Fall 10-31-1997

Analysis of the protocol for the handover in a micro-cell packet switched mobile network

Anna M. Thomas
New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

ANALYSIS OF THE PROTOCOL FOR THE HANDOVER IN A MICRO-CELL PACKET SWITCHED MOBILE NETWORK

by
Anna M. Thomas

The overlay of microcells over macrocells offers new opportunities and is cost effective, which makes the handover in such a micro-cellular environment an important issue. The objective of this study is to analyze and prove the truthfulness of a new protocol proposed, which guarantees no packet loss during the handoff. The study here emphasizes on the data integrity issue, since it is the key factor affecting the throughput performance of the transport layer. It is shown that usage of the Internet Protocol leads to packet loss in the presence of handover. Simulation results reveal that the proposed new protocol preserves the data, even in the presence of handoff. A queuing model is built to illustrate the effect of data integrity on the gateway. Simulation results of this model exhibit the burden on the gateway. The three phases of the analysis mentioned above, are focused in displaying the data integrity issue.

ANALYSIS OF THE PROTOCOL FOR THE HANDOVER IN A MICRO-CELL PACKET SWITCHED MOBILE NETWORK

by
Anna M. Thomas

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Department of Computer and Information Science

October 1997

APPROVAL PAGE

ANALYSIS OF THE PROTOCOL FOR THE HANDOVER IN A MICRO-CELL PACKET SWITCHED MOBILE NETWORK

Anna M. Thomas

Dr. Yeheskel Bar-Ness, Thesis Advisor — Date
Distinguished Professor of Electrical and Computer Engineering, NJIT

Dr. James McHugh, Committee Member — Date
Professor of Computer Science, NJIT

Dr. Michael Hinchey, Committee Member — Date
Assistant Professor of Computer Science, NJIT

BIOGRAPHICAL SKETCH

Author: Anna M. Thomas

Degree: Master of Science in Computer Science

Date: October 1997

Undergraduate and Graduate Education:

- Master of Science in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 1997

- Master of Science in Mathematics,
  Madras Christian College, Madras, India, 1991

- Bachelor of Science in Mathematics,
  Madras Christian College, Madras, India, 1989

Major: Computer Science

Presentations and Publications:

Anna Muthulakshmi Thomas, C.N. Krishnan, S. Srividya,
"Approximation to the Set of Common Functions", Report for the Department of Science and Technology of India, Oct, 1994.

To my dear Bobby

ACKNOWLEDGMENT

I express my deep sense of appreciation to my advisor Dr. Yeheskel Bar-Ness, who provided me with support, guidance and enthusiasm, throughout the course of study. His encouragement in pursuing my field of interest and his constant effort to provide the resources that were needed, are greatly appreciated. I would like to thank Dr. James McHugh and Dr. Michael Hinchey for serving as the committee members. My special thanks are due to Dr. James McHugh, without his encouragement and support this work would not have been made possible

I would like to thank Mr. Nico Van Waes for his suggestions and great help during the course of work. Last but not the least, I would like to thank my husband Bobby Thomas, who was an inspiration and guide in the entire course of my Master's Degree.

TABLE OF CONTENTS

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

LIST OF ABBREVIATIONS

ACK	Acknowledgment
AP	Access Point
BSS	Basic Service Set
CRC	Cyclic Redundancy Check
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
DLC	Data Link Control
DSA	Extended Service Area
DSS	Distribution System Services
ESS	Extended Service Set
ICI	Interface Control Information
IDU	Interface Data Unit
IP	Internet Protocol
MAC	Medium Access Control
MTU	Maximum Transfer Unit
MSDU	MAC Service Data Unit
PDU	Protocol Data Unit
PHY	Physical layer
QoS	Quality of Service
SDU	Service Data Unit
STA	Station
TCP	Transmission Control Protocol
WLAN	Wireless Local Area Networks
WM	Wireless Medium

CHAPTER 1

INTRODUCTION

1.1 The World of Wireless Networks

The vision of the future for telecommunications is, " Information at any time, at any place, in any form". The mobility of human kind has grown immensely and the need to perform various important jobs in mobile has also grown over the years. And hence making wireless networks, the network of the future.

The wireless communication revolution is bringing fundamental changes to communication and computing. Wireless networking does work well as an extension to wired networks. Some wireless technologies extend wired networks by providing high-speed links between networks, whereas other wireless technologies extend traditional networks by giving mobile users easy, ubiquitous access. This access can benefit people who work in environments that require mobility, such as hospitals, restaurants, warehouses, manufacturing plants, and stock exchanges. Products based on mobile user wireless technologies transmit data slowly but over large distances and hence they are most useful for mobile users who need access to low-bandwidth applications on the company network, including e-mail, database applications, and file transfer applications.

Wireless networking is widely being used, because of its mobility, ease of installation in difficult-to-wire areas, increased reliability, long term cost savings etc. In spite of its various advantages, there are some serious concerns in this growing network field that surround the implementation and use of wireless networking. These concerns are radio signal interference, power management, system interoperability, network

security, installation issues, handover etc. This work concentrates on the problem of handoff in a microcell packet switched mobile network.

1.2 The Handover

Handover is a mechanism that transfers support of a mobile from one base station to another as a user moves through the coverage area of a cellular system. The number of cell boundary crossings increases as the smaller cells are deployed to meet the demands for increased capacity.

The steps involved in the handover are,

- Starting in a state where only one cell is supporting the call in question.
- Determining that over-the-air link conditions between the mobile and the old serving cell are deteriorating and that there is a potentially better link to a new candidate cell.
- Informing the candidate cell of the imminent handoff, including parameters needed to identify the mobile and execute the handoff.
- Signaling the mobile to begin executing the handoff.
- New cell beginning to service the mobile
- Mobile beginning to use the new cell.
- Entering the mid-handoffs state(prolonged only in CDMA)
- Mobile discontinuing the use of the old cell.
- Old cell stopping service to the mobile.
- Ending in a state where one cell , the new one , is supporting the call in question.

Each handover requires network resources to reroute the call to the new base station. The switching load decreases as the expected number of handovers are reduced. Another important factor in this handover process is the delay. If handover does not occur quickly, the Quality of Service (QoS) may degenerate below an acceptable level. Minimizing delay also minimizes co-channel interference. During the handover there is a brief service interruption. As the frequency of these interruptions increases the perceived QoS is reduced. The chances of dropping a call due to factors such as the availability of channels increase with the number of handover attempts. As the rate of handover increases, handover algorithms need to be enhanced so that the perceived QoS does not degenerate and to keep the cost under control. The deployment of a multitier system with macrocells overlaying microcells offers new opportunities. The end-user performance and system capacity can be increased with the right usage of the two tiers. For example, stationary users can be assigned to microcells so that they operate at reduced power and cause significantly less interference; when the microcellular capacity is exhausted, the overflow traffic can be assigned to the macrocells.

1.3 The Proposed Problem

Since the handover requires termination and re-establishment of communications and change in network routing, it may result in loss, duplication and misorder of packets. This leads to the performance loss in the higher layer i.e the Transport Layer Protocol. The widely used Transport Layer Protocol, the TCP does not differentiate between the packets

that were lost due to handoff and the packets that were lost due to congestion in the network. And as a result, the congestion recovery mechanism reduces the size of the transmission window, which leads to the reduction in the throughput of the system. It is possible that the retransmissions are also lost. These issues are discussed in [1-3]. The protocol proposed in [4], guarantees that all packets sent by the gateway g to the mobile m and vice versa will be received in order with no loss or duplication, as long as the mobile m moves between cells belonging to the same region. This study analyzes mathematically as well as with simulations, the behavior of this protocol.

The proposed protocol claims to have the following advantages:

- All packets sent by the network layer of m are delivered to the network layer of g in order with no gaps or duplication. i.e it maintains the data integrity.
- There's no minimal impact of handover on the transport protocol.
- Packets do not have to be retransmitted on the radio channels due to handovers. Handover overhead is encountered on the base stations' network where bandwidth is not a scarce resource.
- The Processing burden on the gateway is small
- Multiple overlapping handovers are allowed.

The idea is to prove the following claims using simulations and analytical model.

- Showing the data disintegration using the IP protocol which has no ARQ mechanism, the assumption here is that there are no DLC links between the stations.
- Showing the data integration using the proposed new protocol which has ARQ mechanism and has special set of rules handling the handover problem.

- Developing a Markov Model for the gateway to show the effect of retransmissions on the gateway.

In Chapter 2, the proposed new protocol is stated and discussed. In Chapter 3, the packet loss using the IP protocol is analyzed, supported by simulations. The IP protocol is explained in brief as it is used in the simulation analyses. Here it is assumed that there is no DLC links present between the stations. The DLC protocol used to describe the proposed new protocol here IEEE 802.11. Chapter 4 describes the draft standard of IEEE 802.11 and discusses the simulation results using the proposed new protocol. Chapter 5 provides a Markov Model to show the effect of retransmissions and the burden on the gateway due to the retransmissions. Chapter 6 gives the conclusion of the simulation study of these protocols.

CHAPTER 2

HANDOVER IN A MICRO-CELL PACKET SWITCHED MOBILE NETWORK

2.1 The Proposed New Protocol

When a mobile enters the cell of a base station say b_i , b_i creates a Network layer entity that performs a copy of the protocol for the mobile[4]. This state of the base station, where it is initialized, is the passive state. A base station b_i is in active state, when it receives packets from the gateway g, sends them to the mobile m. It is in connecting state, if it receives packets from b_{i-1} send them to the mobile m. It is in disconnecting state, when it receives frame packets from the gateway g and send them to the next base station b_{i+1} . And it is in Repeat_Handover state when it receives packets from b_{i-1} sends them to b_{i+1}.

Once the base station is in passive state, where the DLC is initialized between the mobile m and the base station b_i a control message JOIN(m, b_{i-1}) is sent from m to b_i. The base station b_i moves from passive state to connecting state when it receives the control message JOIN and sends a LEAVE(m) message to b_{i-1}. When b_{i-1} is in active state, it sends a STOP(m) to the gateway g and enters disconnecting state. In the disconnecting state b_{i-1} sends b_i to all packets destined for m it is receiving from g, until a LAST(m) message is received from g. When a LAST message is received, the b_{i-1} attains the passive state and it also sends a LAST(m) message to b_i.

