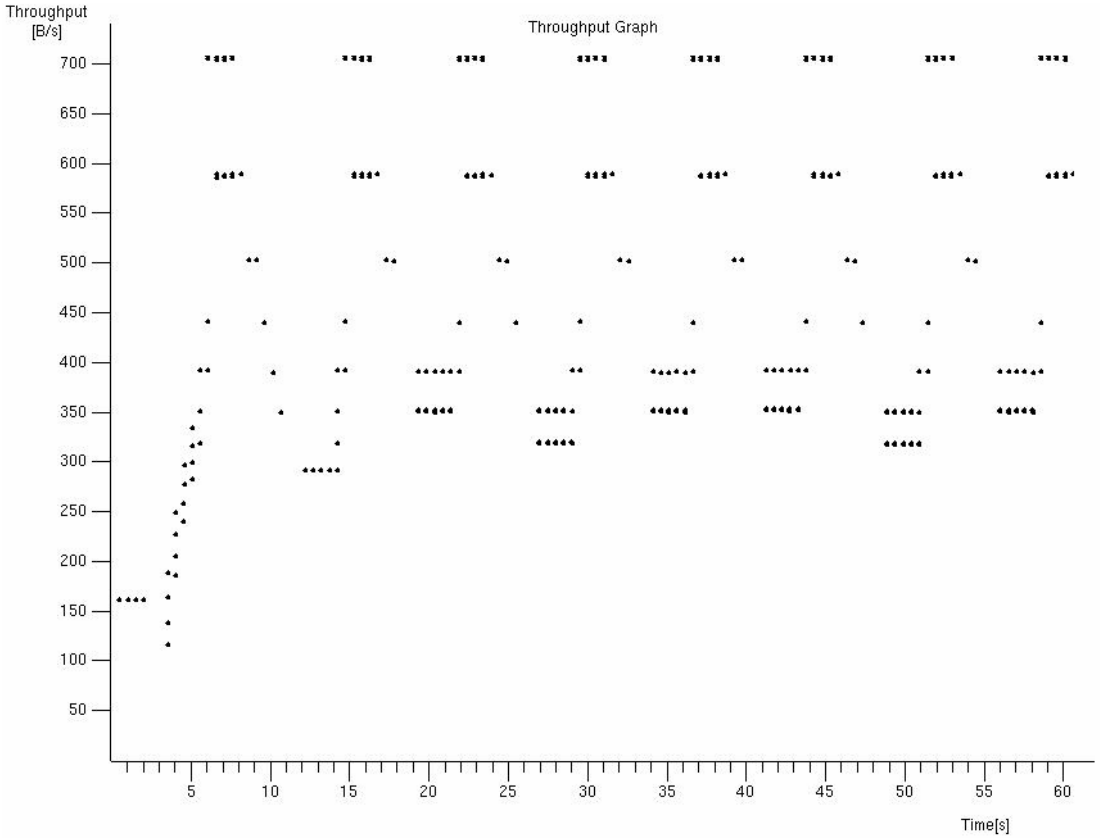


*Figure C-29. TCP throughput graph of 8 forced handoffs with Freeze-TCP in 60s.*

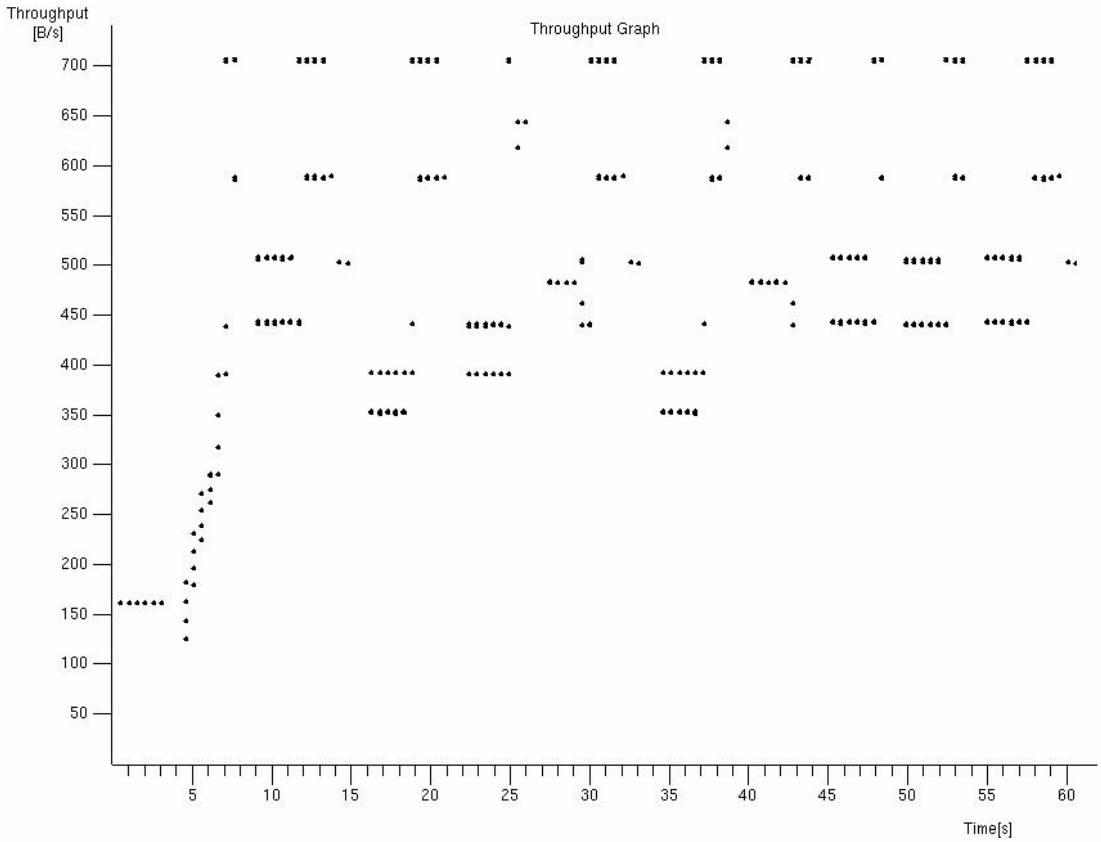


*Figure C-30. TCP throughput graph of 10 forced handoffs with Freeze-TCP in 60s.*

