Protocol Verification in Coexistence Phase of Mobile Internet Protocol Versions 4 and 6

THESIS

Submitted in partial fulfilment
of the requirements for the degree of
DOCTOR OF PHILOSOPHY

by

SUSANNA SAJONI HENRY

Under the Supervision of
Dr. V. Santhosh Kumar

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
2015

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the thesis entitled **“Protocol Verification in Coexistence Phase of Mobile Internet Protocol Versions 4 and 6”** and submitted by **Susanna Sajoni Henry** ID No **2009PHXF430U** for award of Ph.D. Degree of the Institute embodies original work done by her under my supervision.

Signature in full of the Supervisor**:**

Name in capital block letters**: Dr. V. SANTHOSH KUMAR**

Designation**: Assistant Professor, BITS Pilani, Dubai Campus**

Date:

ACKNOWLEDGEMENTS

"What we are is God's gift to us". I am most thankful to almighty for everything given to me.

I wish to express deep sense of gratitude and sincere thanks to respected Director Prof.Dr.R.N.Saha BITS Pilani- Dubai Campus for his valuable guidance, encouragement and moral support to pursue my research.I thank him for providing me with an opportunity of informative 2 weeks stay with Information Science Lab at BITS Hyderabad Campus, Hyderabad.

I would like to express my gratitude to my supervisor, Dr. V. Santhosh Kumar, Assistant Professor, CS Dept, whose expertise, understanding and patience, added considerably to my graduate knowledge. I sincerely thank him for his leadership, availability, scientific and moral guidance during my entire PhD studies. This research would not have been accomplished without his constant support.

I would like to thank the Doctoral Advisory Committee members Prof. S. Vadivel, CS Dept and Prof. B. Vijay Kumar, HOD- CS department for the assistance they provided at all levels of my research project. They provided invaluable supervision, guidance, and inspiration during the past several years. I would also like to thank the DRC members for taking time out to schedule my presentations and seminars.

A very special thanks goes out to my elder brother Thomas Sajo Henry, whose motivation and encouragement made a difference in my life. He provided me with direction, support and became more of a mentor than a brother.

I would like to thank my husband Jobby Zacharia for his love, encouragement and for reminding me at every instant about what is truly important. Special thanks to my elder daughter Anna for her tireless prayers and to my younger daughter Sarah who in her terrible two's did her part to make this even more challenging.

I also thank Mrs. Juliet, my spiritual mother for her prayers that helped me sail through.

I would especially like to thank Dr. R. Swarnalatha, Assistant Professor, EEE Dept, for her untiring show of love, patience and support. I doubt that I will ever be able to convey my appreciation fully, but I owe her my gratitude.

I am sure my late parents are witnessing this……

Susanna Sajoni Henry

ABSTRACT

Verification is often a central issue in the design, development, and configuration of protocols. As new protocols are developed, it becomes important to check that these protocols work properly after implementation. There are numerous studies in the areas of computer networks and telecommunication systems that have shown that time and money can be saved if the protocol verification is done efficiently. Protocol verifications are conducted to evaluate existing or planned protocols, to compare alternative protocols, or to find an optimal protocol for a network. Protocol verification is a process of checking whether the interactions of protocol entities, according to the protocol specification, do indeed satisfy certain properties or conditions which may be either general or specific to the particular protocol system directly derived from the specification. A protocol system consists of a network of protocol entities and communication channels. Protocol entities interact by exchanging messages through channels; messages in transit may be lost, duplicated as well as reordered. The design of communication protocols for computer networks remains a mysterious art with occasional unexpected results and so the topic Protocol Verification requires good attention due to its inherent nature of complexity.

The current work is aimed at building a suitable model for protocol verification in the co-existence phase of mobile internet protocols. The modeling of the protocol is done using a schema diagram. The design functions for each module are formulated in the next step and finally each design function is implemented using programming language C.

The developments in Mobile IPv4 and Mobile IPv6 in recent years have opened up a new horizon for protocol verification. This thesis presents Protocol Verification in Mobile IP which consists of three parts as follows

i. Mobile IPv4.

ii. Coexistence phase: the phase in which Mobile IPv4 and Mobile IPv6 coexists.

iii. Mobile IPv6.

The Coexistence phase refers to the following three phases of Internet Transition Plan:

a. Preparation Phase (year 2008 through 2010): This phase was characterized by experimental use of IPv6, primarily through transition mechanisms and planning activities.

b. Transition Phase (2010 through 2011): characterized by general availability of IPv6 in provider networks, which should be native IPv6; organizations should provide IPv6 connectivity

for their Internet-facing servers, but should still provide IPv4-based services via a separate service name.

c. Post-Transition Phase (2012 and beyond): characterized by a dominance of IPv6-based services and diminishing support for IPv4-based services.

The procedure for protocol verification takes a protocol description as input, and then computes the output as a decision involving "Yes" or "No". A "Yes" implies that the input protocol description has been proven. The other alternative implies that some effort to prove this protocol description has failed. This approach will significantly help network designers in their design and verification phases of mobile internet protocols.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
IP	Internet Protocol
MIP	Mobile Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
MIPv4	Mobile Internet Protocol version 4
MIPv6	Mobile Internet Protocol version 6
ARP	Address Resolution Protocol
HoA	Home Address
HA	Home Agent
FA	Foreign Agent
MN	Mobile Node
CN	Correspondent Node
IETF	Internet Engineering Task Force
DSTM	Dual Stack Transition Mechanism
SOHO	Small Office, Home Office
NAT	Network Address Translation
DNS	Domain Name System
C1 Test	Code 1 Test
FSM	Finite State Machine
CP-Nets	Colored Petri Nets
WAN	Wide Area Network
LAN	Local Area Network
TTL	Time to Live

ICMP	Internet Control Message Protocol
HMAC	Keyed-Hashing for Message Authentication
MD5	Message-Digest Algorithm
TCP	Transmission Control Protocol
MAC	Media Access Control address
RIP	Routing Information Protocol
OSPF	Open Shortest Path First
QoS	Quality of Service
TEP	Tunnel End Point
ALG	Application Layer Gateway
P2P	Peer-to-Peer
SMTP	Simple mail transfer protocol
MTA	Message Transfer Agent
SMTP	Simple mail transfer protocol
SIP	Session Initiation Protocol
MNIP	Mobile Node Internet Protocol Address
HAIP	Home Agent Internet Protocol Address
COIP	Care-Of Address Internet Protocol Address
NAT	Network Address Translator
NAT-PT	Network Address Translation - Port Translation
DNS	Domain Name System
CSV	Comma-separated values file
XML	EXtensible Markup Language

CHAPTER 1

INTRODUCTION

In computer networks, a protocol refers to ‘an agreement between the communicating parties on how communication is to proceed’ [1]. A protocol is described as a set of rules and conventions that describe how to communicate. Verification can be described as a method to provide a guarantee of design correctness. This thesis presents verification of protocols in Mobile IP and its coexistence phase. Coexistence phase is the term coined by many researchers to indicate the duration between IPv4 addresses being exhausted and IPv6 still not deployed [2]. This phase is assumed to prevail for a period of a decade or more than 15 years [3, 4].

1.1 MOTIVATION

Mobile computing has greatly increased in popularity over the past several years, largely due to advances in miniaturization. Today we can get in a notebook PC or even a hand-held computer the power that once required a massive machine. Unfortunately, Internet Protocol was developed back in the era of massive bulky machines when it was not designed to deal gracefully with computers that move around. The ultimate goal of networking is to enable communication between hosts that are not directly connected. Determining what kinds of packets can be exchanged between two hosts connected to a network is a difficult and critical problem facing today’s network designers and operators. One single configuration error or design mistake can cause two hosts unable to communicate under certain failure scenarios, even though the network remains connected; knowing when these vulnerabilities exist is crucial while building a more reliable network. Solving this problem requires knowing far more than the network’s topology and the routing protocols it uses. These tight restrictions together with the increasing demand to ensure that IPv4 and IPv6 coexist in a smooth manner forms the main reason of this thesis to verify these existing network protocols [5].

The solution to all the above-mentioned issues was to define a new protocol especially to support mobile devices, which also adds to the original Internet Protocol. Mobile IP is considered as the only promising solution. This protocol was initially called IP Mobility Support for IPv4 in 1996, which was later updated in 2002 [3]. The formal name as given in that document title is rather long; the technology is now commonly called as Mobile IP by networking people.

Protocol Verification involves the construction of a model of the protocol in order to verify its functionality. The current work focuses on verification of protocols in Mobile IP and its coexistence phase. The present work is motivated by the increasing importance of Mobile IP and its coexistence phase. It is aimed to help network designers significantly in their design and verification phases of mobile internet protocols.

1.2 OBJECTIVES

The work presented here is intended to meet the following specific objectives:

- To perform protocol verification of Mobile IP protocols.
- To identify various modules that will lead to Protocol Verification of Dual Stack.
- To design steps to verify Tunneling in MIPv4 / MIPv6 by performing Encapsulation of IPv6 data packets within an IPv4 data packets and later performing Decapsulation to get the original IPv6 data packet.
- To specify design steps that will translate a MIPv6 data packet to MIPv4 data packet. Thus verifying Translation.
- To analyze MIPv6 by developing a schema diagram that will represent MIPv6 model and to perform protocol verification for correctness proofs.

1.3 SCOPE

This thesis presents a new set of approach that will enable to verify any given protocols. The presence of logical errors due to alterations in the assumptions and environment of the given protocol may lead to one or more statements or entire routines that will never be executed. This causes unnecessary work for the network designers and may result in discarding well developed protocol after its implementation. This thesis presents an approach that allows analyzing important properties of the protocol under consideration. The results obtained eventually help in synthesis of specific systems. It allows improving the correctness and the performance of a system.

The primary purpose of this thesis is to verify protocols running on end hosts in MIPv4-MIPv6 coexistence phase. Verifying a protocol means ensuring that it is free of logical errors prior to deploying and implementing it. The goal of verifying the protocol is to guarantee that protocol does exactly what the designer intended. It is expected this thesis will be useful to network manager to maintain Healthy and Ideal coexistence, MIPv4-MIPv6 environment.

Protocol Verification would be valuable for verifying the intent of the network designer, troubleshooting, and performing “what-if” analysis of failure scenarios.

1.4 SYSTEM DESCRIPTION

This section deals with the system description of the following protocols that are verified in this research work.

- Mobile IPv4
- Coexistence Phase consists of Dual Stack, Tunneling in MIPv4 / MIPv6 and Translation.
- Mobile IPv6

1.4.1 Mobile IPv4

Mobility in IPv4 supports enhancements that allow transparent routing of IP datagrams to mobile nodes in the Internet. Each mobile node is always identified by its home address, regardless of its current point of attachment to the Internet. While situated away from its home, a mobile node is also associated with a care-of address, which provides information about its current point of attachment to the Internet. The protocol provides for registering the care-of address with a home agent. The home agent sends datagrams destined for the mobile node through a tunnel to the care-of address. After arriving at the end of the tunnel, each datagram is then delivered to the mobile node.

- **Terminologies & new architectural entities used in Mobile IPv4.**

This section defines the entities of Mobile IPv4

i. Node: Node is a router or any host.
ii. Agent Advertisement: On attaching a special Extension to a Router Advertisement, resulting Advertisement message formed is called as Agent Advertisement message.
iii. Authentication: The identity of the originator of a message is verified using cryptographic techniques, this process of verification is termed as Authentication.
iv. Correspondent Node: Any node (either mobile or stationary) with which a mobile node is communicating is called as a Correspondent node.
v. Foreign Network: Any network other than mobile node's home network is called as Foreign Network.
vi. Home Agent: Its function is to process and coordinate mobility services.

vii. Foreign Agent: It relays a registration request and reply between Home Agent and Mobile Node and decapsulates datagram for delivery to Mobile Node [6].

viii. Mobility Agent: Home agent or a Foreign Agent is referred as Mobility Agent.

ix. Care-of Address: There are two different types of care -of address, they are 1) foreign agent care-of address and 2) co-located care-of address. The address of Foreign Agent with which mobile node is registered is called as foreign agent care-of address and co-located care-of address is an externally obtained local address that mobile node has associated with one of its own network interfaces.

x. Home Address: It is an IP address assigned to a mobile node. This address is assigned for an extended period of time and it remains unchanged regardless of location of attachment of node to internet.

xi. Home Network: It is a virtual network that has a network prefix which matches to mobile nodes home address. The datagrams destined to mobile nodes home address are delivered here to mobile nodes home network by standard IP routing mechanisms.

The important entities used in Mobile IPv4 is shown in figure 1.1.

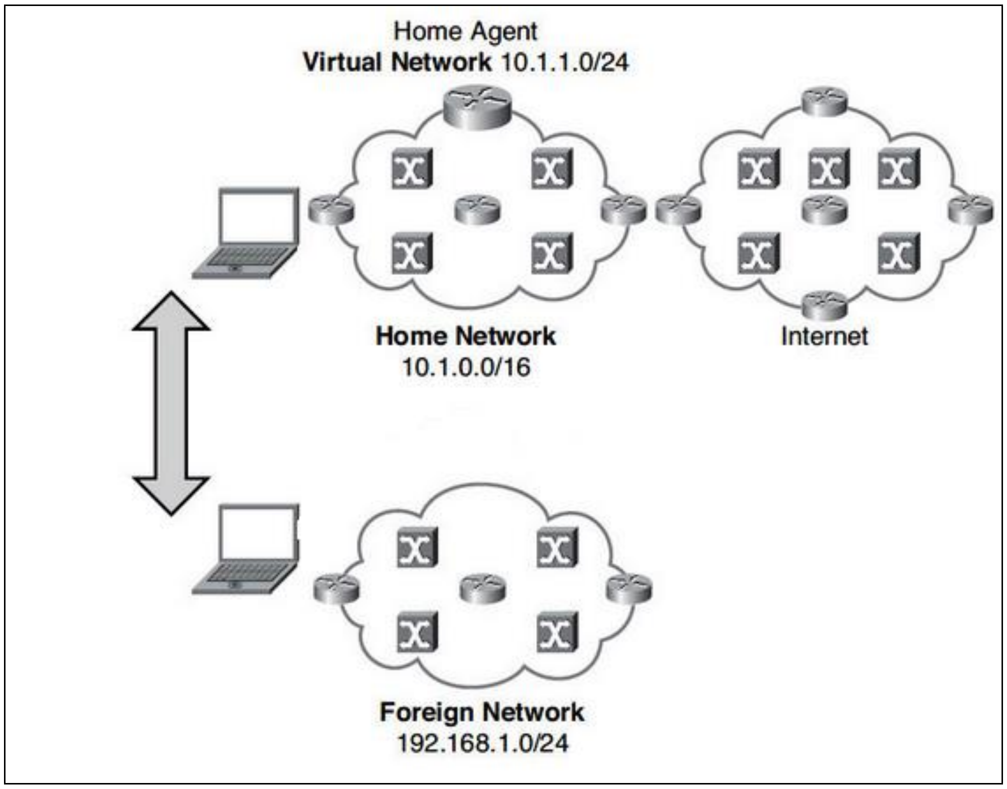


Figure 1.1 Elements of MIPv4

1.4.2 Coexistence Phase

1.4.2 Coexistence Phase: The time span from the period when IPv4 addresses get exhausted until the phase when IPv6 is completely deployed and implemented is termed as Coexistence phase. During this phase, both IPv4 and IPv6 coexist for more than a decade. Dual Stack, Tunneling in MIPv4 / MIPv6 and Translation prevail during Coexistence phase.

i. Dual Stack Architecture

Figure 1.2 provides a clear picture of Dual Stack Architecture. Suppose there are two IPv4-capable devices travelling through an IPv6-capable network. The data exchange between the two devices is assumed to be carried out by means of Dual Stack Transition Mechanism (DSTM). The data packet has IPv4 address which moves through different points in an IPv6 network. Starting from source point P0, the data packet now moves to point P1, which is a DSTM Server. The destination address of this packet is an IPv4 compatible IPv6 address and it travels through an IPv6 enabled network. The IPv6 network is approved as an integrated IPv6/IPv4 network where the border nodes maintain a dual stack of IPv4 and IPv6.The nodes P2 and P3 through which the IPv4 packets are tunneled are IPv6-only enabled nodes. The node P4 is another DSTM server, which can be considered as the end Point, and this is the point where the destination node's IPv4 address is maintained. The nodes P1 and P4 are the routers that act as starting point and end point [7]. These routers are responsible for maintaining both IPv4 and IPv6 stack as shown in figure 1.2