When b_i is in connecting state when it receives the LAST message, it sends a RESUME(m) message to the gateway g and moves from connecting to active state. When g receives the RESUME , it resumes sending packets for the mobile m through b_i. When b_i is not in connecting state, i.e if b_i has received a LEAVE message from another base

b_i is not in connecting state, i.e if b_i has received a LEAVE message from another base station b_{i+1} to which the mobile has moved in the meantime. This kind of LEAVE message makes the base station move to the Repeat_Handover state. The base station b_i sends the packets received from b_{i-1} to b_{i+1} which actually is meant for the mobile. When b_{i-1} receives a LAST message b_i sends a LAST message to b_{i+1} and returns to the passive state. If i=1, then it sends a JOIN(m,nil) control message and enters the active state. Upon entering active state it sends a RESUME(m) to g and proceeds as all other stations.

The gateway algorithm is that , the gateway is in normal state when it sends packets to the base station b_i . When it receives a STOP message from one of the base stations, the gateway sends a LAST(m) message to that base station and enters the suspended state. And upon the receipt of the message RESUME(m) message from some base station, it again goes to the normal state. The formal proof of this algorithm is discussed in [4].

2.2 The Claims of the Proposed New Protocol

The main point that is discussed in [4] is that the data integration . This aspect is considered very important as it greatly affects the throughput performance in the higher layer i.e the transport layer. The proposed new Protocol guarantees the preservation of the packets in the presence of handoff. The protocol is built in such a way that it takes care of single/multiple handoffs without any data disintegration. This protocol requires the use of DLC links between the regional network as the DLC protocols have the ARQ mechanism

to make sure the packet reached the other end. If not, the packets can be re-transmitted in the lower layer as soon as the problem occurred, making the protocol more effective. The processing burden on the gateway is small, since there is only one copy of the protocol between the gateway and the base-station The gateway does not have to deal with the sequence numbers. In case of re-transmissions, the gateway is not burdened much as the base-stations hold responsibility of forwarding the packets in case of any handoffs. The higher layer is not affected much during handoff when using this protocol and hence leading to better throughput performance. And this protocol is simple and fast. The truthfulness of these claims can be found in the following chapters.

CHAPTER 3

THE PACKET LOSS WITH IP PROTOCOL

3.1 The IP Protocol

The TCP/IP is a connectionless, unreliable, best-effort packet delivery system. The Internet Protocol (IP) gives the format for the datagram and the ideas of connectionless delivery. The datagram header contains the source and destination IP addresses, fragmentation control, precedence and a checksum. [8,9]

IP provides 3 important definitions. First, the IP protocol defines the basic unit of data transfer used throughout the a TCP/IP internet. Thus it specifies the exact format of all data as it passes across a TCP/IP internet. Second, IP software performs the routing function, choosing a path over which data will be sent. Third, in addition to the precise formal specification of data formats and routing, IP includes a set of rules that embody the idea of unreliable packet delivery. The rules characterize how hosts and routers should process packets how and when error messages should be generated, conditions under which packets can be discarded. IP is a fundamental part of the design that a TCP/IP internet is sometimes called an IP-based technology.

3.1.1 The IP Datagram

The network layer protocol IP protocol was used in the simulation to show the behavior of the data in the presence of a handover. The data format followed is the IP datagram. The layout of the IP datagram is as follows:

Table 3.1 : The Layout of the IP Datagram

0 4	8	16	19	24	32
VERS	HLEN	SERVICE TYPE	TOTAL LENGTH		
IDENTIFICATION			FLAGS	FRAGMENT OFFSET	
TIME TO LIVE		PROTOCOL	HEADER CHECKSUM		
SOURCE IP ADDRESS					
DESTINATION IP ADDRESS					
IP OPTIONS (IF ANY)				PADDING	
DATA					
. . .					

The 4 bit **version** field is checked before processing a datagram to ensure that it matches with the format that the IP software expects. The **header length** field also 4 bits gives the datagram header length measured in 32-bit words. The header with no option and no padding has the header length equal to 5. **Total length** field gives the length of the IP datagram measured in octets, including the octets in the header and data. The size of the data is Total length-header length. The 8-bit **service type** field specifies how the datagram should be handled and is divided into five sub-fields, 3-bit *precedence* field which specify the precedence of the datagram , 1-bit D field stands for *delay* , when set,

indicating low-delay, 1-bit T field stands for *throughput*, when set, indicating high throughput, 1-bit R filed stands for *reliability* , when set, indicating high reliability and 2 unused bits . To achieve the maximum efficiency, the maximum data size is into one frame. The frame size would be the maximum transfer unit in that network. TCP/IP software chooses a covenant initial datagram size and arranges a way to divide the large datagrams into smaller pieces when the datagram needs to traverse a network that has a small MTU. The small pieces and the process of dividing a datagram is known as fragmentation. Fragmentation normally occurs a the router along the path between the datagram source and its ultimate destination. The router receives a datagram from a network with a large MTU and must send it over a network for which the MTU is smaller than the datagram size. The IP represents the offset of the data in multiples of eight octets, the fragment size must be chosen to be a multiple of eight. Fragments must be reassembled to reproduce a complete copy of the original datagram before it can be processed at the destination. Each fragment has the a datagram header that duplicates most of the original datagram header , except for the bit in the flags filed that shows that it is a fragment, followed by data that can be carried in the fragment and keeping the total length smaller than the MTU of the network over which it must travel.

In TCP/IP internet, once a datagram has been fragmented , the fragments travel as separate datagrams all the way to the ultimate destination where they must be reassembled. The disadvantages of doing this are, datagrams are not reassembled immediately after passing across a network with small MTU, the small fragments must be carried from the point of fragmentation to the ultimate destination. Reassembling at the

destination can be inefficient as when the fragmented pieces go through the network with large MTU capability, only smaller fragments traverse them wasting the large MTU capability. And if the fragments are lost the datagram cannot be reassembled. This is because the receiving machine starts a reassembly timer when it receives an initial fragment. If the timer expires before all fragments arrive, the receiving machine discards the remaining fragments without processing the datagram. This increases the probability of datagram loss when fragmentation occurs because the loss of a single fragment results in loss of the entire datagram.

Three fields in the datagram header, identification, flags, and fragment offset, control fragmentation and reassembly of datagrams. The **identification** field contains a unique integer that identifies the datagram. When a router fragments a datagram, it copies most of the fields in the datagram header into each fragment. The identification field must be copied. The purpose of this field is to let the destination to know which arriving fragments belong to which datagrams. As a fragment arrives the destination uses this field to along with the datagram source address to identify the datagram. This value of the identification filed must be unique for a unique datagram. The IP software keeps a global counter in memory, increments it each time a new datagram is created and assigns the result as the datagram's identification field.

A **fragment offset** specifies the offset in the original datagram of the data being carried in the fragment, measured in units of 8 octets starting at offset zero. To reassemble the datagram the destination must obtain all fragments starting with fragment

offset zero through the fragment with the highest offset. Fragments do not necessarily come in order but the destination reassembles it.

The low order two bits of the 3-bit **flags** field controls fragmentation. If set to 1, it means that do not fragment. The low-order bit in the flags field specifies whether the fragment contains data from the middle of the original datagram or from the end. It is called the more fragments bit. When the IP software in the receiving machine receives the fragments of the original datagram, it knows that there are more fragments following by more field bit set. The total length gives only the total length of the fragment but not the total length of the datagram. Once the more bit is not set, the receiving station understands that there are fragments and it checks to see whether the fragments on hand contain all data needed to reassemble by checking the fragment offset field and the total length fields of the all the fragments.

The **Time to live** counter specifies how long in seconds the datagram is allowed to remain in the internet system. When a datagram is into the internet, the machine sets a maximum time that the datagram should survive. As they processed by the router and hosts this counter must be decremented by 1 and as time passes remove the datagram from the internet when its time expires. The idea here is to prevent the datagram from traveling around the internet forever.

The field **protocol** specifies the format of the data area. The mapping between the high level protocol and the integer value used in the protocol field is administered by a central authority. The field header **checksum** checks the header values. The IP checksum

is formed by treating the header as a sequence of 16-bit integers adding them together using one's complement of the result.

The fields **source address** and the **destination address** contain the 32-bit IP addresses of the datagram's sender and the recipient.

The field labeled **data** shows the beginning of the data. The IP options field if any is of variable length. The Padding field depends on the options selected.

The **options** are not required to be there in a datagram. The purpose of the option field is for the sake of testing and debugging. As mentioned earlier the length is variable depending on which option is selected. The options with single octet long octet code as in Table 2.

Table 3.2 The Option Octet

0	1 2	3 4 5 6 7
COPY	OPTION CLASS	OPTION NUMBER

The option code is divided into 3 fields. The fields of a 1-bit *copy* flag, a 2-bit *option class* and the 5-bit *option number*. The copy flag when set to 1, specifies the option should be copied into all fragments. When set to zero, the copy bit means that the option should be copied into only the first fragment. The option class and option number bit give general class of the option and give specific option in that class.

Table 3.3 Option Code Octet

OPTION CLASS	MEANING
0	Datagram or Network Control
1	Reserved for future use
2	Debugging and Measurement
3	Reserved for future Use

Table 3.4 Option Number codes

Option Class	Option Number	Length	Description
0	0	-	End of option list Used if options do not end at end of header
0	1	-	No Operation
0	2	11	Security and Handling Restrictions
0	3	var	Loose Source Routing
0	7	var	Record Route
0	8	4	Stream Identifier
0	9	var	Strict Source Routing
2	4	var	Internet Timestamp

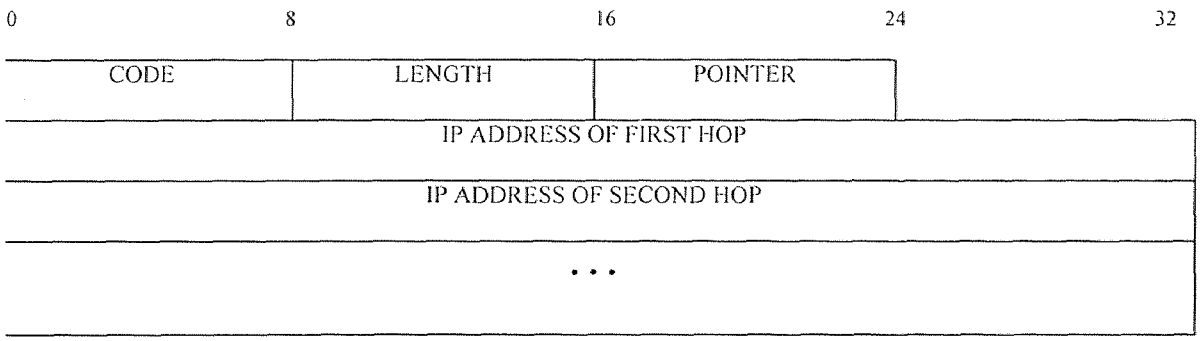
The three main option numbers are 7,9 and 4 which stand for record route, source route and timestamp options. The record route option creates an empty list of IP address and records them down as it goes through traversed routers. The format of the record route option is shown as below. The code field contains the option class and the number.