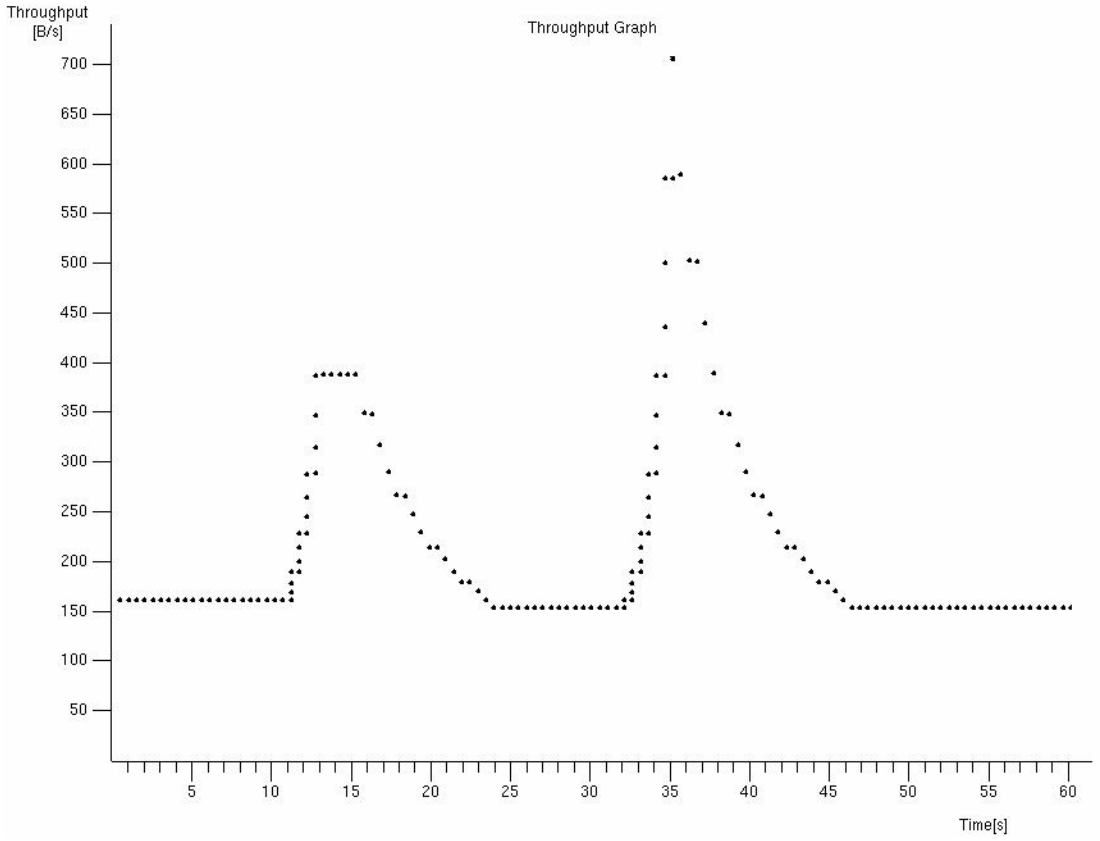


*Figure C-31. TCP throughput graph of 2 SNR handoffs with standard TCP in 60s.*

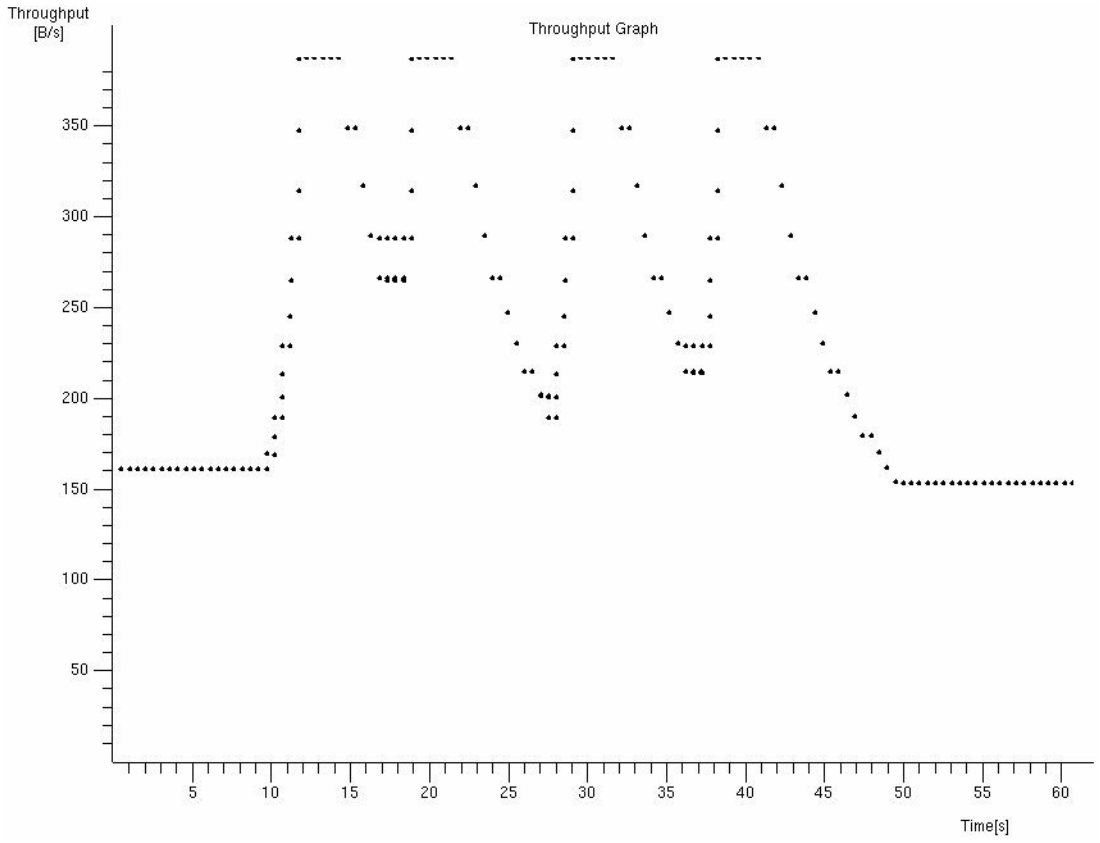