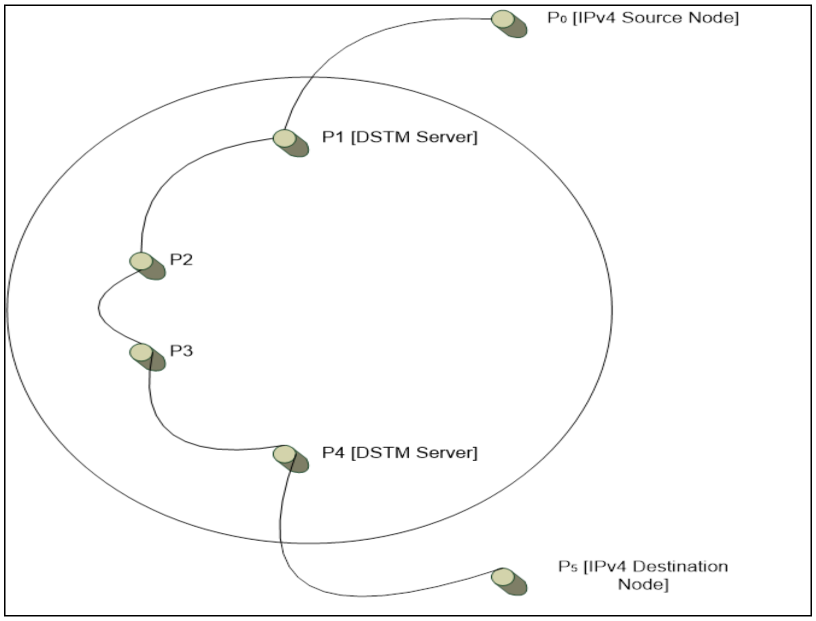


Figure 1.2 Dual Stack Architecture

ii. Tunneling in MIPv4 / MIPv6

Internet Protocol (IP) is a protocol for communicating data across a packet switched network. The network can be wireless and wire-line portions between two nodes, a first node and a second node. IP provides a unique global addressing method for representing the location of nodes in the network. This allows the first node to send data to the second node by using IP address of the second node while sending data. Internet Protocol version 4 (IPv4) uses 32-bit (4-byte) addresses, which limits address space to 4,294,967,296 possible unique addresses. The next generation IP is IPv6, addresses in IPv6 is 128 bits long that supports a larger number of unique addresses equal to 3.4×10^{38}addresses.

Due to differences in address length, the networking equipment that support IPv4 addresses cannot easily read and route IPv6 packets. Thus, an IPv6 message cannot generally be sent over a network that only supports IPv4. This creates a problem for transitioning networks from IPv4 to IPv6 because it can be very expensive to replace networking equipment in order to upgrade the addressing. Thus, Tunneling provides a mechanism to encapsulate IPv6 packets within IPv4 so that they can be carried across IPv4 routing infrastructures thereby reducing the transition complexities.

iii. Translation

Translation refers to the direct conversion of protocols. The IETF has considered Translation as the most sensible transition model. IPv4/IPv6 Translation enables networks to have IPv4 and IPv6 coexist to a certain extent in a balanced manner while transitioning to an IPv6-only network. The IETF recommends that network operators, transit providers, service providers, enterprise networks, small and medium businesses, SOHO (Small Office, Home Office) and residential customers, and any other kind of network that may currently be using IPv4 to perform the following:

1. Obtain an IPv6 prefix,
2. Turn on IPv6 routing within their networks and between themselves and any peer,
3. Upstream, or downstream neighbors, enable it on their computers, and use it in normal processing.

This must be done while leaving IPv4 stable, until a point is reached that any communication carried out could use either IPv4 or IPv6 protocol equally well. At that point, the economic justification for running both becomes controversial, and network operators can justifiably turn IPv4 off. While the new IPv6 protocol is also running stably, shift production towards it. When network and economic conditions permit, remove the old IPv4 protocol, which is now no longer necessary. Translation is described as one of the tools that any network might use to facilitate coexistence and eventual transition. Translation enables the client of one network to access the services of another network and communicate with other network users regardless of their protocol usage. Internet Protocol translation is regarded as one of best transition strategy that can be adapted. Figure 1.3 shown below explains IPv4/ IPv6 Translation.

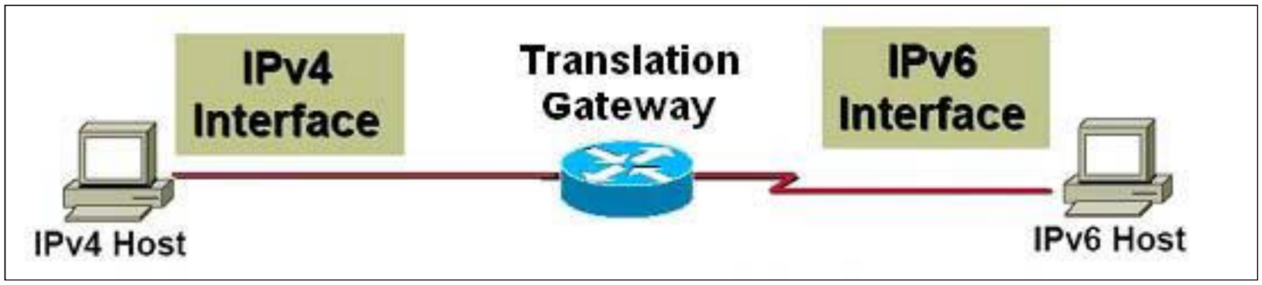


Figure 1.3 IPv4/IPv6 Translations

These mechanisms are expected to play a critical role in IPv4-IPv6 coexistence leading to gradual transition from IPv4 to IPv6. Due to IPv4 address depletion, it is likely that in future, the new clients will be IPv6-only and they will want to connect to existing IPv4-only servers.

Basic functionality of Translation requires the deployment of stateful NAT64 function in devices connecting an IPv6-only network to the IPv4-only network, along with the deployment of DNS64-enabled servers accessible to IPv6-only hosts. These two mechanisms are easily deployable and they do not require any changes in IPv6 client or IPv4 server. Thus Translation is considered as one of the sensible transition mechanisms in the current scenario.

1.4.3 Mobile IPv6

Mobile IPv6 protocol allows mobile nodes to remain reachable while moving around in IPv6 network. Each mobile node is always identified by its home address, regardless of its current point of attachment to the Internet. While situated away from its home, a mobile node is associated with a care-of address, which provides information about the mobile node's current location. IPv6 packets addressed to a mobile node's home address are transparently routed to its care-of address. The protocol enables IPv6 nodes to cache the binding of a mobile node's home address with its care-of address, and to then send any packets destined for the mobile node directly to it at this care-of address. To support this operation, Mobile IPv6 is defined as a new

IPv6 protocol with a new destination option. In this manner, all IPv6 nodes, whether mobile or stationary, can communicate with mobile nodes.

- **Mobile IPv6 terms**

Internet Protocol Version 6 (IPv6):- Started in 1991, the specification was completed in 1997 by the IETF. IPv6 is backward compatible with IPv4 and is designed to fix the shortcomings of IPv4, such as data security and maximum number of user addresses.

Home address: - A home address is an address assigned to the mobile node when it is attached to the home network and through which the mobile node is always reachable, regardless of its location on an IPv6 network.

Home subnet prefix: - Corresponds to a mobile node's home address.

Home link: - This link is configured with the home subnet prefix and this is where the mobile IPv6 device gets its home address.

Mobile node: - A mobile node is an Internet-connected device whose location and point of attachment to the Internet may frequently be changed.

Movement: - The change in the mobile node's attachment to internet so that it is no longer connected to the same link as before.

Correspondent node: - Any IPv6 enable device that intends to have communication with mobile node.

Foreign subnet prefix: - A bit string that consists of some number of initial bits of an IP address which identifies a node's foreign link within the Internet topology

Foreign link: - Any other link which is not mobile node's home link.

Care-of Address: - When a mobile node leaves its home network and attaches to another network, the node will get another address called a care-of address, which is assigned from the newly attached network. Multiple care-of addresses can be assigned to mobile node, but at any instance only one care-of address has binding with home address.

Home agent: - Acts as a support node on the home network for Mobile IPv6 mobile nodes. A home agent is a router that has a proxy function for mobile nodes while they are away from home. The destination addresses of packets sent to mobile nodes are set to the home addresses of the mobile nodes. A home agent intercepts all packets that are addressed to the mobile node's home address, and thus delivered to the home network, on behalf of the mobile nodes.

- **Differences between Mobile IPv4 and Mobile IPv6**

Mobile IP support in IPv6 has benefitted from the experiences gained during development of Mobile IP support in IPv4 [8,9]. Therefore, Mobile IPv6 shares many features from Mobile IPv4. However given below are some important differences between Mobile IPv4 and Mobile IPv6:

i. There is no need for foreign agents; no special local support is necessary to allow a Mobile IPv6 node to operate correctly.
ii. IPv4 mobility defines the default mode of operation through the home agent of mobile node. This tends to stress the mobile node’s home agent; the ability to redirect a correspondent node to the mobile node’s current address is implemented only as an extension of the IPv4 mobility specifications where as IPv6 mobility incorporates route optimization as a fundamental aspect of the protocol due to which home agent does not face any stress as in MIPv4.
iii. Rather than encapsulating packets in tunnels as in IPv4 mobility, most IPv6 packets sent to a mobile node include IPv6 header extensions. There is no need to encapsulate or decapsulate packets for tunnels in MIPv6. This is a significant difference in terms of processing.
iv. Use of IPv6 Neighbor Discovery eliminates the need to rely on ARP or other link layer mechanisms as in IPv4; it also enables functions such as unreachability detection and agent discovery.
v. The traffic in Mobile IPv4 is asymmetric. The tunnel from the home agent to the mobile node is unidirectional. The return traffic from the mobile node back to the corresponding node does not involve the tunnel. The source IP of the packets from the mobile node is the HoA. Since many network edges perform source IP filter, this represent an issue. While for Mobile IPv6, the tunnel between the Home Agent (HA) and the Mobile Node (MN) is bidirectional. This resolves the source IP packet filter that is present in Mobile IPv4.

1.5 RESEARCH GAP

One major shortcomings found in IPv4 is its limited 32 bit address space. The advantage of IPv6 over IPv4 is its 128 bits long address space. IPv6 is derived from IPv4 and in many ways is similar to it. However, IPv6 lacks backward compatibility with IPv4, Hence it requires a long-term development process for transition from IPv4 to pure IPv6 networks, so IPv4 and IPv6 coexists for a long time. The movement from IPv4 to IPv6 is not a straightaway process; it requires developing mechanisms so that IPv4 and IPv6 exist together for at least 20 years and during this transition period IPv4 network will totally disappear. The transition and interoperation between IPv4 and IPv6 must be carried out in such a way that entire deployment is smooth and completely trouble free. The process of transitioning from IPv4 network to new version of internet protocol IPv6 is being considered as one of the current topic in the area of ubiquitous network environment, with a goal to achieve high-level compatibility and clear procedure for easy and autonomous deployment of IPv6. The key to a successful IPv6 transition is compatibility with the large installed base of IPv4 hosts and routers. Maintaining compatibility with IPv4 while deploying IPv6 would be the ultimate goal of any network designers.

The time span from the period when IPv4 addresses get exhausted till the phase when IPv6 is implemented is termed as Coexistence phase. We are currently in the coexistence phase where IPv4 addresses are exhausted and IPv6 is not completely deployed. This scenario poses several configuration problems, such as performance problems, unknown software bugs etc. During this coexistence phase, messages are sent and received on communicating systems to establish reliable communications. The Protocols involved during this transition should therefore specify rules governing the transmission of datapackets. Few protocols suggested during this coexistence phase are Dual Stack, Tunneling, and Translation. In general verification of protocols must be addressed by specifying Data formats for data exchange [10,79], Address formats for data exchange, Address mapping, Routing, Flow control, Sequence control, Direction of information flow, Loss of information - timeouts and retries, Acknowledgements of correct reception of packets, Detection of transmission errors [11,80].

Over any mobile network, agents need means to communicate between themselves. Protocols, although constructed over the arrangement of simple blocks, normally target the yielding of complex goals. They seem simple at a first glance, but hide subtleties that allow them to be exploited. One way of trying to systematically capture such subtleties is through the usage of formal methods to verify any protocol. The maturity of these methods for protocol verification is a fact today. However, these methods are still not able to capture the whole set of protocols being designed. With the convergence to an on-line world, new goals are proposed and new protocols need to be designed. The evolution of protocol verification methods becomes a necessity to keep the pace with this ongoing development.

- It turns out to be rather difficult and time consuming (if not infeasible) to prove these protocols correct. In general, one can state that ‘provable’ protocol versions are often not ready to be directly implemented in practice, due the assumptions and simplifications that were applied in order to deliver a proof.
- Current state of the art formal verification techniques can handle only very restricted configurations. This induces the need for suitable abstraction techniques.
- A related problem is the state space explosion problem that is inherent to model checking: the number of states to be checked grows exponentially in various components of the model, such as the number of variables or the number of parallel components in a concurrent system. The problem is that a system consisting of too many states cannot be automatically checked due to time and memory limitations.

Hence, these approaches suffer from various problems and this clearly obstructs us in our quest for proving correctness of protocols. The present work provides an environment for verification of mobile IPv4, Dual Stack, Tunneling, Translation and mobile IPv6 protocols.

1.6 METHODOLOGY

The approach followed in this research work is as follows. An extensive literature study was carried to become familiar with the concepts and terminology used in the protocol. Then the protocol design was made explicit by specifying a description of the protocol under consideration. While carefully documenting every useful detail of each protocol, models of each protocol were constructed and this was represented by schema diagram. In order to further narrow the scope of this research, next a design was developed from each schema diagram and finally this design was implemented using programming language C. The respective experimental results were noted down. The main motivation for using specifically this method was the popularity of the programming language C which played an important role. The idealized protocol design considered in this thesis has been proven correct by applying C1 test methodology.

This research specifies a basic abstraction of developing the schema diagram that represented the protocol, leading to the design and implementation of each protocol as a way of proving the extensibility of the method applied here. It showed the feasibility of specifications by revisiting a classic protocol MIPv4 , now verified under our framework. Finally Verification under a mixed environment of Mobile IPv4 and Mobile IPv6 was done.

While performing the verification of protocols in this thesis, the complexity of Protocols in coexistence phase (MIPv4-MIPv6) were noted down, each protocol model was verified under different types of assumptions about the environment. Relevant results were recorded and suitable conclusions and/or recommendations based on the above analysis were suggested.

1.7 LIMITATIONS

This section presents the limitations posed while performing Protocol Verification of various protocols. Difficulty arose while modeling the protocol and converting the protocol description to its schema diagram. The limitation in this approach is that a mapping table is used to convert IPv6 address to IPv4 address. This mapping table maps the input IPv6 address to its respective IPv4 address and this design provided only a few options for translation. However to get a wide range of IPv6 addresses translated to IPv4 address, the only solution is to increase the size of the mapping table. The other limitation was that same mapping table could not be used to translate IPv4 address to IPv6 address.

1.8 ORGANIZATION OF THESIS

Chapter 2 contains the necessary background information of this research work; it is divided into 3 subsections involving Mobile IPv4, coexistence phase in MIP and Mobile IPv6.

Chapter 3 deals with verification of Mobile IPv4, First the protocol was described in detail, a schema diagram was developed to represent Mobile IPv4, and design functions were outlined based on schema diagram. Finally, design steps were implementation using programming language C and experimental results were noted down with relevant snapshots.

Chapter 4 presents Protocol verification of Dual Stack. Schema diagram was constructed based on description mentioned and design analysis of Dual stack was carried out using the schema diagram, finally this design was implemented using programming language C. Respective results for each run were noted down with the help of relevant snapshots.

Chapter 5 deals with Protocol Verification of Tunneling in MIPv4/MIPv6 and this chapter is divided into following subsections as follows, modeling of Tunneling using Schema diagram and Design Analysis of Tunneling was carried out. This design was implemented using programming language C and experimental results were noted down for each run.