The length field specifies the total length of the option, including the first three octets. The filed first IP address, second IP address etc. is reserved for recording the traversed addresses. The pointer field specifies the offset within the option of the next available slot. When a machine processes the datagram that has the record route option set, it adds its address to the record route list. To add itself to the list a machine first compares the pointer and length fields. If the pointer is greater than the length, the list is full so the machine inserts the IP address at the position specified by the pointer and increments the pointer by 4.

Source Route Option

This option demands the datagram to travel to travel in a specific path. This is mainly used for testing the networks. There are 2 kinds of source route options. They are the strict source routing and the loose source routing. The format of the source route option is given in the Table 3.

Table 3.5 The Format for Source Route Option

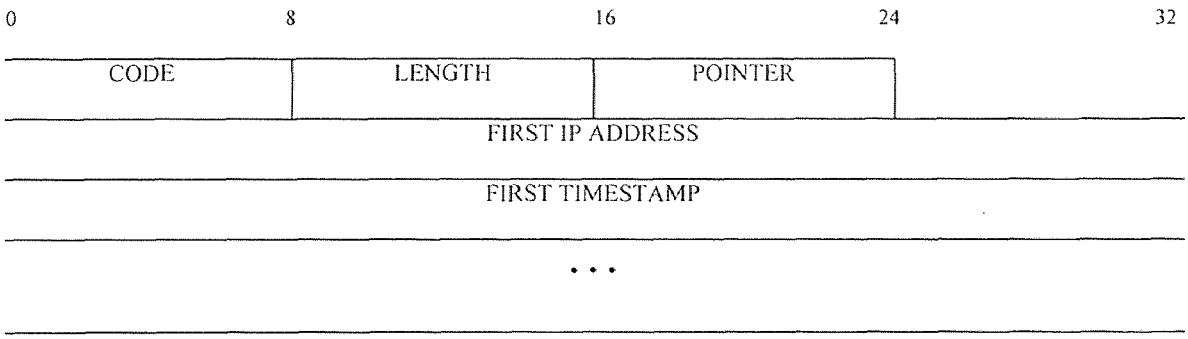


The strict source route option specify the exact path the datagram must take to reach the destination. The path between two successive addresses must be a single network. The loose source routing includes the IP addresses as in strict source routing but this option allows multiple hops between the successive addresses on the list. Both these source routing options require routers along the path to overwrite items in the address list with their local network addresses.

Timestamp Option

The Timestamp option works just like the record route option. The format of the timestamp option is as follows:

Table 3.6 The Format for Timestamp Option



In Table 4 the length and the pointer fields have the same meaning as in the record route option. The 4-bit Oflow filed contains an integer count of routers that was unable to supply the timestamp because the option was too small. The 4-bit flags filed controls the format of the option and how router should supply the timestamps.

The values are:

Table 3.7 The Timestamp Values

Flags Value	Meaning
0	Record timestamps only.
1	Precede Each timestamp by an IP address
3	IP addresses are sent by the sender.

When fragmenting a datagram, a router replicates some IP options in all fragments while it places others in only one fragment.

3.2 Simulation Analysis using the IP Protocol

In this chapter the packet loss issue is examined, using the network layer protocol IP protocol (without any DLC links between the stations) during the handoff. The format of the datagram or the data packet follows the format of the IP protocol. In the data field, the length of the data field was given instead of the actual data itself. A routing table is maintained to update the current base station. The strict source route option is used in the option field. The information on the current base station is found from the option field of the strict source option. The field IP address of the first hop provides this information.

The packet that was read is sent, with a creation failure rate to the respective current base station. Since IP protocol, without any DLC links, is not a protocol that supports ARQ mechanism the packet might get lost in transit. If the transmission is unsuccessful, the next packet is fetched and the procedure continues. If the packet arrives successfully at the base station, the packet is forwarded to the mobile with creation failure rate. The packets that arrived successfully reach the destination, the mobile. The mobile keeps changing cells with the fixed probability. And once the current base station, b_i , is changed, it needs to go through the handoff procedure. At this point, the mobile receives packets from the new base station. The handoff procedure forces the old base station, b_i , to forward the packet to the new base station b_k. The old base station, b_i takes up the job of processing the next packet received from the gateway and sends it to the new base station, b_k . The routing table entry is changed. If the handoff is successful, the procedure continues as before. If not, the packet is lost and the next packet is fetched. In that case, only the transport layer can fix the problem of packet-loss.

The fragmentation is done as described by the IP protocol. If one of the fragments is lost during the transmission, the entire packet has to be retransmitted. This leads to the retransmission of all the fragments all over again. This fact shows that there is no data integrity using this protocol. The results generated for a set of packets sent through the network, show that some packets are missed either because of the lack of ARQ mechanism or because of handoff procedure. And the results shown in the table, show the packet loss information. The packet loss cannot be prevented with this best-effort packet delivery connectionless service. The results are shown in Table 5.

Table 3.8 The Packet Loss

Number of Packets transmitted	The Probability of mobility	The Failure Rate for the ACK	Number of packets lost
50	0.1	0.1	14
50	0.25	0.1	21
50	0.5	0.1	36
50	0.75	0.1	44
50	0.1	0.25	26
50	0.25	0.25	24
50	0.5	0.25	38
50	0.75	0.25	46
50	0.1	0.5	43
50	0.25	0.5	44
50	0.5	0.5	46
50	0.75	0.5	48
50	0.1	0.75	45
50	0.25	0.75	48
50	0.5	0.75	48
50	0.75	0.75	49

CHAPTER 4

THE DATA INTEGRATION CONCEPT

4.1 The IEEE 802.11 Protocol

In order to prevent the packet loss, Cohen et al [4] suggested the use of DLC protocol, as all DLC protocol work with the concept of ARQ mechanism. Since the network considered here is a wireless LAN, the IEEE 802.11 is used in the simulation of this protocol. Since IEEE 802.11 is still in draft standard, a brief and clear understanding of this protocol is given in the following section [16].

IEEE 802.11 is a basic access protocol, Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA).

4.1.1 The Architecture

The 802.11 WLAN consists of two basic components. The interaction of these components defines the WLAN environment. The first component is a station (STA) defined as any device containing an 802.11 compliant Medium Access Control and Physical (PHY) layer interface. The second component is the Wireless Medium (WM), the medium used to implement a Wireless LAN. A Basic Service Set (BSS) is formed by connecting 2 or more stations over the Wireless Medium. The Coordinate Function determines when a station in BSS transmits, receives or senses the medium. The set of services that support transport of MAC Service Data Units (MSDU) between stations within a BSS is called the Station Services (SS).

The BSS can be used as an independent BSS. In this case, the stations in one BSS can communicate with each other but not with the stations of another BSS. The area in which members of a BSS can communicate is called the Basic Service Area. (BSA).

The Extended Service Set (ESS) is essentially a number of connected BSSs. The different BSSs in ESS are connected via Access Point(AP) and distribution system. The AP is a station that can offer Distribution System Services (DSS) in addition to the station Services. This set of services enables the transport of MSDUs between BSSs within an ESS. The area in which the members of the ESS can communicate is called the Extended Service Area (ESA). The connection of the DS with another type of LAN is achieved by a portal. A Portal is gateway or in the case of 802.x secondly LAN a bridge.

4.1.2 Protocol Reference Model

An entity in layer N provides services that can be used by layer N+1. N is the service provider and the N+1 is the service user. In the case of 802.11 standard, the MAC sub-layer provides services to the LLC sub-layer. The Service Access Point (SAP) are the points where layer N services are accessible for layer N+1. The layer N+1 entities exchange Service Data Units (SDU) whereas the layer N entities exchange Protocol Data Unit (PDU). For layer N to send a PDU, layer N+1 sends Interface Data Unit (IDU) to layer N. IDU consists of SDU and Interface Control Information (ICI) field. Once it receives the IDU, the layer N forms a PDU by adding a header to the SDU and sends PDU to the peer layer N entity. This entity passes the SDU upto layer N+1. The standards

describes the services and protocols of the Medium Access Control sublayer of the datalink layer and the Physical layer.

4.1.3 Mobility Types

The 3 types of mobility types are No-Transition, BSS Transition, ESS-transition. No-Transition is either static i.e no motion or local movement within a BSA. BSS-transition is mobility in the case of BSS from one ESS to BSS from within the same ESS . ESS-Transition is the mobility of BSS from one ESS to BSS from another ESS.

4.1.4 Services

There are 7 services specified by this standard, they are association, reassociation, disassociation, distribution, Integration.

Association establishes an initial association between a Station and an Access Point. A station can be associated with only one AP. But the AP can be associated with many stations at the same time. The association can be only initiated by the Station.

Reassociation enables the transition of a station from one AP to another AP within an ESS. This service tells which Station can be reached via which Station. This service can also be initiated by a Station.

Disassociation removes the association. And the attempt to send messages to a disassociated station will be unsuccessful. Theses services are invoked by either the Station or by the AP.

Distribution delivers MSDUs within the DS to the APs of the final destination by using the association information.

Integration enables the delivery of MSDUs between the DS and an existing Network.

Authentication service provides various levels of protection.

Privacy is used to prevent the contents of messages from being read by unintended recipients.

4.1.5 Frame Formats

The Frames have 1) a MAC header which includes the control, address information, sequencing, fragmentation identification and duration information, 2) a Frame Body that contains frame type information 3) IEEE 32-bit CRC

The frame format and fixed field layout is given in Table 6.

Table 4.1 MAC Format (in bytes)

2	6	6	6	6	2	6	0-2304	4
Frame Control	Duration/ Conn. ID	Add 1	Add. 2	Add. 3	Sequence Control	Add. 4	Frame Body	CRC

Frame Control Field:

It has the sub-fields as shown in Table 6.1

Table 4.2 Frame Control Format in bits

2	2	4	1	1	1	1	1	2	1
Protocol Version	Type	Subtype	To DS	From DS	More	Retry	Power Mngmnt.	EP	Rsvd

Protocol version: This 2-bit field is invariant in size and placement across all revisions of the 802.11 standard. The values are assigned sequentially staring with zero.

