Connection Oriented Mobility Using Edge Point Interactivity

A thesis submitted

to Kent State University in partial

fulfillment of the requirements for the

degree of Masters of Science

by

Sandeep Davu

May 2008

Thesis written by

Sandeep Davu

B.Tech, JNTU, India 2002

M.S, Kent State University, 2008

Approved by

Dr. Javed I Khan ,Advisor

Dr. Robert A Walker , Chair, Department of Computer Science

Dr. Jerry Feezel , Dean, College of Arts and Sciences

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

To my dad, I dedicate this book

Chapter 1

Introduction

Connection Oriented mobility requires moving the connection abstraction to new attachment point without disturbing existing applications. Achieving seamless user experience for applications that rely on reliable communication is a demanding task. Most of the existing applications like FTP [27], HTTP [26] use connection oriented communication. Achieving mobility for connection-oriented applications is a challenging task. Single layer solutions either require an infrastructure or significant re-engineering of protocol stack, which are hard to deploy.

This Thesis recognizes the problems in connection oriented mobility and proposes a solution based on cross-layer interaction. The proposed cross layer interaction is secure, flexible and application-aware. In fact the application subscribes to be notified of an event (state change) within a layer. The decision of whether or not to trigger action on the event notification is left to the application.

The rest of the chapter is organized as follows. First section describes connection oriented mobility followed by challenges in connection oriented mobility. Third section emphasizes the importance of cross layer interaction. The following section presents the work done followed by contributions.

1.1 Connection Oriented Mobility

Past decade has seen a drastic change with device communication. With the portability of the devices and good progress in wireless communication, a device can have various network attachment points. With internet widely available mobility has become an integral part of device communication.

The host identity along with the connection dynamics is abstracted to network attachment point. When a network attachment point changes the connection abstraction is no longer valid. All the applications which use connection abstraction are stalled. The challenge lies in not just obtaining a new network attachment point, but in doing so seamless to communicating devices that are using the connection abstraction.

Communication between two devices is logically classified into two categories– connection oriented and connection less protocols. This distinction is made at the transport layer. The layer below the transport layer, called the core network, remains generic. The rationale to adopt such a design principle is to move specialized application-oriented functionalities up into the upper layers and out of the core of the network. The core should be kept as simple and generic as possible, and should only provide general-purpose data transfer services that can be used by all kinds of network applications.

1 Connection Oriented Protocols

Connection Oriented transport protocols should perform a) reliable data transfer -- orderly delivery of data and b) loss recovery -- coping up with any loss during transmission. Most of the applications would want data transfer to be reliable. For

example, a file transfer protocol would require all the transmitted data to be received by the application at the other end and in the order it was sent. The major success of the internet is attributed to reliable data transfer and hence connection oriented protocols. One such widely used protocol is the Transport Control Protocol (TCP) [28]. Added to reliability TCP also has comprehensive congestion control mechanisms [29, 30] for adapting to overload network conditions and recovering from packet loss. More on 80% of today's internet traffic uses TCP.

2 Connectionless Protocols

Connection Oriented Protocols could not scale with multiple simultaneous connections due to the excessive processing to reorder data, congestion control and loss recovery. To overcome the problem of multiple simultaneous connections, connection-less protocols were designed which performs the minimal tasks of a transport protocol. Connectionless protocols do not guarantee reliable data transfer. User Datagram Protocol (UDP) [31] is a connection less protocol which keeps track of the packets and delivers them to the corresponding application. This minimal state design of the transport protocol helped newer applications like voice and video which were otherwise hard to realize.

Though UDP is connection-less, there still exists a demanding need to track the connection state (congestion, packet-loss) at higher layers (Session Layer, Presentation Layer or Application Layer). UDP moved the functionality out of transport layer allowing higher layers to track and adapt to the connection states. Though the resultant approach taken by the layers above UDP may differ from TCP, the common factor of

tracking the connection states could not be truly eliminated. Advanced Session Protocols like RTP [32] and SIP [33] use timestamps or timers to keep track of lost messages.

Irrespective of connection-oriented or connection-less transport used, mobility is hard to achieve. The delays caused with wireless transmission medium and network handoff might misguide connection oriented transport. Current transport protocols require a significant re-engineering of the protocol to dynamically update connection abstraction, whenever the network attachment point changes. Thought connection less transports can be used for mobility solutions, most legacy applications (almost 80%) cannot achieve seamless application space. Unexpected disconnections are problematic with session based protocols (like FTP, SSH, Telnet) which track the connection states.

This thesis focuses on connection oriented mobility, challenges in mobility and a clean end-to-end solution to handle connection oriented mobility. The next section will describe in detail the challenges of connection oriented mobility.

1.2 Challenges in Connection Oriented Mobility

In this section we will present the two major challenges of mobile network – Frequent Disconnections and Handoffs.

1 Frequent Unexpected Disconnections

. Despite the improvement in technology and the wide spread nature of 3G networks a mobile node cannot always be connected due to resource constraints. If the disconnections were to be expected, designing intelligent applications to handle these

disconnections would suffice. Disconnections are rather frequent and on most occasions fast and unexpected. Wireless medium is error prone unlike the traditional wire-line networks and hence the frequency with which these disconnections occur is note worthy.

2 Network Handoff

A most important requirement of internet mobility is handoff -- to allow network applications continue to function as the host changes network attachment point. Most mobile portable devices are wireless and during an impending handoff will experience increasing Bit Error Rates (BER) accompanied by disconnection during handoff. This will confuse the advanced reliable transports like TCP, which in turn invokes robust congestion measures degrading the data transfer rate. Network address change alone can be easily handled by host configuration protocols like DHCP [34]. The real challenge lies in migrating already existing connections seamlessly.

1.3 Cross-Layer Interaction

The fundamental design principle of the internet is the OSI reference model. In the OSI model each layer has distinct functionality and maintains a clear separation between layers. The implementation details and the state transitions in each layer remains within that layer. Each layer performs a specific task and collective effort of all the layers provides end-to-end communication.

Mobility can either be handled in Network Layer or in transport layer. Network Layer Solutions require extensive modification of underlying core network. This approach still

cannot stop advanced transports from reacting to packet delays. This limitation is compensated to some extent by employing application level function to estimate the state information, which is often redundant. Transport layer solutions achieve mobility support by significant re-engineering of the protocol stack. This will allow the network core to be generic and provide general purpose routing for any kind of application. Though this approach is end-to-end, has access to internal states, and provide a cleaner solution, require customizable changes making them difficult, time consuming and impractical to realize.

The ability to access internal state information by layers above will undoubtedly enhance the decision taken in the upper layers. Estimation of internal states of underlying layers are time consuming and in most cases redundant. Moreover, this state information is readily available and could be made available using cross-layer interaction.

1.4 Our Work

This thesis discusses about the protocol extension needed for mobility and justifies the design principle employed. The fundamental approach is to perform simple, light-weight re-organization (meta-engineering) on the protocols of the core networks to enhance them by making them interactive and transparent. This approach though defies the basic principle of layer separation, keeps the network core simple and generalized. Hidden information from the network core is exposed to higher layers for adaptability. Actual protocol extensions can be achieved at the application space by programmable modules

called transientware modules. These modules can then pull-up the protocol's state information needed for adaptation or service extension, perform the required action, and if needed push-down any results or state updates. This mechanism of secure and orderly delivery of internal network state information is called Interactive Transparent Networking (InTraN) [10, 11]. Interactive TCP (iTCP) [35, 36] was developed based on the InTraN mechanism prior to this work. iTCP tracked events related to congestion and used that information to slow down the rate of data transfer during periods of congestion.

This thesis focuses on a systematic cross-layer interaction between networking layers to achieve high performance loss-free handoffs. A Scheme called Interactive Protocol for Mobile Networks (IPMN) is designed based on InTraN. Events of interest from three layers (802.11 MAC, IP, TCP) during mobility and in particular handoff are subscribed. Information from internal states like movement detection, authentication, and association are securely opened up to higher layers. This three layer interaction produces a cross-layer optimized solution for mobile networks. The basic iTCP architecture with a new set of events and transientware modules are designed for the new requirements to prove the adaptability and extensibility of the protocol.

This thesis implemented 2-layer architecture with IP-TCP interaction with the application. The results from FreeBSD based implementation gave over to a more detailed design of a 3 layer scheme using 802.11b MAC layer. An analytical model and performance analysis illustrates our claim of fast-free handoffs in comparison with Mobile IP (MIP) [1]. We propose a connection freeze before handoff and resume the connection after the handoff has been performed. The impending handoff is

communicated to subscribed applications through event notifications. Decision making and customization is performed at the application layer.

1.5 Contributions

. End-to-End protocols are sufficient to provide adequate connectivity and graceful handling of host movement. The technique developed in this thesis makes the following contributions.

1. Developed Interactive Protocol of Mobile Networks (IPMN) [13, 37, 38] technique using the InTraN architecture to achieve rapid loss-free handoffs. Optimized the handoff by using cross-layer interaction between L2-L4-L7.
2. IPMN enabled kernel implementation of L3 and L4 with a rich set of API for event-notifications and information retrieval/pushback mechanisms.
3. Proposed different approaches and reused some of the existing schemes at the application layer based on information attained from L2 illustrating the adaptability, scalability and flexibility of IPMN.

The IPMN has two limitations, rather these are valid assumptions made given the complex applications that surround mobility. We assume that the mobile host is enabled with location tracking application which can initiate an <IP address> search that will let the application know the next point of attachment even before the handoff. Secondly, it requires a kernel change at both the end-points mainly the server. In most cases the server is in a different administrative domain and cannot be modified. We argue that with the

increasing inclination of mobile users any network layer solution will not suffice and pure transport layer solutions are practically not deployable.

1.6 Thesis Organization

In Chapter 2 we will review related work. In Chapter 3 we will elaborate on the driving principle for choosing the architecture and detailed design of Interactive TCP (iTCP). In Chapter 3 we show the IPMN architecture, implementation details and design of the transientware solution. We present initial implementation of IPMN enabled kernel and its performance comparisons with Mobile IP. Later in the chapter we re-evaluate the design and refine the protocol for fine grained adaptation of internal network state information. Chapter 4 summarizes the modeling of the refined protocol along with Mobile IP. Both IPMN and MIP are compared and the results show that IPMN far out weighs the performance even after the little signaling overhead.

Chapter 2

Background and Related Work

This thesis requires background in the functionality of wireless—Data Link layer, IP -- Network Layer Protocol, TCP – Transport Protocol. Problems encountered due to mobility in each of these layers are discussed briefly before detailing handoff in mobility. It is interesting to note how each of these layers are affected by mobility. Solutions to mobility are presented in later section. Related work in protocol engineering is also presented in the following section. The chapter is concluded by discussing why most of the solutions fall short in solving mobility. In the pretext of solving mobility, single layer solutions induce problems in layers above them. Cross-Layer mismatch during mobility is presented in detail and multi-layer solution approach is justified.

2.1 Wireless Data Link Layer

With the enormous growth of wireless communication over the past few years and large spread availability of portable wireless enabled devices, it is inevitable that wireless communication medium will become ubiquitous.Though wireless becomes popular, it is impossible to replace the existing wired medium, rather both the counterparts need to coexist. Wired medium is more reliable and less error prone, while wireless is more error prone due the nature of the communication medium. The bandwidth offered by wireless network is very less compared to the wired counterparts.

Also a wireless network has lot of Bit Error Rate (BER) due to the nature of its transmission. Data transfer will be affected where BERs are high. The data link layer will wait for retransmissions to happen slowing down the rate of transfer. Regions of high BER will most of the times be accompanied by unavailability of service further distorting the transmission.

As we will discuss in the later sections this will not only effect the internal state transition within the layer but might cause advanced transports like TCP to react adversely. TCP accounts all the packet loss to congestion in the network. There is no built in mechanism to distinguish between packet drop due to bit error rate and packet loss due to congestion.

Compounding this problem, mobility also poses its own problem. In most cases problems with mobility and wireless are interrelated. Changing network attachment points will be accompanied with higher BERs and service disconnections causing packet

transmission delays. Furthermore mobility will have frequent changes in network attachment points, leading to connection disruption. The network address change will render all the existing connections stale and useless.

2.2 Network Layer (Internet Protocol)

Internet Protocol is the binding factor for all the networks to remain connected. The main functionality of IP is to route packets correctly from source to destination. This uses special software deployed in the internet called routers. Every host connected to the internet will be provided with an IP address to identify its logical point of attachment. Every IP address has two parts, the network address and the host address. The network address part identifies the network the host belongs to and the host address part uniquely identifies the host. The hierarchical deployment of IP addresses is to easily handle the complex task of routing the packets.

Each packet in the network has a source IP address and a destination IP address in the IP header. Routers in the network do a complex table lookup to forward the packets to the next hop which is either another router or the destination host. IP relies heavily on the IP address and the routing tables to successfully accomplish its task. Advanced routing algorithms use other factors (like link bandwidth, queuing delay) to determine the best possible route from a source to destination.

The enormous and dynamic nature of the internet will not guarantee that every packet from a source to destination will follow the same route. Rather it provides a best-

effort nature where the packets reach their destination using a best possible route available at that instance of time.

Mobility needs a network address change to remain connected if it has moved to a different network. This address change will leave IP layer without any idea where to forward the packets that arrives with the previous header. It must either have a redirect mechanism which needs an infrastructure or a direct modification of the header which will require the core network to be modified.

2.3 Transport Control Protocol

Any advanced and reliable transport mechanism would address three important aspects in connection oriented communication -- Reliable Data Transfer, Loss Recovery and Congestion Control. Here we will discuss one of the most important transport protocol called Transport Control Protocol (TCP) extensively used in today's networking world. TCP establishes a logical connection between the end points of data transfer and maintains state information for reliable data transfer and also loss recovery.

2.3.1 Reliable Data Transfer

Each packet uses a sequence number while data transfer. This sequence number is used to for orderly delivery of data to the application and waits till the packet is acknowledged at the other end. The rate of data transmission is dictated by TCP using the window size. The number of unacknowledged packets in the network at any time is

exactly equal to the window size of TCP. Bandwidth utilization is directly proportional to unacknowledged packets in the network (TCP window size). It also uses a timer based retransmission mechanism for reliable packet delivery. This comes with an induced cost of reducing bandwidth utilization by adaptively adjusting the TCP window size. On receiving packets, TCP reorders them and delivers to the application

2.3.2 Loss Recovery

Due to the uncertainty of the Internet packets may arrive out of order at the receiver. If an out of order packet is received the receiver sends a duplicate ACK to the sender. This is repeated whenever there is an out of order packet. When the TCP on the sender side receives three successive duplicate ACKs it assumes a packet loss and retransmits the lost packet. This mechanism is called the Fast Retransmit [40]. To avoid further loss TCP reduces its window size by half and adaptively increases that linearly on every successful ACK received. This process is continued until the original window size is achieved. This is termed as Fast Recovery. Both Fast Retransmit and Fast Recovery are used to recover and avoid further packet loss.

2.3.3 Congestion Control

TCP heavily paces its state transitions on a set of adaptive timers. The timeout value for retransmission timer is calculated using the Round Trip Time (RTT) between the sender and the receiver. This timer is exponentially backed off with each unsuccessful

retransmission. Window size of TCP is also adjusted to 1 reducing the transmission rate. This technique called Congestion Avoidance stops further degrading the performance of TCP. TCP increases the window size exponentially on every successful ACK until a threshold value is reached. After reaching threshold TCP starts incrementing the window size linearly to avoid flooding more data into the network. This mechanism is termed as slow start [39].

TCP was designed for conventional wired networks to address reliability and orderly delivery of data. An estimate of the network condition based on timers is used for state transitions in TCP. With the wireless communication medium and mobility posing newer set of problems, neither the estimate of the network condition nor the state transitions are valid. Three Duplicate Acknowledgements or Retransmission Timeouts in wireless medium could be due to physical constraints (low Signal strength, out of service area), higher BERs, network handoff). These false state transitions will degrade the transmission rate and in worst cases might even tear down the connection abstraction.

2.4 Mobility

A computer is no longer an immovable end-point that remains in one place for the lifetime of its operation. Today's computing devices are portable and internet is ubiquitous. A device may use more than one network ranging from cable modem, wireless, Wide Area Network (WAN) to Local Area Network (LAN) during the course of the day. Each time the Mobile Node(MN) changes its network attachment points, it will

get a new host identity (IP address). A user still expects a seamless operation for all network applications.

A logical communication link between end-point is uniquely identified using a four tuple <source address, source port, destination address, destination port>. When a network attachment point changes both the host and connection abstraction are no longer valid and hence cannot continue the communication. The connection abstraction has to be updated during the course of the connection whenever there is a change in the network attachment point. The process of remaining connected even during the network address change is termed as handoff. Handoff is classified into two types. Let us look into the both these types in a little detail.

2.4.1 Link Layer (L2) Handoff

In a typical wireless environment, the MN receives packets through a wireless link which requires an Access Point (AP). Each AP will have different signal strengths and covering range depending on the hardware design and the placement of the AP. This might lead to having more than one AP serving the same subnet due to inability and various other physical constraints in the wireless environment. A Link Layer handoff occurs when a MN is moving from one AP to another. If the handoff is with in the same subnet the MN need not be assigned a new IP address. The only thing that will change is the access point that is servicing the MN. All the packets can be routed to the same IP address and the MN will still be receiving the data without any hindrance. These L2 handoffs remain

transparent to the IP layer (L3). The only delay caused here is the duration of the L2 handoff. We will in this thesis only consider the most widely used standard for wireless data communications, IEEE 802.11. As explained in [23] the delays contributing to L2 handoff are broadly classified into three categories -- movement detection, Authentication, Association/Re-Association. Each of these will be explained below.

- **Movement Detection**

Whenever the MN moves from one AP to another AP the MN has to know that it has changed its location. This is done either actively by MN sending timely beacons to know its current status or passively waiting for advertisement beacons from the APs to determine the serving AP. These Agent Advertisements (AA) determine whether MN needs to perform a handoff. Movement Detection is the most time consuming process during L2 handoff.

- **Authentication**

Once a MN determines to perform a handoff using Movement Detection it requires to register with the new AP for maintaining connectivity. Authentication avoids spoofing and helps for secure data delivery. Sophisticated authentication schemes can be incorporated in order to have secure handoffs which make sure that the data is delivered to the correct destination without any pilferage.

- **Association/Re-Association**

Successful authentication will let the MN use the services provided by the AP. AP will build status information for all the MNs it is serving. This status information will be used

to buffer the unacknowledged packets and transfer packets to a new AP in the event of a handoff. During Re-Association the new AP will get the status information from an AP to which the MN was previously registered..

All the three factors will contribute to the handoff latency. Typically handoff latencies range from 1100ms to 1900ms on IEEE 802.11b cards depending on the provider of the card, movement detection alone takes around 1000ms to 1600ms. This indicates L2 handoff detection alone is the single most time consuming process.

2.4.2 Network Layer (L3) Handoff

While performing a L2 handoff if the MN also changes the subnet then the current IP address is no longer valid. The network attachment point of the MN changed and requires to be assigned a new IP address. The change in IP address will render all the active connection abstractions invalid. Performing a L2 handoff alone will not suffice in this case as the network address change is not handled in L2. Logically handling this connection would require a Network Layer (L3) handoff which seamlessly allows data transfer even after the network address change. This thesis concentrates more on the network layer mobility featuring cross-layer optimization and effectively handling mobility at the end-points. Let us first look at the existing solutions for internet mobility.

2.5 Solutions to Mobility

In this section we will look into the solutions that are already proposed and will also comment why we think these solutions are difficult to deploy and use. Solutions that only modify the IP layer and induce indirection in the routing protocol are termed as Network layer solution for mobility. If the solution includes higher layers such as TCP while dealing with mobility we term them as Higher Layer Mobility solutions.

2.5.1 Network Layer Solutions

If the process of maintaining connectivity after changes in network attachment point is handled at IP layer then those solutions are termed as Network Layer Solutions. The layers above IP will not know of the address change. These solutions require an intermediary intervention either in the form of an infrastructure or ip translation/mapping agents. Some of the most popular solutions are discussed below.

- **Mobile IP**

Mobile IP [1] provided a first effective crack of handling mobility. A MN will always be assigned a permanent IP address in one of the networks. This is done by registering to the MN to an agent called the Home Agent (HA). For all communication channels MN uses same IP address regardless to network attachment point. This solves the problem of identifying the MN uniquely but creates a rather complicated problem of routing packets. What will happen to the packets destined to the current IP address? The

MN intended to receive these packets is no longer in the network to which the packets are routed. When ever a MN moves from it's HA to any other location it will have to register itself with a new agent called the Foreign Agent (FA) to obtain a care-of-address (COA) or co-located address. MN also has to update the HA of its current point of attachment with the COA.

To handle discrepancies in the routing protocol, a routing tunnel from the MN's HA to FA is created. Packets from CH reaching MN are routed through this tunnel. Packets sent to the MN's home address are intercepted by its HA, tunneled by the home agent to the MN's COA, received at the tunnel endpoint, and finally delivered to the MN. In the reverse direction, packets sent by the MN generally delivered to their destination using standard IP routing mechanisms, not necessarily passing through the HA. This creates a triangular pattern and hence this system is also referred to as triangular routing. Though this solved the problem of handling stale connections, it introduced lots of communication overhead and longer triangular routing paths.

- **Mobile IP with Route Optimization**

In this enhancement to MIP [2], the CH will have a "cache binding", which allows it to route packets directly to the care of address of the mobile node. Initially, when the CH doesn't have a binding for the MN, it sends the packets destined for the mobile to its home network. These packets are received by the HA, which then sends the MN's COA to the CH. The CH stores this information in a binding for the MN in its binding cache. This would allow CH to create a tunnel to the FA or MN directly instead of waiting for a

tunnel to be created from HA to FA. Packets are encapsulated at CH and the decapsulated and directed to the MN.

The problem with this scheme is that this would imply that every sender on the internet wishing to communicate with a mobile node would have to have software and hardware that is capable of caching these mobility bindings. Today, IPV4 is not capable of supporting cache bindings, and to implement this would thus require major software changes to the sending nodes. So the scalability of this approach is questionable.