Chapter 6 deals with Protocol Verification of Translation and respective experimental results are noted down in the following subsections. **Chapter 7** deals with verification of Mobile IPv6, analysis and modeling was carried out using Schema diagram, followed by its Design Analysis. Finally implementation of Mobile IPv6 was carried out using programming language C and experimental Results were noted down.

Chapter 8 summarizes the work carried out in this thesis, highlights specific contributions and suggests some directions for future work. The appendices consist of the following information:

- The source code that deals with Mobile IPv4 model.
- The source code for Dual Stack, Tunneling and Translation models.
- The source code for Mobile IPv6 model.
- C1 Test methodology.

Relevant articles, books and websites are listed in the **References.** The **list of publications** related to the thesis are given at the end.

CHAPTER 2

LITERATURE REVIEW

This section reviews previous and current literature of the scientific field relevant to this thesis. Section 2.2 presents the description of MIPv4. Section 2.3 covers the analysis of coexistence phase. Section 2.4 deals with MIPv6 protocols.

2.1 INTRODUCTION

Regularly numerous and complex communication protocols are being employed in computer networks of various types. As new protocols are being developed, they must be checked for logical correctness before deploying, implementing and using them. If the protocol is in wide use, many different implementations have to be checked for compliance with a standard. As a result, several protocol verification tools and methods have been built in verifying correctness of communication protocols. Communication protocols can exhibit extremely complicated behavior, and neither informal reasoning nor testing is reliable enough to establish their correctness. Though many researchers in this field have proposed various strategies to relieve this intricate problem when building the tools, the search for an appropriate approach to tackling such a problem still continues [6,25].

Protocol verification is a crucial step to eliminate weaknesses and inaccuracies of effective network protocols. Protocols can be verified against their design and implementation. In simple words, protocol can be verified during the initial phase before the system is implemented and since protocol verification can detect errors at the early stage, unnecessary or incorrect implementation can be avoided. Therefore, protocol verification has the potential to significantly reduce the cost of protocol development and testing.

There are many models and tools to verify network protocols, including, FSM, Colored Petri Nets (CP-Nets), Temporal Logic, Predicate Logic, Estelle Specification, and Path based Approach [6,81] etc. Several studies in this field have recently made significant advances in the generation of test sequences from formal specifications to the development of computer-aided tools, with the aim of improving the effectiveness of verification. However there is not much progress in the techniques for practical verification of communication networks. There is a big gap between verification practice and research results. This big gap is the result of the fact that all verification issues are not being addressed. Hence several researches, empirical studies and a new orientation towards real issues in protocol verification needs to be given higher priority.

The following section deals with detailed literature survey of the protocols relevant for this thesis. Mobile IPv4, coexistence phase which includes Dual stack, Tunneling in MIPv4 / MIPv6, Translation and Mobile IPv6 is described as follows.

2.2 DESCRIPTION OF MOBILE IPV4

This section presents a detailed description of Mobile IPv4.

2.2.1 Introduction

In IPv4, IP address specifies the location of any device on a network and all the datagrams destined to it. This IP address of a particular device gives information about its point of attachment to the internet. Suppose if this device changes its point of attachment, then datagrams destined to it would never be delivered. There are two mechanisms that can be followed in order to maintain continuous correspondence despite of changing its point of attachment; the two mechanisms are as follows

1. The device must change its IP address every time it changes its point of attachment

   or

2. Host specific route must be generated throughout the internet.

The above given mechanisms seem to be unscalable, as in first mechanism, it is impossible to maintain connectivity when device changes its location and second mechanism seems to be an impossible task due to huge number of mobile computers in use.Thus there is a need to introduce a new mechanism that will deal with mobility of computers within the internet. The solution is Mobile IP: A mechanism which allows devices to change their point of attachment without changing their IP address.

Mobile IP (or MIP) is an Internet Engineering Task Force (IETF) standard communications protocol that is designed to allow mobile device users to move from one network to another while maintaining a permanent IP address. When a user leaves the network with which his device is connected (home network) and enters domain of a foreign network, the foreign network uses Mobile IP protocol to inform home network about a care-of address to which all packets for user's device should be sent. Mobile IP is most often found in wireless WAN environments where users need to carry their mobile devices across multiple LANs with different IP addresses.

A common analogy to explain Mobile IP is when someone moves his residence from one location to another. Person moves from Bengaluru to Delhi. Person drops off new mailing address to Delhi post office. Delhi post office notifies Bengaluru post office of new mailing address. When Bengaluru post office receives mail for person it knows to forward mail to person's Delhi address.

2.2.1.1 Goals of Mobile IPv4

Goals of Mobile IPv4 would be to minimize number of administrative messages that are passed over the link in which a mobile device is attached to the internet, moreover as this link is wireless link it would have lower bandwidth and higher error rate as compared to wired networks. Since all mobile devices are battery powered, consumption of power must be as low as possible and message size must be kept as small as possible.

2.2.1.2 Assumptions in Mobile IPv4

It is assumed that mobile devices do not change their point of attachment to the internet more than once in one second, it is also assumed that datagram header holds destination address for respective IP unicast datagrams to be routed to its respective destination. It is assumed that organization that owns machine assigns an IP address to its mobile node so protocols in MIPv4 do not pose any additional constraints on assignment of IP address.

Mobile IPv4 comprises of the following mechanisms.

- Agent Discovery
- Registration
- Routing

2.2.2 Agent discovery

A Mobile Node discovers its Foreign and Home Agents during agent discovery. Agent Discovery comprises of two important aspects, Agent Advertisements and Agent Solicitation as shown in figure 2.1. In absence of agent advertisements, a mobile node can solicit advertisements. This is known as agent solicitation. Mobile IP extends Internet Control Message Protocol (ICMP) Router Discovery as its primary mechanism for Agent Discovery [26]. An Agent Advertisement is formed by including a Mobility Agent Advertisement Extension in an ICMP Router Advertisement message. An Agent Solicitation message is identical to an ICMP Router Solicitation, except that its IP Time to Live (TTL) must be set to 1. The techniques by which mobile nodes, foreign agents and home agents function together to realize Agent

Discovery is mentioned here along with required message formats. The procedures described below assume that link-layer connectivity has already been established. No authentication is required for Agent Advertisement and Agent Solicitation messages. They may be authenticated using IP Authentication Header [27].

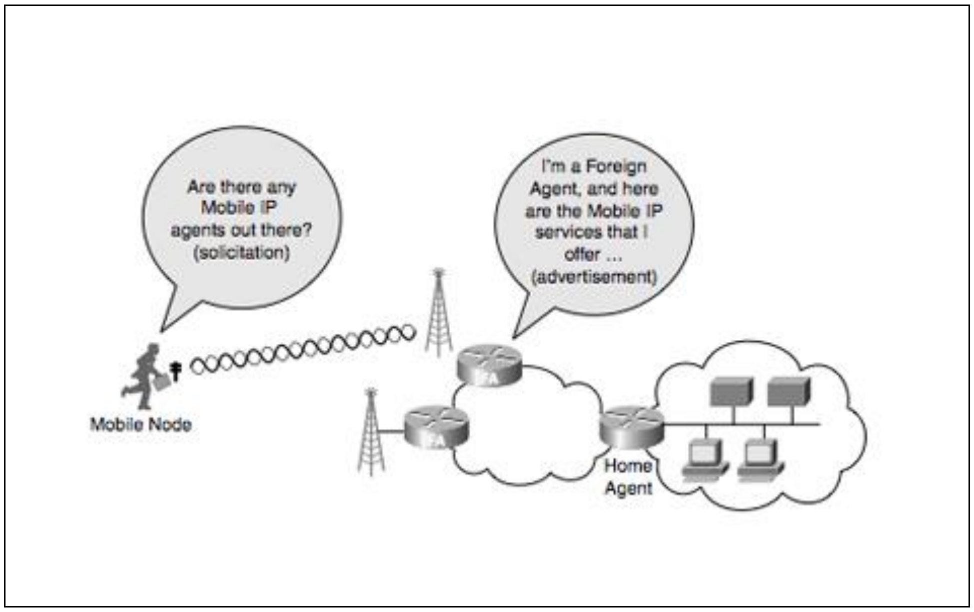

Are there any Mobile IP agents out there? (solicitation)
I'm a Foreign Agent, and here are the Mobile IP services that I offer ... (advertisement)
Mobile Node
FA
FA
Home Agent

Figure 2.1 An illustrative representation of Agent Discovery

2.2.2.1 Agent Advertisement

Mobility agents advertise their services on a link by transmitting Agent advertisements. A mobile node uses these agent advertisements to decide its current point of attachment to the Internet or to an organization's network. In case of multiple numbers of mobile nodes, a foreign agent might be too busy to attend all additional mobile nodes, in such situations a mobile node must send out agent advertisements continuously such that foreign agent will know whether mobile node which is registered with it is within same network or has moved out of its range.

An agent advertisement is an Internet Control Message Protocol (ICMP) router advertisement that has been extended to also carry a mobility agent advertisement extension. This ICMP Router Advertisement is extended to carry a Mobility Agent Advertisement Extension and, optionally, a Prefix-Lengths Extension, One-byte Padding Extension along with other Extensions. Agent Advertisement messages are not a generalized method for exchanging routing information. They are a support mechanism only, used to inform hosts about existence of routers. The structure of MIP Agent Advertisement message is as shown in figure 2.2.

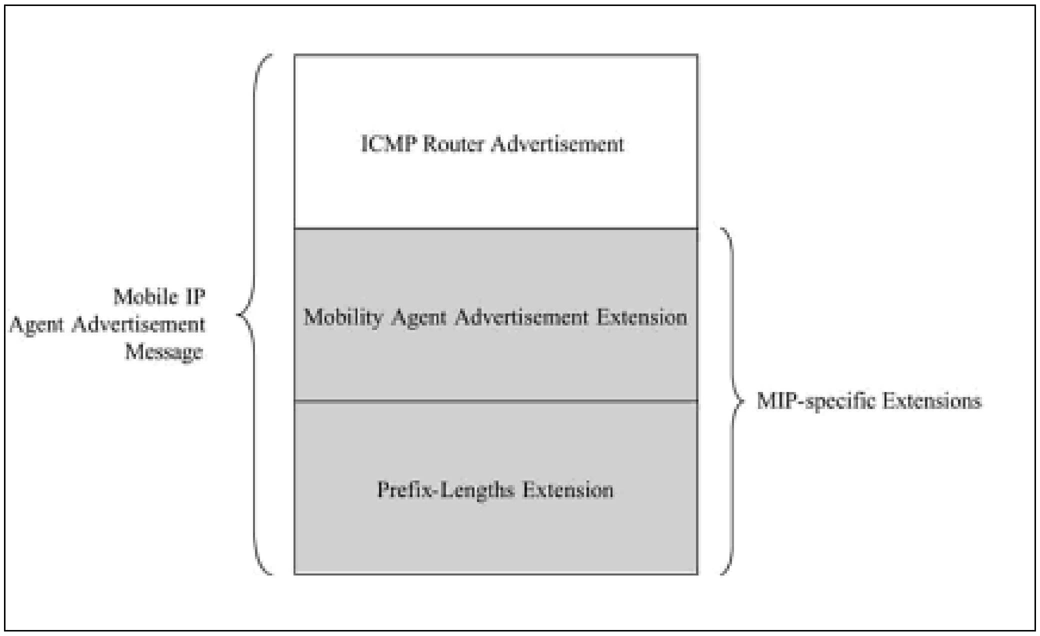

ICMP Router Advertisement
Mobility Agent Advertisement Extension
Prefix-Lengths Extension
Mobile IP Agent Advertisement Message
MIP-specific Extensions

Figure 2.2 Structure of MIP Agent Advertisement message.

2.2.2.2 Addressing and Pattern of Agent Advertisement and Agent Solicitation Messages

ICMP Agent Advertisement messages are sent frequently by IP routers to notify hosts of their presence and characteristics. When a host is new to a network, it may send an Agent Solicitation message to find out what routers are present. This will prompt listening routers to send out Agent Advertisements. Agent Advertisements use “all devices” multicast address (224.0.0.1), since they are suggested for hosts to hear, If local network does not support multicast, messages are instead sent out broadcast (to address 255.255.255.255) while Agent Solicitation messages use “all routers” multicast address (224.0.0.2). Agent Advertisement and Agent Solicitation messages are sent out multicast for efficiency.

Thus on summarizing, Agent discovery is an alternative to manual configuration of a host's default router. Finally, note that when Mobile IP is implemented, Router Advertisement messages are used as the basis for Mobile-IP-aware routers to send Agent Advertisements. One or more special extensions are added to regular Router Advertisement format to create an Agent Advertisement.

2.2.2.3 Role of Mobile node in Agent Discovery The role of mobile node is explained by following features,

1.Visiting Policies of a Mobile node

To enforce visiting policies that require exchanges of authorization, mobile node should register through foreign agent even while it is able to acquire its own co-located care-of address on receiving Agent Advertisement. This has effect of forcing mobile node to register through foreign agent, so foreign agent could then monitor/enforce the policy [27].

2.Algorithm for movement of a Mobile node

When mobile node discovers that it has moved to a new foreign network, it should register with a suitable care-of address on that foreign network. On the other hand, mobile node must not register more frequently than once per second on average. There are two primary algorithms for mobile nodes to detect when they have moved from one subnet to another.

Algorithm 1: This algorithm depends on lifetime field of Agent Advertisement. A mobile node should record the Lifetime received in any Agent Advertisements, until that Lifetime expires. The mobile node now assumes that it has lost its contact with agent if it fails to receive another advertisement from same agent within specified Lifetime. The mobile node may attempt registration with other agent if it has previously received an Agent Advertisement for which the Lifetime field has not yet expired, else it should attempt to discover a new agent with which mobile node has to register.

Algorithm 2: The second algorithm uses network prefixes. The mobile node utilizes the Prefix-lengths to determine whether or not a newly received Agent Advertisement was received on the same subnet as mobile node's current care-of address. If the prefixes differ then mobile node assumes that it has moved. Mobile node should not use this algorithm if it is currently using a foreign agent care-of address unless both current agent and new agent include the Prefix-Lengths Extension in their respective Agent Advertisements. Similarly, if a mobile node is using a co-located care-of address, it should not use this method of move detection unless the new agent includes Prefix-Lengths Extension in its Advertisement and mobile node knows network prefix of its current co-located care-of address. If the result of this algorithm indicates that mobile node has moved on expiration of its current registration, mobile node may now choose to register with foreign agent sending new Advertisement with different network prefix rather than re-registering with its current care-of address.

3.Returning home

When a mobile node receives an Agent Advertisement from its own home agent it detects that the mobile node has returned to its home network. Now mobile node should configure its routing table correctly for its home network. And finally mobile node should deregister with its home agent.

4.Rollover and re-registration

Mobile node should re-register if it finds two successive values of sequence number in Agent Advertisements from foreign agent with which it is registered and if the second value is less than first and between the range of 0 to 255, but re-registration is not necessary if second value is less than first and if it is greater than or equal to 256 in which case the sequence number is assumed to roll over its maximum value (0xffff) [26].

Despite performing the above mentioned roles, every mobile node must employ Agent Solicitation. Mobile node sends Agent Solicitation message in the absence of Agent Advertisements and whenever it is not possible to determine care-of address through a link-layer protocol. The frequency at which a mobile node sends solicitations is restricted by mobile node. The mobile node sends Solicitation messages once in every three seconds, while searching for an agent; the mobile node may send 3 initial solicitations at a maximum rate of one per second, and later the rate at which the solicitation message sent is decreased to reduce overhead on local link. The maximum interval is chosen appropriately based upon characteristics of media over which the mobile node is soliciting. This maximum interval is restricted to be at least one minute between solicitations. Care must be taken such that mobile node must not increase the rate at which it sends solicitations as it searches for an agent. Once it receives an affirmation that it has moved to a new link and after successfully registering with an agent, mobile node can now increase the rate at which it will send solicitations when it next begins searching for a new agent with which to register. The solicitation rate can increase to a maximum rate within the limits of nominal values. Mobile nodes must randomize their solicitation times around these nominal values as specified for ICMP Router Discovery [26].