*Figure C-32. TCP throughput graph of 4 SNR handoffs with standard TCP in 60s.*

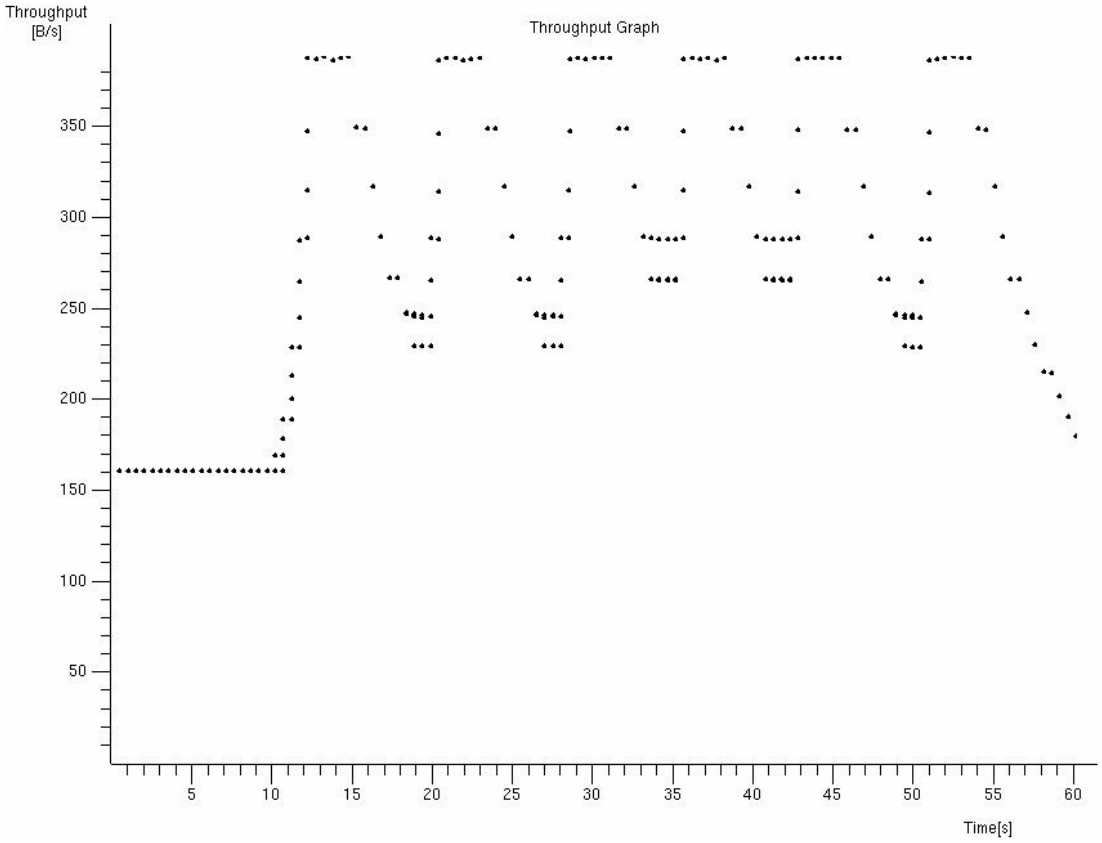


*Figure C-33. TCP throughput graph of 6 SNR handoffs with standard TCP in 60s.*