- **Reverse Address Translation**

The RAT (Reverse Address Translation) architecture [4] is based on the Network Address Translation (NAT) protocol. NAT is very largely deployed and used in the industry. IN RAT the registration is done using the existing applications and the routing of packets is done by address translation. It uses packet re-direction service between CH and MN to support IP mobility. Thus the mobile node requires no modifications. However, both the techniques require functional enhancement of the layers involved.

- **Hint based handoffs**

Another approach is for network layer to wait for hints from L2. Hint based handoffs in [3] uses triggers from L2 as hints to inform L3 of a lower-layer handoff initiation, eliminating the cost and complexity of advertisement based movement detection algorithms. [20] also uses assistance form link layer to perform fast Mobile IP handoff through MAC bridging. Other proposals took a different direction by suggesting a deployment scheme of MIP based on existing infrastructure.

2.5.2 Higher Layer Solutions

Higher layer solutions handle mobility at the end-points. The network core remains unchanged and does not require infrastructural deployments. Thought there are many higher layer solutions we will only concentrate on solutions in Transport layer.

- **Indirect TCP**

TCP with split connection tries to split the connection into wired and wireless part and uses different techniques in wired and wireless transport separately. This would let advanced transport layers to employ different state transitions for wired and wirelss part of the connection independently. This leads to the violation of true end-to-end paradigm and much more complex infrastructure. One such approach is Indirect-TCP [5, 9] which splits the wired and wireless parts of the connection using the Base Station (BS) as a common end point. The state information of the connection is transferred between the old BS and new BS transparently during handoffs. MSOCKS[8] employs a similar connection redirection using split connection proxy.

- **TCP-Redirect**

TCP-Redirect[12] is based on an idea- same as ours, renewing the connection to handle the new IP address. But their scheme bases its idea on the conventional timer based approach and hence lacked the robustness and scalability of inter-layer interactivity of our scheme.

- **TCP Migrate**

In Migrate[22] the network communication is well-modeled by a session abstraction, an architecture based on system support for a flexible session primitive. *Migrate* works with mobile "suspend/resume" operation of legacy applications and provide enhanced functionality for mobile-aware, session-based network applications, enabling adaptive operation of mobile clients and allowing Internet servers to support large numbers of intermittently connected sessions.

- **Connection Migration**

[41] presents soft state synchronization and migrating the connection to achieve application-seamless fault-tolerance and load balancing for high-performance servers. A set of co-located servers migrate connections between themselves using transport level soft state synchronization. This work is based on TCP migrate and uses suspend/resume mechanism to achieve connection migration.

- **Freeze-TCP**

Freeze-TCP [6], though does not handle mobility directly, but aids the TCP performance by freezing the connection during periods of disconnection. The main idea behind Freeze-TCP is to move the onus of signaling an impending disconnection to the client. A mobile node can certainly monitor signal strengths in the wireless antennas and detect an impending handoff; and in certain cases, might even be able to predict a temporary disconnection (if the signal strength is fading, for instance). In such a case, it

can advertise a zero window size, to force the sender into the ZWP mode and prevent it from dropping its congestion window. If a zero window probe is lost, the sender does not drop the congestion window. This will avoid a packet loss. Freeze TCP is incorporated into our architecture to ensure that there are no packet losses.

2.6 Protocol Engineering

Active[35, 36, 42] and Programmable networks[43] took a new approach -- to solve problems in traditional networks by deploying customizable code into the core network. Programmable networks tried to solve the problems by modeling programmable application interfaces to the network core. These interfaces can be used by the applications to program the communication channel. Active Networks tries to dynamically create services in the network core by injecting customizable programs into the network elements. Special packets called capsules [44] carry snippets of code to the network elements which can create activate and deploy services. Though this approach was envisioned as a generic solution to most network related issues, it is facing a lot of challenges (security, multi user execution, resource sharing and scalability).

2.7 Cross Layer Mismatch

Network Layer Handoff is potentially disruptive to the transport connections. The most common way of handling this disruption is to add an indirection to the routing

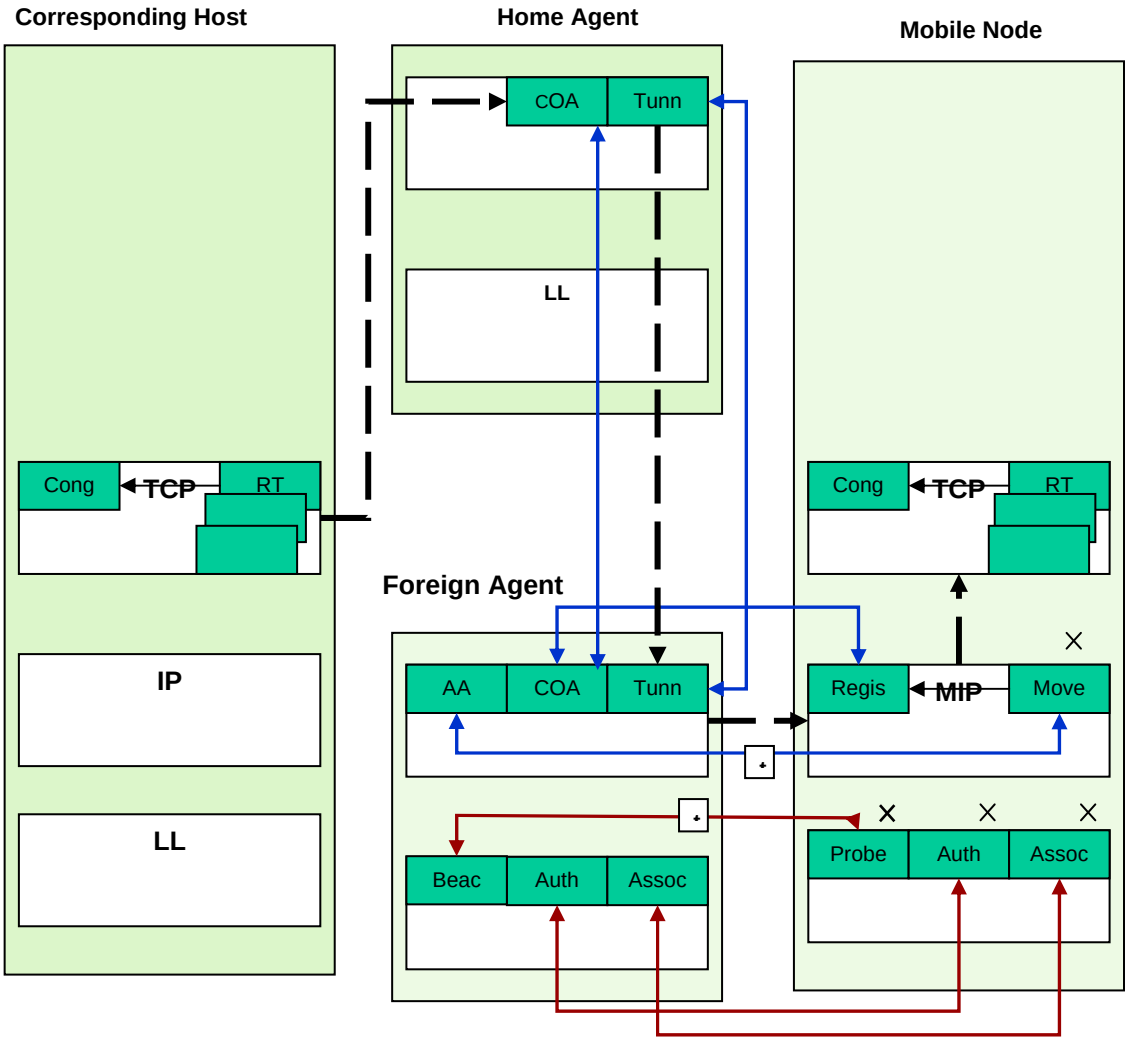


Figure 1. Mobile IP's Complex Handoff Procedure.

mechanism in the end-to-end paradigm. Mobile IP (MIP) [1] uses this approach by adding an indirection to the routing mechanism in the form of home agent (HA) and foreign agent (FA). Mobile IP only requires changes to IP layer and it remains hidden to the higher layers. From individual layer point-of-view it seems to handle the reconnection quite effectively. However, when the cross-layer impact is taken into account the problem seems to be much more compounding. Though sophisticated transports at higher layer are

adaptive, but most of the advanced transport schemes (such as TCP) use timer based adaptation.

Figure 1 shows MIP scheme and the potential points where the information exchange between layers would have been helpful to achieve faster handoff is denoted by 'x'. In MIP scheme the movement detection in network layer is performed based on the periodic beacons received from the outgoing and incoming agents. This requires FA to periodically and continually broadcast a small signal. The next issue is to circumvent the static identity attached routing. This is done by registering the MN to a new foreign agent and updating the same at the HA. To disable the MN's identity based routing a tunnel is created from MN's HA to FA, where the actual IP packet rides inside another packet from HA to the FA. The return path from MN to Corresponding Host (CH) is directly routed through the FA without requiring any intervention of HA. Thus, the lower bound of delay is heavily dependent on the periods.

On the other hand, the TCP on the upper layer heavily depend on a set of timers to pace its state transitions. The delay caused by MIP's indirection mechanism adversely reacts with the entire TCP dynamics. These delays could be easily mistaken as congestion by TCP and the various congestion control algorithms are invoked which then further slows down the rate at which the data transfer proceeds. Such adverse cross-layer reaction remains tolerable only in local area networks, but causes severe performance degradation for connection oriented applications across larger networks. It is not clear that such cross-layer interaction (particularly with TCP) has been well considered at the design time of MIP.

Chapter 3

Interactive Protocol for Mobile Networks – 2 Layer Approach

This chapter presents the solution based on recently developed Interactive Transparent Networking (InTraN) architecture. The design principle of InTraN architecture is explained briefly before presenting the proposed mobility solution. Interactive Protocol for Mobile Networks (IPMN) suggests cross layer interactivity between the networking layers to achieve optimal performance gains. This event based protocol avoids the use of infrastructure in the network hence hugely reducing the deployment costs along with increased scalability.

3.1 Introduction

Classic IP protocol was designed long time ago with no vision about node mobility. Its routing mechanism relies on IP address semantics to deliver packets to a destination node whose location is assumed to be fixed. The same argument also applies to TCP, whose congestion control mechanism assumes that the path between the two endpoints is 'wired'.

Since the advent of wireless technology and the tremendous growth of mobile networking applications, a persistence need has emerged to remedy the TCP/IP stack and make it *mobile networking compatible*.

On the IP level, we need to be able to deliver packets to a mobile node regardless of its current point of attachment, and on the TCP level we need to be able to identify the reasons behind packet loss and react to them differently; if loss was due to congestion on the wired link, we let TCP run its native course–invoke the appropriate congestion control procedure. However, if the loss was due to radio disturbance on the wireless link or due to handoff disconnection, TCP should retransmit as soon as possible without any rate throttling.

In contrast to the approaches presented in the previous chapter, the proposed solution has several distinguishing characteristics. First the mobility reactions are *event based* as opposed to *periodicity dependent*. Secondly, the solution does not require any significant overhauling of the existing protocols rather works in Application Layer (L7). The InTraN enables networking layer protocol events to be orderly and securely subscribed and responded at L7 without requiring functional modification of the existing protocols- with much simpler protocol meta-engineering. This is an interesting diversion from previous thinking that application layer solutions may slow down the overall performance.

On the other hand- since mobility tracking and corresponding reconnection responses can be generated at application layer- its deployment is much easier and also the strategies can be made much more sophisticated and powerful. Indeed the scheme we show

incorporates and extends the strengths of several previous approaches. It could thus offer a dramatically improved handoff and normal time data transport performance- far out weighing the cost.

3.2 Interactive Transparent Networking (InTRAN)

During the past few years, a new paradigm called *Interactive Transparent Networking* was developed. It provides a framework for researchers to deploy protocol solutions/modifications without embedding custom codes within the network layer itself. InTRAN proposes using inter-layer *interactivity* and state *transparency* to be able to

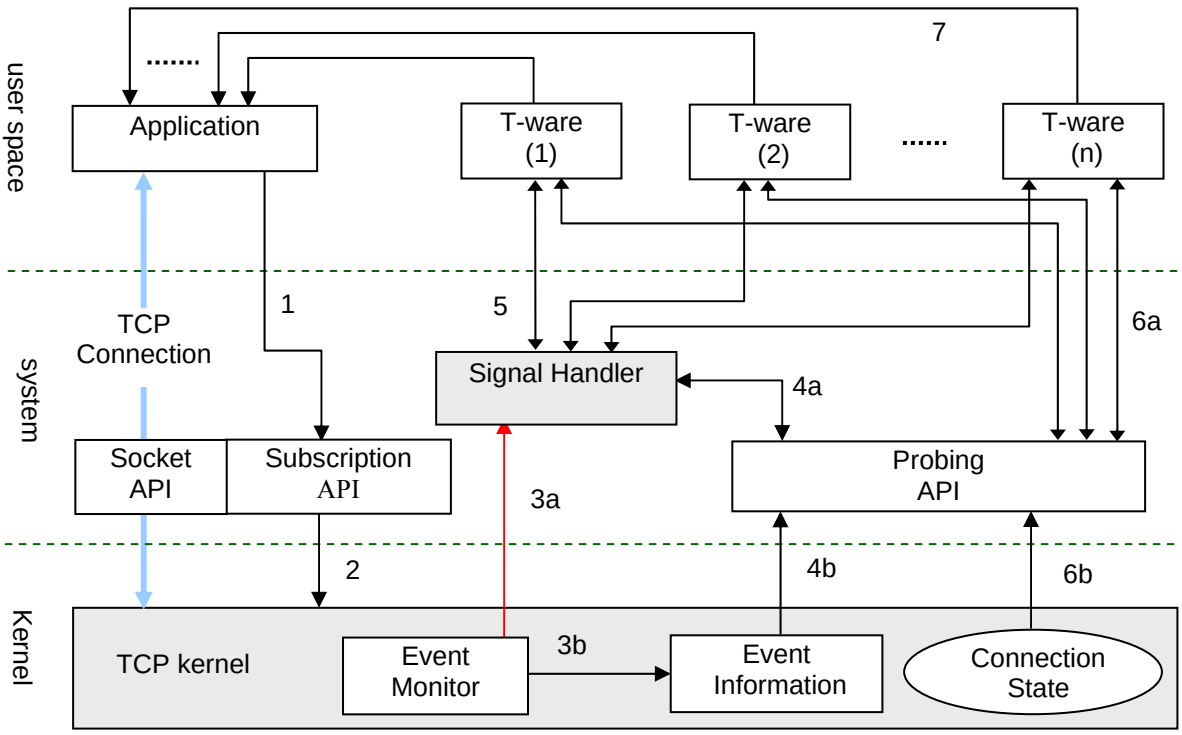


Figure 2. TCP interactive extension and API.

perform these solutions at the upper layers –e.g. at the application layer. In this section we briefly define the new paradigm; more information can be found in [10] and [11].

Within the interactive-transparent framework, we provide means to allow programs in upper layers to subscribe with protocols in lower layers to be notified when certain events (state transitions) occur. Subscribers can then pull up the required service state information, perform the actual action by programmable components running in the upper layer –called *Transientware* modules—and then create handles to push down the generated actions (state modifications) into the target network layer. A subscriber application should be interested in certain events that might occur in the target protocol, and by subscribing to events, the subscriber wishes to be notified when any one of the events has occurred. In Figure-2 we show the general architecture of a TCP-based interactive protocol. Upon opening the socket, an adaptive application may bind a Transientware module to a designated TCP event by subscribing with the kernel. This is represented by arrows 1 and 2 in Figure-2. The binding is optional; if the application chooses not to subscribe, the system defaults to the silent mode identical to TCP classic. When the event occurs in TCP, the kerenl sends a signal to the upper layers (3a) and at the same time it saves the event information (3b). A special handler catches the signal and probes the kernel for the event type (4a, 4b), The handler then invokes the appropriate Transientware module to serve the event (5),. A Transientware module can also use the probing API to access the kerenl state (6a, 6b) or to pass some information to the subscriber application itself (7).

A extensive API set for subscribing to events with appropriate permissions have been identified. In this table for each system call we list its prototype, the level of its caller (user or system), its potential caller (application, signal handler, or *Transientware modules*), and a brief description about its functionality. Some of these functions are designed for the network administrator (root process) to manage event subscription by granting priority levels and access permissions for the user process. This presents a limited security model for InTRAN.

Since InTraN exposes the internal state of the protocol to entities running in the user space, it must address the correctness and safety issues of the underlying protocol appropriately. The T-ware modules with access modes that are Read-Write should be carefully examined as malicious sate update can lead to potential instability of the system. InTRAN proposes a security model which allows controlled access to protocol's variables while maintaining system stability. The danger may come from two sources: (1) a flaw in the protocol design (e.g., wrong type declaration), and (2) a malicious T-ware module of type Read-Write. When a system is running with a dangerous combination, the operating system can optionally activate a guarding program that verifies any malicious attempts to update state variables. If the update is safe, it is allowed to proceed. But, if the update may cause instability in the system (i.e., it is attempting to change a timer or an index variable) then the write operation is blocked immediately and the offending T-ware module is shut down. The guarding program itself is simple and can be implemented as an operating system utility. Basically, it needs to know a set of values that are safe on any

internal variables. This way, the integrity of the InTraN-enabled system can be preserved even in the presence of potential design flaws.

3.3 Interactive Protocol for Mobile Networks (IPMN)

The InTRAN framework works on a set of events that can be subscribed. The first step in designing IPMN [13, 37, 38] was to identify a set of events in each layer that are of interest during mobility. Once identified, the action codes (T-ware modules) for each of the events are modeled. The basic idea of our scheme is to enable the MN to obtain a new IP from the future FA before handoff is performed, replace the 'source IP' field in the TCP/IP stack of the MN with the new IP, and relay the new IP to the fixed host (FH). Once it receives the new IP, the CH immediately switches to the new IP by replacing the '*destination IP*' field in the TCP/IP stack with the new IP.

The assumption made in this scheme is that MN will have a pre assigned IP address for the AR it will be visiting even before a handoff is performed. Using interactivity the application can subscribe to L2 layer events that will notify the application of an impending handoff giving ample time to start some intelligent mobility-aware applications that will fetch an IP address. Once a new IP address is obtained, the process of handoff is initiated by MN

1. *Freeze the TCP connection by advertising a zero window to the FH.*
2. *Perform actual L3 handoff by replacing the IP fields in the TCP/IP stack at both the MN and the FH with the new IP address.*

3. Wakeup TCP by advertising a nonzero window to the FH.

We can benefit from interactivity again by allowing this IP-lookup module to probe L2 for the identity of the next Access Router (e.g., its IP address). There are a number of previous works like [4], [17], and [21] based on DHCP, that have shown excellent schemes that can support handoff pre-processing. More and more intelligent location based schemes are also emerging that can either augment the existing schemes or totally replace them. We can re-model these schemes –or some aspects of them—with the

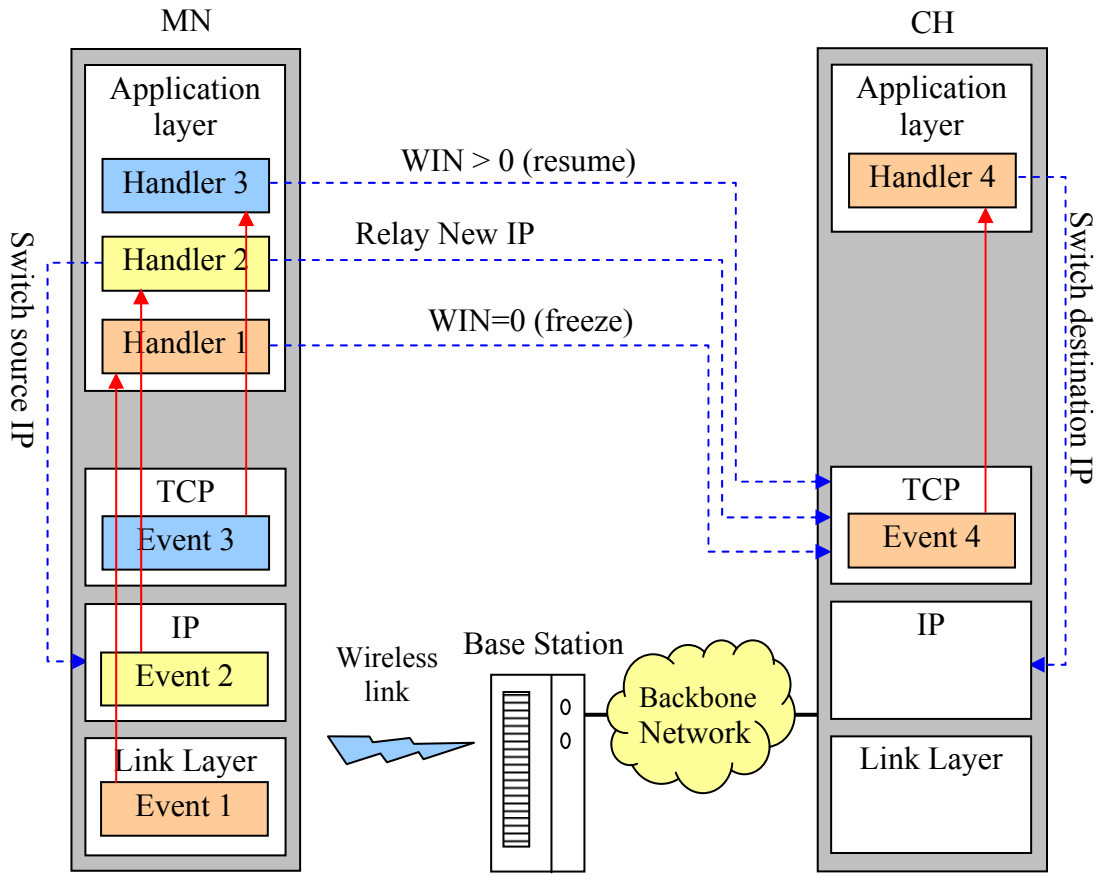


Figure 3. IPMN-Full architecture and event sequencing.