The mobile nodes must process all the Agent advertisements. When multiple methods of agent discovery are in use, mobile node should first attempt registration with agents including Mobility Agent Advertisement Extensions in their advertisements, thereby maximizing the predictable registrations which lead to minimized registration attempts.

2.2.2.4 Role of Foreign agent in Agent Discovery

A foreign agent must implement Agent advertisements irrespective of whether these mobility agents can be discovered or cannot be discovered by link-layer protocol. However these Agent advertisements are sent only as response to a specific agent solicitation or if site policy requires registration with the agent, but for any other situation, there is no need to send agent advertisements. All foreign agents must process packets that they receive addressed to Mobile-Agents multicast group, at address 224.0.0.11. All foreign agents must respond to agent solicitations sent by mobile node to 224.0.0.11.

Foreign agents must follow the rules given below:

i. Care should be taken that the rate at which foreign agent sends Agent advertisements is limited, and maximum rate should be an optimum value such that advertisements do not occupy a significant amount of network bandwidth.
ii. IP source address of router solicitation message received by foreign agent must not be same as address of any neighbor node.
iii. Foreign agent must be configured in such a way that it must send agent advertisements only in response to an Agent Solicitation message.

The sequence number in Agent Advertisements varies from 0 to 0xffff. After booting, an agent must use number 0 for its first advertisement. Each following advertisement must use the sequence number one greater, with exception that the sequence number 0xffff must be tracked by sequence number 256. In this way, mobile nodes can decide a decrease in sequence number that occurs after a reboot from a reduction that results in rollover of sequence number after it attains value 0xffff.

For smooth operation of algorithms designed for mobile nodes, it is suggested that on a particular subnet either all of foreign agents must not include prefix-lengths extension or all of foreign agents must include prefix-lengths extension. Every agent advertisement contains one or more router address. A foreign agent must route all data packets it receives from mobile node irrespective of whether the foreign agents address is present in router addresses [28].

2.2.2.5 Mobility agents Considerations

All mobility agents must send Agent Advertisements irrespective of whether these mobility agents can be or cannot be discovered by a link-layer protocol and they are supposed to process the packets that they receive but are addressed to Mobile-Agents multicast group, at address

224.0.0.11 (“all devices” multicast address). A mobile node may send an Agent Solicitation to 224.0.0.11 while all mobility agents should respond to Agent Solicitations.

The following considerations must be maintained by foreign agent and Home agent:

i. The foreign agent and home agent MUST limit rate at which it sends broadcast or multicast Agent Advertisements; the maximum rate SHOULD be chosen so that Advertisements do not consume a significant amount of network bandwidth.

ii. The foreign agent and home agent that receive a Router Solicitation MUST check that IP Source Address is not address of a neighbor.

iii. The foreign agent and home agent MAY be configured to send Agent Advertisements only in response to an Agent Solicitation message.

At this juncture, a needfull question arises and it states as: How does a mobile node determine when it is in home network ??.

There are two cases to consider.

- If Home network is a Virtual Network
- If Home network is Not a Virtual Network

Case a: If home network is a virtual network, home network has no physical realization external to the home agent itself. In this case, there is no physical network link on which Agent Advertisement messages are sent to advertise home agent. Mobile nodes for which this is home network are always treated as being away from home.

Case b: If home network is not a virtual network, then home agent for any mobile node should be located on the link identified by mobile node's home address. In this way, mobile nodes on their own home network will be able to determine that they are indeed at home.

Every mobile node is supposed to implement Agent Solicitation. Solicitations are usually sent in absence of Agent Advertisements and only when a care-of address has not been determined through a link-layer protocol or other means. The rate at which a mobile node sends solicitations is limited by mobile node. The mobile node may solicit more often than once every three seconds. This mobile node will send three initial solicitations at a maximum rate of one per second while searching for an agent. After this, the rate at which solicitations are sent must be reduced so as to limit overhead on the local link. Subsequent solicitations are usually sent using a binary exponential backoff mechanism, doubling interval between consecutive solicitations, up to a maximum interval. The maximum interval is chosen appropriately based upon characteristics of the media over which mobile node is soliciting. This maximum interval should be at least one minute between solicitations [29].

While still searching for an agent, it must be ensured that mobile node does not increase the rate at which it sends solicitations unless it has received a positive indication that it has moved to a new link. After successfully registering with an agent, the mobile node is allowed to increase rate at which it will send solicitations when it next begins searching for a new agent with which to register. The increased solicitation rate may revert to maximum rate, but then must be limited. In all cases, recommended solicitation intervals are nominal values. Mobile nodes are expected to randomize their solicitation times around these nominal values. Now these mobile nodes are expected to process received Agent Advertisements. A mobile node can distinguish an Agent Advertisement message by examining the number of advertised addresses and IP Total Length field. The presence of a Mobility Agent Advertisement Extension identifies this message as an Agent Advertisement. If there is more than one advertised address, the mobile node picks up the first address for its initial registration attempt. If registration attempt fails with a status code indicating rejection by foreign agent, then the mobile node might retry its attempt with each subsequent advertised address in turn.

2.2.3 Registration

The process in which a mobile node registers its current location with the Foreign Agent and Home Agent is termed as Registration. The following figure 2.3 explains all the steps involved during Registration.

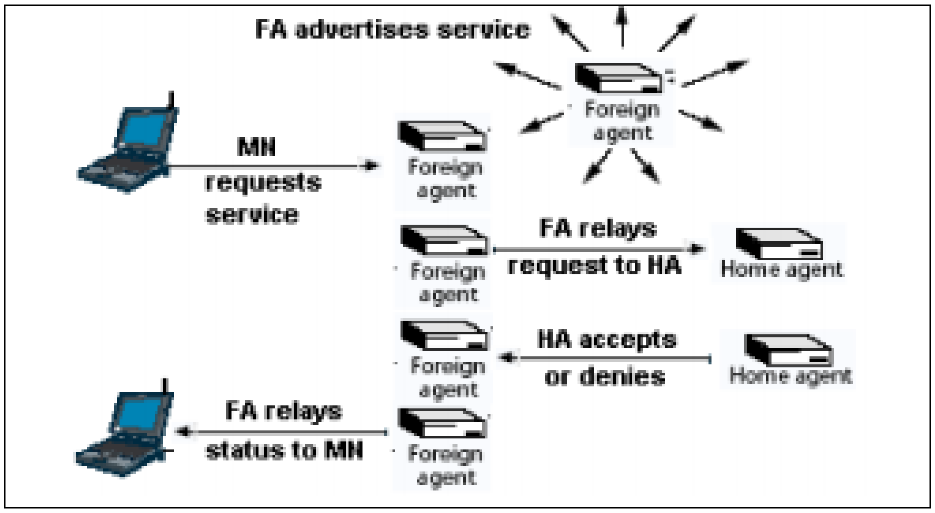


Figure 2.3 Registration in MIPv4

Registration procedure offers several optional capabilities which are listed below,

i. In case if mobile node is not configured with the information of its home address then Registration procedure enables mobile node to discover its home address.
ii. A copy of each datagram will be tunneled to each active care-of address such that Mobile nodes maintain multiple simultaneous registrations.
iii. While retaining other mobility bindings mobile nodes deregister specific care-of addresses.
iv. If mobile node is not configured with any information about home agent, in such case Registration procedure enables mobile node to discover the address of a home agent.

2.2.3.1 Registration Overview

There are following rules for registration procedures

Rule1: Mobile node MUST register via foreign agent if it is registering a foreign agent care-of address.

Rule2: If a mobile node is using a co-located care-of address, and receives an Agent Advertisement from a foreign agent on the link on which it is using this care-of address, mobile node should register via that foreign agent (or via another foreign agent on this link).

Rule3: Mobile node must register directly with its home agent if it is using a co-located care-of address.

Rule4: Mobile node must register directly with its home agent if a mobile node has returned to its home network and is (de)registering with its home agent.

There are two different procedures in Registration defined in Mobile IP

1) Registration through a foreign agent that relays registration to mobile node's home agent.

2) Registration with mobile node's home agent.

The rules mentioned above determine which of two registration procedures can be used in any particular circumstance. Both registration procedures involve the exchange of Registration Request and Registration Reply messages, but there is a major difference between the two Registration procedures. The difference is between numbers of messages used. When registering via a foreign agent; the registration procedure requires following four messages [27].

i. To begin registration process mobile node sends a Registration Request to prospective foreign agent.
ii. Foreign agent relays this Registration Request to home agent after processing it.

iii. Home agent sends a Registration Reply to foreign agent to grant or deny Request.

iv. The disposition of Request is informed to mobile node and finally foreign agent processes Registration Reply and then relays it to mobile node.

2.2.3.2 Authentication

Home agents and mobile nodes must be able to perform authentication. Registration messages between a mobile node and its home agent must be authenticated with an authorization-enabling extension, e.g., Mobile-Home Authentication Extension. This extension must be the first authentication extension; other foreign-agent-specific extensions may be added to message after mobile node computes authentication. Each mobile node, foreign agent, and home agent must be able to support a Mobility Security Association for mobile entities, indexed by their SPI and IP address. There are few algorithms that support Authentication, They are HMAC (HMAC: Keyed-Hashing for Message Authentication) and MD5 Message-Digest Algorithm [30, 31].

2.2.3.3 Registration Request

A mobile node registers with its home agent using a Registration Request message. In order to create mobility binding between home agents and mobile node, Registration request is sent by mobile node to register with its home agent. The Request may be relayed to home agent by foreign agent through which mobile node is registering, or it may be sent directly to home agent wherein mobile node registers a co-located care-of address.

2.2.3.4 Registration Reply

Reply messages contain necessary codes to inform mobile node about status of its Request and also lifetime granted by home agent, Registration Reply message is returned to mobile node by a mobility agent that had sent Registration Request message.

The Lifetime is covered by an authentication extension that enables authorization by home agent. As this extension contains authentication data that cannot be computed by foreign agent, therefore foreign agent is not supposed to increase Lifetime in registration request selected by mobile node. Similarly home agent should not increase Lifetime selected by mobile node in Registration Request, since doing so could increase it beyond maximum Registration Lifetime allowed by foreign agent. If Lifetime received in Registration Reply is greater than that in Registration Request, Lifetime in Request must be used. And if Lifetime received in Registration Reply is less than that in Registration Request, the Lifetime in Reply must be used.

2.2.3.5 Role of Mobile Node in Registration

For registration of a mobile node, every mobile node must be configured (statically or dynamically) with a netmask and Mobility Security Association for each of its home agents along with its home address and IP address of one or more of its home agents. If in case of any pending registration, mobile node maintains following information.

i. Link-layer address of foreign agent to which Registration Request was sent, if applicable,
ii. IP Destination Address of Registration Request,
iii. Care-of address used in registration,
iv. Identification value sent in registration,
v. Originally requested Lifetime, and
vi. Remaining Lifetime of pending registration.

Whenever a mobile node detects a change in its network connectivity, it should initiate registration. The mobile node determines whether it is at home or it is away from home. A mobile node operates without support of mobility functions when it is at home. The home agent creates a mobility binding for mobile node when it is away from home. Similarly home agent deletes its previous mobility binding with mobile node on finding out that it is at home.

A mobile node should register itself with foreign agent when it finds that foreign agent has rebooted or when current registration's Lifetime is near expiration. Mobile node is not allowed to attempt for new registration if its current registration has not expired and it is still receiving Agent Advertisements from foreign agent with whom it is currently registered. Mobile node can register with a different agent when transport layer specifies excessive retransmissions and in absence of link-layer specifications.

1.Values supplied by Mobile Node in Registration Request Message

Mobile node must supply values in fields of registration request message as given below

a. IP Time to Live:

When IP Destination address is set to "All Mobility Agents" multicast address, IP TTL field must be set to 1 else a suitable value should be chosen in accordance with standard IP practice [32].

b. IP Source Address:

i. IP source address must be the care-of address when registering on a foreign network with a co-located care-of address

ii. In case if mobile node does not have a home address, the IP source address must be 0.0.0.0.

iii. In all other circumstances, the IP source address must be mobile node's home address.

c. IP Destination Address:

i. If mobile node is registering directly with its home agent then destination address must be set to IP address of its home agent

ii. If the IP address of agent with which mobile node is registering is unknown to mobile node then the "All Mobility Agents" multicast address (224.0.0.11) must be used and to deliver datagram to correct agent, the mobile node must use agent’s link layer unicast address.

iii. The mobile node may use dynamic home agent address resolution to automatically determine IP address of its home agent if mobile node does not know IP address of its home agent. In this case, IP Destination Address is set to subnet-directed broadcast address of mobile node's home network. Though mobile node may use home agent address in body of Registration Request when registering via a foreign agent, it must not use Destination IP address if mobile node is registering via foreign agent.

iv. When mobile node is registering with a foreign agent, the address of agent recovered from IP source address of corresponding agent advertisement must be used. This address may not appear as an advertised care-of address in agent advertisement.

d. Lifetime field of Registration request message

i. When mobile node deregisters all care-of addresses upon returning home, the lifetime field is set to zero.

ii. With lifetime set to zero, mobile node may send a registration request to home agent requesting to delete mobility binding.

iii. When mobile node is registering with a foreign agent, lifetime should not exceed the value inRegistration Lifetime field of Agent Advertisement message received from foreign agent. The default lifetime is set to 1800 seconds.

2.Registration Replies received by Mobile node

Registration Replies will be received by mobile node in response to its Registration Requests. Registration Replies generally fall into three categories:

i. Registration request accepted,

ii. Registration request denied by the foreign agent, or

iii. Registration request denied by the home agent.

i. Registration request accepted: When a mobile node is registered on a foreign network, it should re-register itself before its Lifetime expires. Mobile node must maintain remaining lifetime and original lifetime of pending registration request. When mobile node receives a valid Registration Reply, mobile node must decrease the remaining lifetime of registration with same amount by which home agent decreased the originally requested Lifetime. This procedure is equivalent to the mobile node starting a timer for granted Lifetime at the time it sent Registration Request, even though granted Lifetime is not known to mobile node until Registration Reply is received. This procedure ensures that mobile node will re-register before home agent expires and deletes registration.

ii. Registration request denied by foreign agent: Registration request may be denied due to long lifetime. The Lifetime field in Registration Reply will contain maximum Lifetime value. In this case mobile node may attempt to register with the same agent using a Lifetime in Registration Request that must be less than or equal to the value specified in Registration Reply.

iii. Registration request denied by the home agent: Registration request may be denied due to registration identification mismatch. In this case, Identification field in Registration Reply will contain a value that allows mobile node to synchronize with home agent. Before issuing any future Registration Requests, mobile node must adjust the parameters it uses to compute Identification field based upon information in Registration Reply.

There is one more reason that registration request may be denied, that is due to unknown home agent address. In this case, Home Agent field will contain unicast IP address of home agent returning the Reply. Mobile node may then attempt to register with this home agent in future Registration Requests.

3.Registration Retransmission

If registration reply is not received within a reasonable time, another Registration Request may be transmitted. For each retransmission a new registration Identification is chosen which counts as a new registration. The maximum time until a new Registration Request is sent should not be greater than requested Lifetime of Registration Request. The minimum value should be large enough to account for the size of the messages, twice the round-trip time for transmission to home agent, and at least an additional 100 milliseconds to allow for processing the messages before responding. The round-trip time for transmission to home agent will be at least as large as time required to transmit messages at the link speed of mobile node's current point of attachment. Some circuits add another 200 milliseconds of satellite delay in total round-trip time to the home agent. The minimum time between Registration Requests must not be less than 1 second.

2.2.3.6 Role of foreign agent in Registration

The foreign agent plays a typically inactive role in Mobile IP registration. If it is not detectable by link-layer then the foreign agent has to advertise its presence by sending periodic Agent advertisement messages. Foreign agent decapsulates datapackets to provide the mobile node with care-of address. The responsibility of foreign agent is to transmit a Registration Request, only if the request is being relayed from a mobile node to its home agent. Foreign agent is not allowed to transmit a registration reply in case it denies its service to the mobile node. And if mobile nodes lifetime is expired, foreign agent must not generate a registration request. A foreign agent also must not initiate a Registration Request message that requests for deregistration of a mobile node; however, it must dispatch well-formed (de)Registration Requests originated by a mobile node.