Type and Subtype: These fields identify the function and interpretation of a frame. The 3 types are control, data and management,

To DS, the 1-bit field is used to specify whether a frame is supposed to stay within the current BSS or has to enter the Distribution System It is et to 1, when the frame is data frame and is entering the Distribution System. It is set to zero otherwise.

From DS is set 1 when the frame is a Data frame and is exiting the Distribution System. Otherwise it is zero. The following table gives the description of the different To/From subfield combinations.

More, the 1-bit field indicates that the AP holds additional frames buffered for the station identified by the destination address of the frame. This bit may only be set by an AP, a station must always transmit this bit as zero.

The 1-bit **Retry** field indicates whether the its a retransmission or not.

Power Management, the 2-bit field shows the state of the power management after the completion of transmission of the frame. The values are as follows

Elements Present, the 1-bit field , if set 1, the frame body contains 1 or more elements.

Duration or Connection ID, the 6-byte field in the frame control contains the value to update the Network Allocation Vector during the Contention period, when the Distributed Co-ordinate function is present. It contains a connection Identifier during the contention free period.

Address Fields: The address fields can be either an individual address which is associated with a particular station on the network or a group address a multi-destination address associated with more stations in a network like multi-cast address, broadcast address. These address fields can be of one of the following types.

BSS identifier, which uniquely identifies each BSS.

Destination Address field identifies the end system for which the frame is intended.

Source Address field identifies the station from which the frame was initiated.

Receiver Address: identifies the address of the intended recipient of a wireless transmission.

Transmitter Address: field identifies the address of the transmitter of a wireless transmission.

Sequence Control: The 16-bit filed contains 2 subfields, the 12-bit Dialog Token and the 4-bit Fragment Number. The dialog Token subfield contains an incrementing value. The value is increased by one for the initial transmission of an MSDU. The same value must be used for all fragments of the same MSDU in case of fragmentation. The value is not increased for retransmission of the same MSDU or its fragments. It basically

represents the sequence number. The Fragment number is for numbering the fragments. This number does not change in the case of retransmission.

Duration the 16-bit field is used to broadcast a value that updates the Network Allocation Vector(NAV).

Frame Body: This is a variable length field that varies between zero and 2304 bytes. This is were the data is carried.

CRC: The CRC is a 4-byte field which covers the MAC header and data. The encoding is defined by generating the polynomial,

The Control Frames are RTS Frame Format, CTS Frame Format, ACK Frame Format, Poll Frame Format. The RTS Frame Format is the Request To send Frame Format. The Frame format looks as below

Table 4.3 RTS Frame Format

2	6	6	6	4
Frame Control	Duration/ Conn. ID	Dest. Add	Source Add.	CRC

And the **CTS Frame** Format is the Clear to Send Frame format. Once a station receives a request to send (RTS) from another station, this station sends a Clear To Send (STS) Frame as reply. And has the same frame format as the **RTS Frame** Format except there is no source address field. The **ACK Frame** Format has the same format as the CTS frame

format. And this frame is sent as a response to receiving any frame successfully. And the format of the poll frame format is the pretty much the same as RTS frame format except that the BSS ID is used in place of the Dest Address.

Data Frame: The format is base equivalent to the MAC frame format. The address fields of the data frame is dependent upon the values of the To DS and From DS. The address field contents of a data frame are given below.

The Management Frame has a similar format as the general frame format in table 7. And the management frames are Beacon Frame Format, Disassociation frame format, Association request frame format, Association Response Frame Format, Resassociation request frame format, Resassociation Response Frame Format, Probe Request Frame Format, Probe Response Frame Format, Privacy Request Frame Format, Privacy Response Frame Format, Authentication Frame Format

4.2 Simulation Analysis using IEEE 802.11 Protocol

After the brief discussion of the data frame formats, the understanding of the simulation of the proposed new protocol to prevent packet loss becomes easier. Since the proposed new protocol assumes the pre-existence of DLC links between the stations, the protocol used in this case is the DLC protocol IEEE 802.11 and the data frame format follows that of the IEEE 802.11 standard. The data packets are forwarded to the current base station. In this case, since 802.11 is a protocol with ARQ mechanism, a packet that is not received by the base station is reported back to the gateway and the packet is sent out to

the base station again. The procedure repeats until the transmission is successful. When the current base station receives the packet, it is forwarded to the mobile. If the mobile changes cells and thus requires the current base station to be changed. The mobile has to receive packets from the new current base station. The old base station hands off the packets that were handled by it to the new base station. This handoff procedure is different from the one discussed in Chapter 3. When the next packet is fetched from the file, the packet uses the updated routing table and uses the new current base station for the transmission of the packets. This procedure makes sure that the packets are not lost in transit because of the way in which this protocol is designed. And hence preserving the data integrity. This simulation is essentially to show the integrity of the data with this protocol when compared with other protocols. The data integrity is an essential issue as this reduces the throughput performance of the transport layer and also burdens the gateway. The results of this simulations in Table 7 show zero packet loss for the same fixed probabilities for the mobility and the fixed failure rates for the acknowledgment as in Table 5. It can be noted that all the packets (including the fragmented packets) sent over the regional network, reach the destination even in the presence of handover.

Table 4.4 Data Integrity

Number of Packets transmitted	The Probability of mobility	The Failure Rate for the ACK	Number of packets lost
50	0.1	0.1	0
50	0.25	0.1	0
50	0.5	0.1	0
50	0.75	0.1	0
50	0.1	0.25	0
50	0.25	0.25	0
50	0.5	0.25	0
50	0.75	0.25	0
50	0.1	0.5	0
50	0.25	0.5	0
50	0.5	0.5	0
50	0.75	0.5	0
50	0.1	0.75	0
50	0.25	0.75	0
50	0.5	0.75	0
50	0.75	0.75	0

CHAPTER 5

THE BURDEN ON THE GATEWAY IN CASE OF PACKET LOSS

5.1 The Markov Model

To illustrate the effect of prevention of packet loss on the gateway is illustrated by developing a Markov as in Figure1. The main consideration here is that the packet loss due to any other transmission errors are overlooked and assumed not present. The only case of packet loss considered are those due to the handoff procedure. The packets that are lost during the handoff should be retransmitted over the network to the mobile which creates an extra burden on the gateway of processing the packet again. On the other hand using the proposed new protocol, the packet loss due to handoff is guaranteed to be prevented and the gateway's burden is reduced as it does not have to reprocess the packets.

The model lends itself to solution through Markovian processes owing to its unit-step characteristics. The restrictions governing this solution are: one line, one server, independent exponential and probabilistic arrivals, first-come first served discipline.

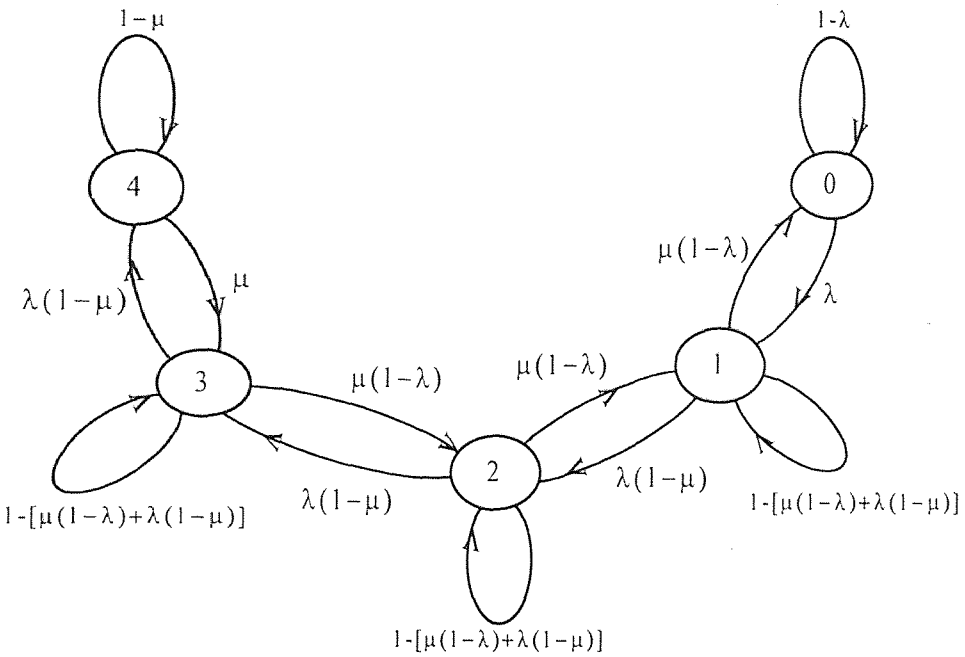


Figure 5.1 The Transition Diagram

The states stand for the number of packets that are in the queue, which in this case has a limited memory of 4 packets. If one of the packets fails to reach the destination because of handoff, the packet has to be requeued into the gateway. This is shown in the Figure 5.1 as a loop back to the same state as there are no new packets arriving or departing. When a packet is serviced successfully, the state changes from i to i-1. In the same way when a packet is received the state changes from i to i+1. In the set-up of the problem, the values assigned to the problem are given as below:

λ = The Arrival Rate expressed as a probability

μ_1 = The Probability of Success

μ_2 = The Departure Rate expressed as a probability

$\mu = \mu_1 * \mu_2$ = The Departure Rate with the Probability of Success

$a = 1 - \lambda$ = No packets (i.e No Arrival)

$b = 1 - \mu$ = No Transmission - Packet Miss (Handoff)

$c = \lambda (1 - \mu)$ = Arrival But No Transmission

$d = \mu (1 - \lambda)$ = Service But No Arrival

$e = 1 - [\mu (1 - \lambda) + \lambda (1 - \mu)]$

= Packet Miss (Handoff)

Here μ_1, is 1 for the proposed new protocol and is between 0 and 1 for the IP protocol. Suppose that the gateway had two packets. From the diagram it can be seen that there are three paths depicting what may occur during the next unit time. One path goes back to one packet, the other loops two and thus remains the same and the last goes to three which indicates that the number of packets is increased by one. The probability of various events are along these paths. For instance, going from two to three in one step has a probability of $\lambda (1 - \mu)$. Going from two to three in two steps would be, $P^2_{2,3}$ [(2-2-3) V (2-3-3)] = $1 - [\mu (1 - \lambda) + \lambda (1 - \mu)] * \lambda (1 - \mu)] + \lambda (1 - \mu)] * 1 - [\mu (1 - \lambda) + \lambda (1 - \mu)]$, where (2-2-3) is the joint probability of the individual events going from state two to state two and state two to state three. (2-3-3) is the joint probability of the individual events going from state two to state three and state three to state three. These two are mutually exclusive events. And the figure is disjointed at state 4 as there is no way that four packets getting reduced to 0 packet in one single step.