Table 1. **IPMN-Full Events and Sequence Handling**

Node	Event	Layer	Event tracked	Action taken by event handler
Mobile Node	1	LL	L2 handoff has been initiated.	Advertises a zero window to the CH to force the FH to stop transmission.
	2	IP	A new IP has been assigned to the MN from the future BS.	Call the `switch_ip()` system call. This will replace the source IP filed in the IP header of the MN and will send a segment to the CH with TCP option = SWITCH_IP to replace the destination IP field on the CH.
	3	TCP	The 'SWITCH_IP' segment has been ACKed.	Advertises a non-zero window to the CH to resume transmission.
Fixed Host	4	TCP	A special TCP segment received with TCP option=SWITCH_IP.	Get the new IP number from the options part of the segment, then call the `switch_IP()` system call which replaces the new IP in the destination IP field of the IP header.

interactive paradigm to achieve handoff pre-processing. Furthermore, we believe that since the MN can obtain a new IP before handoff, this pre-processing will not impact handoff latency. The design of IPMN led to two implementation paths. IPMN-Half-- A light weight protocol interactivity with application level IP relay message. IPMN-Full – heavy weight, robust, protocol interactivity with TCP level IP relay message. IPMN-Full will have events for identifying an incoming TCP message with IP address. Both the architectures are discussed separately along with event and sequence handling.

- IPMN-Full

At the MN, when the link layer detects signal fading and initiates L2 handoff (event 1), it signals the subscribing application. When the event is received at the application layer, a T-ware module (handler 1) is activated immediately; this module securely updates TCP

internal state variable (win_size = 0). This leads the TCP to advertise a zero window to the CH stopping further packet transmission. Table-1 describes the corresponding events and their handling sequences at each endpoint This would normally cause the CH to stop transmission. Figure-2 describes the conceptual architecture of IPMN. When the MN gets a new IP from the future network (event 2), it activates (handler 2) which transmits the future IP to the FH at TCP level through a system call. The new IP is sent in a special TCP segment with 'option=SWITCH_IP'.

At the CH, When TCP recognizes this option (event 4) it activates (handler 4) which then triggers a switch_ip() system call to replace the 'destination IP' field in the TCP/IP stack with the newly received IP number. In the mean time, at the MN (handler 2) also makes a similar system call which changes the 'source in its own TCP/IP stack. When the 'SWITCH_IP' segment is ACKed at the MN (event 3), the MN advertises a non-zero

Table 2. **IPMN-Half Events and Sequence Handling**

Node	Event	Layer	Event tracked	Action taken by event handler
Mobile Node	1	LL	L2 handoff has been initiated.	Advertises a zero window to the CH. The freeze mechanism of TCP will force the CH to stop transmission.
	2	IP	A new IP has been assigned to the MN from the future BS.	Send a special message to the peer application on the CH. The message carries the new IP just assigned to the MN.On receving the message, Switch_IP() system call is invoked to replace the destination IP field in IP header with new IP.
	3	LL	Handoff Completed	Advertises a non-zero window to the FH. This will unfreeze the connection and enable the FH to resume transmission.

window to the FH which allows it to resume transmission.

- IPMN-Half

Both the event and the handler are in the application layer.Figure-4 describes the architecture and Table-2 describes the corresponding events and their handling sequences. Events 1 and 3 have the same meaning and handling as in the previous 'Full' mode. Event 2, however, is handled differently; when (handler 2) is activated, it makes a system call to probe the TCP layer for the new IP number. It then sends this IP number to the correspondent node using a normal `write()` socket operation. Naturally, since the correspondent node can '`read()`' the new IP directly from the socket, it does not have to catch any events or activate handlers. When the correspondent node receives the message it strips off the IP number and makes a `switch_ip()` system call—as in the

Table 3. API Extension set of IPMN

System Call	Description
`Subscribe(evt)`	Subscribes with a network layer protocol for event (evt).
`GetEventInfo()`	Used by a global *signal handlers* which catches all signals from the kernel to probe the kernel and retrieve event information and then activate appropriate event handler (Transientware module).
`Relay_IP(IP_addr)`	Let TCP transmit a special segment carrying the new IP to the other end.
`Switch_Source_IP (IP_addr)`	Changes the source IP address in local TCP/IP stack. Used by MN only.
`Switch_Dest_IP (IP_addr)`	Changes the destination IP address in local TCP/IP stack. Used by FH only.
`Freeze_TCP()`	Advertise a zero window to the other end (the FH) to force it to halt transmission.
`Resume_TCP()`	Advertise a non-zero window to the other end (the FH) to allow it to resume transmission.

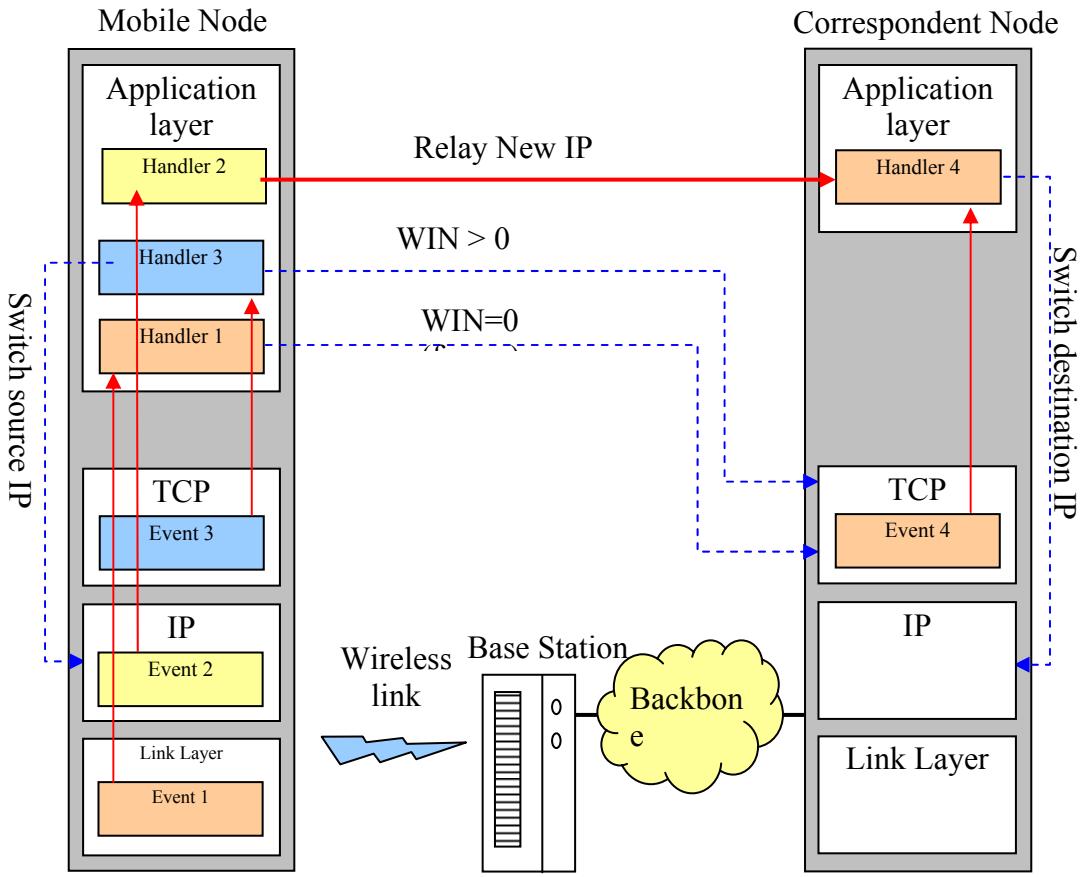


Figure 4. IPMN-Half architecture and event sequencing.

previous mode—to change the 'destination IP' number in the TCP/IP stack. The remaining procedure is identical to the 'Full' mode. Advertising a zero window to the CH to temporarily freeze the TCP connection was proposed in [6] to improve TCP performance over wireless networks.

We adapted this part of their solution in our interactive scheme avoid packet loss during handoff. Although this will slightly disrupt the service while handoff is being performed, but since we avoid packet loss, the CH will not resort to congestion control procedures

avoiding unnecessary retransmissions and sender rate throttling. As we show later, this will definitely improve TCP performance and save network resources.

The purpose of this current implementation is to experiment the basic idea of physically changing the IP address at both end-points whenever the MN changes network attachment point. Therefore, this version of IPMN, only implements the three-step L3 handoff procedure shown above. IPMN tracks three events at the MN and one event at the corresponding host (CH).

3.4 IPMN – 2 Layer Implementation

In this chapter, the two layer implementation will be presented. The rationale behind implementing only 2 layer interactivity out of the 3 layer architecture is due to unavailability of easy access to source code of 802.11 (wireless MAC). We have implemented the scheme on FreeBSD-4.5 by extending the kernel source code. We call the extended implementation *BSD-interactive*. We have created a number of system calls that implement the system's API. The API list is shown in Table-3. The first two system calls are standard in the interactivity service. They allow demanding user applications to subscribe with lower network protocol to be notified when certain events occur. We configured a *Signal Handler* with the OS to catch all signals and filter them. Whenever an IPMN relevant signal is detected, the signal handler uses the GetEventInfo() function to retrieve event type from the network protocol which generated the signal. The signal handler then activates the appropriate *Transientware* module at the application level (e.g.

handler 1 in figure-3) to handle the event. The rest of the API functions in the table are used by these event handlers as described earlier in the IPMN architecture.

3.5 Conclusion

This chapter explained the importance of cross layer interaction. The chapter also dealt with the architecture of InTRAN and how events can be subscribed and internal state variables can be securely modified. Any internal state access should be controlled and InTRAN provides an excellent framework. IPMN is discussed in depth and the two implementation paths IPMN-Full and IPMN-Half are explained. The incorporation of already existing schemes into IPMN design shows the flexibility of the architecture.

Chapter 4

Experimental Setup and Performance Analysis

This chapter explains the experimental setup and performance evaluation. To evaluate performance IPMN is compared against Mobile IP (MIP). Deployment of MIP infrastructure configuration, BSD-Interactive enabled kernel for IPMN and the exact experimental scenarios are discussed. The experiments are performed with various types of traffic – voice, WWW, FTP. Comparison results indicate that IPMN performance gains far outweigh MIP. The inability of MIP to achieve performance gains is discussed.

4.1 Experimental Setup

We used three machines with AMD 1.6 GHz processor (BS1, BS2, and BS2) as our Base Stations and a laptop with Intel P-II processor as our MN. The (GW) machine was our gateway to the Internet and was also used to configure each one of the Base Stations as a separate subnet. Each subnet has four IP numbers per subnet. We installed FreeBSD-4.5 on all BS machines, the MN, and the CN. For Interactivity experiments we installed the BSD-interactive on the MN and the CN only.

For the MIP experiments we installed the MIP implementation of the Portland State University [1]–also known as PSUMIP—on the MN and the three BS machines. One of the three Base Stations machine (BS1) was configured as the Home Agent (HA), and the other two (BS2 and BS3) were configured as Foreign Agents. Figure-4 explains the experiment testbed. For MIP signaling to work correctly, the time must be synchronized on all machines that run the MIP daemons. ntpd utility in FreeBSD is used

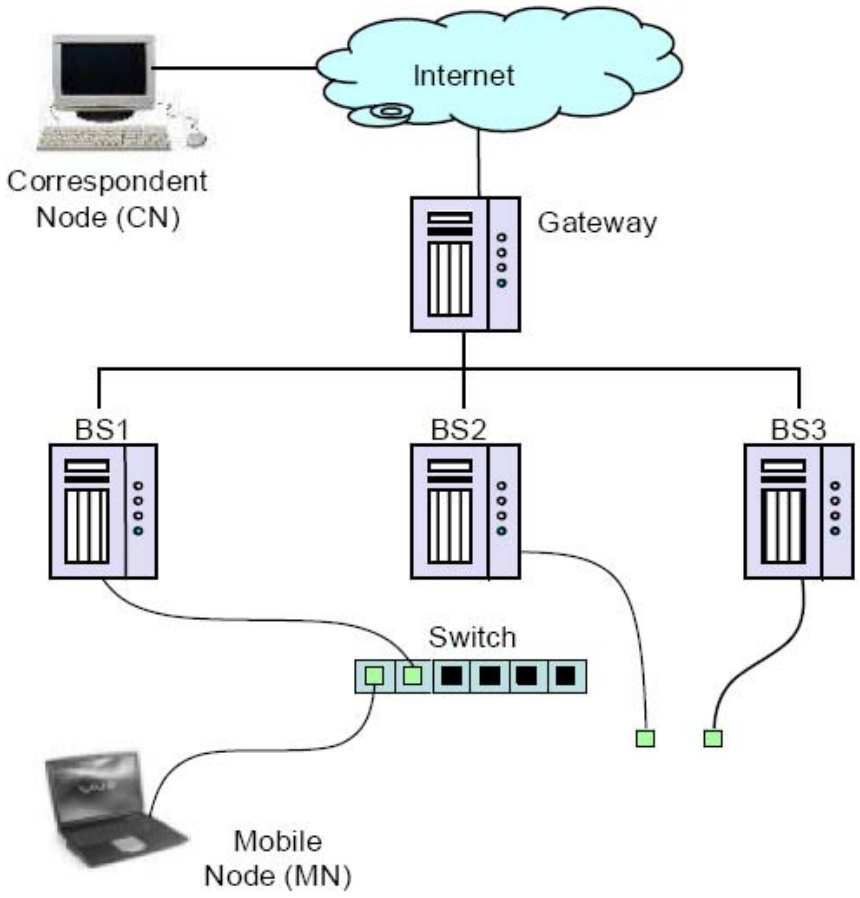


Figure 5: **Experimental Setup**

Table 4. Correspondent Node Locations

Node name	Location	IP number	Average RTT	Hops from MN
Local	Kent, Ohio	131.123.36.11	1 ms	3
Virginia	Chantilly, VA	66.94.95.236	90 ms	19
Texas	Huston, Texas	70.241.64.99	183 ms	26

to synchronize with three STATUM 2 external time servers. We used the simplest possible MIP configuration to reduce unnecessary overhead.

We wanted to test our interactive scheme against MIP and compare performance by measuring application level voice delay, handoff latency, and jitter. We have run all experiments by placing the CN in three locations, one locally (in our lab) and two remotely; in Texas and Virginia. Table-4 shows the three nodes and their route characteristics.

The CN generated voice traffic based on the NetSpec Source Models [45]. We also let the MN move along the cyclic path BS1→BS2→BS3→BS2→BS1… etc. In each run, we let server program at the CN transmit voice traffic until 5 Mbytes has been received at the MN. We configured the MN to perform handoff every 2 minutes. We used a switch to simulate L2 wireless handoff; for example, in Figure-5 the MN is connected to BS1 through the switch. Table-5 gives the API set that we use for IPMN. To perform L2 handoff from BS1 to BS2, we manually unplugged BS1 from the switch and instantly plugged BS2 to an empty port in the switch. We kept the MN connected to the switch all the time.

4.2 Traffic Characteristics

In order to model real-world traffic, we used a tool called NetSpec [45] which was developed at The University of Kansas—to generate traffic at the CN. Netspec offers several source models which can generate traffic for Telnet, FTP, Video, voice, and WWW . Each of the traffic characteristics is used to test the performance of IPMN against MIP. A little overview on different traffic patterns – their characteristics and empirical models used to simulate the traffic pattern will be discussed in this section.

4.2.1 Voice Traffic Characteristics

In NetSpec., voice has been characterized by a constant bit rate (CBR) source. Sampling rate is 8 kHz and each sample is 8 bits. This gives the standard bit rate of 64 Kb/sec for acceptable voice quality. Call arrivals are modeled by a Poisson process with fix hourly rates within one-hour periods. This means that the interarrival time between two calls is exponentially distributed. The probability density function of exponential distribution is given by:

$$f_X(x) = \lambda e^{-\lambda x} \qquad , \lambda = 1/mean$$

Figure-6 shows an example of call arrivals with λ=1 over 5 hours sampling, and Figure-7 shows an example of call duration over 5 hour sampling with three values of λ: λ_1=0.004167, λ_2=0.003333, λ_3=0.002777. The inverse of these λs, we get mean call durations (3, 4, and 5 minutes respectively).

At the call level, the source is presented to the network as a constant-bit stream. To generate a 64 Kb/sec voice stream, talk bursts were generated by a 144-byte blocks separated by 18 ms silence periods.

4.2.2 WWW Traffic Characteristics

WWW traffic is modeled at two levels: *call* level and *session* level. The call level models the interarrival times of multiple sessions.

The session level on the other hand models the traffic patterns within each single session,

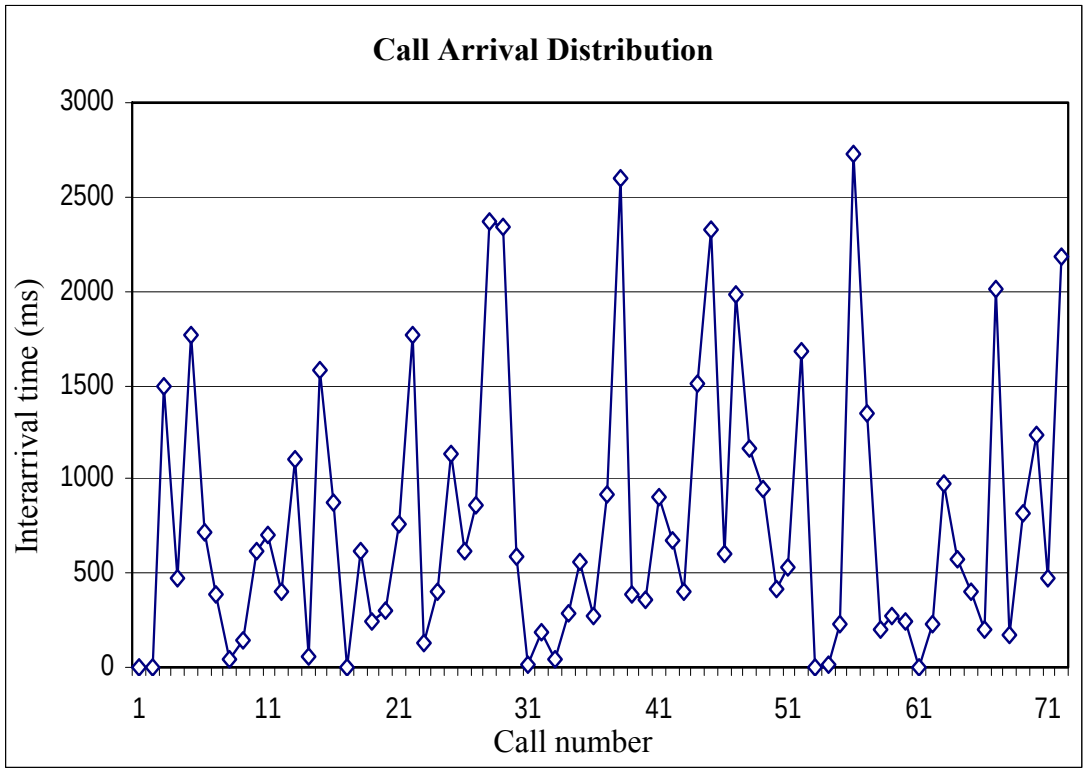


Figure 6. Call interarrival sampling over 5 hours for λ = 1 for Voice Traffic.

such as number of files being transferred and the size of each file. Request arrivals are modeled by a homogenous Poisson process within one-hour intervals. The interarrival time between two requests is exponentially distributed. The probability density function of exponential distribution is:

$$f_X(x) = \lambda e^{-\lambda x} \qquad , \lambda = 1/mean$$

X is the interarrival time between two requests (in seconds). The distribution of document transfer size is a Pareto distribution. The probability mass and cumulative distribution functions of a Pareto distribution is:

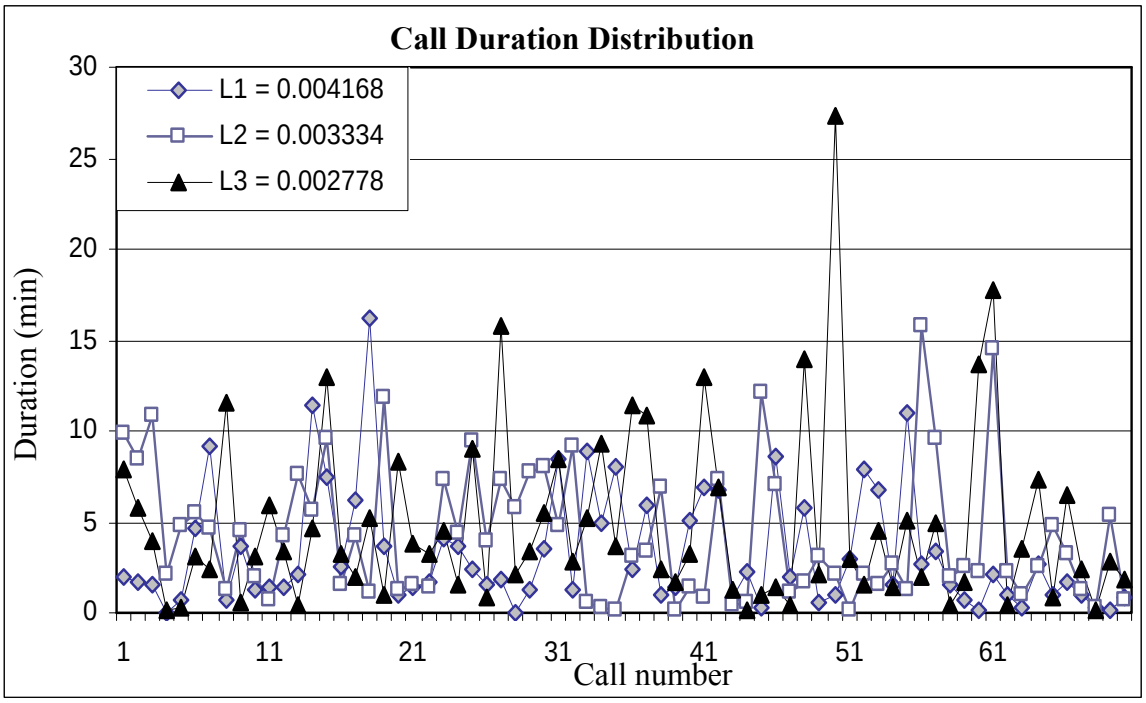


Figure 7. Call duration sampling over 5 hours for Voice Traffic.

$$F_X(x) = 1 - \left(\frac{k}{x}\right)^{\alpha} \qquad 0.4 \le \alpha \le 0.63, \quad k \le 21$$

$$f_X(x) = \alpha k^{\alpha} x^{-\alpha-1}$$

Due to the heavy tail resulting from Pareto distribution, WWW traffic contributes to a portion of the self-similarity of network traffic. We ran the experiment with two values of λ, λ_1=0.000001 and λ_2=0.000005. Since λ is the inverse of the Mean, smaller λ produces longer interarrival times. Figure-8(a) shows a sample of the interarrival times for one of the runs on Alquds node. The upper row of Figure-8(b) plots the arrival times in milliseconds of the first 30,000 fragments that arrived at the *Local* node. The second row in the figure plots the arrival times of the first 10,000 fragments on the remote *Al-Quds* node.Figure-8 shows the document size distribution of λ_1 and λ_2 which correspond to the shape parameters α_1=0.4 and α_2=0.55 respectively.