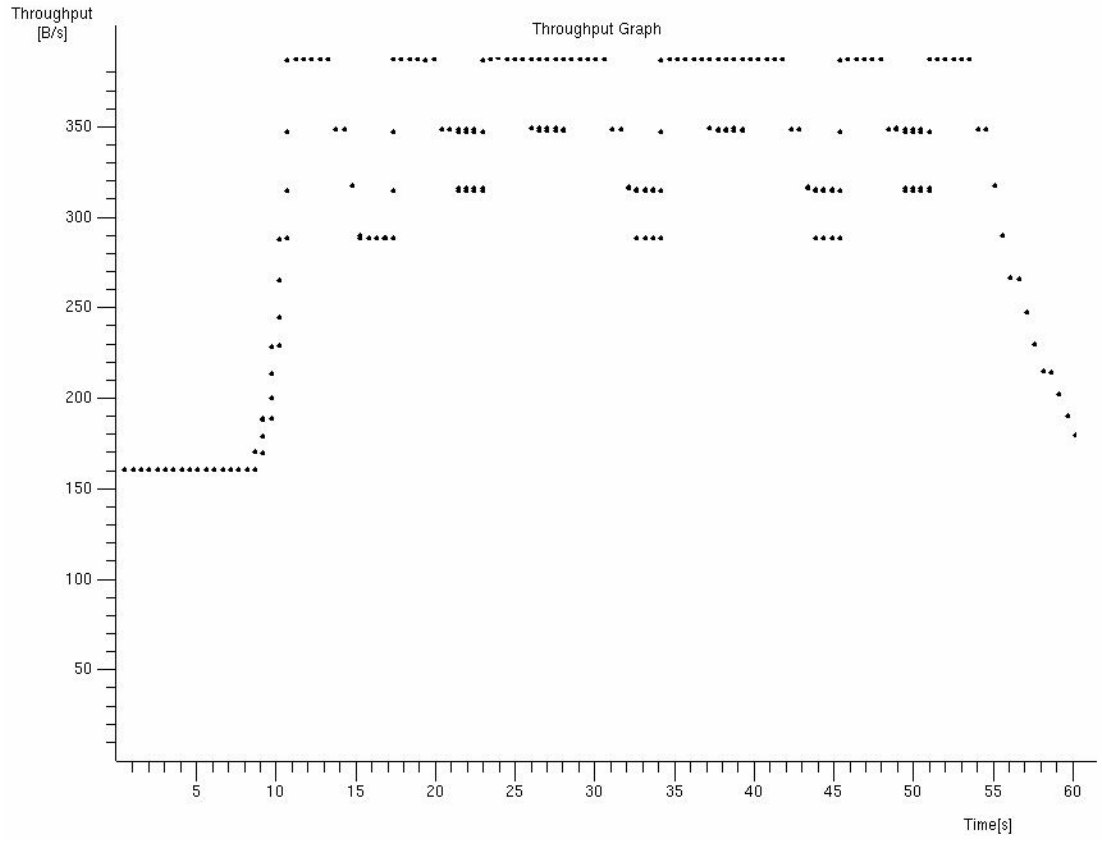


*Figure C-34. TCP throughput graph of 8 SNR handoffs with standard TCP in 60s.*

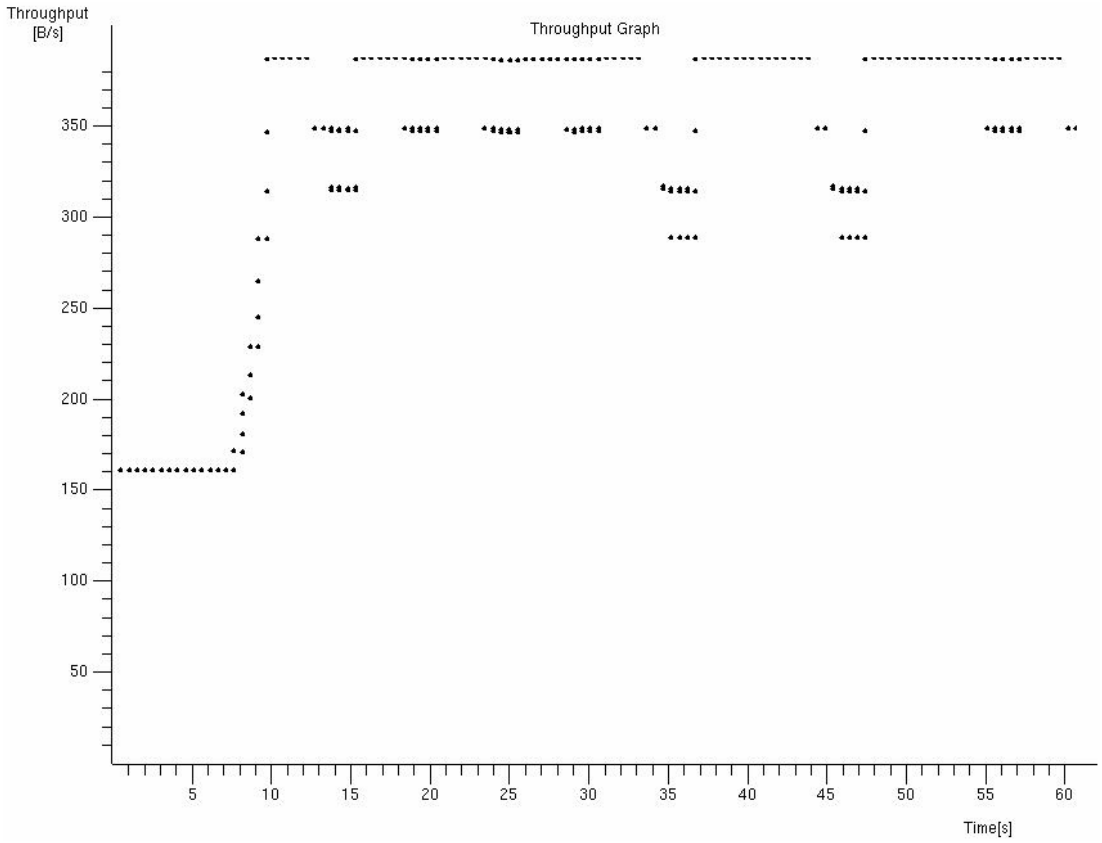