The transition matrix for the transition diagram Figure 5.1 is given as follows

$$P = \begin{pmatrix} a & \lambda & 0 & 0 & 0 \\ d & e & c & 0 & 0 \\ 0 & d & e & c & 0 \\ 0 & 0 & d & e & c \\ 0 & 0 & 0 & \mu & b \end{pmatrix}$$

The transition matrix contains the same probabilities as the transition diagram. If the gateway has three packets the probability of having two packets after one transmission is $P^1_{3,2} = \mu (1- \lambda)$. Here the starting state are the rows and the finishing states are the columns. The steady state of the matrix P^n is found, where all the entries of a row j equals the that of any row k.

5.2 The Model Simulation

For the simulation various values for λ and μ are chosen and used in the transition matrix P. With this matrix, the steady state power n is calculated. Once it reaches the steady state, column i, of any row gives the probability P_i of being in state i. i.e. it gives the probability of having i packets in the queue. The expected value of the average number of packets E[k] is found out using $\sum kP_k$. Here $0 \leq k \leq Q$ where Q is the maximum number of packets (i.e. the memory size of the gateway). The parameter E[k] explains the burden on the gateway. The simulation was done for a number of packets between 4 and 20. Figure 2 shows the burden on the gateway when λ=0.3 and μ=0.45. It can be noted from the Figure 2. that when there is no packet loss i.e when the probability of success is 1(or

converging to 1), E[k] does not increase much even if the number of packets are increased. As the probability of success converge to zero, the burden on the gateway continues to increase as the number of packets grows. Results are shown in the following Figures.

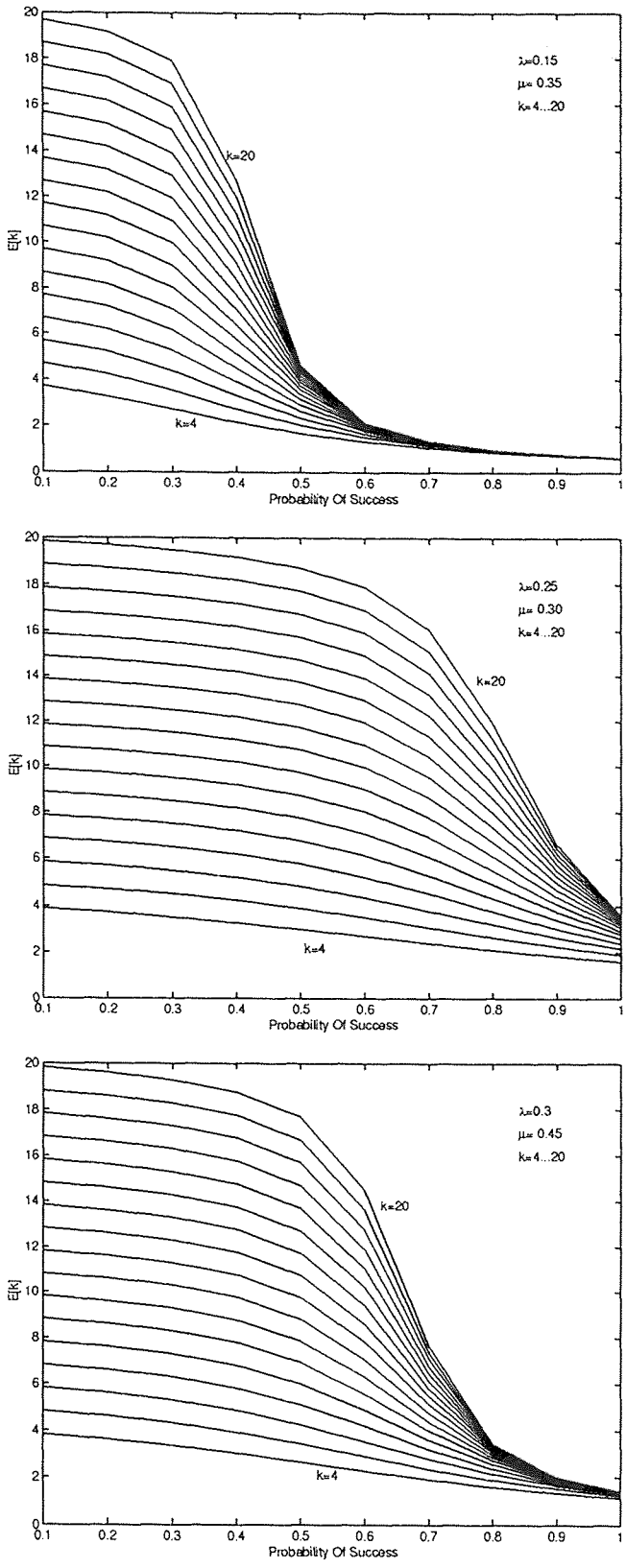


Figure 5.2: Effect of μ and λ on the gateway's burden.

5.3 The Results and Discussions

Though it is a well known fact that the probability of success being 1 is hardly possible, it is assumed that the packet loss considered here is only due to handoff with no transmission errors. The dimension of the transition matrices are 4 X 4 through 20 X 20 for the packet sizes 4 through 20. The graphs show that when the probability of success tends towards one, the value of E[k] does not change much.. But when the probability of success tends towards zero, E[k] values for larger k is very high when compared with that of smaller values of k. This shows the burden on the gateway, when it involves more number of retransmissions due to handoff. Since the proposed new protocol always has the probability as 1 (assuming that there is no other loss due to transmission error) the behavior of this protocol even for the gateway with larger number of packets is falls within an acceptable range. IP like protocol without any ARQ mechanism and with less reliable handoff procedure, tends to have a greater probability of packet loss, leading to burden on the gateway.

In conclusion to this chapter, it can be noted that the behavior of the IP protocol tends to be like the proposed new Protocol as the Probability of Success μ_1 tends to 1. The E[k] value is close and comparable with the proposed new protocol case when μ_1 is closer 1. The drastic increase in E[k] value can be noted as μ_1 is closer to zero. The increase in the expected value shows the burden of the gateway when there are more retransmissions.

CHAPTER 6

CONCLUSION

The Simulations were done for two cases where the gateway g sends packets to the mobile m. In the first case the protocol used is the IP protocol (assuming that there wasn't any DLC links between the stations) where there is no ARQ mechanism. And in the other case, the proposed new protocol is used. The former case showed the disintegration of the data packets and the latter case showed the preservation of the data packets in the presence of handoff. Since the data integrity is an important issue which affects the throughput performance in the transport layer, the use of this proposed new protocol is an efficient way to handle the handoff problems in the microcells. And this protocol as mentioned earlier is fast and does not involve a long delay as the DLC sessions are already present between the stations. The data forwarding is done between the base stations and the handoff procedure does not involve the gateway for the retransmission of the packets. The only jobs that are handled at the gateway are STOP and RESUME messages. It can be concluded that the proposed new protocol is an efficient and simple solution for handling handovers in a micro-cell packet switched mobile network. The further work on this protocol is to build and simulate the regional network described in this thesis and measure the delay and throughput in the presence of handover.