4.2.3 FTP Traffic Characteristics

Like www traffic, FTP traffic is modeled at two levels: call level and session level. The call level models the interarrival times of multiple sessions. The interarrival time between two FTP sessions is exponentially distributed. During a single FTP session multiple data items of varying sizes are transferred. NetSpec used a fixed distribution to model the number of items per session called (ftpNOfItems), and a fixed distribution to model items' sizes called (ftpItemSize).

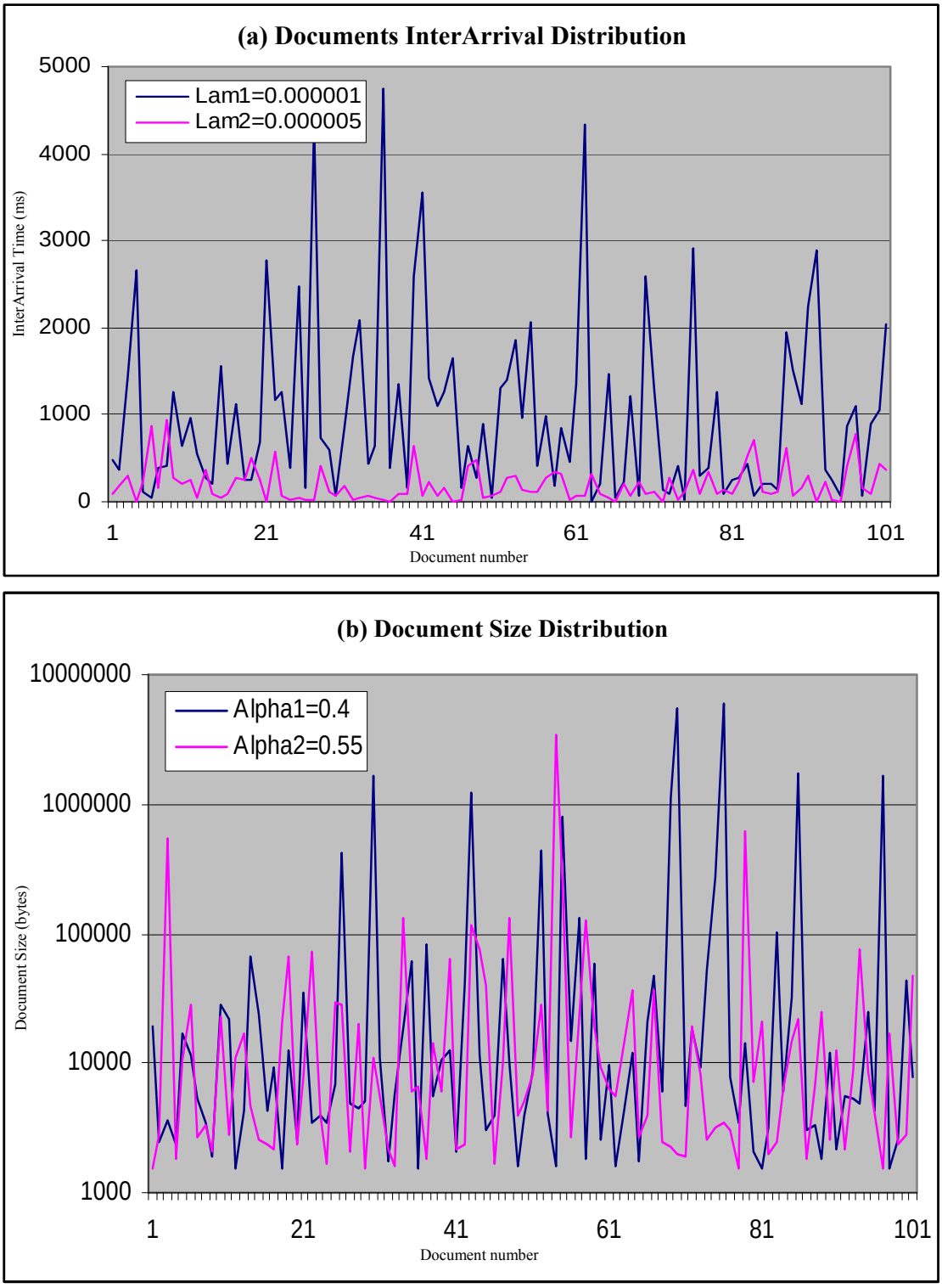


Figure 8. Document Inter-arrival and size distribution of the first 100 documents.

Table 5. Handoff Latencies (in ms) of the first five handoffs

Handoff	Local		Virginia		Texas	
	IPMN	MIP	IPMN	MIP	IPMN	MIP
1	106	12654	114	58669	202	51359
2	107	7124	106	24975	193	33187
3	111	1524	106	22672	195	29099
4	115	48945	111	77414	195	63523
5	109	1008	121	30772	200	41676
Avg	**110**	**14251**	**112**	**42900**	**197**	**43769**

4.3 Performance Analysis

Extensive experimentation has been done for both IPMN and MIP. The results were promising and IPMN had very good performance gains which are discussed shortly. The analysis has two parts. Handoff – the time from where the application losses connection of the network to the time we get back the connection. Traffic analysis—the delays in message arrival times at the receiver in our case the Mobile Node (MN). As the number of handoffs during the transmission increase MIP starts to react adversely to the applications.

4.3.1 Handoff Latency on Voice Traffic

One of the key features of the interactive scheme is short handoff latency. After running the experiment several times on the three nodes we have observed a big difference –up to two orders of magnitude—in handoff latency between IPMN and classic MIP. Table-5 shows the handoff latency of the first five handoffs on the three nodes for both MIP and IPMN runs. IPMN managed to perform handoff in 110 to 200 milliseconds on average while MIP needed between 14 to 44 seconds. This substantial reduction in handoff latency highlights the advantage of event-based protocols like IPMN over timer-based protocols like MIP. The former allows protocols in different layers to interact and pass events and new state information –like the new IP number in our case— to upper layers instantly. This enables peer protocols to respond immediately, therefore cutting down unnecessary overhead time. Timer-based protocols on the other hand usually use a timer-based periodic probing mechanism to discover state changes.

For example, in this particular implementation of MIP that we have tested, the foreign agent sends beacon signals (agent advertisements) to discover MN movement every 60 seconds! A best case scenario will happen if L2 handoff was performed right before the arrival of a beacon signal. Therefore, this process will take half of that time on average –i.e. 30 seconds. Adding to that communication and address registration overhead we can easily reach the 40 seconds average especially on the two remote nodes.

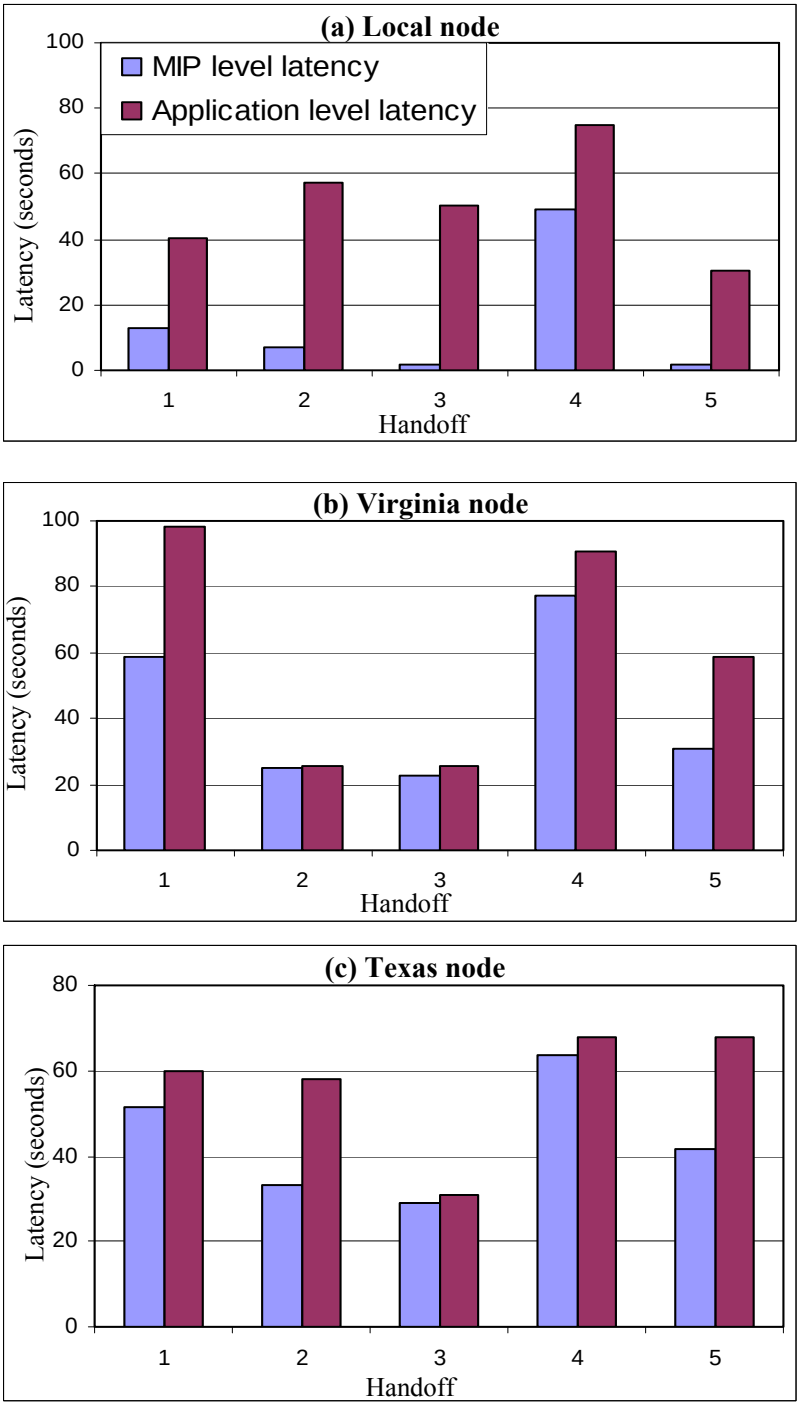
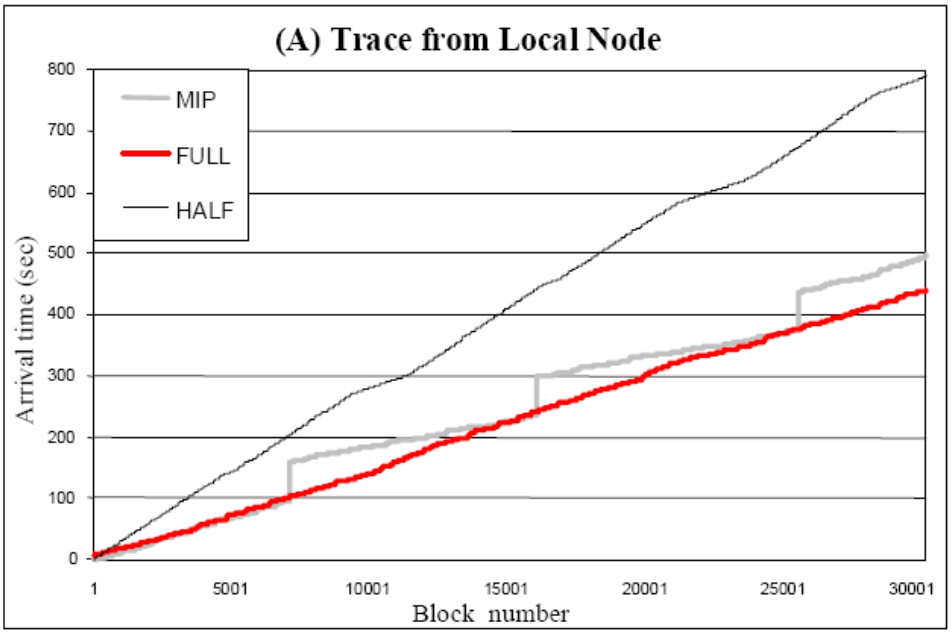


Figure 9. Handoff latencies on both MIP level and application level.

Actual latency in MIP was even longer; by the time MIP recovers and becomes ready to resume service, TCP has already timed out and will probably need even more time to discover MIP recovery and then resume communication on its own level –and at the application level as well. We have observed this behavior of MIP by also registering the time when application level communication resumed after each handoff was completed. Figure-9 demonstrates this property by comparing the handoff latencies of the first 5 handoffs at the application level and at the MIP level. For example, on the Texas node – figure-9 (c), when handoff 2 was performed, the application level suffered 58 seconds of service disruption, even though the MIP was responsible for 33 seconds delay only. While in some cases, TCP overhead was small –like handoffs 2 and 3 on the Virginia node which had less than 2 second of TCP overhead, in other cases –like handoffs 2, 3, and 5 on the Local node, TCP added up to 50 seconds! The application level latencies highlight the fact that even if the handoff is performed by MIP, higher layers like TCP, RTP and application protocols like HTTP, FTP which uses their own timer mechanisms will falsely identify handoff as network congestion and reduce the data transfer.



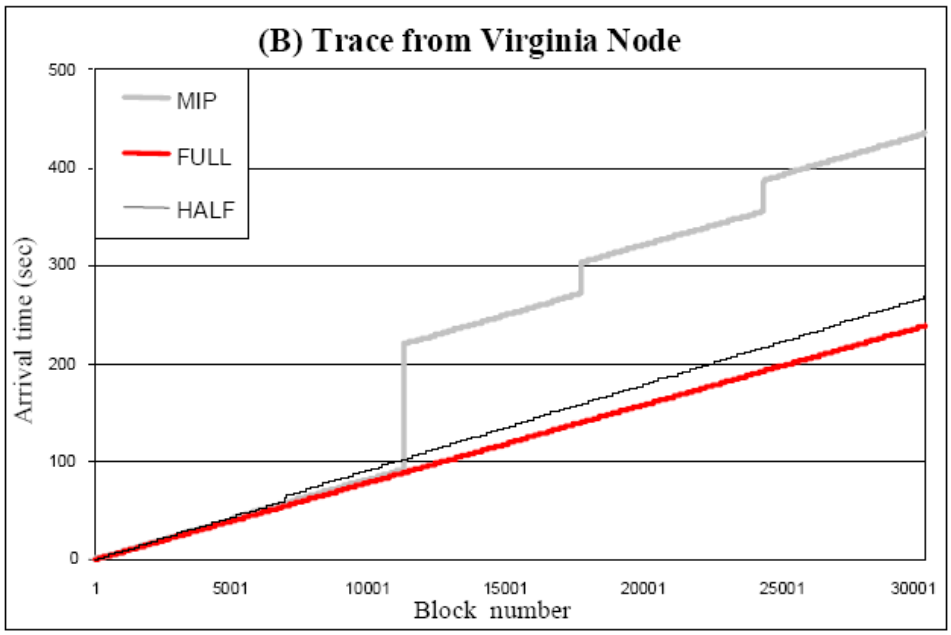


Figure 10. The arrival time of each 144-bytes block (talk burst) at the MN from (a) Texas node and (b) Virginia node.

4.3.2 Voice Stream Arrival Delay

Now, we show application level performance by observing stream arrival delay. At the MN, we kept a log file to register the arrival time of each 144-bytes block (talk burst) in the voice stream. Figure-10 plots the arrival times of the first 20,000 blocks at the MN from the two remote nodes: Local and Virginia.

IPMN dramatically outperformed MIP on two levels: Firstly, in general, most blocks were delivered faster with IPMN due to shorter triangulation-free path that they had to travel to reach the MN as well as to smaller overhead. Secondly, IPMN plots were much smoother than MIP since the latter suffered from longer disruption of TCP service due to longer handoff delays. This can be observed in the Virginia plots. After each handoff event, we see the impact of TCP's slow start behavior on the plot. These step jumps and the impact of TCP dynamics created jitter on the voice stream –as we show in the next section. One last observation is the impact of connection speed. The CH in Virginia was transmitting on a 350 kbps DSL connection; MIP needed about 430 seconds to transmit all 20,000 blocks while IPMN-Full managed to transmit in 230 seconds while IPMN-Half transmitted the messages in 260 seconds.–a 2.2 minutes difference.

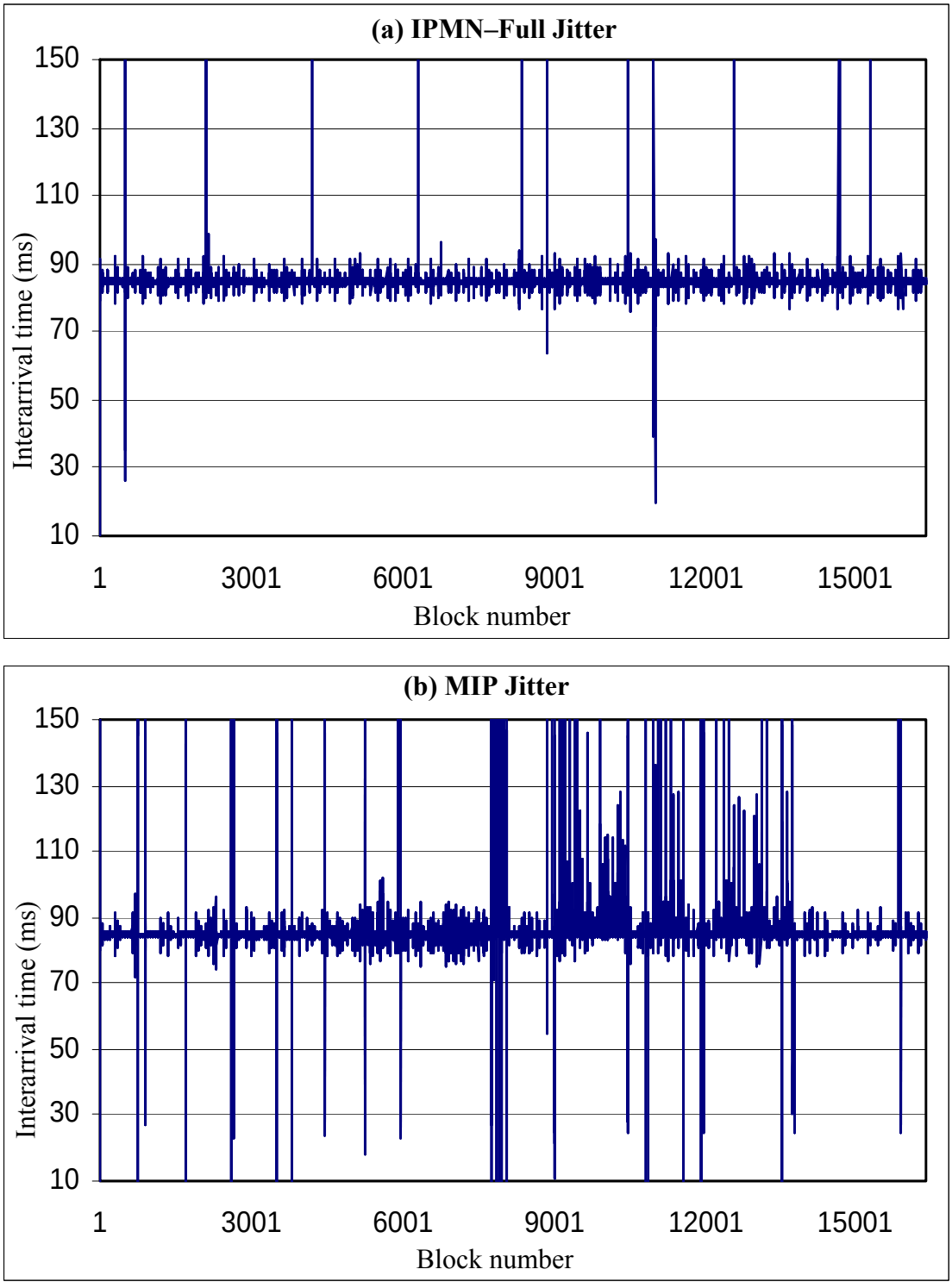


Figure 11. Block interarrival times at the MN.

Table 6. Handoff Latency of mip and ipmn on local and Al-Quds in millisconds.

Handoff Latency (ms) / Local				
	λ_1		**λ_2**	
Handoff	IPMN	MIP	IPMN	MIP
1	108	28687	87	56939
2	90	53636	90	43696
3	88	74728	97	51294
4	92	54417	86	28632
average	**94.5**	**52867**	**90**	**45140.25**
Handoff Latency (ms) / Al-Quds				
	λ_1		**λ_2**	
Handoff	IPMN	MIP	IPMN	MIP
1	92	39426	90	30109
2	88	97468	92	42197
3	87	37481	92	51342
4	95	57487	90	39894
average	**90.5**	**57965.5**	**91**	**40885.5**

4.3.3 Jitter on Voice Stream

Figure-11 plots the interarrival times of the first 16000 blocks arriving at the MN from the Texas node, (a) on IPMN, and (b) on MIP. On IPMN almost all blocks were delivered at (75 to 90) ms apart, except (mainly) those that faced a handoff –only 22 blocks were delayed for more than 100 ms. .

In Figure-11 we show a maximum of 150 ms on the y-axis to be able to see the mainstream case. Average interarrival time for all blocks on IPMN was 85.57 ms. On MIP the situation is different; about 177 blocks in the stream faced more than 100 ms

interarrival –10 of these blocks faced more than 8000 ms delay—and average interarrival time for all blocks was 129 ms.