*Figure C-35. TCP throughput graph of 10 SNR handoffs with standard TCP in 60s.*

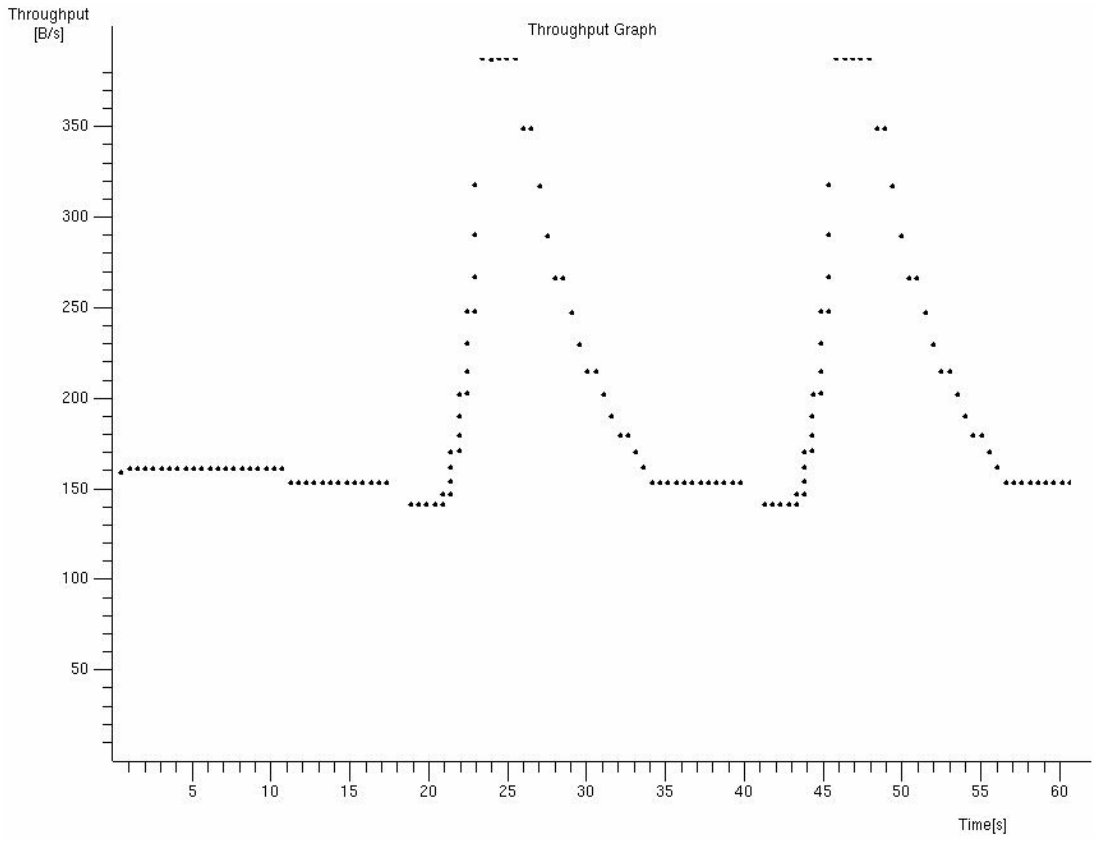


*Figure C-36. TCP throughput graph of 