APPENDIX A SIMULATION PROGRAM I

```
/*************************************************************************
Packet handling with IP protocol in the regional network described.
*************************************************************************/
#include<stdio.h>
#include <stdlib.h>
#include<math.h>
#include<fcntl.h>

int i,count,cnt,hndof,chk,kf;
int mob_base, cur_base;
int frg_data[15],f_oset[15];

FILE *in_file, *ofile, *out_file;

struct inp_rec
        {
         char ver[5];
         char hlen[5];
         char svc[9];
         char ttlngth[17];
         char nwln1[1];
         char identi[17];
         char flag[4];
         char frgmnt[14];
         char nwln2[1];
         char ttl[9];
         char prtcol[9];
         char chksum[17];
         char nwln3[1];
         char src_ip1[9];
         char src_ip2[9];
         char src_ip3[9];
         char src_ip4[9];
         char nwln4[1];
         char dst_ip1[9];
         char dst_ip2[9];
         char dst_ip3[9];
         char dst_ip4[9];
         char nwln5[1];
         char opcopy[2];
         char opclass[3];
```

```
        char opnum[6];
        char oplngth[9];
        char pnter[9];
        char pad[9];
        char data[5];
       char nwln6[1];
      };

struct opt_add
        {
        char ip1[9];
        char ip2[9];
        char ip3[9];
        char ip4[9];
        char nl[1];
         };

struct opt_add ipadd[10];
struct inp_rec ar_pac[15];

int bintodec();          /* the binary to decimal converter */
void processfile();      /* module acting as the gateway */
void resume();           /* procedure which decides the resume operation after handoff */
int fragment();          /* fragments the packet if it has to be! */
int endstation();        /* acts as an endstation for the fragmented packets */
void sendpkt_base1();  /* module acting as the base station 1 */
void sendpkt_base2();  /* module acting as the bas station 2 */
void sendpkt_base3();  /* module acting as the bas station 3 */
void sendpkt_base4();  /* module acting as the bas station 4 */
void sendpkt_base5();  /* module acting as the bas station 5 */
void sendpkt_base6();  /* module acting as the bas station 6 */
void sendpkt_base7();  /* module acting as the bas station 7 */
void sendpkt_base8();  /* module acting as the bas station 8 */
void sendpkt_base9();  /* module acting as the bas station 9 */
void sendpkt_base10(); /* module acting as the bas station 10 */
void sendpkt_mobile(); /* module which decides the handoff */
void routing_table();
int randomize();
void receivepkt(); /* module acts as a mobile */

main()
{
  int j;
  cnt=0;
  hndof=0;
```

```
   in_file=fopen("in_ip.dat","r");
   out_file=fopen("output","w");

   if(in_file==NULL){
     (void)printf("Cannot open %s\n", FILE_NAME);
     exit(8); }
        i = 1;

    ofile=fopen("outf","w");

    processfile();
} /* end main */

void processfile()
{
   int version,hdrlen,service;
   int k,tot_length,identif,flg,fragm_offset;
   int act_data[10],data_val,tm_to_live,protocol,checksum;
   int src_add1, src_add2, src_add3, src_add4;
   int dst_add1, dst_add2, dst_add3, dst_add4;
   int opt_copy,opt_class,opt_numb,opt_length,opt_pnter,padding;
   int num_ips,mtu, ii;
   int dest1[15],dest2[15],dest3[15],dest4[15];
   struct inp_rec pkt;
   char *fr_dt, nullp;

   fr_dt = (char *) malloc(sizeof(char)*10);

 if(nullp=fgets(pkt.ver,5,in_file)!=NULL)
  {
    pkt.ver[4]='\0';
    fgets(pkt.hlen,5,in_file);
    pkt.hlen[4]='\0';
    fgets(pkt.svc,9,in_file);
    pkt.svc[8]='\0';
    fgets(pkt.ttlngth,17,in_file);
    pkt.ttlngth[16]='\0';
    fgets(pkt.nwln1,2,in_file);
    fgets(pkt.identi,17,in_file);
    pkt.identi[16]='\0';
    fgets(pkt.flag,4,in_file);
    pkt.flag[3]='\0';
    fgets(pkt.frgmnt,14,in_file);
    pkt.frgmnt[13]='\0';
```

```
        fgets(pkt.nwln2,2,in_file);
        fgets(pkt.ttl,9,in_file);
        pkt.ttl[8]='\0';
        fgets(pkt.prtcol,9,in_file);
        pkt.prtcol[8]='\0';
        fgets(pkt.chksum,17,in_file);
        pkt.chksum[16]='\0';
        fgets(pkt.nwln3,2,in_file);
        fgets(pkt.src_ip1,9,in_file);
        pkt.src_ip1[8]='\0';
        fgets(pkt.src_ip2,9,in_file);
        pkt.src_ip2[8]='\0';
        fgets(pkt.src_ip3,9,in_file);
        pkt.src_ip3[8]='\0';
        fgets(pkt.src_ip4,9,in_file);
        pkt.src_ip4[8]='\0';
        fgets(pkt.nwln4,2,in_file);
        fgets(pkt.dst_ip1,9,in_file);
        pkt.dst_ip1[8]='\0';
        fgets(pkt.dst_ip2,9,in_file);
        pkt.dst_ip2[8]='\0';
        fgets(pkt.dst_ip3,9,in_file);
        pkt.dst_ip3[8]='\0';
        fgets(pkt.dst_ip4,9,in_file);
        pkt.dst_ip4[8]='\0';
        fgets(pkt.nwln5,2,in_file);
        cnt +=1;
    }
else exit(0);

    version = bintodec(pkt.ver);
    hdrlen  = bintodec(pkt.hlen);
    service = bintodec(pkt.svc);
    tot_length = bintodec(pkt.ttlngth);
    identif=bintodec(pkt.identi);
    flg = bintodec(pkt.flag);
    fragm_offset=bintodec(pkt.frgmnt);
    tm_to_live = bintodec(pkt.ttl);
    protocol = bintodec(pkt.prtcol);
    checksum  = bintodec(pkt.chksum);
    src_add1 = bintodec(pkt.src_ip1);
    src_add2 = bintodec(pkt.src_ip2);
    src_add3 = bintodec(pkt.src_ip3);
    src_add4 = bintodec(pkt.src_ip4);
    dst_add1 = bintodec(pkt.dst_ip1);
```

```
dst_add2 = bintodec(pkt.dst_ip2);
dst_add3 = bintodec(pkt.dst_ip3);
dst_add4 = bintodec(pkt.dst_ip4);



if(version!=4)
 fprintf(ofile,"It's not the Current Version\n");

if(hdrlen<5)
 fprintf(ofile,"Headerlength too small\n");
if(hdrlen==5)
 {
 fscanf(in_file,"%5s",pkt.data);
 fgets(pkt.nwln6,2,in_file);
 cur_base = 1;
 mob_base = 1;
     }

if((service==1) || (service==2))
 fprintf(ofile,"Unused bits are set\n");

if(flg==4)
 fprintf(ofile,"Unused bit is set\n");

 if(hdrlen>5)
{
 {
 fgets(pkt.opcopy,2,in_file);
 pkt.opcopy[1] = '\0';
 fgets(pkt.opclass,3,in_file);
 pkt.opclass[2] = '\0';
 fgets(pkt.opnum,6,in_file);
 pkt.opnum[5] = '\0';
 fgets(pkt.oplngth,9,in_file);
 pkt.oplngth[8] ='\0';
 fgets(pkt.pnter,9,in_file);
 pkt.pnter[8] ='\0';
 }
     opt_copy   = bintodec(pkt.opcopy);
     opt_class  = bintodec(pkt.opclass);
     opt_numb   = bintodec(pkt.opnum);
     opt_length = bintodec(pkt.oplngth);
     opt_pnter  = bintodec(pkt.pnter);
```

```
    padding   = bintodec(pkt.pad);

  num_ips = (opt_length-3)/4;
  for(k=0; k<num_ips; k++)
  {
        fgets(ipadd[k].ip1,9,in_file);
        ipadd[k].ip1[8] ='\0';
        fgets(ipadd[k].nl,2,in_file);
        fgets(ipadd[k].ip2,9,in_file);
        ipadd[k].ip2[8] = '\0';
        fgets(ipadd[k].ip3,9,in_file);
        ipadd[k].ip3[8] ='\0';
        fgets(ipadd[k].ip4,9,in_file);
        ipadd[k].ip4[8] = '\0';
        dest1[k] =  bintodec(ipadd[k].ip1);
        dest2[k] = bintodec(ipadd[k].ip2);
        dest3[k] = bintodec(ipadd[k].ip3);
        dest4[k] = bintodec(ipadd[k].ip4);
        cur_base = dest4[k];
        mob_base = dest4[k];
    }

  fgets(pkt.pad,9,in_file);
  pkt.pad[8]  = '\0';
  fgets(pkt.nwln6,2,in_file);

     fscanf(in_file,"%5s",pkt.data);
 } /* endof if hdrlen<5 */

data_val = atoi(pkt.data);
mtu = 250;

  if(tot_length!=(data_val+(hdrlen*4)))
    fprintf(ofile,"Total_length field is Wrong\n");

     if(tot_length>mtu)
 {   if(flg==2)
  fprintf(ofile,"Cannot Fragment\n");
  else
  count = fragment(hdrlen,data_val,mtu);
  }
     else
      {
       count = 0;
       frg_data[0]=tot_length;
```

```
            f_oset[0]=0;
          }

          kf=0;
          chk = 0;



if(count>0)
for(kf=0; kf<=count; kf++)
          {sprintf(fr_dt,"%d",frg_data[kf]);
          strcpy(pkt.data,fr_dt);
          resume(kf,pkt);}
else
          resume(0,pkt);
          processfile();
      }

void resume(t,pkt)
struct inp_rec pkt;
int t;
{
int res;
routing_table();
if(hndof==1)
{res=randomize(10);
if(res==1)
if(count>0)
processfile();}
hndof=0;
if(cur_base==1)
sendpkt_base1(t,pkt);
else if(cur_base==2)
sendpkt_base2(t,pkt);
else if(cur_base==3)
sendpkt_base3(t,pkt);
else if(cur_base==3)
sendpkt_base3(t,pkt);
else if(cur_base==4)
sendpkt_base4(t,pkt);
else if(cur_base==5)
sendpkt_base5(t,pkt);
else if(cur_base==6)
sendpkt_base6(t,pkt);
else if(cur_base==7)
```

```
sendpkt_base7(t,pkt);
else if(cur_base==8)
sendpkt_base8(t,pkt);
else if(cur_base==9)
sendpkt_base9(t,pkt);
else if(cur_base==10)
sendpkt_base10(t,pkt); }

 void sendpkt_base1(t,pac1)
struct inp_rec pac1;
int t;
{
int success;
success = ack();
if(success==0)
processfile();
else
sendpkt_mobile(t,pac1);
}

void sendpkt_base2(t,pac2)
struct inp_rec pac2;
int t;
{
int success;
success = ack();
if(success==0)
processfile();
else
sendpkt_mobile(t,pac2);
}

void sendpkt_base3(t,pac3)
struct inp_rec pac3;
int t;
{
int success;
success = ack();
if(success==0)
processfile();
else
sendpkt_mobile(t,pac3);
}

void sendpkt_base4(t,pac4)
```

```
struct inp_rec pac4;
int t;
{
int success;
success = ack();
if(success==0)
processfile();
else
sendpkt_mobile(t,pac4);
}

void sendpkt_base5(t,pac5)
struct inp_rec pac5;
int t;
{
int success;
success = ack();
if(success==0)
processfile();
else
sendpkt_mobile(t,pac5);
}

void sendpkt_base6(t,pac6)
struct inp_rec pac6;
int t;
{
int success;
success = ack();
if(success==0)
processfile();
else
sendpkt_mobile(t,pac6);
}

void sendpkt_base7(t,pac7)
struct inp_rec pac7;
int t;
{
int success;
success = ack();
if(success==0)
processfile();
else
sendpkt_mobile(t,pac7);
```

```
}

void sendpkt_base8(t,pac8)
struct inp_rec pac8;
int t;
{
int success;
success = ack();
if(success==0)
processfile();
else
sendpkt_mobile(t,pac8);
}

void sendpkt_base9(t,pac9)
struct inp_rec pac9;
int t;
{
int success;
success = ack();
if(success==0)
processfile();
else
sendpkt_mobile(t,pac9);
}

void sendpkt_base10(t,pac10)
struct inp_rec pac10;
int t;
{
int success;
success = ack();
if(success==0)
processfile();
else
sendpkt_mobile(t,pac10);
}

void sendpkt_mobile(t,packet)
struct inp_rec packet;
int t;
{
int val,tr_fls,l;
val = randomize(2);
if((val==1))
```

```
mob_base=mob_base+1;
if(cur_base==mob_base)
{
 if(count==0)
 {receivepkt(packet);
 hndof = 0;
 processfile();}
 else
  {
   tr_fls=endstation(t,packet);
    if(tr_fls==1)
   { hndof=1; resume(t,packet); }
   if(t==count)
    for(l=0; l<=count; l++)
    receivepkt(ar_pac[l]);
    else ;}
}
else{
 hndof=1;
 resume(0,packet);
 }
}

void routing_table()
{
cur_base=mob_base;
}

int randomize(upb)
int upb;
{
long rv;
int value;
rv=0;
rv = random();
value =0;
value = rv % upb;
return(value);
} /* end of procedure randomize */

int ack()
{
int value1, succ;
value1=randomize(4);
if(value1>0)
```

```
succ = 0;
else succ = 1;
return(succ);
}



void receivepkt(pack)
struct inp_rec pack;
{
fprintf(out_file,"%d PACKET\n",cnt);
fprintf("%d PACKET\n",cnt);
fprintf(out_file,"%s%s%s%s%s\n",pack.ver,pack.hlen,pack.svc,pack.ttlngth);
fprintf(out_file,"%s%s%s%s\n",pack.identi,pack.flag,pack.frgmnt);
fprintf(out_file,"%s%s%s%s\n",pack.ttl,pack.prtcol,pack.chksum);
fprintf(out_file,"%s%s%s%s%s\n",pack.src_ip1,pack.src_ip2,pack.src_ip3,pack.src_ip4);
fprintf(out_file,"%s%s%s%s%s\n",pack.dst_ip1,pack.dst_ip2,pack.dst_ip3,pack.dst_ip4);
fprintf(out_file,"%s%s%s%s%s%s%s\n",pack.opcopy,pack.opclass,pack.opnum,pack.oplngt
h,pack.pnter,pack.pad);
fprintf(out_file,"%s%s%s%s%s\n",ipadd[0].ip1,ipadd[0].ip2,ipadd[0].ip3,cur_base);
fprintf(out_file,"%s\n",pack.data);
} /* end of procedure receivepkt */

int bintodec(ps)
char *ps;
    {
     int vers,len,I,tmp1,decval;
     double tmp=0.0;

     len = strlen(ps);

     for(I=len-1; I>=0; I--)
     {
        tmp1=ps[I]-48;
     if(tmp1==1)
        tmp = pow(2.0,(len-(I+1.0)))+tmp;
     }
     decval = (int)(tmp);
     return(decval);
    }


int fragment(hl,data,maxtr)
int hl,data,maxtr;
   {  int ii,val1,val2;
```

```
        ii=0;
        hl = hl*4;
        f_oset[0]=0;
        while(data>maxtr)
        {
        val1=maxtr-hl;
        val2 = val1/8;
        f_oset[i+1] = val2+f_oset[i];
        val1 = 8*val2;
        frg_data[ii] = val1+hl;
        data = data-val1;
        ii +=1;
        }
        frg_data[i] = data+hl;
    return(i);
  }

endstation(k,pack)
struct inp_rec pack;
int k;
{
int flag;
if(k==chk)
      { flag=0;
        strcpy(ar_pac[k].ver,pack.ver);
        strcpy(ar_pac[k].hlen,pack.hlen);
        strcpy(ar_pac[k].svc,pack.svc);
        strcpy(ar_pac[k].ttlngth,pack.svc);
        strcpy(ar_pac[k].identi,pack.identi);
        strcpy(ar_pac[k].flag,pack.flag);
        strcpy(ar_pac[k].frgmnt,pack.frgmnt);
        strcpy(ar_pac[k].ttl,pack.ttl);
        strcpy(ar_pac[k].prtcol,pack.prtcol);
        strcpy(ar_pac[k].chksum,pack.chksum);
        strcpy(ar_pac[k].src_ip1,pack.src_ip1);
        strcpy(ar_pac[k].src_ip2,pack.src_ip2);
        strcpy(ar_pac[k].src_ip3,pack.src_ip3);
        strcpy(ar_pac[k].src_ip4,pack.src_ip4);
        strcpy(ar_pac[k].dst_ip1,pack.dst_ip1);
        strcpy(ar_pac[k].dst_ip2,pack.dst_ip2);
        strcpy(ar_pac[k].dst_ip3,pack.dst_ip3);
        strcpy(ar_pac[k].dst_ip4,pack.dst_ip4);
        strcpy(ar_pac[k].opcopy,pack.opcopy);
        strcpy(ar_pac[k].opclass,pack.opclass);
        strcpy(ar_pac[k].opnum,pack.opnum);
```

```
        strcpy(ar_pac[k].oplngth,pack.oplngth);
        strcpy(ar_pac[k].pnter,pack.pnter);
        strcpy(ar_pac[k].pad,pack.pad);
        strcpy(ar_pac[k].data,pack.data);
        chk++; }
         else  flag=1;
return(flag); }
```