4.3.4 Handoff on WWW Traffic

One of the most important features of our interactive protocol is its short handoff latency. After running the experiment several times on both nodes we have observed a big difference –up to three orders of magnitude—in handoff latency between IPMN and classic MIP. Table 6 shows the handoff latency of the first four handoffs in (ms) of the two protocols for the two values of λ. IPMN managed to perform handoff in 90 milliseconds on average while MIP needed almost one minute. This substantial reduction in handoff latency highlights the advantage of event-based protocols like IPMN over timer-based protocols like MIP. The former allows protocols to interact and pass events and new state information –like the new IP number in our case—to upper layers instantly. This enables peer protocols to respond immediately cutting down overhead time. Timer-based protocols on the other hand usually use a timer-based periodic probing mechanism to discover state changes. For example, in this particular implementation of MIP that we have tested, the MN sends beacon signals to discover new FA assignments every 60 seconds!

A best case scenario will happen if L2 handoff was performed right before periodic beacon signal arrives. Therefore, this process will take half of that time on average –i.e. 30 seconds.Adding to that communication and address registration overhead

we can easily reach the 40 seconds average especially on the two remote nodes. Actual latency in MIP was even longer; by the time MIP recovers and becomes ready to resume service, TCP has already timed out and will probably need even more time to discover MIP recovery and then resume communication on its own level –and at the application level as well. We have observed this behavior of MIP by also registering the time when application level communication resumed after each handoff was completed.

4.3.5 WWW Message Arrival Times

For both WWW and FTP evaluation we kept two log files at the MN: one that registered the arrival time of each document received and one that regarded the incoming stream as a whole sequence and registered the arrival time of each 1 KB-fragments. Figure-12 plots the arrival time of each 1 KB fragment at the MN. The figure shows four plots for each node / λ pair. In each plot we show MIP and IPMN runs with their associated handoff events as vertical lines on the x-axis. As we can easily observe, the IPMN dramatically outperformed MIP –especially with the remote node—on two levels: Firstly, in general, most fragments were delivered faster with IPMN due to shorter triangulation-free path that they had to travel to reach the MN –this was not a relevant factor with the Local node plots since all machines were in the same room! Secondly, IPMN plots were much smoother than MIP since the latter suffered from longer disruption of TCP service due to long handoff delays.

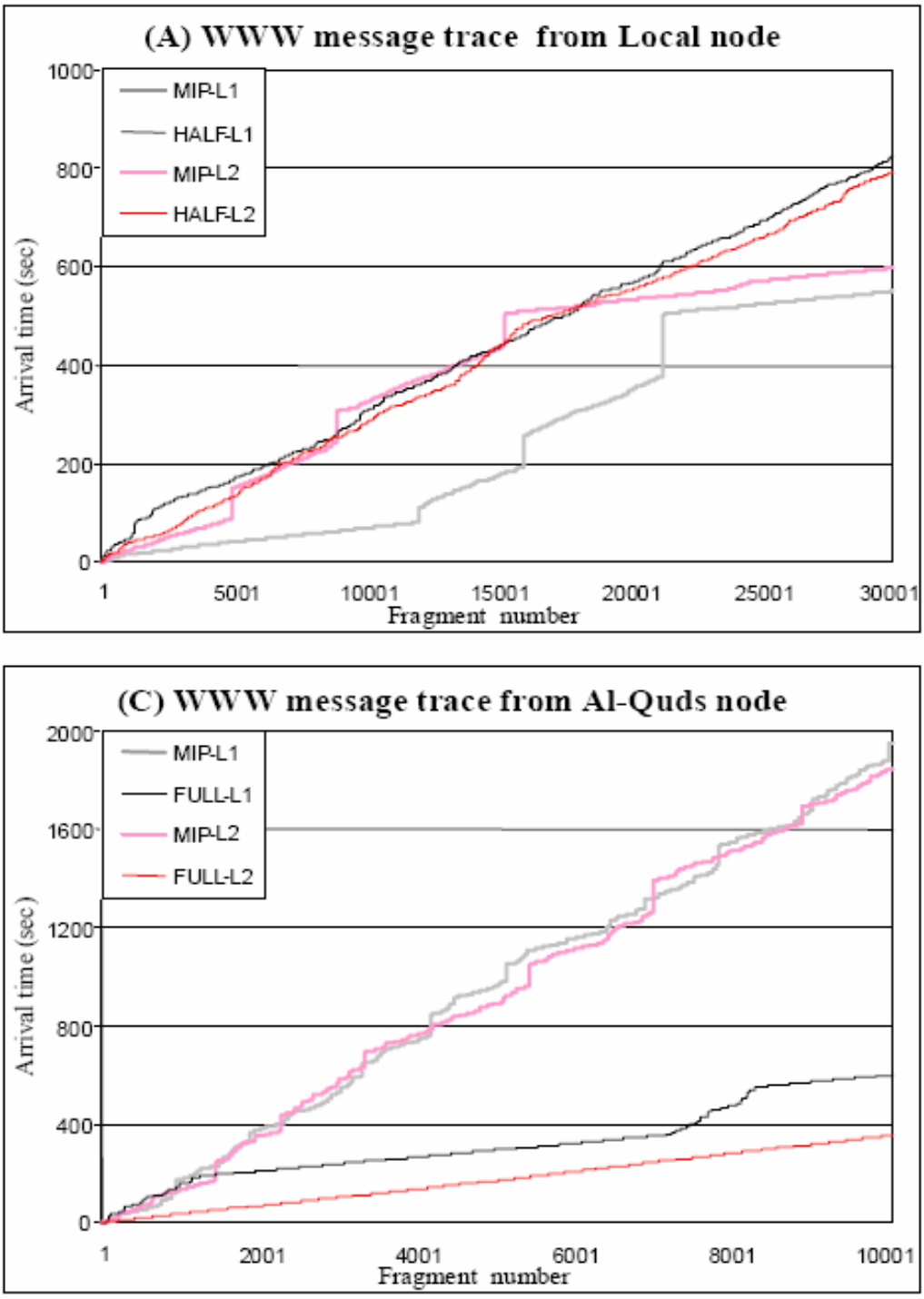
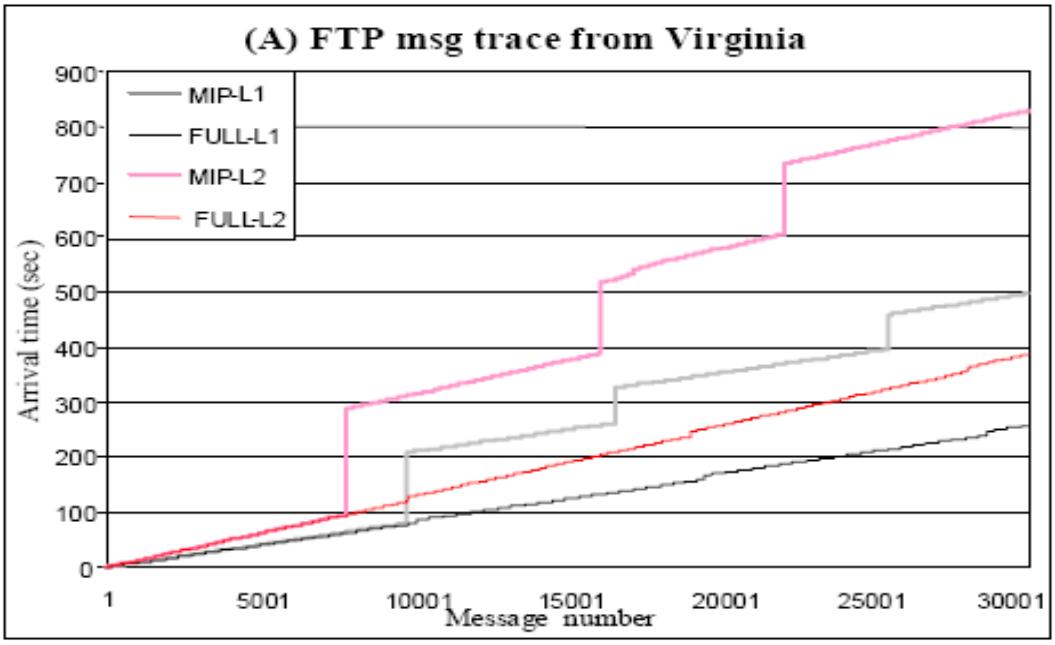


Figure 12. WWW Message Arrical Times at the MN



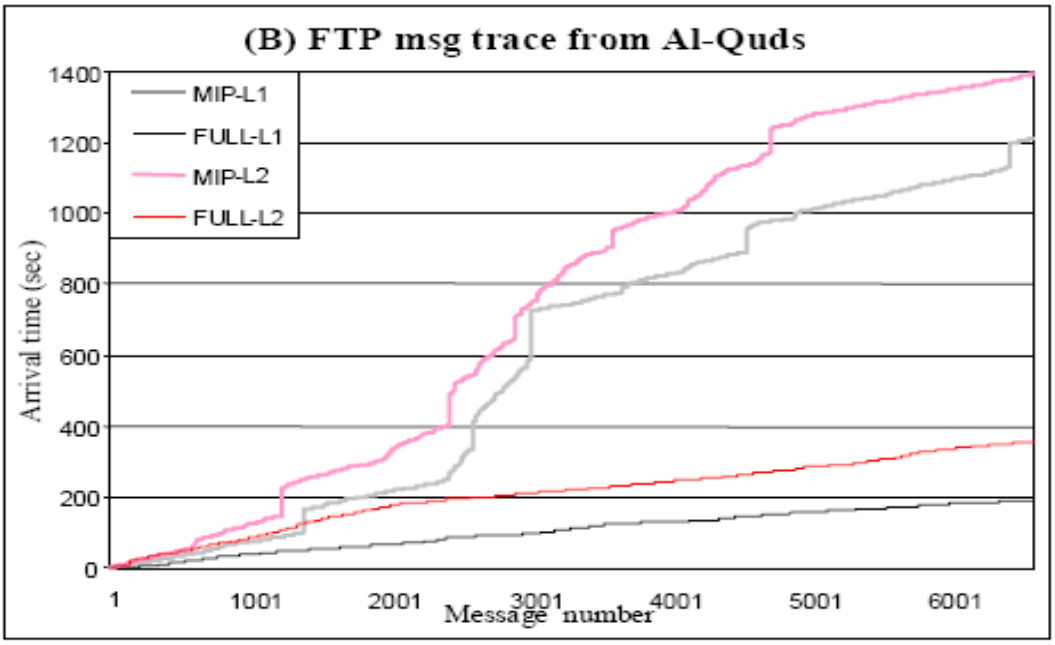


Figure 13. FTP message arrival times at the MN

In each plot there are four runs: two for MIP and two for IPMN (all plots shows IPMN-Full). For each mode we generated WWW traffic using the two values of interarrival parameter of $\lambda 1 = 0.000001$ and $\lambda 2 = 0.000005$ —shown in the figure as L1 and L2. Again, on the Local plot (A), MIP seems to outperform IPMN especially with L1 traffic. On the Al-Quds plot (C), MIP's performance degrades with each handoff. After each handoff event, we see the impact of TCP's slow start behavior on the plot. This can be specifically important if we were dealing with voice or video streams which prefer lower jitter.

4.3.6 FTP Message Arrival Times

We show FTP traffic trace in Figure-13. Plot (A) of the figure shows the arrival times in seconds of the first 30,000 1 KB fragments that arrived at the mobile node from the Virginia node, plot (B) traces the first 6,500 fragments from Al-Quds node. As in the WWW case, there are four runs in each plot: two for MIP and two for IPMN with the same interarrival parameters L1 and L2 that were used before. These plots further confirm the obvious advantage of IPMN over MIP as we saw with Voice and WWW traffic.

4.4 Conclusion

In all the experimental scenarios IPMN performed better. Only one graph for IPMN-Half is compared as we thought IPMN-Full was more appropriate and truly cross-layer interactive. Its interesting to notice that IPMN-Full performed better than IPMN-Half in Figure-10 even though there is lot of interactivity. The trade-off between signaling overhead and the performance gains can clearly be observed. With a minimal signaling overhead huge performance gains can be achieved.

Chapter 5

IPMN 3-Layer Modeling

In the previous chapter all the experiments were performed assuming that a LL signal has been received. The experimental results were encouraging and so in this chapter we will present a 3-Layer modified design. Model the delay characteristics of signal handlers for all the events. MIP is also modeled for comparing performances. Since 802.11 is the most widely used wireless LAN we have modeled the 3-layer design considering 802.11 as the Data Link Layer. We have identified the handoff delay from each layer and identified a technique in the application to minimize this delay.

5.1 IPMN 3-Layer Architecture

In a wireless scenario L2 handoff is initiated by the MN when the Signal/Noise ratio and Signal strength of the current Access Point (AP) falls behind a certain threshold. To have a better understanding of the architecture let us first explain briefly the two wireless scenarios existing and how our architecture robustly handles both while performing a L2 handoff.

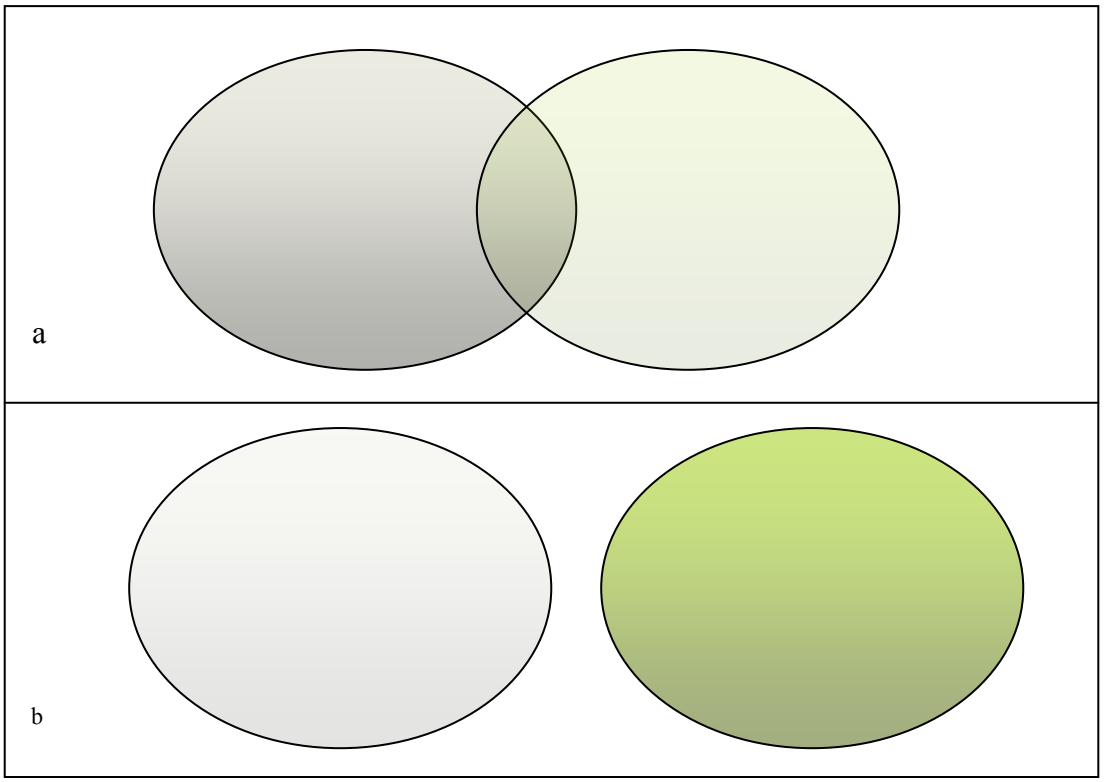


Figure 14. Cell Boundaries a) Overlapping b) Non-Overlapping

When an MN receives service from more than one access point at any point of time then their areas of coverage are overlapping, which generally happens at the boundaries of the cells and this is called *Overlapping cell boundaries.* Architectural view of overlapping cell boundaries is shown in Figure-14(a) and will have better performance. Applications probing the link layer will get useful information about the next possible AP that would serve MN after the handoff. This information aids faster L3 handoffs which would be discussed in detail later in the section.

At times MN may encounter temporary layer2 disconnections, encountered due to disjoint coverage areas. This type of coverage is called *Non-Overlapping cell boundaries*

as shown in Figure-14(b). Probing the link layer will not yield any useful information. So the Higher Layer has to wait until the LL handoff is performed. Architecture

A vertical handoff will include LL handoff and network handoff without disrupting the existing connections. In this section we will formalize the handoff procedure for both overlapping and non-ovelapping scenarios. The IPMN described is IPMN-Full as this showed more performance gains in the previous chapter and more protocol-interactive. This chapter and all the later chapters will address IPMN-Full as IPMN.

. L2 handoff consists of three steps a) probing, b) authentication and c) re-association explained in detail in the next section. The handoff procedure is started when the application receives a L2 trigger notifying an impending handoff and saves the cell boundary condition for application to use this information if needed. To detect an impending handoff anything from a simple technique like rate of depletion in the signal strengths to a very complex GPS based movement tracking techniques can be employed. Application upon receiving the signal shall invoke a transient-ware module forcing TCP layer to advertise a zero window temporarily interrupting the data transmission [6]. This approach, though interrupts the data transmission does not tear down the connection. It will also probe L2 to get the boundary condition of the AP's as this will be a deciding factor for choosing one of two handoff algorithm – IPMN Fast Handoff or IPMN Normal Handoff. From this point the notifications for subscribed events will be handled through different transient-ware based on the information available to the application from previously handled signal (overlapping or non-overlapping cell boundary condition). Each of these cases is discussed below separately

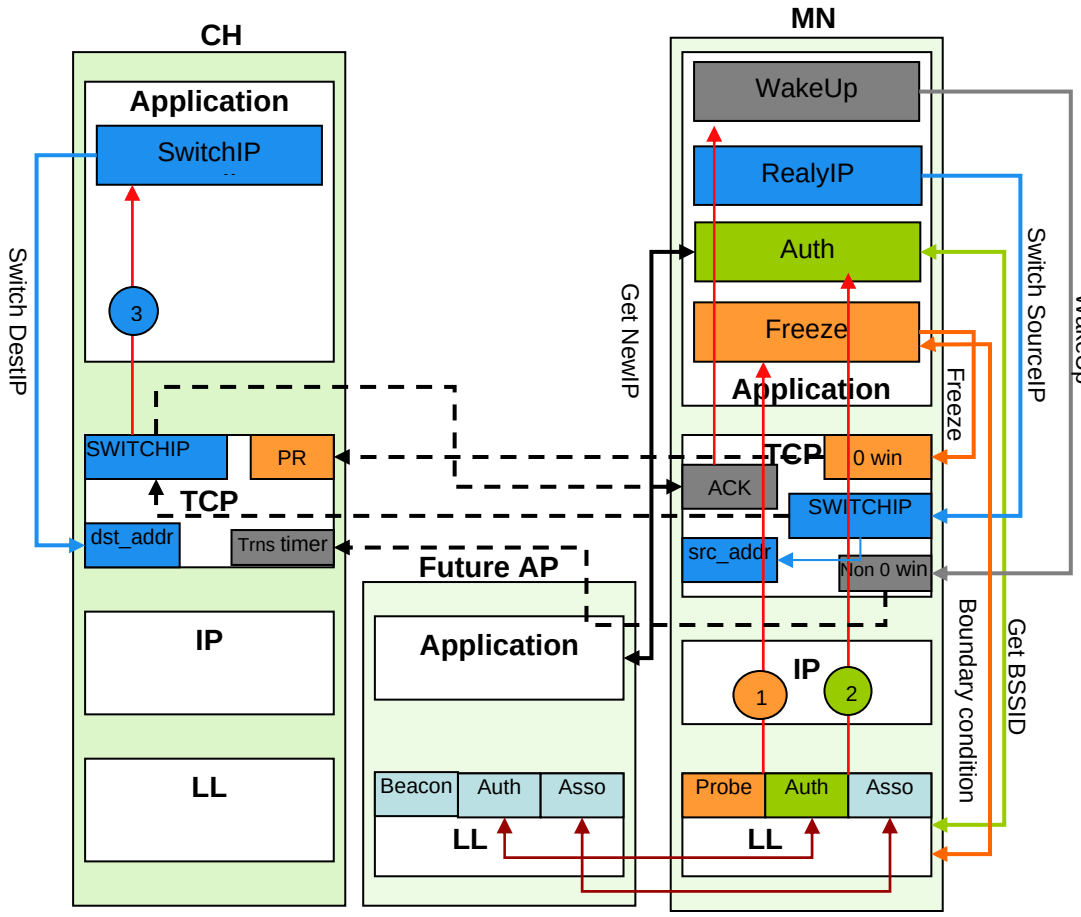


Figure 15. IPMN-Fast Handoff Architecture and event sequencing for overlapping cell boundary condition.

- **IPMN Fast Handoff**

Completion of L2 authentication process will trigger an event notification and save the information of the PAP. Application upon receiving this event will invoke a transient-ware which will probe L2 and get PAP information. This information is used to register the MN with PAP and get an IP address in parallel with L2 association process. Once the IP address is assigned, the same is propagated to the other end-point using TCP. After getting the address the same connection is renewed by manipulating the TCP/IP stack at

both the end points. The application unfreezes the connection after manipulating the TCP stack. Since IP address is attained almost in parallel with L2 handoff we reduce the handoff latency by a great extent. Freezing the connection during handoff avoids congestion control algorithms and increasing the performance of the transport layer.

- **IPMN Normal Handoff**

This algorithm is triggered in non-overlapping cell boundary cases. Here the application will be waiting for a different event to proceed further. Normal Handoff will let L2 complete the handoff process and wait for the IP layer to detect the change of IP address and trigger the change to the application. The event handler for this event will propagate the IP address to the other endpoint and connection is renewed by manipulating the TCP stack followed by unfreezing the connection to resume the data transmission.

During the handoff process the application will not be able to handle any new connections. This temporary failure of new connections will remain until all the existing connections are restored and operational. But with minimal handoff time the number of connection attempts to a moving node would be negligible and acceptable.