2 SNR handoffs with Freeze-TCP in 60s.*

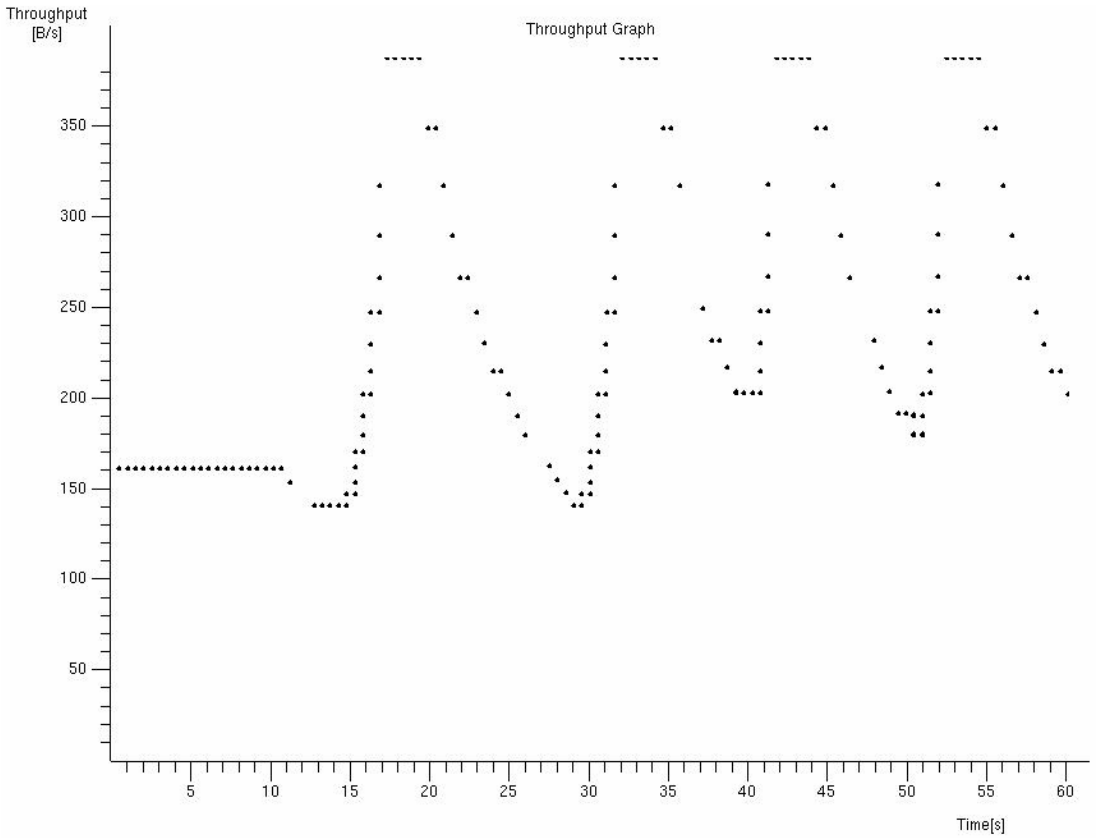


*Figure C-37. TCP throughput graph of 4 SNR handoffs with Freeze-TCP in 60s.*