APPENDIX B SIMULATION PROGRAM II

```
/*************************************************************************
*  Packet handling with the proposed new protocol for the regional network described.
*   To compile this program : cc -o hnd1 hndof_new.c  -lm
*   To Run this program     : hnd1 <input_file> <output_file>
*
*************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include<math.h>
#include<fcntl.h>
#include<malloc.h>

int b[15],i, last_bs, cur_base,mob_base,count;
FILE *in_file,*out_file;

 struct inp_rec
        { char ver[3];
          char tpe[3];
          char sbtp[5];
          char tods[2];
          char frds[2];
          char mre[2];
          char rtry[2];
          char pwrmng[3];
          char ep[2];
          char rsvd[2];
          char nwln1[1];
          char cnnid[49];
          char nwln2[1];
          char add1[49];
          char nwln3[1];
          char add2[49];
          char nwln4[1];
          char add3[49];
          char nwln5[1];
          char dlgtkn[13];
          char frgnum[5];
          char nwln6[1];
          char add4[49];
          char nwln7[1];
```

```
        char frmbdy[4];
        char nwln8[1];
        char crc[33];
        char nwln9[1]; };

int bintodec();          /* converts  binary to decimal */
int fragment();          /* fragments the packet if it has to be! */
struct inp_rec readfile();/* module that does the gateway's jobs */
void resume();           /* procedure which decides the resume operation after handoff */
void sendpkt_base1();  /* module acting as the base station 1 */
void sendpkt_base2();  /* module acting as the bas station 2 */
void sendpkt_base3();  /* module acting as the bas station 3 */
void sendpkt_base4();  /* module acting as the bas station 4 */
void sendpkt_base5();  /* module acting as the bas station 5 */
void sendpkt_base6();  /* module acting as the bas station 6 */
void sendpkt_base7();  /* module acting as the bas station 7 */
void sendpkt_base8();  /* module acting as the bas station 8 */
void sendpkt_base9();  /* module acting as the bas station 9 */
void sendpkt_base10(); /* module acting as the bas station 10 */
void sendpkt_mobile(); /* module which decides the handoff */
void join();
void leave();
void stop();
void last();
void routing_table();
int randomize();
int ack();
void dropkt_base();
void receivepkt();

int mtu,frg_data[10],f_oset[10];
main()
{
int
j,version,type,subtype,dis_syst_2,dis_syst_fr,more,retry,powermn,ele_p,reserved,conn_id
,bss_add,dest_add,src_add,rece_add,dial_tkn,flg_no,chk_sum,data_val;
int kf,frg_cnt;
char *fr_dt;

   struct inp_rec pkt;

   in_file = fopen("inpnew.dat","r");
   out_file=fopen("oput","w");

   if(in_file==NULL){
```

```
        (void)printf("Cannot open %s\n", FILE_NAME);
        exit(8);}


            i=1;
            count=0;

            fr_dt = (char *) malloc(sizeof(char)*10);

lab:        pkt = readfile();
            count++;

            version =bintodec(pkt.ver);
            type    =bintodec(pkt.tpe);
            subtype =bintodec(pkt.sbtp);
            dis_syst_2=bintodec(pkt.tods);
            dis_syst_fr=bintodec(pkt.frds);
            more    = bintodec(pkt.mre);
            retry   = bintodec(pkt.rtry);
            powermn  = bintodec(pkt.pwrmng);
            ele_p   = bintodec(pkt.ep);
            reserved = bintodec(pkt.rsvd);
            conn_id  = bintodec(pkt.cnnid);
            bss_add  = bintodec(pkt.add1);
            cur_base=bss_add;
            mob_base = bss_add;
            dest_add = bintodec(pkt.add2);
            src_add  = bintodec(pkt.add3);
            rece_add = bintodec(pkt.add4);
            dial_tkn = bintodec(pkt.dlgtkn);
            flg_no   = bintodec(pkt.frgnum);
            data_val = atoi(pkt.frmbdy);
            flg_no   = bintodec(pkt.frgnum);
            chk_sum  = bintodec(pkt.crc);

routing_table();
if(data_val>256)
frg_cnt = fragment(34,data_val,256);
if(frg_cnt>1)
for(kf=0; kf<=frg_cnt; kf++)
{sprintf(fr_dt,"%d",frg_data[kf]);
 printf("%d\n",frg_data[kf]);
 strcpy(pkt.frmbdy,fr_dt);
 resume(pkt);}
else
```

```
 resume(pkt);
goto lab;
}

struct inp_rec readfile()
{
  char nullp;
  struct inp_rec pktn;

  if(nullp=fgets(pktn.ver,3,in_file)!=NULL)
  {
    pktn.ver[2]='\0';
    fgets(pktn.tpe,3,in_file);
    pktn.tpe[2]='\0';
    fgets(pktn.sbtp,5,in_file);
    pktn.sbtp[4]='\0';
    fgets(pktn.tods,2,in_file);
    pktn.tods[1]='\0';
    fgets(pktn.frds,2,in_file);
    pktn.tods[1]='\0';
    fgets(pktn.mre,2,in_file);
    pktn.mre[1]='\0';
    fgets(pktn.rtry,2,in_file);
    pktn.rtry[1]='\0';
    fgets(pktn.pwrmng,3,in_file);
    pktn.pwrmng[2]='\0';
    fgets(pktn.ep,2,in_file);
    pktn.ep[1]='\0';
    fgets(pktn.rsvd,2,in_file);
    pktn.rsvd[1]='\0';
    fgets(pktn.nwln1,2,in_file);
    fgets(pktn.cnnid,49,in_file);
    pktn.cnnid[48]='\0';
    fgets(pktn.nwln2,2,in_file);
    fgets(pktn.add1,49,in_file);
    pktn.add1[48]='\0';
    fgets(pktn.nwln3,2,in_file);
    fgets(pktn.add2,49,in_file);
    pktn.add2[48]='\0';
    fgets(pktn.nwln4,2,in_file);
    fgets(pktn.add3,49,in_file);
    pktn.add3[48]='\0';
    fgets(pktn.nwln5,2,in_file);
    fgets(pktn.dlgtkn,13,in_file);
    pktn.dlgtkn[12]='\0';
```

```
    fgets(pktn.frgnum,5,in_file);
    pktn.frgnum[4]='\0';
    fgets(pktn.nwln6,2,in_file);
    fgets(pktn.add4,49,in_file);
    pktn.add4[48]='\0';
    fgets(pktn.nwln7,2,in_file);
    fgets(pktn.frmbdy,4,in_file);
    pktn.frmbdy[3]='\0';
    fgets(pktn.nwln8,2,in_file);
    fgets(pktn.crc,33,in_file);
    pktn.crc[32]='\0';
    fgets(pktn.nwln9,2,in_file);
  } else exit(0);
return(pktn); }

void resume(pktr)
struct inp_rec pktr;
{ if(cur_base==2147483647)
sendpkt_base1(0,pktr);
else if(cur_base==2147483646)
sendpkt_base2(0,pktr);
else if(cur_base==2147483645)
sendpkt_base3(0,pktr);
else if(cur_base==2147483644)
sendpkt_base3(0,pktr);
else if(cur_base==2147483643)
sendpkt_base4(0,pktr);
else if(cur_base==2147483642)
sendpkt_base5(0,pktr);
else if(cur_base==2147483641)
sendpkt_base6(0,pktr);
else if(cur_base==2147483640)
sendpkt_base7(0,pktr);
else if(cur_base==2147483639)
sendpkt_base8(0,pktr);
else if(cur_base==2147483638)
sendpkt_base9(0,pktr);
else if(cur_base==2147483637)
sendpkt_base10(0,pktr);  }

void sendpkt_base1(jn1,pac1)
struct inp_rec pac1;
int jn1;
{
int success;
```

```
success = ack();
if(success==0)
resume(pac1);
else
if(jn1==1)
leave(pac1);
else  sendpkt_mobile(pac1);
}

void sendpkt_base2(jn1,pac2)
struct inp_rec pac2;
int jn1;
{
int success;
success = ack();
if(success==0)
resume(pac2);
else
if(jn1)
leave(pac2);
else  sendpkt_mobile(pac2);
}

void sendpkt_base3(jn1,pac3)
struct inp_rec pac3;
int jn1;
{
int success;
success = ack();
if(success==0)
resume(pac3);
else
if(jn1)
leave(pac3);
else sendpkt_mobile(pac3);
}

void sendpkt_base4(jn1,pac4)
struct inp_rec pac4;
int jn1;
{
int success;
success = ack();
if(success==0)
resume(pac4);
```

```
else
if(jn1)
leave(pac4);
else sendpkt_mobile(pac4);
}

void sendpkt_base5(jn1,pac5)
struct inp_rec pac5;
int jn1;
{
int success;
success = ack();
if(success==0)
resume(pac5);
else
if(jn1)
leave(pac5);
else sendpkt_mobile(pac5);
}

void sendpkt_base6(jn1,pac6)
struct inp_rec pac6;
int jn1;
{
int success;
success = ack();
if(success==0)
resume(pac6);
else
if(jn1)
leave(pac6);
else sendpkt_mobile(pac6);
}

void sendpkt_base7(jn1,pac7)
struct inp_rec pac7;
int jn1;
{
int success;
success = ack();
if(success==0)
resume(pac7);
else
if(jn1)
leave(pac7);
```

```
else sendpkt_mobile(pac7);
}


void sendpkt_base8(jn1,pac8)
struct inp_rec pac8;
int jn1;
{
int success;
success = ack();
if(success==0)
resume(pac8);
else
if(jn1)
leave(pac8);
else sendpkt_mobile(pac8);
}

void sendpkt_base9(jn1,pac9)
struct inp_rec pac9;
int jn1;
{
int success;
success = ack();
if(success==0)
resume(pac9);
else
if(jn)
leave(pac9);
else sendpkt_mobile(pac9);
}

void sendpkt_base10(jn1,pac10)
struct inp_rec pac10;
int jn1;
{
int success;
success = ack();
if(success==0)
resume(pac10);
else
if(jn1)
leave(pac10);
else sendpkt_mobile(pac10);
}
```

```
void sendpkt_mobile(packet)
struct inp_rec packet;

{
int success;
int val;

success = ack();
if(success==0)
resume(packet);

else
{
val = randomize(3);
if((val==2)||(val==1))
mob_base=mob_base-1;

if(cur_base==mob_base)
receivepkt(count,packet);
else
join(cur_base,mob_base,packet);
 } /* end else */
}

void join(cur_base,mob_base,pktt)
struct inp_rec pktt;
{
int jn;
if(mob_base==2147483647)
{jn=1; sendpkt_base1(jn,pktt); }
else if(mob_base==2147483646)
{jn=1; sendpkt_base2(jn,pktt); }
else if(mob_base==2147483645)
{jn=1; sendpkt_base3(jn,pktt); }
else if(mob_base==2147483644)
{jn=1; sendpkt_base3(jn,pktt); }
else if(mob_base==2147483643)
{jn=1; sendpkt_base4(jn,pktt); }
else if(mob_base==2147483642)
{jn=1; sendpkt_base5(jn,pktt); }
else if(mob_base==2147483641)
{jn=1; sendpkt_base6(jn,pktt); }
else if(mob_base==2147483640)
{jn=1; sendpkt_base7(jn,pktt); }
```

```
else if(mob_base==2147483639)
{jn=1; sendpkt_base8(jn,pktt); }
else if(mob_base==2147483638)
{jn=1; sendpkt_base9(jn,pktt); }
else if(mob_base==2147483637)
{jn=1; sendpkt_base10(jn,pktt);}
 }

void leave(pktt)
struct inp_rec pktt;
{dropkt_base(pktt); }

void stop()
{last();}

void last()
{routing_table();}

void routing_table()
{cur_base=mob_base;}

int randomize(upb)
int upb;
{
long rv;
int value;
rv=0;
rv = random();
value =0;
value = rv % upb;
return(value);
} /* end of procedure randomize */

int ack()
{
int succ,value;
value = randomize(10);
if(value==5)
succ = 0;
else succ = 1;
return(succ);
}

void dropkt_base(p1)
struct inp_rec p1;
```

```
{stop();
 sendpkt_mobile(p1); }


int fragment(hl,data,maxtr)
int hl,data,maxtr;
    {  int ii,val1,val2;
         ii=0;
         f_oset[0]=0;
         while(data>maxtr)
         {
         val1=maxtr-hl;
         val2 = val1/8;
         f_oset[ii+1] = val2+f_oset[ii];
         val1 = 8*val2;
         frg_data[ii] = val1+hl;
         data = data-val1;
         ii +=1;
         }
         frg_data[ii] = data+hl;
     return(ii);
   }

void receivepkt(count,pack)
struct inp_rec pack;
{
int j;
fprintf(out_file," PACKET %d\n",count);
fprintf(out_file,"%s%s%s%s%s%s%s%s%s%s\n",pack.ver,pack.tpe,pack.sbtp,pack.tods,
pack.frds,pack.mre,pack.rtry,pack.pwrmng,pack.ep,pack.rsvd);
fprintf(out_file,"%s\n",pack.cnnid);
fprintf(out_file,"%s\n",pack.add1);
fprintf(out_file,"%s\n",pack.add2);
fprintf(out_file,"%s\n",pack.add3);
fprintf(out_file,"%s%s\n",pack.dlgtkn,pack.frgnum);
fprintf(out_file,"%9s\n",pack.add4);
fprintf(out_file,"%s\n",pack.frmbdy);
fprintf(out_file,"%s\n",pack.crc);
} /* end of procedure receivepkt */

 int bintodec(pkt_string)
 char *pkt_string;
    {
         int vers,len,I,tmp1,decval;
         double tmp=0.0;
```

```
    len = strlen(pkt_string);

    for(I=len-1; I>=0; I--)
{
    tmp1=pkt_string[I]-48;
    if(tmp1==1)
       tmp = pow(2.0,((double) (len-(I+1))))+tmp;
    }
    decval = (int)(tmp);
    return(decval);
}
```