As shown in Table-6 five events are trapped from three layers and corresponding 5 transientware processes are invoked in order to provide mobility support. Figure-15 gives the architecture and event sequencing for overlapping cell boundary condition where the 802.11 Access Points (APs) coverage areas overlap. This is called IPMN Fast Handoff. Figure-14 gives the architecture of overlapping cell boundary condition. Probing in L2 is initiated when the Signal-to-Noise Ratio (SNR) of the current AP falls below a certain

Table 7. IPMN-3 Layer Events and Sequence Handling

Layer	Events
LINK LAYER	Successful Probing
	Successful Authentication
IP LAYER	IP Address Change
TCP LAYER	Recv TCP_OPTION = *SWITCHIP*
	Ack for TCP_OPTION = *SWITCHIP*

threshold. At the MN, the subscribing application (L7) is notified of an impending handoff after probing (event 1) is completed.

When the event is received at the L7, a Transientware module (Freeze Handler) is activated immediately; this module makes a system call *Freeze* which lets TCP advertise a zero window temporarily interrupting the data transmission. Freezing the connection during handoff avoids TCP congestion control algorithms by invoking the persist timer. L7 probes L2 to get information about the cell boundary condition by invoking the *Boundary Condition* system call and performs handoff accordingly. In overlapping case successful L2 authentication (event 2) triggers L7 after the authentication process is complete and saves the Access Point's (Base Station) information locally. Auth Handler probes L2 again to identify the cell boundary condition (if the probe yields at least one AP then the cell boundaries are overlapping) using the *Get BSSID* system call. Here we assume that the current connection is still valid and extract information about the Future Access Point (FAP).

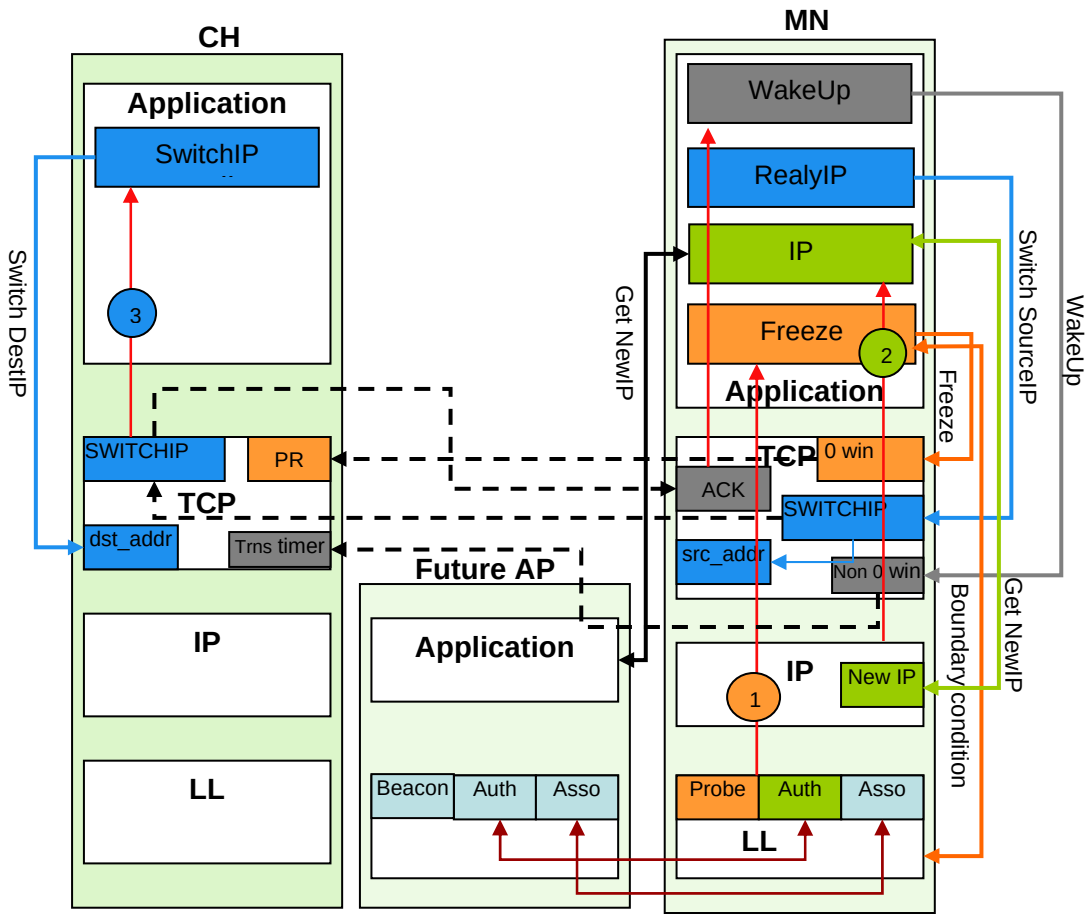


Figure 16. IPMN-Normal Handoff Architecture and event sequencing for non-overlapping cell boundary condition.

MN uses this information to register with Future AP and attain an IP address (L3 handoff) in parallel with L2 handoff. Depending on the various security features that L7 might incorporate, the decision of initiating RelayIP Handler is left to L7. RelayIP Handler upon initiation transmits the already attained IP address to the CH through a system call *RelayIP* and also manipulates the '*SourceIP*' field in the TCP substrate. A special TCP segment with option = SWITCH_IP' is sent to the CH. On reception of this special segment at CH (event 3), SwitchIP Handler is activated, which manipulates the

'*DestinationIP*' field in the TCP substrate with the newly received IP address. MN on reception of an 'ACK' for 'SWITCH_IP' segment (event 4), invokes Wakeup Handler allowing MN to resume the communication by calling the *UnFreeze* system call. *UnFreeze* advertises a non-zero window to the CH.

In non-overlapping case IPMN Normal Hanoff algorithm is invoked. The MN after probing the cell boundary condition the Freeze Handler waits until L2 handoff is completed and a DHCP IP address is assigned. IP Handler is invoked when L7 receives event2 which probes IP layer to get the IP address. The same process as explained in the previous case follows when RelayIP Handler is initiated. Tough the handoff process is comparable to that of MIP, both the application and TCP adapt to the temporary limitation of the lower layer (L2). The event triggering mechanism avoids any time lapse between the actual occurrence of the event and the action needed to be performed. This still outweighs all timer based mechanisms by a huge factor.

5.2 Formulation of Handoff Latency for IPMN

In this section we model the latency of our handoff scheme. To evaluate our model we also model the handoff latency of MIP. The handoff process is thus divided into Link Layer handoff and Higher Layer handoff latencies. Each of these latencies is computed separately and combined later depending on the cell boundary scenario. In IPMN cell boundary condition plays an important role as the total handoff latency will be reduced by performing LL and higher layer handoffs in parallel to each other. MIP has to wait till

the LL handoff is completed and hence induces greater delay than that of MIP. We will discuss individual latencies of each of the layer starting with 802.11 LL.

5.2.1 Link Layer Handoff Latency (802.11)

Link layer (802.11) handoff can be classified into 3 categories according to [14, 18, 23] a) Probing, b) Authentication and c) Association/Re-Association. Each of these categories would constitute a delay parameter to L2 handoff latency.

- **Probing**

Probing is done by a MN to determine the availability of a wireless network and perform a L2 handoff. A MN sends a probe request and waits for a reply governed by two timers. *MinChanneltime*(T_e) – minimum time that a MN needs to wait after sending a probe request. If there is no activity in the channel for *MinChanneltime* then the channel is considered empty. *MaxChanneltime* (T_u) – time for MN to get a response from a used channel. In [14] Velayos and Karlsson et al., tried to estimate the various 802.11 handoff delays and tried to optimize the handoff latency. If u and e are the number of used and empty channels respectively, then the total time to probe the network called the Scan Delay S_d can be computed as in Eq 1(a).

$$S_d = u * T_u + e * T_e \quad \text{.....1(a)}$$

Furthermore both T_u *and* T_e send probe requests twice so as to minimize the possibility of these probe packets being lost. Values of T_u *and* T_e are directly dependent of the transmission delay T_d governed by the load of the channel.

$$T_u = 2 * T_d + MaxChannelTime$$
$$T_e = 2 * T_d + MinChannelTime \quad \text{.....1(b)}$$

The transmission delay T_d can bmodeled as a function (Eq 1(b)) as the contention for the channel and retransmission (both being the critical parts of T_d) always follow the randomized binary exponential backoff. Equations 1(a), 1(b) and 1(c) will give the total probing delay.

$$T_d = \int_0^{\infty} f_t(t)\,dt \quad \text{.....1(c)}$$

The transmission delay T_d can be modeled as an integral function as there is no control over the transmission medium. We assume that all transmission delays will follow the same integral function. The function itself may vary as each medium has its own delay characteristics. From here on the Transmission delay will use this integral function. The change in channel will be indicated by a different function that will relate to channel characteristics. We will not go into the details of the functions as it is out of scope of our work.

- **Authentication**

This is the process that validates whether the MN can use the services of an AP. If a mutually acceptable level of authentication has not been established between an AP and MN then, association/re-association would not be established. The MN after probing the channels and obtaining the list of Access Points in range tries to prioritize them based on numerous criteria. The MN will send an authentication request to the first entry in the list of prioritized APs and waits for an authentication reply.

$$A_d = \sum_{i=1}^{n} (2 * T_d + P_{Ta}) Where \ \ 1 \le n \le u \qquad 1(d)$$

Equation 1(d) gives the total time taken to authenticate the AP A_d as a function of time of transmission in that channel T_d and actual authentication time P_{Ta} and n is the number of successful channels scanned and returned by probing. i gives the number of attempts made before the successful authentication. If the reply indicates a successful authentication then re-association procedure is started. If there is an unsuccessful authentication then the same procedure is repeated with the next entry until successful authentication.

- **Re-Association**

After successful authentication the MN's state information is transferred from the old AP to new AP. Re-association is helpful in knowing the current attachment point of the MN as it is moving from AP to AP. Re-Association delay (RA_d) is the request/response sequence *RTT* plus the time to exchange state information between the old and the new AP(P_{Tr}).

$$RA_d = 2 * T_d + P_{Tr} \qquad .....1(e)$$

5.2.2 Handoff Latency in Higher Layers

There are 5 system calls that are used to invoke handlers. The latencies of these system calls range from 10µs to 100µs. Each system call will have latency Syc_d given by a random variable X between the aforementioned intervals.

$$Syc_d = X\mu s \quad where \ 10 \le X \le 100 \qquad 2(a)$$

In the same way the triggering latencies are between 5 and 20µs which is represented as Tg_d and governed by a random variable Y ranging form 5 to 20.

$$Tg_d = Y\mu s \quad where \ 5 \le Y \le 20 \qquad .....2(b)$$

MN would directly contact the Future AP to pro-actively register itself and get an IP address. This delay termed as proactive registration delay(PR_d) constitutes of the RTT

Table 8: Handoff delay for Overlapping and Non-Overlapping cell boundaries

Overlapping cell boundaries		Non-overlapping cell boundaries	
Link Layer	Higher Layers	Link Layer	Higher Layers
probing		probing	
Handoff trigger		Handoff trigger	
Authentication	Freeze system call	Authentication	Freeze system call
	cellBoundary system call		cellBoundary system call
Auth trigger		Auth trigger	
Reassosiation	NewBSSID system call	Reassosiation	
	GetNewIP system call		NewIP trigger (MN)
	RealyIP system call		ProbeNewIP system call
	NewIP trigger (CH)		RealyIP system call
	switchIP system call		NewIP trigger (CH)
	Ack TCP OPT signal		switchIP system call
	Unfreeze system call		Ack TCP OPT signal
			Unfreeze system call

between the MN and the PAP through the present AP $T1_d$.

$$PR_d = 2 * T1_d \quad \text{.....2(c)}$$

The transmission delay $T1_d$ can be modeled as an integral function as there is no control over the transmission medium.

We assume that all transmission delays will follow the same integral function. The function itself may vary as each medium has its own delay characteristics. We will not go into the details of the functions as it is out of scope of our work.

$$T1_d = \int_0^{\infty} f1_t(t)dt \quad \text{.....2(d)}$$

We have observed roundtrip times for various demographic distances up to 400 miles and observed the fact that it is practically impossible to have access points whose demographic distance is more than 400 miles. This implies that RTT latency would fall between 100µs to 10ms depending on the distance. Table 8 gives the handoff delay in both overlapping and non-overlapping cases. The shaded region gives the possible overlap L2 and L3 handoff.

From Eqs 1(d), 1(e), 2(a) and 2(b) we have the delay of overlapping cell boundary given by Equation (3).

$$T_0 = S_d + Tg_d + \max(A_d, Syc_d) + 2 * Tg_d + \max(RA_d,\ 3 * Syc_d + PR_d + Ep_d) + Syc_d \quad \text{…3}$$

The max function used can be referred to Table 2 as those processes that are done in parallel in both L2 and TCP layers. Where *max(a1, a2)* gives the maximum value of *a1* and *a2*, and represents the overlapping latencies. EP_d is the Round Trip Time between end points of the connection governed by Relay Handler.

In Non-Overlapping cell boundaries the probe delay S_{dno} prolongs until it hears from at least one AP, also taking into account the time to traverse between the coverage areas. From Eqs 1(d), 1(e), 1(f), 2(a) and 2(b) we have the total handoff delay for non-overlapping cells T_{no} is as given in Equation (4).

$$T_{n0} = S_d + Tg_d + \max\left(A_d, Syc_d\right) + 2*Tg_d + RA_d,\ 4*Syc_d + PR_d + Ep_d + Syc_d \quad \text{.....4}$$

The formulation of handoff delay is also well depicted in the timing diagram (Figure 6) for both overlapping and non-overlapping cell boundary scenarios. Figure 6(a) shows the overlapping case and figure 6(b) gives the non-overlapping scenario and clearly explains why we need a *max* function while formulating the handoffs. Clearly the ovelapping case will perform better as most of the L2 and L3 handoff happen simultaneously. Event E1 (impending handoff) happens at the same time in both the cases. The difference is observed in the ocuurance of the subsequent event. Event E2 (authentication) occurs only in overlapping case. E2 will trigger E3 on the other end-point. The corresponding communications in overlapping case, though appear to be complex yeilds complementing results. It is interesting to know that E2 and E3 will invoke their respective handlers which does similar jobs. In non-overlapping case the occurance of E2 is not of importance to the application. Rather it waits for E5 to occur (notification of change in IP address). This will send TCP_OPTION which triggers E3 at the other end-point. Table 2 shows the order of events and the correspondng handlers and the order of execution. L2 and L3 events that can be executed simultaneously are shown in the shaded region of the table. In overlapping and non-ovelapping cell boundary cases the time of simultaneous handoff process varies substantially. In overlapping cell boundary the connection still

remains valid for further communication and hence can acheive optimized results due to extended conncetion. Since the architecture is L7 based, re-synchronization of the channel to the original wireless network characteristics can be easily acheived. The values of these variables can be stored in L7 and can be re-associated easily. The process of scanning in L2 starts when the signal to noise ratio falls below a certain threshold. In this case all the commincation is halted and the handoff procedure is started. To obtain optimal results the scanning has to be interleaved intelligently with the data transfer. For this to happen the scanning has to start even before the SNR falls below a threshold.

5.3 Formulation of Handoff Latency for Mobile IP

Mobile IP [1, 15] handles mobility in the network layer transparent to higher layers. Mobile IP uses its own timer based movement detection algorithms to detect mobility maintaining layer separation form Link Layer. Mobile IP introduces mobility agents in the form of Home Agents (HA) and foreign Agents (FA). Every agent broadcasts an Agent Advertisement (AA) specific to a particular agent along with the life time of the advertisement. If the AA is not received again before the AA life time has expired then the MN confirms that it has moved out of its current serving area. When MN receives an AA with different specifics it is confirmed that it entered a new service area and the handoff procedure is started. Movement detection has a lot of latency and is timer dependent. Frequent AA result in consuming lot of network bandwidth, while non frequent AA delays movement detection and have an adverse effect on higher layers.

Registration and tunneling follows movement detection which is achieved through inter-agent communication. This is done by IAPP between the out going and incoming agents. Typically the delays for registration and tunneling are governed by the Transmission delays.

Since Mobile IP operates independently of Link Layer, it cannot detect the handoff until L2 has completed its handoff procedure. Link Layer delay L_d will be addition of all the individual delays of L2.

$$L_d = S_d + A_d + RA_d \quad \text{.....5}$$

Movement detection is based on agent advertisement's life time. The frequency of these agent advertisements will directly impact the handoff latency. Firstly MN needs to detect that it has moved out of its present agent's service area. This movement detection depends on the AA life time x_1 of the present agent and the transmission delay T_d. After knowing that it is no more being serviced by the old agent it waits for the new Agent advertisements. In a worst case scenario the MN has to wait until AA life time along with the transmission delay T_d of the new agent. For non-overlapping cell boundaries the movement detection will also have the time taken to move between cells unserviced (N_d) in addition to all the above stated latencies. So Movement Detection Latency M_d will be

$$M_d = x_1 + 2*T_d + x_2 + N_d \quad \text{.....6}$$

where T_d is the same as that of Link Layer and N_d depends on the movement parameters of the MN like speed orientation of MN's movement. In this modeling we tried to keep this a constant of 1 second for performance analysis.

MN gets a temporary care-of-address (COA) when it registers with the FA. The COA needs to be updated at HA so that the packets arriving at HA can be rerouted to MN's current COA. Registration delay R_d would be the transmission delay between MN and HA through FA T_{d2} given by equations (7) and (8).

$$T_{d2} = \int_0^{\infty} f2_t(t)dt \quad \text{.....7}$$

$$R_d = 2 * T_{d2} \quad \text{.....8}$$

Tunneling is a mechanism by which the packets can be re-routed through a new agent using encapsulation. The HA intercepts the packets destined to the MN and tunnels them to the agent (FA) whose COA is registered with the HA, encapsulates the packets and route them to the FA. FA then decapsulates the packets and sends it to the MN. Tunneling delay Tu_d is equal to the inter-agent transmission delay IT_d as given in equation (9)

$$Tu_d = 2 * IT_d \quad \text{...9}$$

And IT_d will be as shown in equation (10)

$$IT_d = \int_0^{\infty} fw_t(t)dt \quad \text{...10}$$

More over once the tunnel is established this tunneling delay will remain for each packet until the whole transmission process is over or the MN returns to its HA where tunneling is not required. The total handoff delay due to mobile IP would be H_d given by equation (11).

$$H_d = L_d + M_d + R_d + IT_d \quad \text{.....11}$$

5.4 Conclusion

This thesis reinforces the fact that solutions to handoff cannot be addressed in a single layer. A vertical approach, considering all the discrepancies in each layer and addressing them will result in better performance gains. A LL handoff procedure is explained, handoff delay is calculated. IPMN handoff delay is calculated based on the cell boundary condition.

Chapter 6

IPMN 3-Layer Modeling

IPMN 3-layer scheme is realized and handoff latency for IPMN and MIP are modeled in the previous chapter. In this chapter the handoff latencies are experimentally analyzed. First only the handoff is compared in ideal conditions without any cross traffic. Then Message arrival times for ftp, voice, www and time critical traffic are analyzed in cross traffic scenarios to simulate network congestion and BER to simulate a error prone 802.11 wireless access LAN. The huge performance gains observed in IPMN are attributed the cross-layer design and event based architecture.

6.1 Handoff Latency Comparison between IPMN and MIP

We have simulated two versions of MIP- MIP1 with AA lifetime of 100ms and MIP2 with AA lifetime of 1s. The reason for this is to see whether MIP can outperform IPMN. MIP2 is the recommended practice for MIP as MIP was not designed to handle micro-mobility.

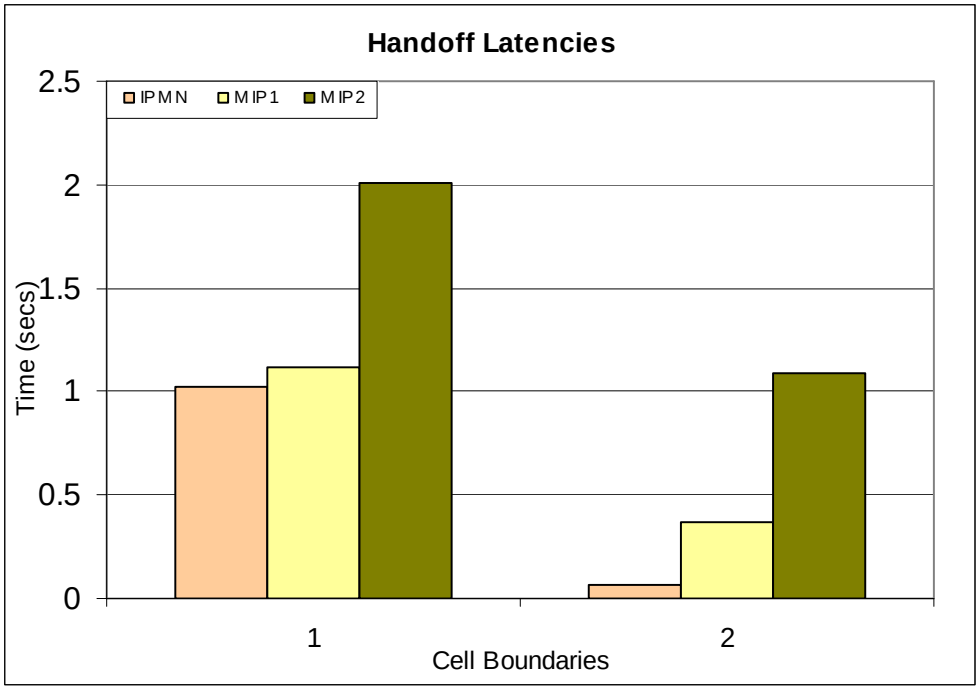


Figure 17. Handoff Latencies of Mobile IP and IPMN. MIP1 has a AA lifetime of 100ms and MIP2 1sec.

Figure-17 presents handoff delays of MIP and IPMN for overlapping and non-overlapping cell boundary conditions. We simulated the handoff for 100 times and the figures shown in the graph are an average of these 100 runs. The right side of the graph shows the overlapping cell boundary condition while the left side shows the non-overlapping case.