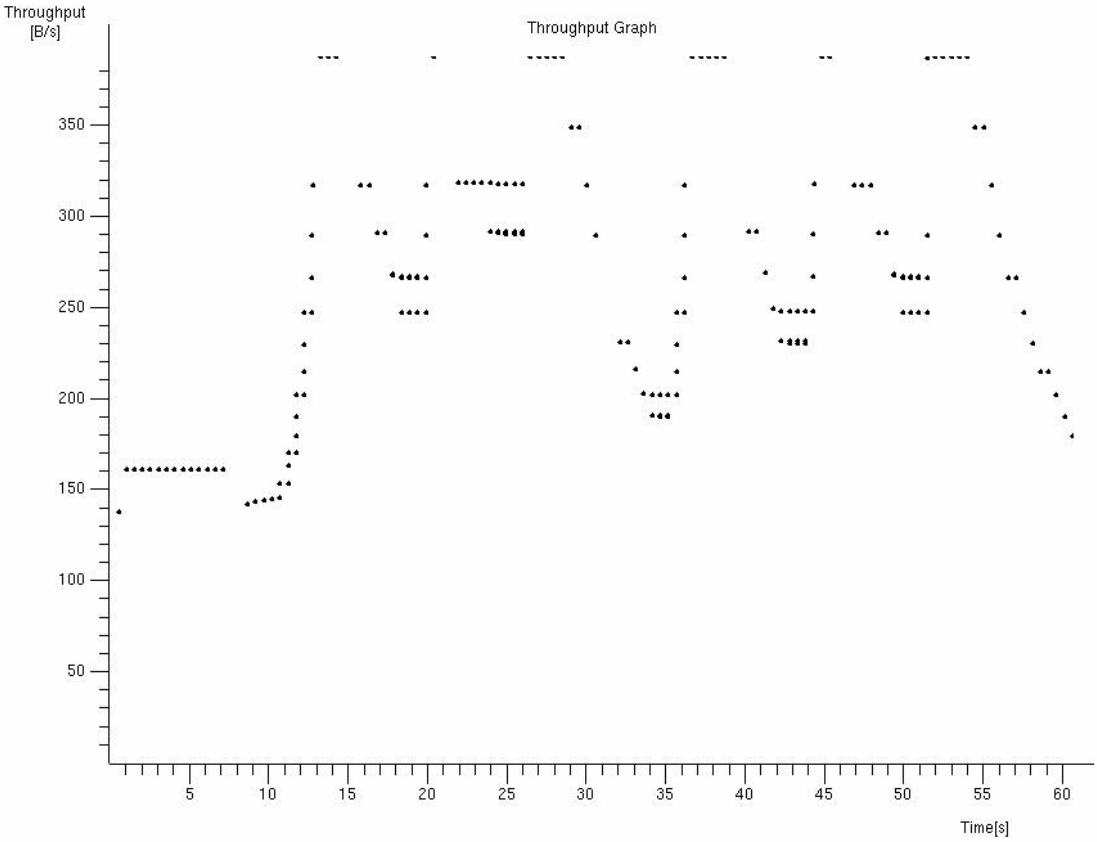


*Figure C-38. TCP throughput graph of 6 SNR handoffs with Freeze-TCP in 60s.*

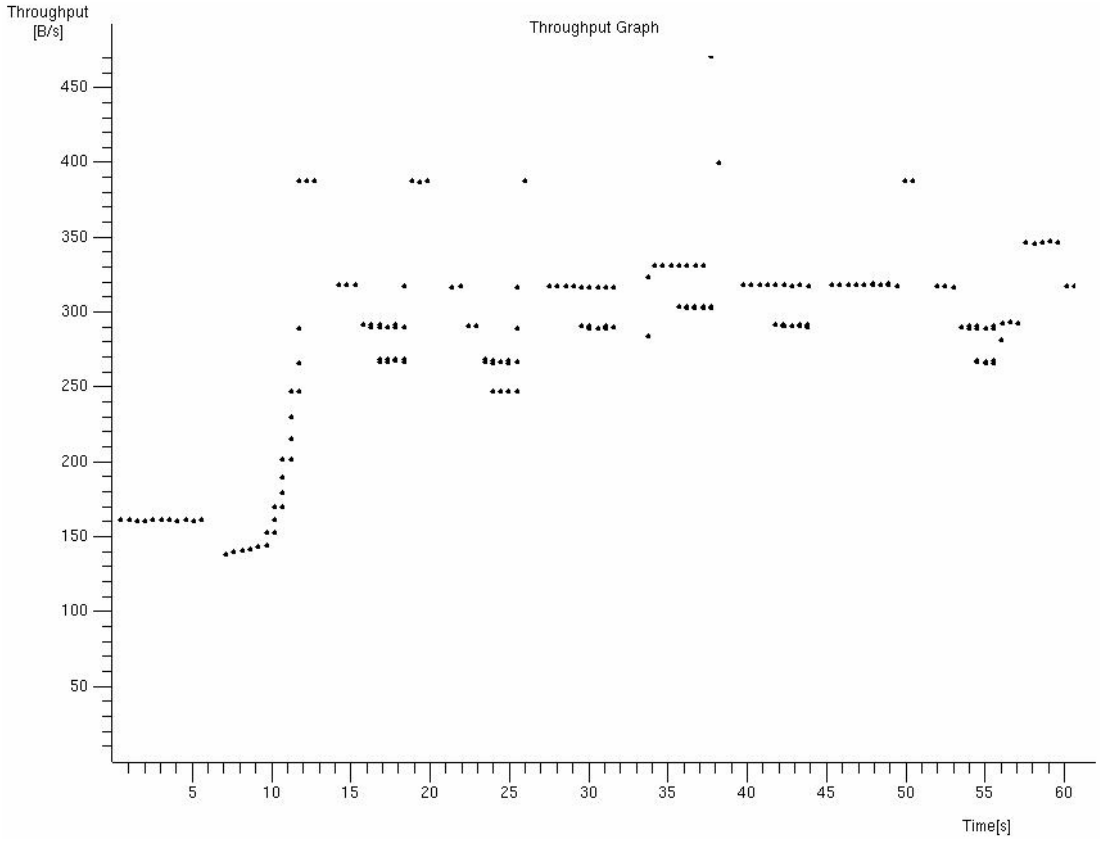