A foreign agent must not communicate a Registration Reply except when sending a Registration Reply received from a mobile node's home agent. In particular, a foreign agent must not generate a Registration Reply because a mobile node's registration Lifetime has terminated [33].

2.2.4 Routing in MIPv4

Routing refers to the method in which datagrams move from a mobile node of one network to the mobile nodes or home agents or (possibly) foreign agents of another network. The pictorial representation of Routing is given in figure 2.4.

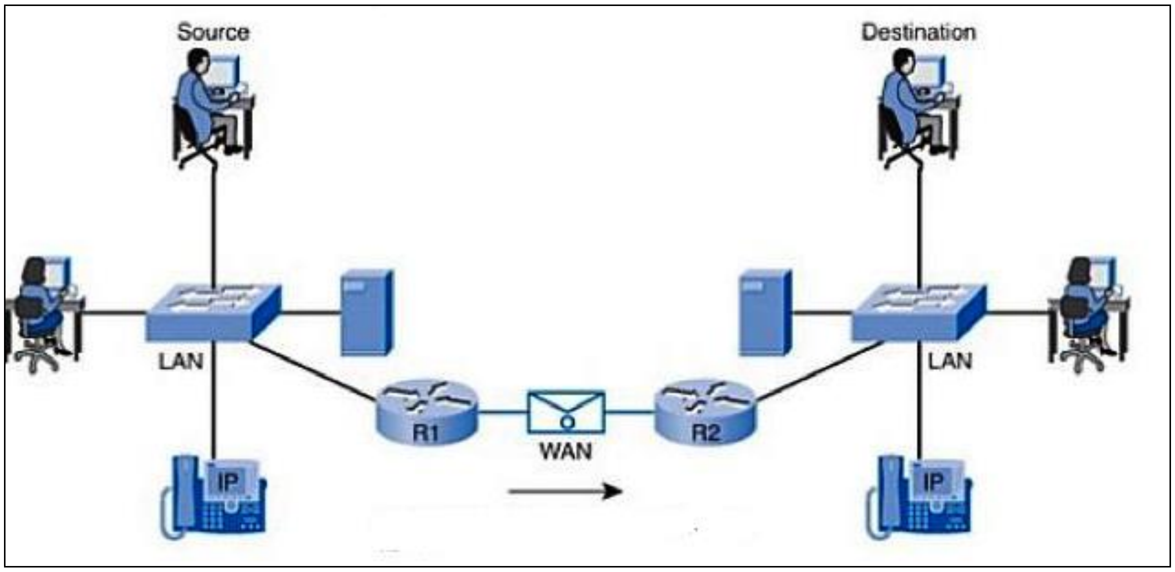


Figure 2.4 Illustration of Routing in MIPv4

Once host name is resolved to an IP address, the IP packet is sent to the host. This host also termed as mobile node notifies its home agent of its current location using registration procedure as mentioned in the previous topic. Once the Registration procedure is complete, next step is Routing which involves elements like TCP/IP host and an IP router. The host and router need to determine on how a datapacket is forwarded. To make these determinations, IP layer consults a routing table stored in memory. Routing table entries are created by default when TCP/IP initializes and additional entries are added either manually by a system administrator or automatically through communication with routers.

2.2.4.1 Types of Delivery

There are two types of delivery based on whether the final destination is located on a directly attached network. IP packets use at least one of the two types of delivery, Direct and Indirect Delivery as shown in figure 2.5.

Direct delivery: When a mobile node forwards a packet to final destination on a directly attached network it is termed as direct delivery. The mobile node encapsulates datapacket in a frame format for Network Interface layer (such as Ethernet or Token Ring) addressed to destination’s MAC address.

Indirect delivery: When a mobile node forwards a packet to an intermediate node which can be an IP router because final destination is not on a directly attached network. The IP node encapsulates the IP packet in a frame format for Network Interface layer (such as Ethernet or Token Ring) addressed to the IP router’s MAC address.

IP routing is a combination of direct and indirect deliveries. In the following figure, when sending packets to node B, node A performs a direct delivery. When sending packets to node C, node A performs an indirect delivery to Router 1, and Router 1 performs an indirect delivery to Router 2, and then Router 2 performs a direct delivery to node C.

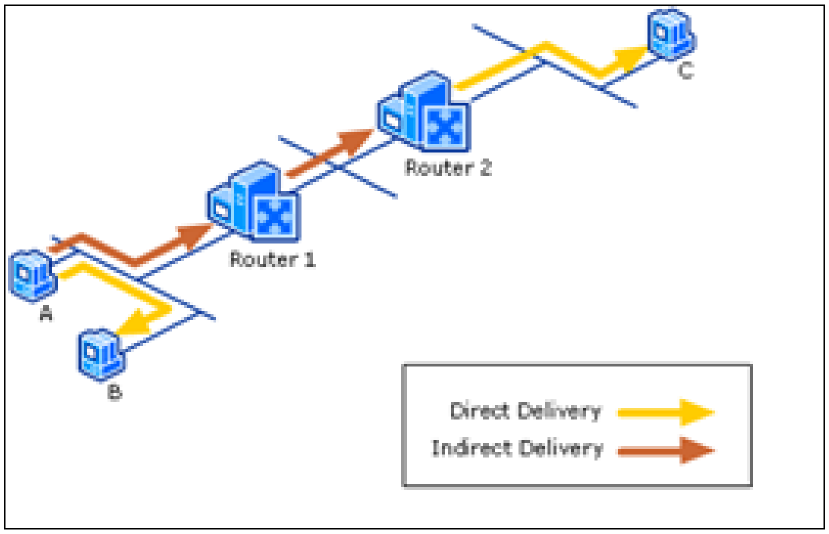


Figure 2.5 Direct and Indirect Delivery

2.2.4.2 IP Routing Table

The information about IP networks is stored in a file called Routing Table which stores records on how data packets can be reached to its respective destination (either directly or indirectly). Since routing is an important aspect for all mobile nodes, every node loading the TCP/IP protocol has a routing table. There are a series of default entries according to configuration of the node and additional entries can be entered either manually through TCP/IP utilities or dynamically through interaction with routers.

When a data packet is to be forwarded, the routing table is used to determine next-hop IP address and next-hop interface. The next-hop IP address for a direct delivery will be the destination IP address in the data packet. For an indirect delivery, the next-hop IP address will be the IP address of a router. The next-hop interface identifies the physical or logical interface, such as a network adapter, that is used to forward the packet to either its destination or the next router.

2.2.4.3 Routing Table Entry Types

Routing table contains the following entries:

i. Network ID: A specific subnet is identified by Network ID. It can be a summarized route, or an IP address for a host route.

ii. Network mask: The pattern that is used to match a destination IP address to network ID.
iii. Next hop: The IP address of next hop.
iv. Interface: An indication of which network interface is used to forward the IP packet.
v. Metric: A number used to indicate cost of route so the best route among possible multiple routes to the same destination can be selected. A common use of metric is to indicate the number of hops (routers crossed) to network ID.

Entries in routing table can be used to store following types of routes:

i. Directly attached network ID: It is the route for network IDs that are directly attached.For directly attached networks, the Next Hop field can be blank or contain the IP address of interface on that network.
ii. Remote network ID: It is the route for network IDs that are available across other routers and are not directly attached. For remote networks, Next Hop field is IP address of a local router.
iii. Host route: It is the route to a specific IP address. Host routes allow routing to occur on a per-IP address basis. For host routes, network ID is IP address of specified host and network mask is 255.255.255.255.
iv. Default route. The default route is designed to be used when a more specific network ID or host route is not found. The default route network ID is 0.0.0.0 with a network mask of 0.0.0.0.

2.2.4.4 Route Determination Process

The following process is used to determine which routing table entry is used to find next hop address and interface:

i. For every individual entry in a routing table, a bit-wise logical AND operation is performed between destination IP address and network mask. The result is then compared with network ID of the entry for a match.
ii. All matching routes are compiled to form a list. In this list route that has longest match (the route with the largest number of bits that match the destination IP address) is chosen. The longest matching route is the most direct route to the destination IP address. In case of multiple matching entries (for example, multiple routes to the same network ID), the router uses the lowest metric to select the best route. If multiple entries have the same

longest match and the lowest metric, any one of them, either the longest match or the lowest metric is selected as the routing table entry.

iii. The end result of route-determination process is a single route in the routing table that yields a next-hop IP address and interface.

If route-determination process fails to find a route, a routing error occurs which is sent to upper layer protocol, such as TCP or UDP.

2.2.4.5 Maintenance of Routing Table Entries

On large internetworks, the network administrators encounter a challenge on how to maintainrouting tables on their routers such that traffic flow chooses the best path to travel and also fault tolerance is maintained. This requires every router be configured with remote network IDs or default route thus resulting in a capable routing between routers in the network. There are two approaches of maintaining routing table entries.

i. Manual

Remote network IDs are not discovered by static routers and so must be configured manually. Static routing depends on manual administration of routing table. If a static router fails, neighboring routers do not sense the fault and inform other routers. It proves that static routers are not fault-tolerant. Thus Static IP routers have routing tables that do not change unless manually changed by a network administrator.

ii. Automatic

Remote network IDs are discovered by dynamic routers and automatically entered into the routing table. The routing table is updated dynamically through exchange of routing information between routers. Dynamic routing employs use of routing protocols, such as Routing Information Protocol (RIP) and Open Shortest Path First (OSPF). If a dynamic router goes down, the fault is detected by neighboring routers, which send the changed routing information to other routers in the internetwork. It proves that Dynamic routers are fault-tolerant. Thus Dynamic IP routers have routing tables that change automatically based on exchange of routing information with other routers [34].

2.2.4.6 Role of Mobile node in Routing

A mobile node operates without the support of mobility services when it is connected to its home network. That is, it operates in same way as any other (fixed) host or router [26]. The mobile node chooses a default router by the following rules when it is registered on a foreign network

Rule 1: Mobile node may use its foreign agent as a first hop router if it is registered using a foreign agent care-of address. The MAC address of foreign agent can be achieved from foreign agent's Agent advertisement message. Else, Mobile node must choose its default router from among the router addresses advertised in ICMP Router Advertisement portion of that Agent Advertisement message.

Rule 2: If mobile node is registered directly with its home agent using a co-located care-of address, then mobile node should choose its default router from among those advertised in any ICMP Router Advertisement message that it receives. If mobile node's externally obtained care-of address matches IP source address of Agent Advertisement within the network prefix, the mobile node may also consider that IP source address as another possible choice for IP address of a default router. The network prefix may be acquired from Prefix-Lengths Extension in Router Advertisement.

2.3 COEXISTENCE PHASE:

The new generation of internet protocol IPv6 will replace IPv4 very soon in order to acquire sufficient IP addresses and guarantee the quality of service (QoS). The IPv6 networks will most likely be through gradual deployment, resulting in IPv4/IPv6 coexisted networks environment with interconnected IPv4 and IPv6 networks. The network's packet filters, routing policies, and packet transformations all must be taken into account to even ask the simple and very important question of "can MIPv4-MIPv6 coexist?" Thus the following section deals with the coexistence phase. The time span from the period when IPv4 addresses get exhausted till the phase when IPv6 is implemented is termed as Coexistence phase. This phase requires a long-term development process for transition from IPv4 to pure IPv6 networks, so IPv4 and IPv6 coexists for a long time. The movement from IPv4 to IPv6 is not a straightaway process; it requires developing mechanisms so that IPv4 and IPv6 coexist together for at least 20 years and during this transition period IPv4 network will totally disappear. The transition and interoperation between IPv4 and IPv6 must be carried out in such a way that entire deployment would be smooth and completely trouble free. The process of transitioning from IPv4 network to new version of internet protocol IPv6 is being considered as one of the current topic in the area of ubiquitous network environment, with a goal to achieve high level compatibility and clear procedure for easy and autonomous deployment of IPv6. The key to a successful IPv6 transition is compatibility with the large installed base of IPv4 hosts and routers. Maintaining compatibility with IPv4 while deploying IPv6 would be the ultimate goal of any network designers. During this transition, network designers have specified three major protocols that are identified to persist. They are mentioned as follows

- Dual Stack
- Tunneling and
- Translation

During this transition or Coexistence phase messages are sent and received on communicating systems to establish reliable communications. The Protocols involved during this transition should therefore specify rules governing the transmission of datapackets.

2.3.1 Description of Dual Stack

This section presents Dual stack mechanism

2.3.1.1 Dual Stack Transition Mechanism DSTM

Basically, Dual Stack Transition Mechanism DSTM consists of the DSTM server, the DSTM Tunnel End Point (TEP), and DSTM hosts, DNS-ALG (Application Layer Gateway). The DSTM server performs following two functions,

a. Assign temporary IPv4 global addresses to DSTM hosts,
b. Maintain the mapping between the allocated IPv4 address and the permanent IPv6 address of each DSTM host.

The function of DSTM TEP is to tunnel all the IPv4 packets to DSTM hosts. And therefore DSTM TEP is located on the boundary of IPv6 and IPv4 networks. The working of DSTM is explained in detail by the following steps,

- **Step 1**: The IPv4 host (i.e. the host within the IPv4 network) attempts to start a conversation with the DSTM host by sending a DNS query message of ‘A’ record for the DSTM host. This is sent to the IPv4 DNS Server.
- **Step 2:** Since the IPv4 DNS server does not have DNS information about the DSTM host, it forwards this message to the DNS-ALG.

- **Step 3**: The DNS-ALG on the DSTM server translates the ‘A’ record into the ‘A/AAAA’ record and forwards it to the IPv6 DNS server [35].
- **Step 4**: The IPv6 DNS Server sends a reply message with ‘AAAA’ record to the DNS-ALG.
- **Step 5**: The IPv6 DNS Server requests for IPv4 address corresponding to its IPv6 address, this request is sent to DSTM Server.
- **Step 6**: If the DSTM server receives the DNS reply message only with ‘AAAA’ record, it assigns a temporary IPv4 global address to the DSTM host from its IPv4 address pool. The DSTM server holds the mapping between the allocated IPv4 address and the IPv6 address of the DSTM host in its mapping table and sets the lifetime timer with the value of the amount of the time during which the allocated IPv4 address is considered to be valid.
- **Step 7**: And then the DSTM server dynamically registers the allocated IPv4 address to the IPv6 DNS server [36].
- **Step 8**: And sends a DSTM address allocation message to the DSTM host to notify the allocated IPv4 address, the IPv6 address of the DSTM TEP, and the lifetime.
- **Step 9**: When the DSTM host receives the DSTM address allocation message from the DSTM server, it caches the allocated IPv4 address and the IPv6 address of the DSTM TEP and sends a DSTM address allocation acknowledgement message to the DSTM sever to notify that it is successfully processed.
- **Step 10**: After the DSTM server receives the acknowledgement from the DSTM host, the DSTM server requests for a reply with IPv4 address.
- **Step 11**: The DNS-ALG on the DSTM server translates the DNS reply message of ‘AAAA’ record into ‘A’ record with the allocated IPv4 address and sends it to the IPv4 DNS server.
- **Step 12**: When the IPv4 DNS server receives the reply message with ‘A’ record, it forwards this message to the IPv4 host.
- **Step 13**: After the IPv4 host receives the DNS reply message with the DSTM host’s IPv4 address, it sends IPv4 packets to the DSTM host through the DSTM TEP.
- **Step 14**: When the DSTM TEP receives IPv4 packets from the IPv4 host, it searches its mapping table for the mapping entry and sends a message to the IPv6 address pool

requesting for DSTM host's IPv6 address corresponding to the destination IPv4 address of the IPv4 packet.

- **Step 15**: If the DSTM TEP finds the mapping entry from its mapping table, then it encapsulates the IPv4 packet in an IPv6 packet and replies the tunneled packet to the DSTM host through the DSTM TEP.
- **Step 16**: When the DSTM TEP receives this message, it caches the mapping between the IPv4 address and the IPv6 address of the DSTM host in its mapping table and sets its lifetime timer. The DSTM TEP encapsulates the IPv4 packet in an IPv6 packet by using the mapping information and sends it to the DSTM host.
- **Step 17**: The DSTM host decapsulates the IPv6 header to get the IPv4 packet. With the cached DSTM TEP's IPv6 address, the DSTM host encapsulates the IPv4 packet in an IPv6 packet and sends it to the DSTM TEP. The tunneled packet is directed the DSTM TEP.
- **Step 18:** And finally the DSTM TEP decapsulates the IPv6 header and forwards it to the IPv4 network [37]. All the steps involved to realize Dual stack is shown figure 2.6 and table 2.1.