Let us first compare the huge difference observed between both the scenarios. In non-overlapping case the physical connection breaks down and there can be no communication until the physical medium is available again. This is also incorporated as a constant of 1s in our case. So a delay of 1s is straight away incurred in the non-overlapping case. Furthermore the huge jump of MIP2 is due to the delay in MIP's

handoff. MIP1 seems to have the same handoff latency as IPMN in the left hand side of the graph. This performance gain is achieved by sending beacon signals every 100ms utilizing the bandwidth more for movement detection rather than actual communication. So the performance gain in handoff attained reduces the bandwidth available. Also the huge amount of traffic generated and the processing time spent on communication would rather degrade the overall performance, in an attempt to improve the handoff delay.

In overlapping case as shown on the right side of the graph IPMN handoff has very little overhead. MN can perform L3 handoff while still performing a L2 handoff. This simultaneous process drastically reduces the handoff delay in IPMN. MIP1 and MIP2 have larger delays which cannot handle time critical traffic.

As can be seen the performance of MIP is critically dependent on its AA life time period.

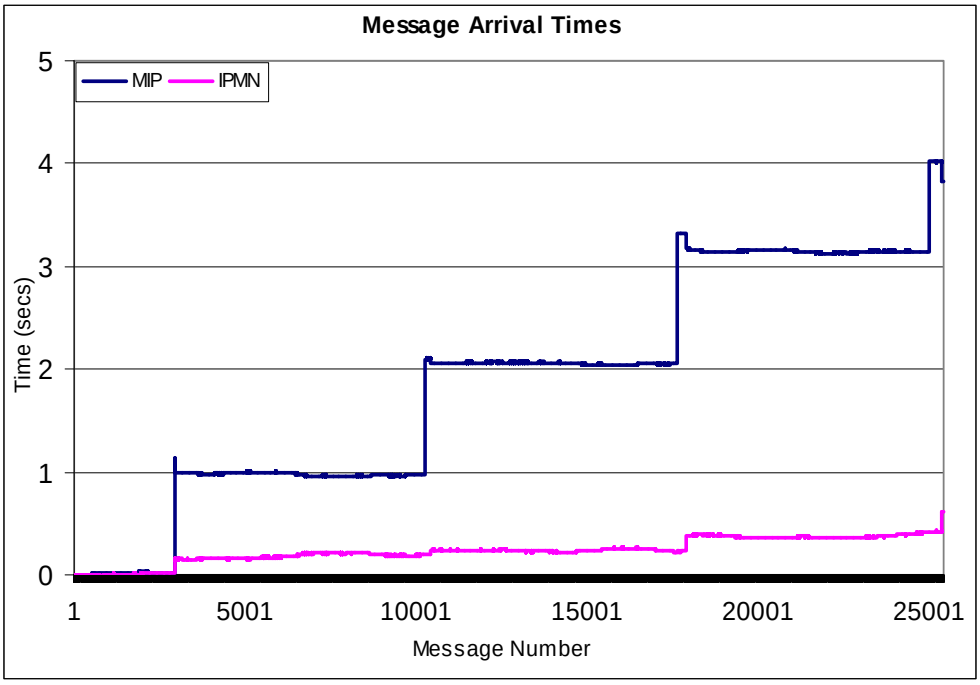


Figure 18. Message Arrival Times in an ideal channel.

So we plotted both for AA Life Time = 1 sec (MIP2), the typically recommended, and 100 ms (MIP1), which is upper limit. Figure-17 gives the pictorial representation of handoff delays of MIP1, MIP2 and IPMN. Though MIP1 has considerable decrease in the handoff delay, its periodicity degrades the effective throughput. MIP2 is the recommended practice, but fails to deliver deadline sensitive traffic. In overlapping scenario IPMN has a handoff delay of 68ms--way better than that of MIP2 (recommended practice) having this delay equal to 1.15s on average. Similarly in non-overlapping scenario IPMN has 1.07s handoff delay while MIP2 has 2.1s on average for 100 handoffs.

To show the effectiveness of IPMN's handoff we also simulated the message arrival times at MN with a message size of 1KB and MN would receive 30,000 of such messages. Handoff is scheduled for every minute for both MIP and IPMN. Figure-18 depicts the characteristics of MIP and IPMN in an ideal channel. MIP has longer jumps than IPMN and degraded performance along with the deployment costs. We also took into account the Bit Error Rates (BER) which can be even upto10% of the transmission rates. [18] suggests that BER can be modeled as a Log-Normal distribution. We have used their claim to simulate the BER model by incorporating the delay aspects into the RTT using a Log-Normal distribution.

$$f_t(t) = \frac{1}{\sqrt{2\prod}\sigma t} e^{-\frac{1}{2\sigma^2}[\log_2(t)-m]^2} \quad where \quad m = scale \quad parameter \qquad .....12$$

We evaluated our scheme and MIP on different traffic patterns from voice to WWW to controlled time sensitive traffic generated using the models given by NetSpec [19]. The

delay between the actual arrival time and expected arrival time was plotted in each case for both MIP and IPMN. To give a better understanding we have also plotted the No Handoff case (where there is no handoff) along with the MIP and IPMN cases. The delay in this No Handoff case is only due to BER and/or congestion. The following subsections will compare the performance of IPMN and MIP in FTP, Voice, WWW and Time Critical Video Stream traffic patterns. IPMN adapts to all these patterns effectively.

6.2 FTP Message Arrival Times

Ftp follows an exponential distribution for Session inter-arrival times and a log-normal distribution for the item sizes in Session Level. We used these density functions to generate cross traffic by incorporating these delays in RTT measurements. We used λ = *0.05* for exponential distribution *mean = 6* and *standard deviation = 2* for Log-Normal distribution.

$$f_X(x) = \lambda e^{-\lambda x} \qquad , \lambda = 1/mean$$

$$f_t(t) = \frac{1}{\sqrt{2\prod}\sigma t} e^{-\frac{1}{2\sigma^2}[\log_2(t)-m]^2} \quad where \quad m = scale \quad parameter$$

Figure-19 shows the performance analysis of MIP IPMN to that of Normal case. Here the congestion is generated due to FTP transfers suddenly initiated while the data is being transferred to the MN. The burst indicated in Figure-19 is the delay caused in packet arrival times due to FTP cross traffic. This has been induced as congestion into the packet

arrival times in all the three cases. As evident, in MIP this delay compounds whenever there is a handoff. The handoff in MIP causes congestion in higher layers and cumulatively worsens the packet arrival time. This is evident from the Figure-19. After every handoff there is an exponential increase of packet arrival times. This tends to be linear as the transmission progresses. Our metric was packet arrival times with respect to expected packet arrival. MIP triggers TCP congestion control every time a handoff is initiated. In contrast, IPMN does not have this delay because of the explicit event notification and the persist timer. Overall MIP takes around 5 seconds more than IPMN.

6.3 Voice Message Arrival Times

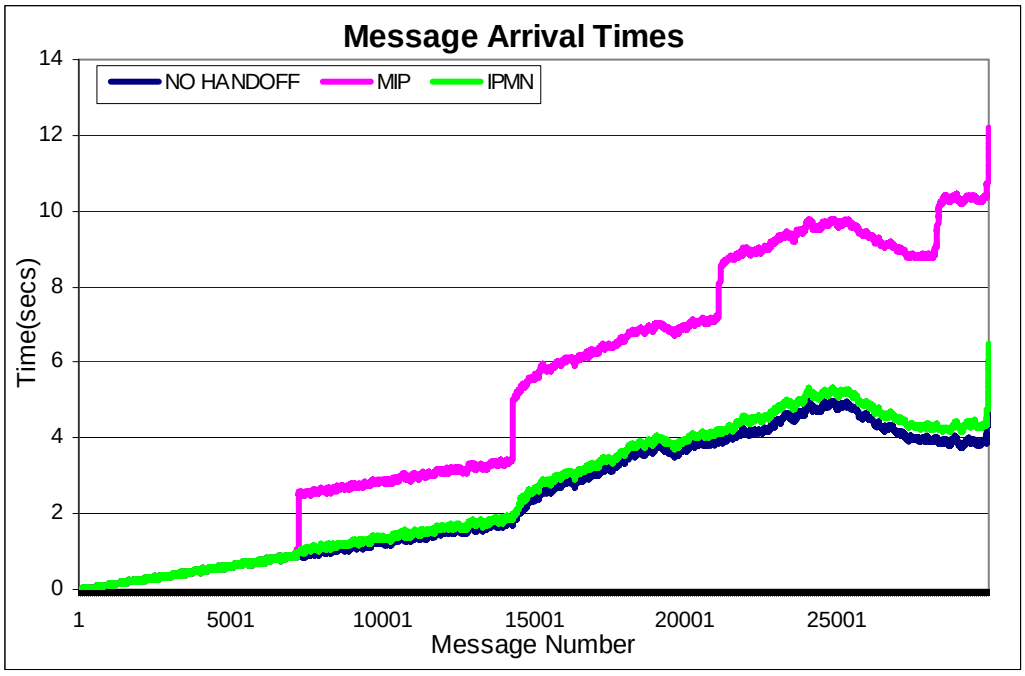


Figure 19. Message Arrival Times with 10% BER and FTP Cross-Traffic.

Voice has a Constant Bit Rate (CBR) traffic characteristic and its typical sampling rate is 8 kHz and each sample consisting of is 8 bits. This gives the standard bit rate of 64 Kb/sec for acceptable voice quality. Call inter-arrival times are modeled to be exponentially distributed

$$f_X(x) = \lambda e^{-\lambda x} \qquad ,\lambda = 1/mean$$

Figure-20 shows the performance when the congestion is due to background voice calls. We have used $\lambda = 0.00333$ and incorporated the congestion factor in packet arrival times. As observed in Figure-20 MIP has lot of performance degradation due to cross traffic. It also suffers a ripple effect and the packet arrival times tend to increase after every handoff. Handoff also has delay induced as congestion into packet arrival times. The

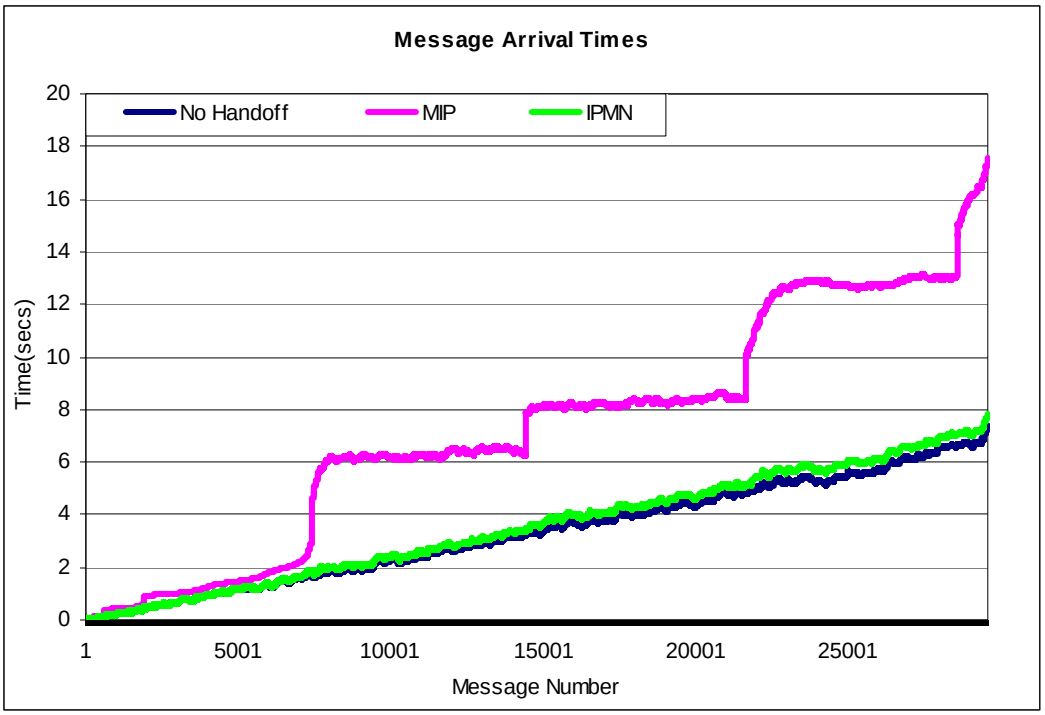


Figure 20. Message Arrival Times with 10% BER and Voice Cross Traffic.

difference in the packet arrival times of MIP and IPMN is almost 10 seconds in this case which gives a clear indication of how MIP would fail delivering voice traffic. IPMN on the other hand has no congestion buildup due to handoff. Compared to IPMN is again almost linear and smooth. MIP cannot handle real time traffic like voice and video while IPMN delivers the same efficiently with very little or no quality degradation.

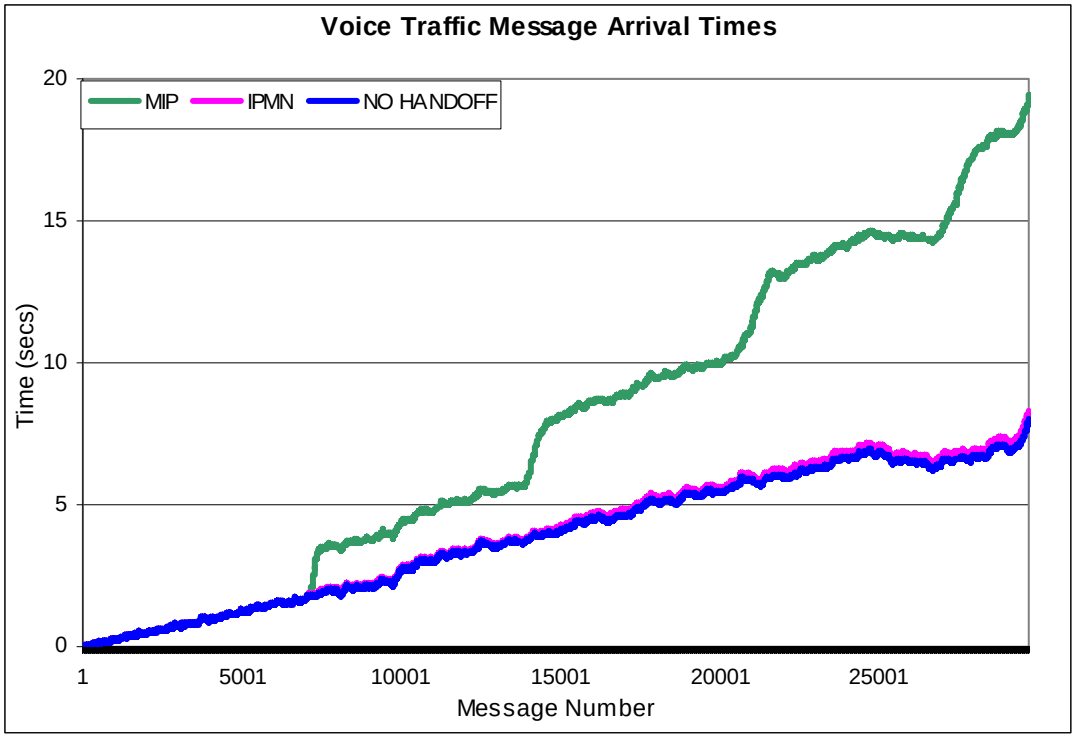


Figure 21. Message Arrival Times of Voice Traffic and Voice Cross-Traffic.

Figure-21 shows the performance of voice traffic for IPMN and MIP along with the No Handoff case. As shown in the figure MIP has longer handoff jumps than IPMN. This is due to the time MIP spends in its handoff procedure. Since voice is a constant bit rate with 144kb bursts MIP stabilizes with 12 sec delay which is almost similar to that in Figure-20. Moreover this delay will have an adverse effect on the TCP timers. TCP

misunderstands this delay as congestion and starts its congestion control algorithms. We used $\lambda = 0.00333$ and incorporated the congestion factor in message arrival times. MIP has lot of performance degradation when carrying voice traffic.

Figure-22 gives the jitter for voice traffic in both MIP and IPMN case. Clearly when there is a handoff the IPMN has lot of jitter. This would again be due to TCP's adverse reaction with MIP. MIP takes a longer time to settle down to its original rate of transmission. In IPMN there is a large jitter during handoffs but recovers quickly after successfully completing the handoff procedure. Since voice is a constant bit rate even MIP recovers reasonably well. Figure-21 shows the jitter caused due to one handoff. This consists of 4000 messages taken from the message arrival times. Just to show the

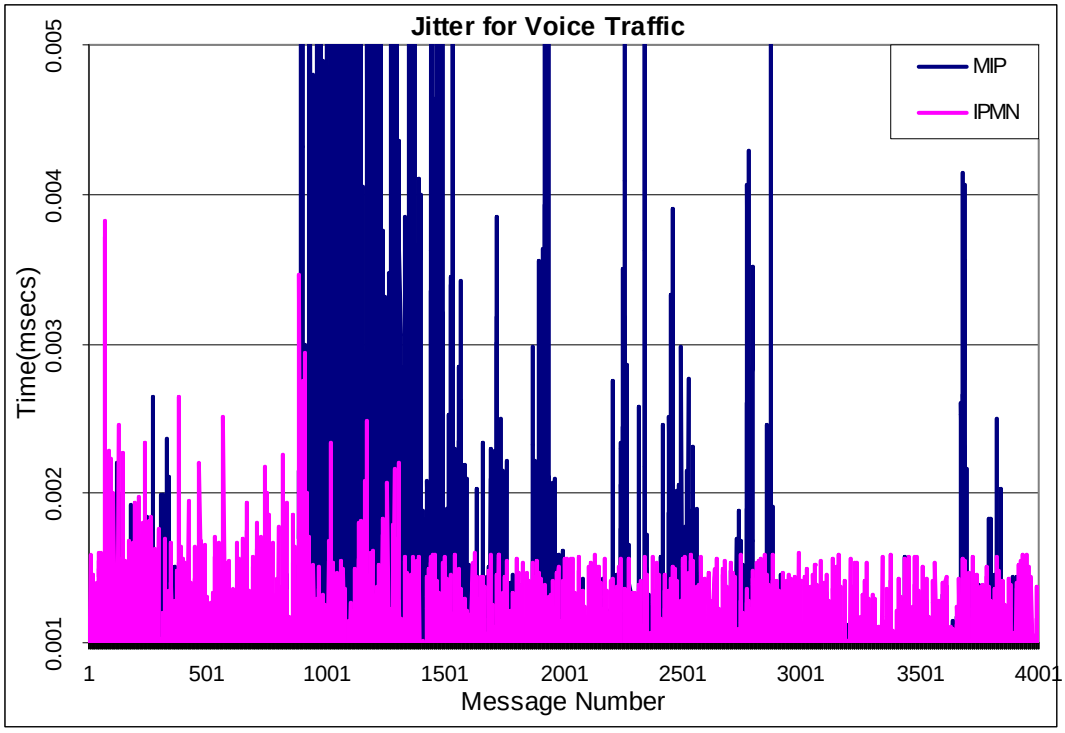


Figure 22. Jitter In Voice Traffic with Voice Cross-Traffic.

behavior during handoff we picked those messages consisting of a handoff some where in between those 4000 handoffs. MIP has jitter even after the handoff due to control algorithms.

6.4 WWW Message Arrival Times

Interactive traffic has the document size as a Power Law or Pareto distribution which is heavy tailed. The probability density function and cumulative distribution function of a Pareto distribution. Since this is a heavy tailed distribution WWW possesses self similarity in network traffic.

$$p_t(t) = \alpha k^{\alpha} t^{\alpha-1} \quad where \quad k = scale\ \ parameter\ \alpha = shape\ \ parameter$$

Call Level of the WWW traffic has a mean request time of 5.75 and can be modeled as a homogeneous Poisson distribution over one hour period implying that the inter-request time is an exponential model. We have generated the distribution using the probability density function and the value of $\alpha=0.45$. This delay was also introduced as congestion into the packet arrival times in between the data transfer shown in Figure-23 as burst. MIP has longer handoff jumps and as a result introduces its own congestion along with the heavy tailed nature of cross traffic. This leads to longer arrival times of packets and the delay creeps up as every handoff will add more and more latency in packet arrival times into WWW interactive traffic. MIP takes 7 seconds more than IPMN for the same data transfer.

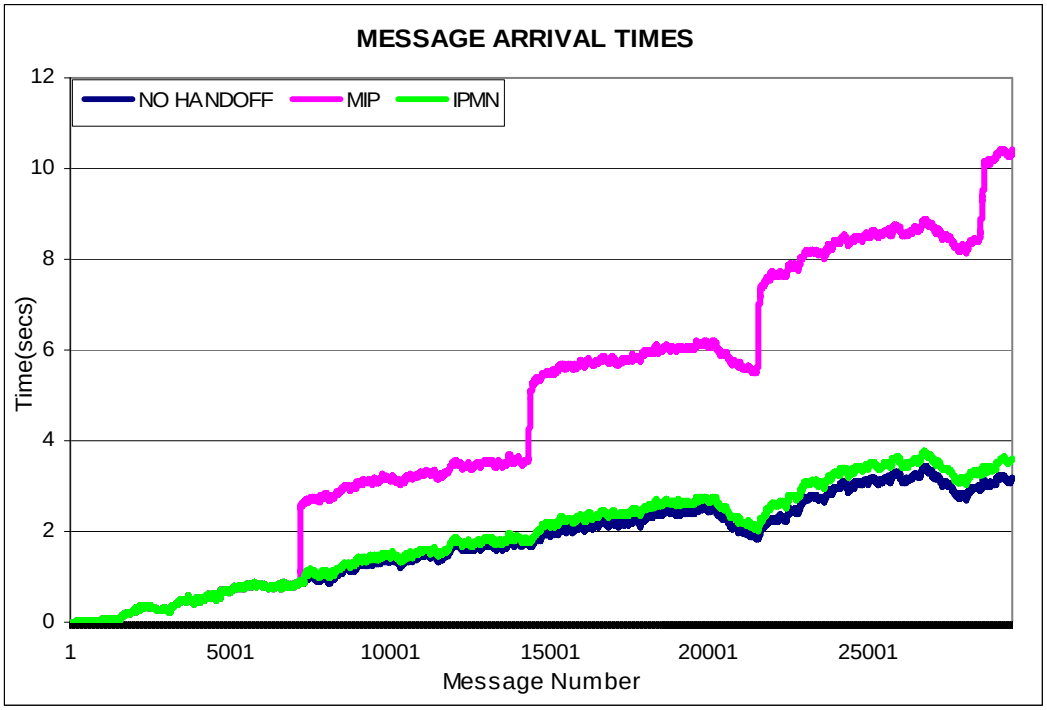


Figure 23. Message Arrival Times with 10% BER and Random Cross-Traffic.