*Figure C-39. TCP throughput graph of 8 SNR handoffs with Freeze-TCP in 60s.*

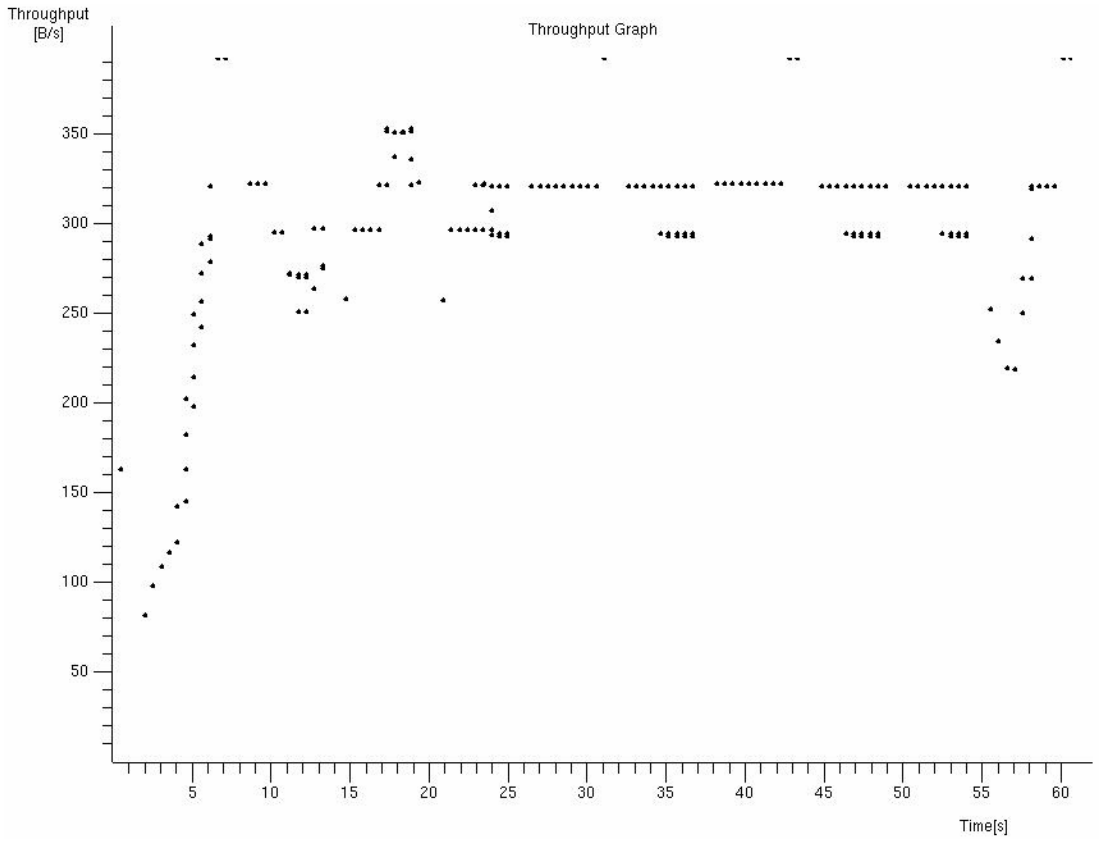


Figure C-40. TCP throughput graph of 10 SNR handoffs with Freeze-TCP in 60s.