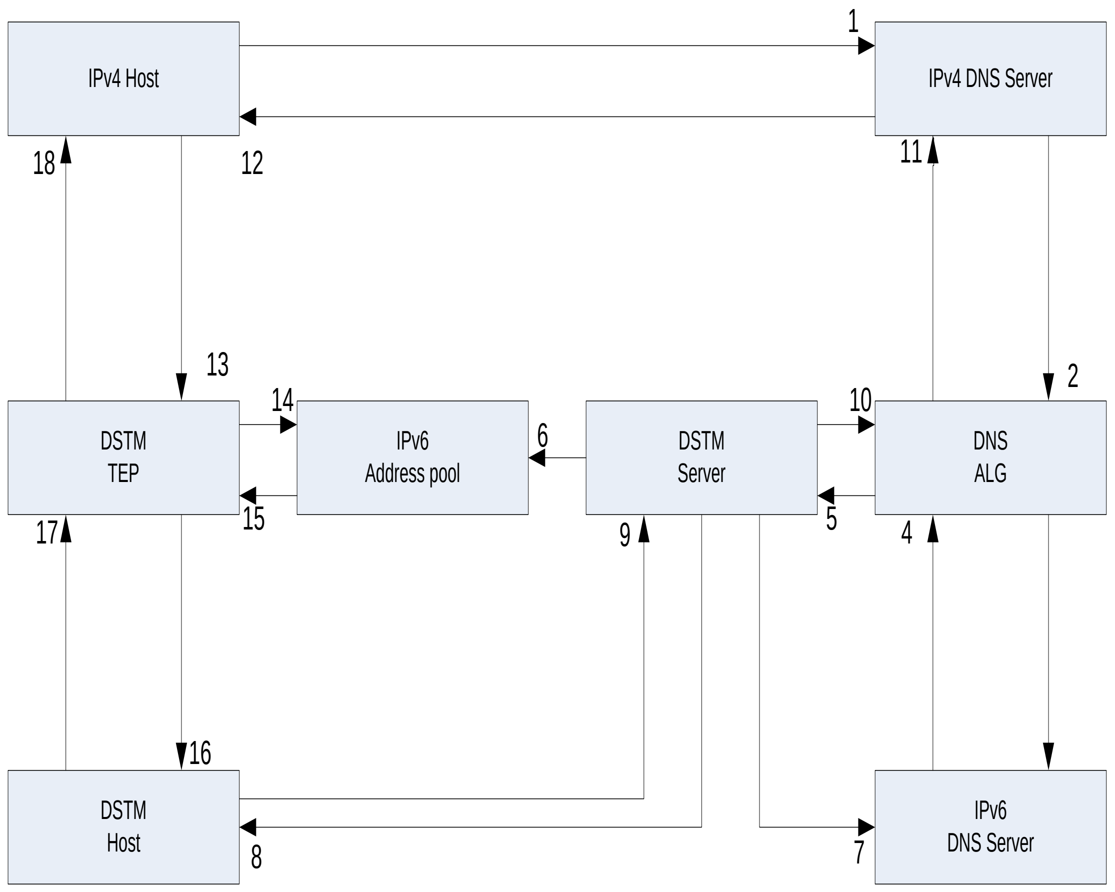


Figure 2.6 Dual Stack Transition Mechanism

Table 2.1 Description of Dual Stack Transition Mechanism

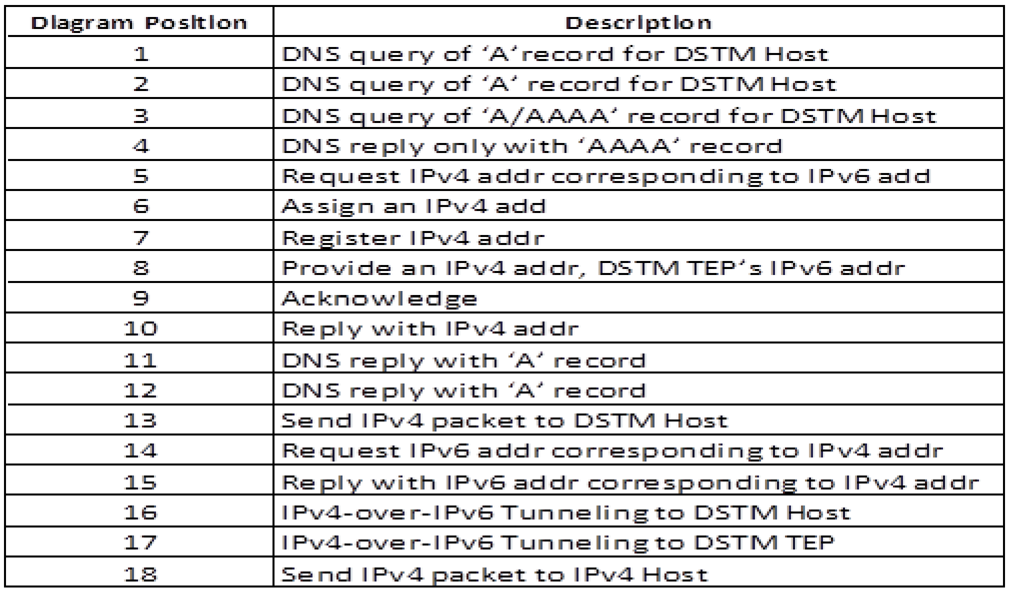


Diagram Position	Description
1	DNS query of ‘A’ record for DSTM Host
2	DNS query of ‘A’ record for DSTM Host
3	DNS query of ‘A/AAAA’ record for DSTM Host
4	DNS reply only with ‘AAAA’ record
5	Request IPv4 addr corresponding to IPv6 add
6	Assign an IPv4 add
7	Register IPv4 addr
8	Provide an IPv4 addr, DSTM TEP’s IPv6 addr
9	Acknowledge
10	Reply with IPv4 addr
11	DNS reply with ‘A’ record
12	DNS reply with ‘A’ record
13	Send IPv4 packet to DSTM Host
14	Request IPv6 addr corresponding to IPv4 addr
15	Reply with IPv6 addr corresponding to IPv4 addr
16	IPv4-over-IPv6 Tunneling to DSTM Host
17	IPv4-over-IPv6 Tunneling to DSTM TEP
18	Send IPv4 packet to IPv4 Host

In case if the lifetime of the allocated IPv4 address is expired, the following changes happen, the DSTM server removes the mapping entry from its mapping table and the allocated IPv4 address gets removed from the DSTM host. Then registered IPv4 address is dynamically deleted from the IPv6 DNS server and finally the lifetime of a DSTM server's mapping table entry gets expired.

If the DSTM host wants to use the allocated IPv4 address even after the timer expiration, it can make a request to extend the lifetime by sending a DSTM address extension request message to the DSTM server. The DSTM server sends a DSTM address extension acknowledgement message to the DSTM host after the extension of the lifetime. If the DSTM TEP knows the mapping entry is still in use, it sends a DSTM binding request message to the DSTM server to refresh the entry.

2.3.1.2 Benefits of Dual Stack

The benefits of dual stack are that it can be deployed on hosts, routers that use the same interface as IPv4. Therefore during transition from IPv4 to IPv6, a lot of changes do not have to be countered. It also deals with most of the address selection and DNS resolution issues which allows host to continue to reach IPv4 resources. It does so while adding IPv6 functionality which is simple to deploy. It also allows backwards compatibility. Some of the fore mentioned benefits are that it is available on most platforms and is quite user friendly.

2.3.2 Description of Tunneling

This section deals with the mechanisms involved in Tunneling.

2.3.2.1 Tunneling Mechanisms:

In present scenario, IPv4 routing infrastructure does not have any functional problems. The only major problem faced in IPv4 is its lack of sufficient address, in other words, due to insufficient address there is an urgent requirement to shift to IPv6. Compared to IPv4, the most obvious advantage of IPv6 is its larger address space. IPv4 addresses are 32 bits long and number about 4.3 billion. On the other hand, IPv6 addresses are calculated to be 2^ 128 bits long and the result number is 39 digits long given by 340 282 366 920 938 463 463 374 607 431 168 211 456 which is read out aloud as 340 undecillion. So obviously there is a need to shift to IPv6. Deploying IPv6 involves IPv6 routing infrastructure and building an IPv6 routing infrastructure will require ample amount of time. While the IPv6 infrastructure is being deployed, the existing IPv4 routing infrastructure can remain functional and can be used to carry IPv6 traffic. Tunneling provides a way to utilize an existing IPv4 routing infrastructure to carry IPv6 traffic [38].

Tunneling can be carried out in a number of ways as follows:

1. Router-to-Router Tunneling: IPv6/IPv4 routers interconnected by an IPv4 infrastructure can tunnel IPv6 packets between themselves.
2. Host-to-Router Tunneling: IPv6/IPv4 hosts can tunnel IPv6 packets to an intermediary IPv6/IPv4 router that is reachable via an IPv4 infrastructure.
3. Host-to-Host Tunneling: IPv6/IPv4 hosts that are interconnected by an IPv4 infrastructure can tunnel IPv6 packets between themselves.
4. Router-to-Host Tunneling: IPv6/IPv4 routers can tunnel IPv6 packets to their final destination IPv6/IPv4 host.

As there is a need to configure tunneling endpoints, router-to-router is mostly used. However all the above cases can be used to configure tunneling.

2.3.3 Description of Translation

This section deals with the detailed description of Translation in MIPv4 / MIPv6

2.3.3.1 Translation Goals

Whether a person is having an IPv4 stack and connectivity or IPv6 stack and connectivity, in an ideal situation, any network operator would wish to create a system and an application that should be able to interchange or swap datagrams with any other system running the same application.

So the question raised here is, what are different kinds of connectivity that can be easily supported? What kinds are harder? And what technologies are needed to at least pick the low-hanging fruit. All applications today fall into two main categories [39].

- Client/Server Application: Client/server describes the relationship between two computer programs in which the client makes a service request and server fulfills the request. In networking, the behavior of the applications is that connections are initiated from client software systems to server software systems.

  Examples include mail handling between an end user and his mail system, the web, and DNS name resolution.

- Peer-to-Peer (P2P) Application: A P2P application is an application that uses the same endpoint to initiate outgoing sessions to peering hosts as well as accept incoming sessions from peering hosts. P2P Applications can be broadly classified into two categories:
    - Peer-to-peer infrastructure applications and
    - Peer-to-peer file exchange applications

1. Peer-to-peer infrastructure applications:

An important characteristic of these applications is that they use short-lived sessions which means they open sessions when they are needed and close them when they are done. Examples of "infrastructure applications" include Simple mail transfer protocol (SMTP) between Message transfer agents (MTAs), Network News, and Session Initiation Protocol (SIP).

2. Peer-to-peer file exchange applications:

These are applications that open sessions between systems and leave them open for long periods of time, and in situations where short-lived sessions are important, these applications can get required information about reliability of peers from history or by reputation. Examples of these include Limewire, BitTorrent, and UTorrent.

2.3.3.2 Classification of Proposed solution for IPv4/IPv6 translation

Currently, the proposed solutions for IPv6/IPv4 translation are classified [40] into

- Stateless translation and
- Stateful translation.

1. Stateless Translation

For stateless translation, translation information is carried in the address and configuration information is carried in the translators which permit the initiation of both IPv4->IPv6 & IPv6->IPv4 sessions. Stateless translation supports end-to-end address transparency and has better scalability compared with stateful translation [41].

2. Stateful Translation

For stateful translation, the translation state is maintained between IPv4 and IPv6 address/port pairs, enabling IPv6 systems to open sessions with IPv4 systems.

The following points must be considered before choosing stateless or stateful translation.

(i) No matter the direction is IPv6->IPv4 or IPv4->IPv6, a technology is needed that will permit systems acting as clients to be able to open sessions with other systems acting as servers. This in fact is stateless Translation. Majority of accesses will be to servers in a carrier infrastructure which optimizes access to them. However, if the complexity is reduced and a stateless algorithm cannot be developed, a stateful algorithm is acceptable.

(ii) Whether the direction is IPv6->IPv4 or IPv4->IPv6, a technology is needed that will permit peers to connect with each other and if this was stateless it would be an ideal case. However, if the complexity is reduced and a stateless algorithm cannot be developed, a stateful algorithm is acceptable.

(iii) Stateless or Stateful algorithm is not required in some cases to enable connections to the hosts where hosts are purely clients.

IPv4/IPv6 translation cases are classified into 4 types:

1. Interoperation between an IPv6 network and the IPv4 Internet
2. Interoperation between an IPv4 network and the IPv6 Internet
3. Interoperation between an IPv6 network and an IPv4 network
4. Interoperation between the IPv6 Internet and the IPv4 Internet

Whether the communication is initiated by IPv6 side or the IPv4 side each translation case is sub divided into 2 scenarios which ultimately leads to 8 scenarios:

1. IPv6 network to the IPv4 Internet
2. IPv4 Internet to an IPv6 network
3. IPv6 Internet to an IPv4 network
4. IPv4 network to the IPv6 Internet
5. IPv6 network to an IPv4 network
6. IPv4 network to an IPv6 network
7. IPv6 Internet to the IPv4 Internet
8. IPv4 Internet to the IPv6 Internet

1. An IPv6 Network to the IPv4 Internet: Due to the lack of IPv4 addresses or due to other technical and economic constraints, the network is IPv6-only, but the hosts in the network require communicating with the global IPv4 Internet. Figure 2.7 explains An IPv6 Network to the IPv4 Internet.

Ex: Wireless NW, enterprise N/W, sensor N/W.

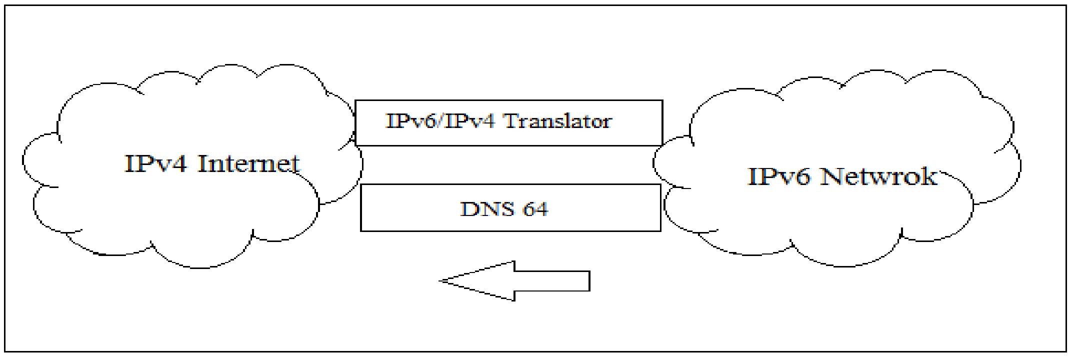


Figure 2.7 An IPv6 Network to the IPv4 Internet

Both stateless and stateful solutions can support this Scenario.

2. The IPv4 Internet to an IPv6 Network: When the enterprise networks or ISP networks adopt Scenario 1, the IPv6-only users will not only want to access servers on the IPv4 Internet but also will want to setup their own servers in the network that are accessible by users on the IPv4 Internet. Thus, with a translation solution for this scenario, the benefits would be clear. Not only could servers move directly to IPv6 without trudging through a difficult transition period, but they could do so without risk of losing connectivity with the IPv4-only Internet. Figure 2.8 explains The IPv4 Internet to an IPv6 Network.