We have generated the messages using the probability density function and the value of $\alpha=0.87$. Figure-24 gives the message arrival times of MIP, IPMN and No Handoff cases. MIP lags behind IPMN in the data transfer for about 15s which is a considerable duration. This delay is acceptable for WWW traffic as this kind of traffic is not always time critical but the security of the data is important. As evident from Figure-24 MIP has longer and longer delays towards the end of data transfer which is due to the heavy tailed nature of the WWW traffic.

6.5 Interactive steering Message Arrival Times

We provide analyses for another interesting class of traffic- interactive video steering. It typically has time sensitive short control message in one direction, and high

$$f_t(t) = \frac{1}{\sqrt{2\prod}\sigma t} e^{-\frac{1}{2\sigma^2}[\log_2(t)-m]^2} \quad where \quad m = scale \quad parameter$$

bandwidth video traffic in the other. Interactive steering traffic is modeled as Telnet traffic with a delay due to video cross traffic along with random cross traffic delay induced into the message arrival times. Delay in this case is due to real-time video traffic which follows a variable bit rate. In order to produce high quality motion pictures the inter frame period must be at least 33ms (30 frames/sec). There are three types of coded

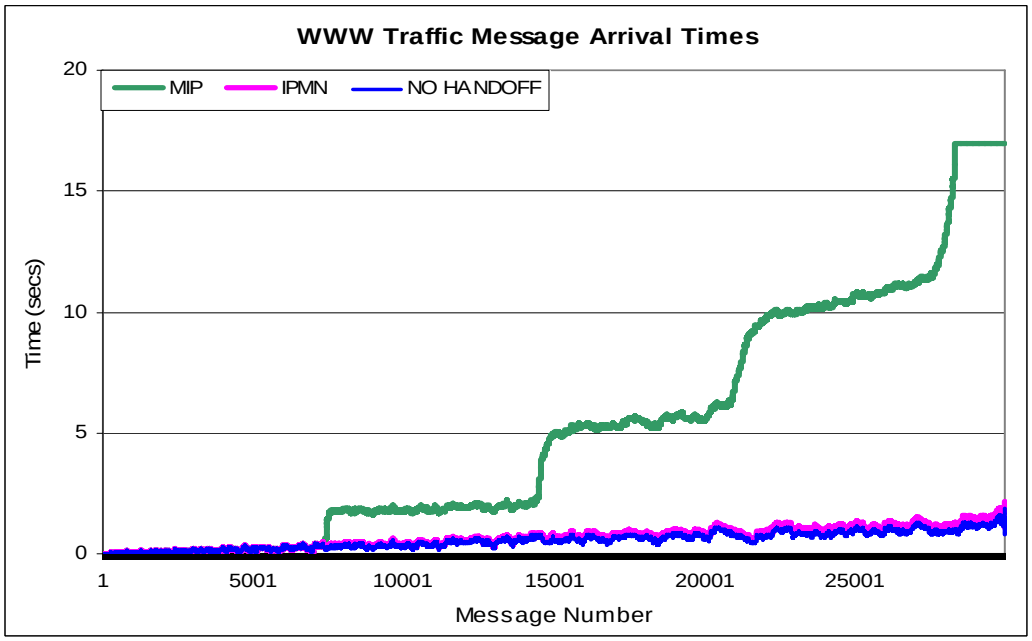


Figure 24. WWW Traffic Arrival Times with and Random Cross-Traffic.

Table 9. Mean and Standard Deviation of Coded Frames in Video Traffic

Frame Type	Mean	Standard Deviation
I-Frame	5.1968	.20160
P-Frame	3.7380	0.5961
B-Frame	2.8687	0.2675

frames called Intra-coded (I-frame), Prediction (P-frame) and Bi-directional (B-frame) and occur in a particular sequence.

The sizes of all these three frames follow Log-Normal distribution with different means and standard deviations given by Table-9. We have generated the messages using the

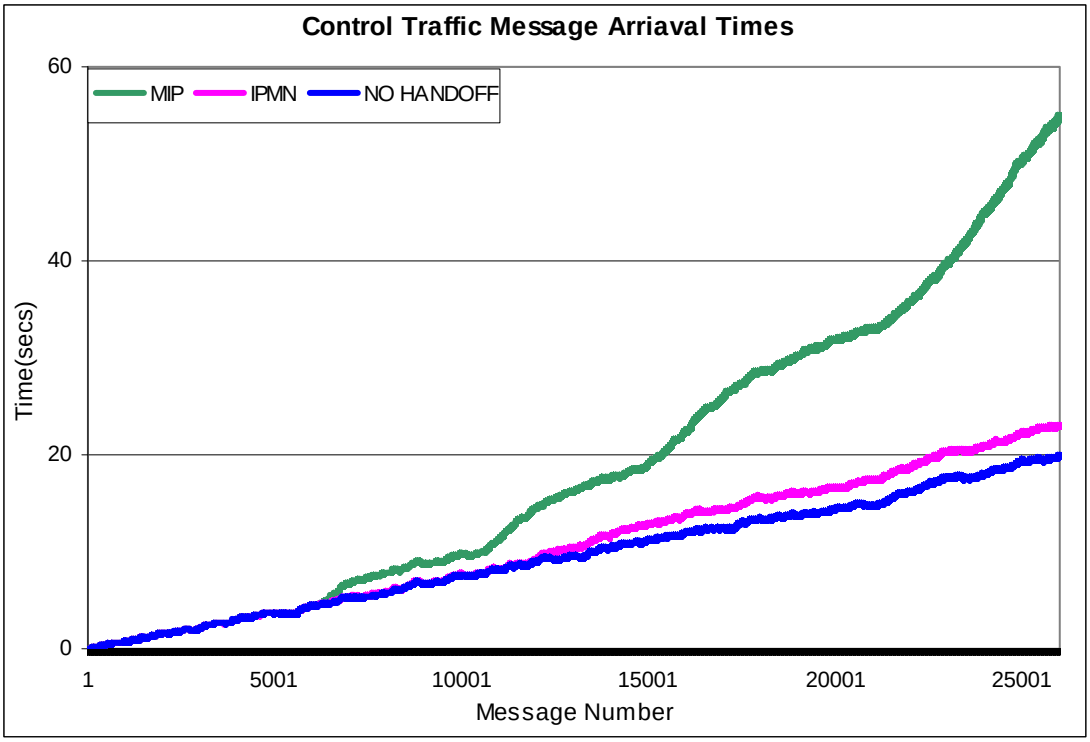


Figure 25. Control Traffic Arrival Times with Random Cross-Traffic.

probability density function and the value of $\alpha=0.9$. Also the various delays of video streaming are incorporated as Log-Normal distribution. Figure-25 illustrates the climbing delay in MIP.

This is due to congestion the system develops while transferring this time critical data. Even in this case though MIP alone handles this effectively by caching the packets at the HA and re-routing it locally, it has delay characteristics that TCP may react adversely. Even the slightest delay would have a ripple effect on the consequent data transfer. The congestion keeps on building even after a handoff because of the amount of data transfer required. A slight deviation in IPMN can also be noticed due to huge amount of incoming data. Though IPMN does not introduce the delay the after effect of the small delay caused due to handoff will trigger retransmission timeouts after the handoff is

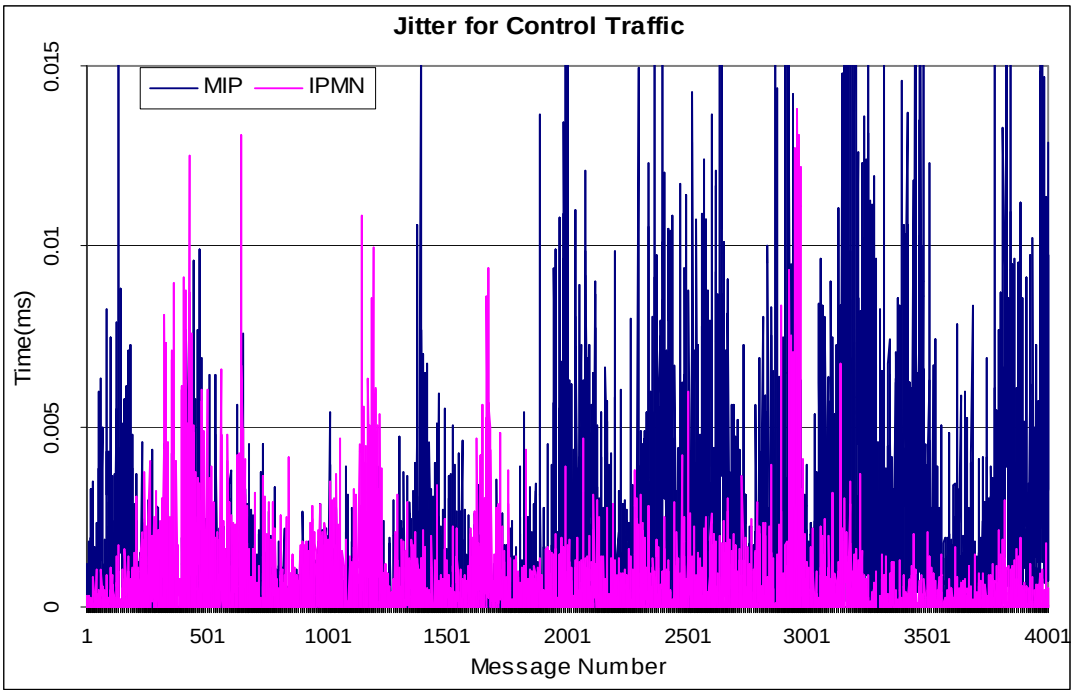


Figure 26. Jitter in Control Traffic with Random Cross-Traffic.

performed.

This is due to the enormous amount of incoming traffic generated that monopolizes the bandwidth during the course of this data transfer. MIP fails drastically to handle this quality of data transfer. IPMN completes the data transfer 30s before MIP and deviates only by 2s from the No Handoff case. Figure-26 clearly depicts that MIP is not at all compatible to handle control data. Jitter keeps on compounding and MIP reaches a state where congestion events are repeatedly triggered in TCP. MIP reaches a point where this data will not be deliverable at all. Even here we have 4000 messages encompassing 2 handoffs. As depicted MIP will take longer to recover from handoff and the jitter developed will continue for a longer time even after the handoff. The second handoff is even worse where MIP takes much longer time to recover. In this event of recovering it encounters more congestion and practically cannot recover to deliver within the time constraints.

Chapter 7

Conclusion

During handoff a smooth transition of the change in the network attachment point is a challenging job. Most time consuming state during this process is detecting a L3 handoff after a L2 has been performed. There are already effective mechanisms that speed up both L2 and L3 handoffs individually. The challenge lies in not trying to speed up individual layer handoffs but to attain rapid handoff by optimizing the individual handoffs in overlapping time interval. To attain such a task cross-layer optimization technique was employed.

802.11 MAC layer handoff is meticulously studied for individual processes that are most time consuming. Events that might be useful to the application to realize interactivity are identified. A subset of these event set is useful during the course of a L2 handoff. Using the subscribe-notify mechanism the 802.11 events are either used to access the hidden state information from L2 layer or to manipulate the internal information of other layers, in this case TCP (L4) and IP (L3).

In other words a L3 handoff is started during a L2 handoff attaining parallelism wherever possible drastically reducing the handoff times. The only way to realize such a time sensitive process is through cross-layer optimization.

In this thesis we have presented an 802.11 based wireless infrastructure-less high performance mobility protocol which uses event driven end-to-end notification mechanism to handle mobility. The event based architecture performs well than timer based approach reducing the reaction time. Timer based approach are bound by the duration of the timer. Too much of the timer interval reacts slowly to the situation at hand while too little will ramp up unnecessary traffic and network congestion. used or manipulated to achieve rapid loss-free handoff. We have also identified more events in 802.11 to make the interactive solution for 802.11.

The thesis also identifies a complete set of events for 802.11 that could be used other than for mobility to MIP requires extensive infrastructure. It requires new software at one end-point (MN), one new entity at MN's traditional base station from which it gets identity (HA) and also a battery of network deployed entities (FAs). Copies of FA have to be available ahead of time in all parts of the Internet where the mobile node might move. In comparison, the IPMN does not require HA, does not require the battery of pre-deployed FAs. However, IPMN requires the sending end-point to be mobility aware. MIP does not need this. Naturally, there will be mobility scenarios where one will be more appropriate than the other.

In scenarios where infrastructure is available and the corresponding host is in a different administrative domain, deploying MIP seems to be a better option. It will require a change in the one end-point (MN) and agents deployment within the infrastructure. But, it might not always be possible to have access to an administrative

access to infrastructure, rather accessing both the end-points is feasible. In such scenarios IPMN is definitely an unquestionable choice. When both options are feasible it is worthwhile to consider the tradeoff. Tradeoff for IPMN is that it offers higher performance on several counts. Besides, the difference in deployment scenarios, the IPMN offers blazingly fast handoff (because it is based on local event) compared to MIP (which is based on remote timer/beacon). IPMN also uses simplified routing. A packet is directly routed in both directions (does not have to go through HA). Just the hop count is reduced approximately by half. There is also no tunneling. This is quite expensive generally and requires packing/unpacking. This substantially adds not only delay but also jitter to packets. IMPN altogether eliminates it. With the elimination of infrastructure it also drastically reduces the deployment and maintenance costs. This range of performance advantage is likely to make IPMN a candidate technology for connection oriented mobility. This is currently very difficult on MIP.

As discussed earlier Cross-Layer interaction has many advantages. The inherently hidden information in one layer can be used by other layers to achieved optimized results. The layered approach is simple to implement and maintain but with application demanding more optimized network solutions cross-layer interaction will become more and more popular. The complexity of cross-layer optimization will increase if each layer has access to every other layer. This would lead to a chaos rather than optimization.

The cross layer optimization that we proposed is a cleanest way of achieving inter-layer communication. All the communication happens through the application and

allows the application to selectively control the communication and dynamically change the way each layer can communicate. This allows robustness to the architecture.

There are still a lot of issues that need to be addressed to have a complete solution. Some major issues that are identified are security, how to avoid attacks. There is secure way of how to allow the applications controlled access to the internal states. In [23] this has been well documented. Given that the application has control, there can be various security measures that can be integrated with the applications on a per connection basis.

For instance what will happen if the authentication fails after the connection freeze? This situation can be tackled easily because the logic is maintained by the application at all times. There are 3 possible ways this could be handled. 1) The application waits only for a minimal amount of time and unfreezes the connection. Since there is no physical connection established the connection will eventually be dropped 2) The Application can freeze the connection and wait until a successful authentication is done. For larger file transfers it might be a good idea to retain the connection instead of dropping the connection and starting all over again. 3) Authentication failure can be subscribed and the application can decide based on the information available from link layer and the nature of the application that is running. Similarly there are many scenarios where the system can fail. But due to ease of handling all these scenarios in the application it makes it more robust and easy to use.

8 References

[1] Perkins C., “IP Mobility Support,” RFC2002, IETF, Oct 1996.

[2] Charles Perkins, David B. Johns “Route Optimization in Mobile IP”, IETF, February 1999.

[3] Fikouras N. and Görg C., “Performance Comparison of Hinted and Advertisement Based Movement Detection Methods for Mobile IP Hand-offs,” In Proc. of the European Wireless 2000, Dresden, Germany, September 2000.

[4] Singh R., Tay Y., Teo W., and Yeow S., “RAT: A Quick (And Dirty?) Push for Mobility Support,” 2nd IEEE Workshop on Mobile Comp. Systems and Applications, pp. 32, Feb. 1999.

[5] A. Bakre., and B.R. Badrinath., “ Handoff and system support for Indirect TCP/IP, Proc. Second Usenix Symp. On Mobile and Location –Independent Computing 1995.

[6] Goff T., Moronski J., Phatak D., Gupta V., "Freeze-TCP: A True End-to-End TCP Enhancement Mechanism for Mobile Environments," INFOCOM'00, Tel-Aviv, Israel, pp. 1537-1545, 2000.

[7] Gustafsson E., et al. “Mobile IPv4 Regional Registration” draft-ietf-mobileip-reg-tunnel-05, IETF, September 2001.

[8] Maltz D., and Bhagwat P., “MSOCKS: An architecture for transport layer

mobility". In *Proc. IEEE Infocom '98,* March 1998".

[9] Bakre A., and Badrinath B.R., "I-TCP; Indirect TCP for Mobile Hosts", In Proc. Of 15th International conference on Distributed Computing Systems, May 1995.

[10] Javed I. Khan and Raid Y. Zaghal, Symbiotic Rate Adaptation for Time Sensitive Elastic Traffic with Interactive Transport, Journal of Computer Networks, Elsevier Science March 2006.

[11] Javed I. Khan and Raid Y. Zaghal, Interactive Transparent Networking: Protocol Meta-modeling based on EFSM, Journal of Computer Communications, Elsevier Science, May 2006.

[12] Funato D., Yasuda K., and Tokuda H., "TCP-R: TCP Mobility Support for Continuous Operation", Proc. International Conference on Network Protocols, 1997.

[13] R. Y. Zaghal, S. Davu, and J. I. Khan, An Interactive Transparent Protocol for Connection Oriented Mobility – Performance Analysis with Voice Traffic, IEEE Wireless and Optical Communications, WiOPT05, 3rd IEEE Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, Riva Del Grande, Italy, April, 2005, pp.219-228.

[14] Velayos H., Carlson G., "Techniques to Reduce 802.11b MAC Layer Handover", Technical Report, April 2003

[15] Perkins C., "Mobile IP, Design Principles and Practices," Addison Wesley, 1998.

[16] Davu S., "Handoff Model of Mobile IP", Technical Report, Kent State University,

2005.

[17] Postel J.B., “Transmission Control Protocol”, RFC 793, September 1981.

[18] Carvalho M., Aceves J., “Delay Analysis of IEEE 802.11 in Single-Hop Networks”, ICNP, November 2003.

[19] Lee B., Frost V., “Wide Area ATM Network Experiments Using Emulated Traffic Sources”,Technical Report, Jan. 1998.

[20] Yokota H, et al., “Link Layer Assisted Handoff Method over Wireless LAN Networks,” Proc. of MOBICOM ’02, Sept. 2002.

[21] Carvalho M., Aceves J., “Delay Analysis of IEEE 802.11 in Single-Hop Networks”, ICNP, November 2003.

[22] Alex Snoeren., Hari Balakrishna., “An End-to-End Approach To Host Mobility”, 6th International Conference on Mobile Computing and Networking, MOBICOM 2000.

[23] Mishra A., Shin M., Arbaugh W., “An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process,” Dept. of Computer Science, University of Maryland, technical report number CS-TR-4395.

[24] Okoshi, T. Mochizuki, M. Tobe, Y. Tokuda, H. “ MobileSocket: toward continuous operation for Java applications,” ICCCN, 1999

[25] Xiliang Zhong, Cheng-Zhong Xu, Haiying Shen, "A Reliable and Secure Connection Migration Mechanism for Mobile Agents," ICDCSW, 2004

[26] Fielding R., Gettys J., Mogul J., “Hypertext Transfer Protocol -- HTTP/1.1”,

RFC 2616, IETF, June 1999

[27] Postel J., Reynolds J., “File Transfer Protocol”, RFC 959, IETF, Oct 1985

[29] Jacobson V., “Congestion Avoidance and Control,” Computer Communication Review, vol. 18, no. 4, pp. 314-329, Aug. 1988.

[30] Jacobson V., “Modified TCP Congestion Avoidance Algorithm,” end2endinterest mailing list, April 1990.

[31] Postel ., “User Datagram Protocol”, RFC 768, IETF, Aug 1980

[32] Schulzrinne H., Casner S., Frederick R., and Jacobson V., "RTP: A Transport Protocol for Real-Time Applications," RFC 3550, July 2003.

[33] Rosenberg J., Schulzrinne H, “Session Initiation Protocol”, RFC 3261, IETF, June 2002

[34] Droms R., “Dynamic Host Configuration Protocol”, RFC 2131, IETF, Mar 1997

[35] Khan J., Zaghal R., and Gu Q., “Symbiotic Streaming of Elastic Traffic on Interactive Transport,” IEEE ISCC'03, Antalya, Turkey, July 2003.

[36] Khan J., and Zaghal R., “Jitter and Delay Reduction for Time Sensitive Elastic Traffic for TCP-interactive based World Wide Video Streaming over ABone,” Proc. of the 12th IEEE-ICCCN 2003, Dallas, Texas, Oct. 2003, pp.311-318.

[37] Davu S., Zaghal R., and Khan J., An Infrastructureless End-to-End High Performance Mobility Protocol for Connection Oriented Applications using Interactive Transparent Networking, Proceedings of the 2005 IEEE International Conference on Electro Information Technology, 2005, Lincoln Nebraska, May 2005.

[38] Javed I. Khan, Sandeep Davu, & Raid Y. Zaghal, High Performance Mobility without Agent Infrastructure for Connection Oriented Services, Journal of Pervasive and Mobile Computing, Elsevier Science, February 2008

[39] Allman, M., Paxson, V. and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999.

[40] Floyd, S. and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 2582, April 1999.

[41] Alex Snoeren., David G Anderson., Hari Balakrishna., Fine-Grained Failover Using Connection Migration, Proc. of the Third Annual USENIX Symposium on Internet Technologies and Systems (USITS), March 2001

[42] Calvert, K. et al, “Architectural Framework for Active Networks”, Active Networks Working Group Draft, July 1998.

[43] Biswas, J., et al., " The IEEE P1520 Standards Initiative for Programmable Network Interfaces” *IEEE Communications Magazine,Special Issue on Programmable Networks*, October, 1998.

[44] Wetherall, D., Guttag, J. and Tennenhouse, D., “ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols”, Proc. IEEE OPENARCH'98, San Francisco, CA, April 1998.

[45] Jonkman R., Evans J., and Frost V., "Netspec: A Tool for Network Experimentation and Measurement", University of Kansas, 1998. http://www.ittc.ku.edu/netspec/