REFERENCES

1. R. Caceres and L. Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments", *IEEE J. Selected Areas Of Communications*, 13(1995) 850.

2. A. Desimone, M.C. Chuah and O.C Yue, "Throughput Performance of Transport-Layer Protocols over Wireless LANs, *Proc. 1993 GLOBECOM* (1993) 542-549.

3. P. Manzoni, D. Ghoshal and G. Serazzi, "Impact of Mobility on TCP/IP: An Integrated Performance study, *IEEE J. Selected Areas of Communications*, 13(1995)858.

4. R. Cohen, B.V. Patel and A. Segall, "Handover in a Micro-Cell Packet Switched Mobile Network", *Wireless Networks*, 2(1996) 13-25.

5. G. Pollini, K.Meier-Hellstern and D.J Goodman, "Handover Protocols between Metropolitan Area Networks, *Proc. 1992 GLOBECOM*, 1(1992) 11-15.

6. G. Pollini, A Catalog of Handover Algorithms for Cellular Packet Switch. *Tech. Rep. WINLAN-TR-48*, Rutgers University, NJ, 1993.

7. A.S. Acampora and M. Naginesh, "An Architecture and Methodology for Mobile-Executed Handoff in Cellular ATM Networks, *IEEE J. Selected Areas in Communications*, 12 (1994) .

8. W. Richard Stevens, *TCP/IP Illustrated Volume 1-The Protocols*, Addison Wesley Publishing Co., Massachusetts, 1994.

9. A. Tanenbaum, *Computer Networks*, Third Edition, Prentice Hall PTR, Prentice Hall Inc., New Jersey, 1996.

10. P.T. Davis and C.R McGuffin, *Wireless Local Area Networks-Technology, Issues and Strategies*, McGraw Hill Inc., New York, 1995.

11. J. Geier, *Wireless Networking Handbook*, New Riders Publishing, Indianapolis, Indiana, 1996.

12. D. Gross and Carl M. Harris, *The Fundamentals of Queueing Theory*, 2nd Ed., Wiley Series in Probability and Mathematical Statistics, New York, 1985.

13. R.L. Disney and P.C. Kiessler, *Traffic Processes in Queueing Networks-A Markov Renewal Approach*, The Johns Hopkins University Press, London, 1987.

14. N. Zhang and J.M Holtzman, "Analysis of Handoff Algorithms Using Both Absolute and Relative Measurements", *IEEE Transactions on Vehicular Technology*, 45 (1996).

15. J.M Holtzman, A. Sampath, "Adaptive Averaging Methodology for Handoffs in Cellular Systems", *IEEE Transactions on Vehicular Technology*, 44 (1995).

16. M.A. Visser, " Performance Study of the IEEE 802.11 Wireless LAN Standard", *Research Thesis, Department of Electrical Engineering, Delft University of Technology*, June 1995.

17. D. Sarobinski, "New Call Blocking Versus Handoff Blocking in Cellular Networks", *Research Thesis, Department of Electrical Engineering, Israel Institute of Technology*, December 1995.