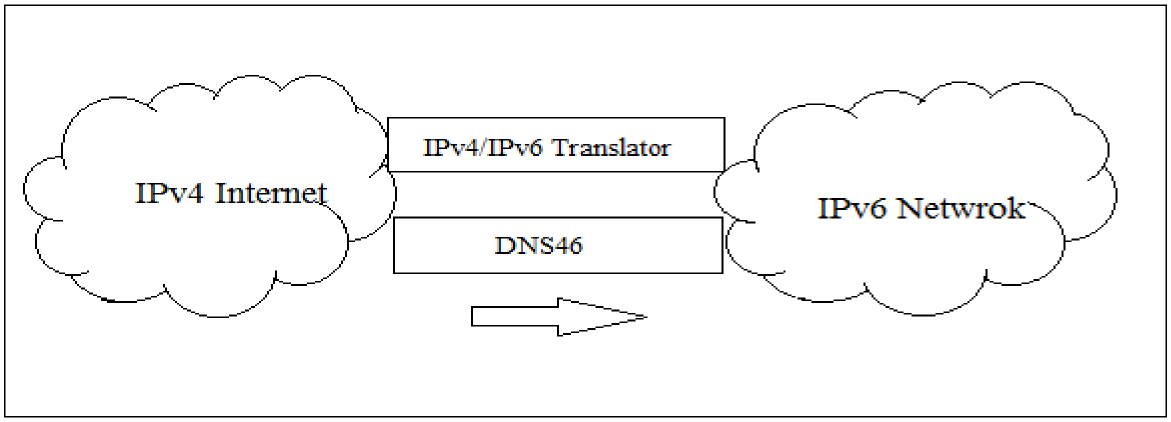


Figure 2.8 The IPv4 Internet to an IPv6 Network

Stateful translation such as NAT-PT can be used in this scenario, but it requires a tightly coupled DNS Application Level Gateway (ALG) in the translator [42, 43].

The stateless translation solution also can work in this Scenario, since it can support IPv4-initiated communications with a subset of the IPv6 addresses (IPv4-translatable addresses) in an IPv6 network.

3. The IPv6 Internet to an IPv4 Network: An inheritance of IPv4 network is required to provide services to IPv6 hosts. In this case, a Network Specific Prefix assigned to the translator will give the hosts unique IPv4-converted IPv6 addresses. There is no need to synthesize AAAA from A records, since static AAAA records can be put in the regular DNS to represent these IPv4-only hosts. Figure 2.9 explains the IPv6 Internet to an IPv4 Network.

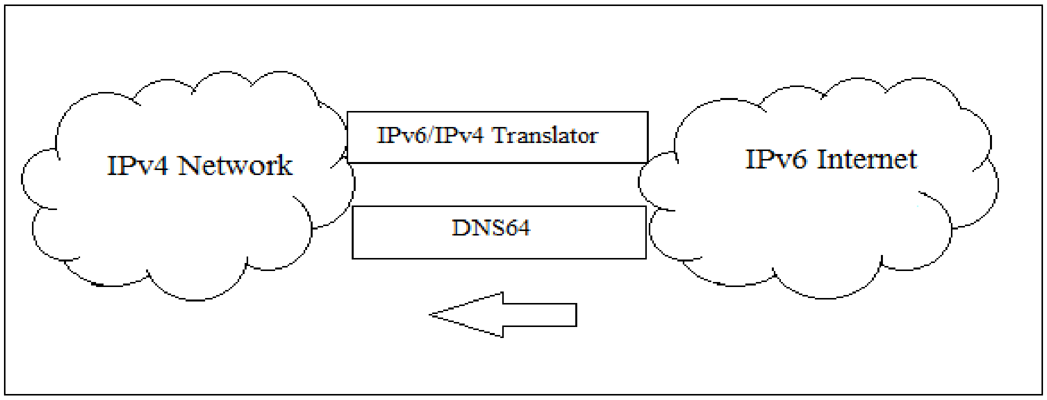


Figure 2.9 The IPv6 Internet to an IPv4 Network

Stateless translation will not work for this scenario, as stateless can only support a subset of the IPv6 addresses. However, IPv6-initiated communication can be achieved through stateful translation.

4. An IPv4 Network to the IPv6 Internet: Due to technical or economic constraints, the network is IPv4-only, and IPv4-only hosts (applications) may require communicating with the global IPv6 Internet. This scenario will probably only occur when we are well past the early stage of the IPv4, and the IPv4/IPv6 transition has already moved to the right direction. Figure 2.10 explains An IPv4 Network to the IPv6 Internet.

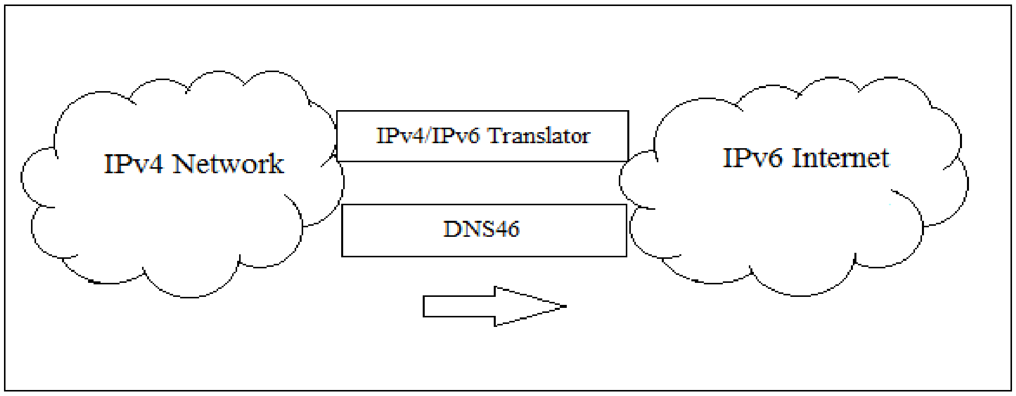


Figure 2.10 An IPv4 Network to the IPv6 Internet

Stateful translation such as NAT-PT can be used in this scenario, but it requires a tightly coupled DNS ALG in the translator.

5. An IPv6 Network to an IPv4 Network: In this scenario, both an IPv4 network and an IPv6 network are within the same organization. The IPv4 addresses used are either public IPv4 addresses or private addresses. The IPv6 addresses used are either public IPv6 addresses. The stateful and stateless translation schemes apply here. Figure 2.11 explains an IPv6 Network to an IPv4 Network.

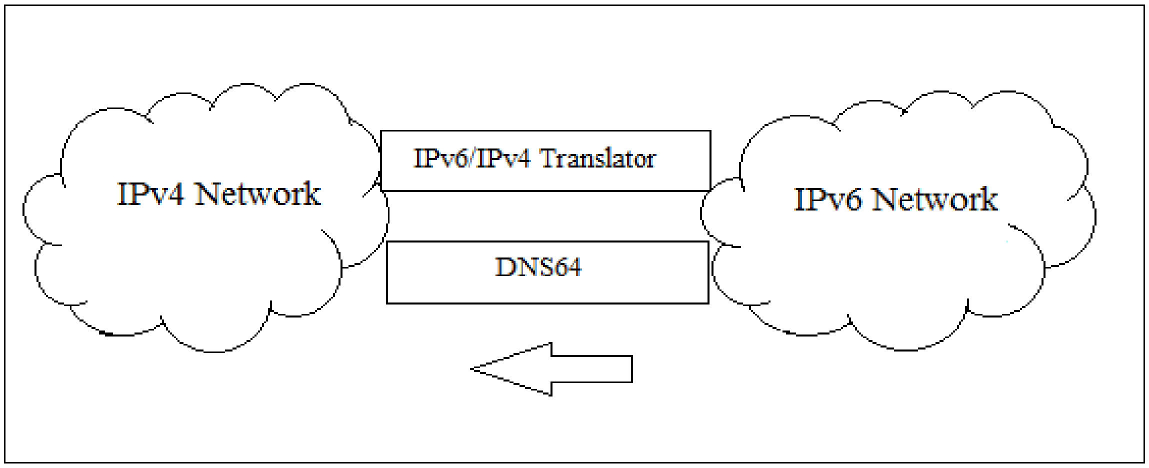


Figure 2.11 An IPv6 Network to an IPv4 Network

6. An IPv4 Network to an IPv6 Network: This is another scenario when both an IPv4 network and an IPv6 network are within the same organization. The IPv4 and IPv6 addresses used are either public IPv4 addresses or private addresses or ULAs (Unique Local Addresses). Figure 2.12 an IPv4 Network to an IPv6 Network.

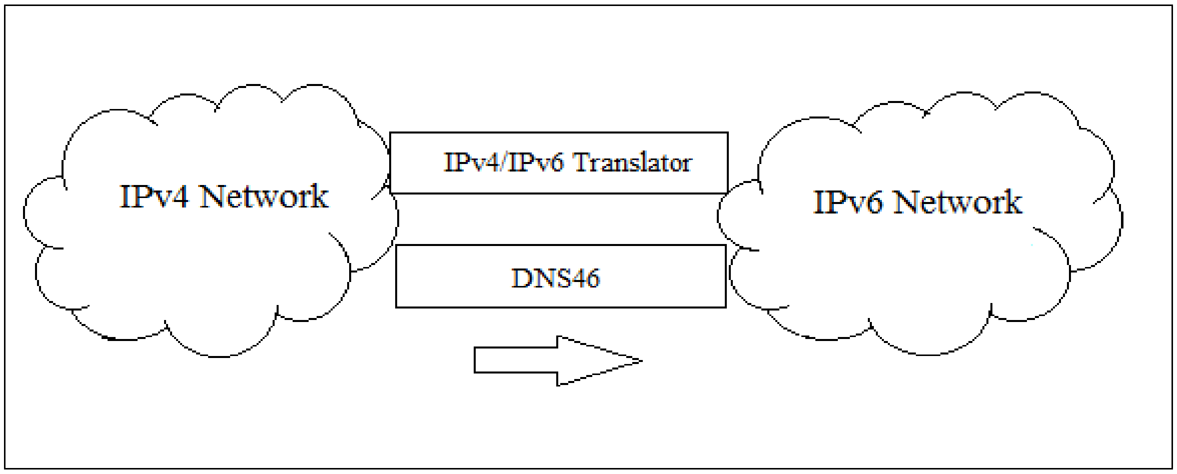


Figure 2.12 An IPv4 Network to an IPv6 Network

7. The IPv6 Internet to the IPv4 Internet: Any IPv6-only host or application on the global Internet can initiate communication with any IPv4-only host or application on the global Internet. Due to the huge difference in size between the address spaces of the IPv4 Internet and the IPv6 Internet, there is no viable translation technique to handle unlimited IPv6 address translation. Figure 2.13 explains the IPv6 Internet to the IPv4 Internet.

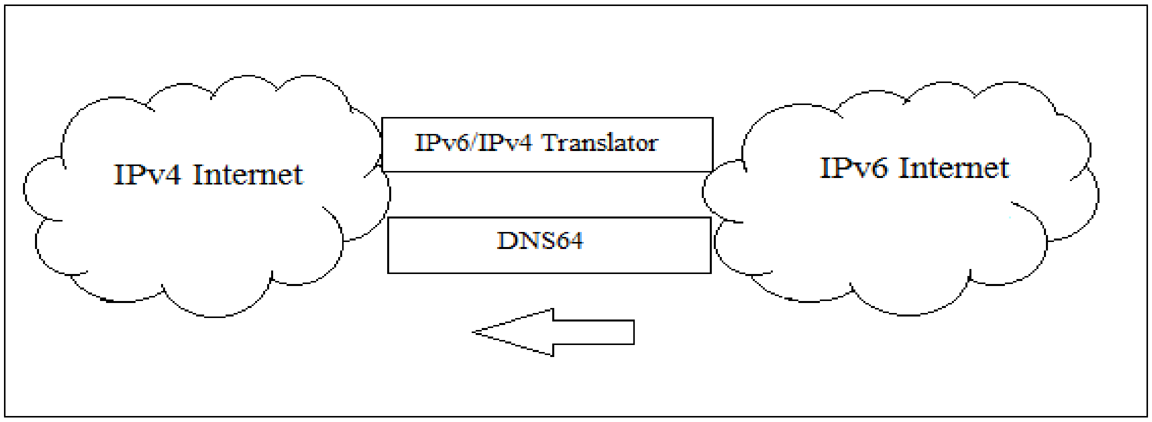


Figure 2.13 The IPv6 Internet to the IPv4 Internet

8. The IPv4 Internet to the IPv6 Internet: This case is very similar to Scenario 7. The analysis and conclusions for Scenario 7 also apply for this scenario. Figure 2.14 explains The IPv4 Internet to the IPv6 Internet.

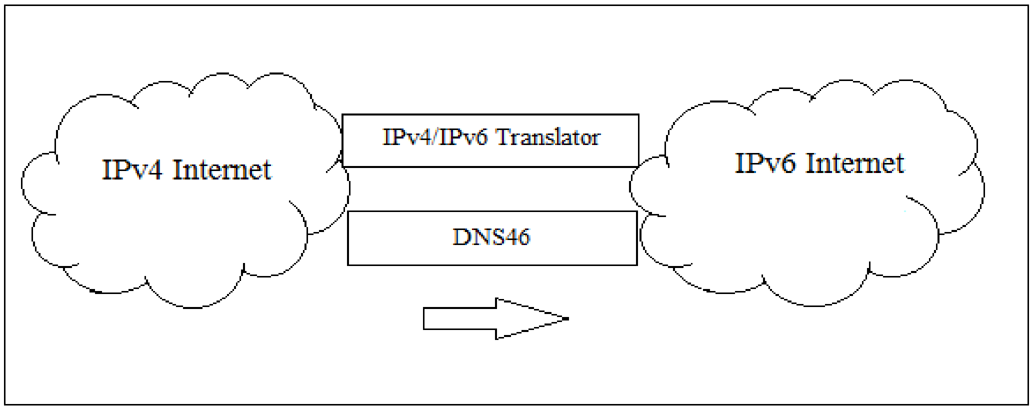


Figure 2.14 An IPv4 Internet to the IPv6 Internet

2.4 Mobile IPV6

The following section deals with the detailed description of MIPv6

2.4.1 Introduction

Mobile IPv6 is a protocol developed as a subset of Internet Protocol version 6 (IPv6) to support mobile connections. MIPv6 is an update of the IETF (Internet Engineering Task Force) Mobile IP standard designed to authenticate mobile devices (known as mobile nodes) using IPv6 addresses [8].

When a mobile node moves from one network to another, or when a notebook computer is used in more than one location, one way to accommodate the changes in the network is to use two or more sets of network configuration to match each location. This solution seems to be rather unachievable task. However, another solution is to use a mobile IP protocol to allow the computer to use one IP address for all its communications across any network to which it is attached and this solution seems to be a practical option.

Mobile IPv6 provides mobility support for IPv6. It allows you to keep the same internet address all over the world, and allows applications using that address to maintain transport layer and upper-layer connections when changing locations. It allows mobility across homogenous and heterogeneous media. MIPv6 allows a mobile node to transparently maintain connections while moving from one subnet to another. Each device is identified by its home address although it may be connecting to through another network. When connecting through a foreign network, a mobile device sends its location information to a home agent, which intercepts packets, intended for the device and tunnels them to the current location. For example, Mobile IPv6 facilitates node movement from one Ethernet segment to another as well as it facilitates node movement from an Ethernet segment to a wireless LAN cell, with the mobile node's IP address remaining unchanged in spite of such movement.

2.4.2 Basic Operation

A mobile node uses a home address when communicating with other nodes. When a mobile node moves from one network to another network, the node sends a message called a Binding Update to its home agent. The message includes the care-of address and the home address of the mobile node. Such information is called binding information since it binds a care-of address to the home address of a mobile node. When a home agent receives the message and accepts the contents of the message, the home agent replies with a Binding Acknowledgment message to indicate that the Binding Update message is accepted. The home agent creates a bidirectional tunnel connection from its address to the care-of address of the mobile node. A mobile node also creates a bidirectional tunnel connection from its care-of address to the home agent when it receives the acknowledgment message. After the successful tunnel creation, all packets sent to the home address of the mobile node are intercepted by the home agent at the home network and tunneled to the mobile node. Also, all packets originated at the mobile node are tunneled to its home agent and forwarded from its home network to destination nodes.

If a peer node supports the route optimization mechanism defined in the Mobile IPv6 specification, the mobile node and the peer node can communicate directly without detouring through the home agent. To optimize the route, the mobile node sends a Binding Update message to the peer node. After it receives the message, the peer node sends packets directly to the care-of address of the mobile node. The packets also contain a Routing Header that specifies their final destination that is set to the home address of the mobile node. The packets are routed directly to the care-of address of the mobile node. The mobile node receives the packets and finds that the packets have a Routing Header and performs Routing Header processing, which involves swapping the destination address in the packet's IPv6 header and the home address carried in the Routing Header. The mobile node forwards the packets to the final destination that is the home address at this point, and the packets are delivered to the mobile node itself. When the mobile node sends packets to the peer node, the mobile node sets its care-of address as a source address of the packets and inserts its home address into a Destination Options Header. The peer node swaps the care-of address and the home address when it receives those packets, and processes the packets as if they were sent from the home address. The operations involved in MIPv6 is categorized as

- Home Agent Operations
- Correspondent Node Operations
- Mobile Node Operations

2.4.2.1 Home Agent Operations

The function of home agent is to record information about each router and maintain the Home Agents entry List, this list is used as address discovery mechanism [9]. Home Agent is responsible for the following operations

1. Care-of Address Registration
2. Care-of Address De-Registration:
3. Processing Mobility Headers
4. Processing Data packets:
5. Handling Reverse-Tunneled Packets:

1. Care-of Address Registration

When Binding Update requests the registration of the mobile node's primary care-of address following tests are performed

a. The node implementation test,

b. Configuration test to act as home agent,

c. Setting status field by returning Binding Acknowledgement

The home address of the mobile node is provided by the Home Address field as received in the Home Address option. Binding update should be rejected if the home address being IPv6 address with respect to the home agent's current Prefix List and Binding Acknowledgement is returned to the mobile node. Create a new entry or update existing Binding Cache if home agent accepts the Binding Update. The node serving as the home agent for Binding cache will be marked as home registration. The Binding Cache is excluded from the normal cache replacement policy and will not be removed from the Binding Cache until the expiration of the Lifetime period.

Few factors that the lifetime of the Binding Cache entry depends on are as follows,

i. The lifetime for the binding cache must not be greater than the lifetime value specified in the Binding update and it must not be greater than the remaining valid lifetime for the subnet prefix in the mobile node's home address specified with the Binding Update.

ii. The remaining preferred lifetime must not have any impact on the lifetime for the Binding Cache entry and should be removed when the valid lifetime of the prefix associated with it expires.

iii. The home agent can decrease the lifetime for the binding cache and resulting lifetime is stored by it in Binding cache entry, and this entry is deleted after the expiration of the lifetime.

The rules for selecting the Destination IP address (and possibly routing header construction) for the Binding Acknowledgement to the mobile node are the same.

2. Care-of address de-registration

When the mobile node returns home or knows that it will not have any care-of addresses in the visited network, the binding is de-registered and this is done at home network, it chooses to omit the Home Address destination option in which case the mobile node's home address is the source IP address of the de-registration Binding Update.

3. Processing Mobility Headers

There are certain rules followed by every IPv6 home agents when processing Mobility Headers.

i. Mobility Header messages must be discarded by the node if verified checksum is invalid,

ii. Mobility Header field must have a known value else a binding error message will be issued with status field set to 2.

iii. Payload Proto field must be IPPROTO_NONE (59 decimal) else the message will be discarded Source Address of the packet. The Pointer field in the ICMP message should point at the Payload Proto field,

iv. The Pointer field in the ICMP message should point at the Header Len field if the length of Header Len field in this header is less than the total length .The total length of the message is divisible by 8 and a message with ICMP Parameter Problem, Code 0, is sent to the Source Address of the packet [44].

4. Processing Data packets:

Packets containing a Home Address option must be forwarded only if the given home address is a unicast routable address. Mobile nodes can include a Home Address destination option in a packet if they believe the correspondent node has a Binding Cache entry for the home address of a mobile node. The packet will be processed normally if the Next Header value of the Destination Option is 135 for mobility header and 51 for auxiliary header. While sending Packets to a Mobile Node the sending node should examine its Binding Cache Before sending any packet, if the sending node has a Binding Cache entry for this address, the sending node should use a type 2 routing header to route the packet to destination node.

5. Handling Reverse-Tunneled Packets:

A binding must be established between the mobile node and a correspondent node in order to avoid traffic from the mobile node to the correspondent node entering the reverse tunnel unless as follows:

i. Using IPv6 encapsulation [45] the tunneled traffic arrives to the home agent's address .

ii. Reverse-tunneled packets are discarded unless accompanied by a valid encapsulating Security protocol (ESP) header depending on the security policies used by the home agent. The support for authenticated reverse tunneling allows the home agent to protect the home network and correspondent nodes from malicious nodes hidden as a mobile node.

iii. The home agent verifies that the Source Address in the tunnel IP header is the mobile node's primary care-of address when a home agent decapsulates a tunneled packet from the mobile node. Otherwise, any node in the Internet could send traffic through the home agent and escape ingress filtering limitations. This simple check forces the attacker to know the current location of the real mobile node and be able to defeat ingress filtering. If the reverse-tunneled packet is protected by ESP in tunnel mode this check is not necessary [46, 47].

2.4.2.2 Correspondent Node Operations

The Operations carried out by Correspondent Node is given below

- Receiving Packets with Home Address Option
- Sending Packets to a Mobile Node

i. Receiving Packets with Home Address Option

The Correspondent node must process this option in a manner consistent with exchanging the home address field from the home address option into the IPv6 header and replacing the original value of the source address field there. After all IPv6 options have been processed, the upper layers can process the packet without the knowledge that it came originally from a CoA or that a home address option was used. Packets containing a home address option must be dropped if the given home address is not a unicast routable address. Mobile nodes can include a home address destination option in a packet if they believe the Correspondent Node has a binding cache entry for the home address of a Mobile Node. Packets containing a home address option must be dropped if there is no corresponding binding cache entry. A corresponding binding cache entry must have the same home address as appears in the home address destination option, and the currently registered Care-of Address must be equal to the source address of the packet. These actions are not done for packets that contain a home address option and a Binding Update. If the packet is dropped due these conditions, the CN must send the binding error message.

ii. Sending Packets to a Mobile Node

Before sending any packet, the sending node should examine its binding cache for an entry for the destination address to which the packet is being sent. If the sending node has a binding cache entry for this address, the sending node should use a type 2 routing header to route the packet to this mobile node (the destination node) by way of its Care-of Address. For example, if there are no additional routing headers in this packet beyond those needed by MIPv6, the Correspondent Node could set the fields in the packet's IPv6 header and routing header as follows:

a. The destination address in the packet's IPv6 header is set to the Mobile nodes home address (the original destination address to which the packet was being sent).

b. The routing header is initialized to contain a single route segment, containing the mobile nodes Care-of Address copied from the binding cache entry. The segments left field is, however, temporarily set to zero.

c. If, on the other hand, the sending node has no binding cache entry for the destination address to which the packet is being sent, the sending node simply sends the packet normally, with no routing header.

d. If the destination node is not an MN (or is an MN that is currently at home), the packet will be delivered directly to this node and processed normally by it. However, if the destination node is an MN that is currently away from home, the packet will be intercepted by the MN’s HA and tunneled to the MN’s current primary CoA.

2.4.2.3 Mobile Node Operations

The operations carried out by a Mobile Node is as follows

i. Receiving Packets While Away from Home

ii. Transmission of packets while away from home

i. Receiving Packets While Away from Home: A mobile node will receive packets that are addressed to its home address while away from home. They will use one of the following two methods:

a. Packets that do not have a Binding Cache will be transmitted to the home address, thereby pulled by the home agent and tunneled.
b. Packets that do have a Binding Cache and contain the mobile node’s present care-of address will be sent. The packet is addressed to the node’s care-of address.

For the first method, the node must check if the IP address of its home agent is the same as IPv6 source address of the tunneled packet. A binding update may also be sent by the mobile node to the original sender of the packet in this method. The received packet must all also be processed by the mobile node by IPv6 encapsulation standards. For the second method, the packet will be processed normally by protocols of the upper-layer in the mobile node like it was addressed to the node’s home address.

A packet that is addressed to itself and is received by a node will follow the next header chain of headers and process them accordingly. The following checks will be performed when it encounters a type 2 routing header. The node must delete the packet when any of the following checks fail. 1.The routing header’s length field is precisely 2.

2.The router header segments left field is 1.

3.The routing header’s Home Address field is one of the node’s home addresses.

After fulfilling the above checks, the node switches the IPv6 Home Address field with the destination field in the routing header. It then resubmits the packer for processing to the IP.

ii. Transmission of packets while away from home**:** A mobile node continues using its home address even when it’s away from home. It may also use more than one care-of addresses. To send a packet while being away from home, a mobile node can choose from the following addresses to use as a source of the packet:

1. For most packets, the home address will be used as the IP source address of the mobile node. The mobile node must use the home address to send packets that are part of transport level connections while the mobile node is at home. Similarly, the mobile node should use its home address after changing its location while still being a part of the transport level connections. The mobile node may send the packets directly to the corresponding node depending on the existence of binding. In case of absence of a binding, the mobile node must use reverse tunneling.
2. Alternatively, a mobile node can also decide to use one of its care of addresses as the IP source address of the packet i.e., not using the home address option of the packet. This may especially be useful for short-term communication, which may easily be re-established should it fail. Since no extra options will be used during reply or query, using the care-of-address as the source will have lower overheads compared to using the home address of the mobile node..
3. While away from home, the home address option MUST not be used by the mobile node while communicating with link-local connections. Similarly, the mobile node may not use the home address for IPv6 Neighbor Discovery.

Packets sent while away from home don't require any special Mobile IPv6 processing. Similarly, no special IPv6 processing is required if the mobile node uses any address other than the home address as the source while away from home. The packet is addressed and sent the same way that any normal IPv6 packet would be.

2.4.3 Security Considerations

Understanding the potential threats and harms present in any protocol will lead to take measures in considering the security issues. Some of the Internet security systems in widespread use are IPsec, Transport Layer Security (TLS) and Secure Shell (SSH). Internet Protocol Security (IPsec) is a protocol suite for securing Internet Protocol (IP) communications by authenticating and encrypting each IP packet of a communication session. IPsec is an end-to-end security scheme operating in the Internet Layer of Internet Protocol Suite also IPsec protects any application traffic across an IP network thus IPsec is the most used security scheme. IPsec can be used in protecting data flows between a pair of hosts (host-to-host), between a pair of security gateways (network-to-network), or between a security gateway and a host (network-to-host). In order to decide what protection is to be provided for an outgoing packet, IPsec uses the Security

Parameter Index (SPI), an index to the security association database (SADB), along with the destination address in a packet header, which together uniquely identify a security association for that packet. A similar procedure is performed for an incoming packet, where IPsec gathers decryption and verification keys from the security association database.

The Neighbor Discovery protocol (NDP) that exists in Internet protocol suite is used with IPv6. It operates in Link Layer of the Internet model and is responsible for address auto configuration of nodes, discovery of other nodes on the link, determining the Link Layer addresses of other nodes, duplicate address detection, finding available routers and Domain Name System (DNS) servers, address prefix discovery, and maintaining reachability information about the paths to other active neighbor nodes.

CHAPTER 3

PROTOCOL VERIFICATION OF MOBILE IPV4

Since IPv4 was not built with mobility in mind, Mobile IPv4 was designed as an extension to the base IPv4 protocol to support mobility. Mobile IP resolves the issue of mobility by assigning the mobile node a temporary address at each new location, maintaining the MN’s original IP address, and creating and storing a covering between the two addresses with a router in the mobile node’s original network [48].

This chapter deals with development of schema diagram, formulation of design steps for each module from schema diagram, implementation of each design step using C and interpretation of the experimental results, for the Mobile IPv4 Model.

3.1 INTRODUCTION

The outline of operation of Mobile IPv4 protocol is given in following steps:

i. Mobility agents (i.e., foreign agents and home agents) publicize their existence via Agent Advertisement messages. A mobile node may optionally solicit an Agent Advertisement message from any locally attached mobility agents through an Agent Solicitation message.

ii. When mobile node obtains these Agent Advertisements, it decides whether it is on its home network or a foreign network.

iii. When mobile node identifies that it is located on its home network, it operates without mobility services. If returning to its home network from being registered elsewhere, mobile node deregisters with its home agent, through exchange of Registration Request and Registration Reply message with it.

iv. When a mobile node detects that it has moved to a foreign network, it obtains a care-of address on foreign network. The care-of address can be determined from a foreign agent's advertisements (a foreign agent care-of address).

v. The mobile node operating away from home then registers its new care-of address with its home agent through interchange of a Registration Request and Registration Reply message with home agent, probably via a foreign agent.

vi. Datagrams sent to mobile node's home address are captured by its home agent, tunneled by home agent to mobile node's care-of address, received at tunnel endpoint (either at a foreign agent or at the mobile node itself), and finally delivered to mobile node [23].

vii. In reverse direction, datagrams sent by mobile node are generally delivered to their destination using standard IP routing mechanisms, not necessarily passing through home agent.

Figure 3.1 gives the pictorial representation of operations involved in Mobile IPv4.

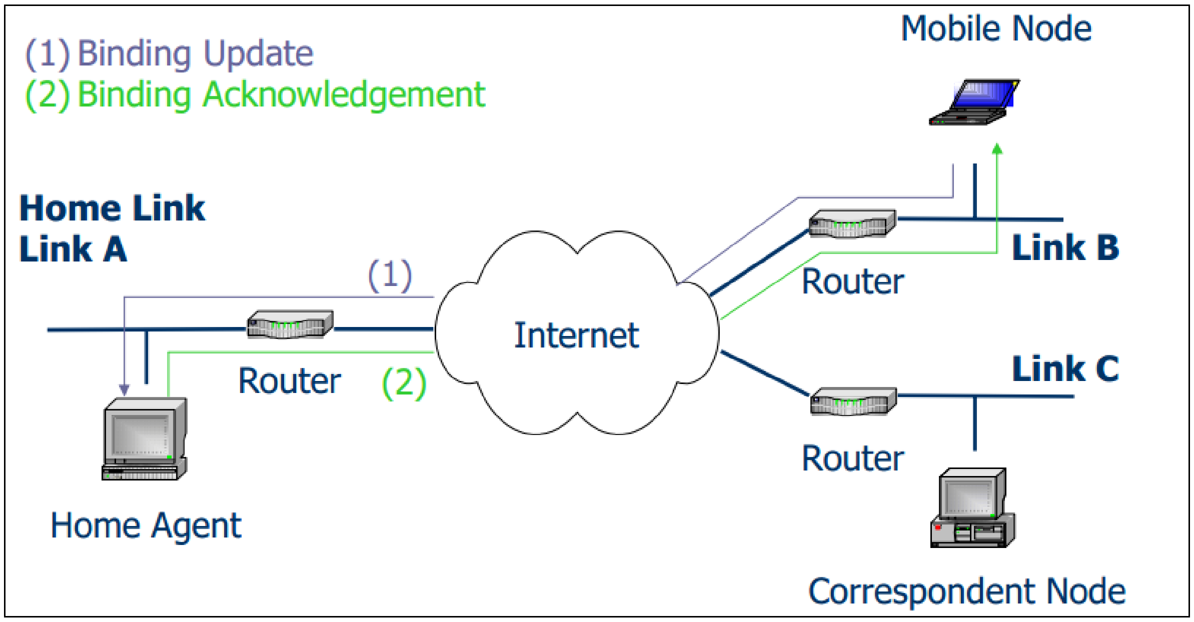


Figure 3.1 Operations in MIPv4

3.1.1 Agent Discovery in Mobile IPv4

The process for a mobile node to discover a home agent or a foreign agent and to receive information from these mobility agents is known as Agent Discovery. Home agent or foreign agents advertise their services and system information to mobile nodes via Agent advertisement messages. These Agent advertisement messages may be periodically broadcast to all mobile devices or it can be delivered in any manner which is appropriate for a network provider.

3.1.2 Registration in Mobile IPv4

In Registration phase of Mobile IP, mobile nodes communicate to their home agent the information about their current reachability via Registration messages. A mobile node, (optionally) a foreign agent and home agent exchange this information with help of registration messages which are data packets that contain relevant information. Whenever a mobile node visits a foreign network, it requests for forwarding services, then mobile nodes inform their home agent of their current care-of address, and finally mobile node also renews a registration that is due to expire, and/or deregister when they return home. Thus the process of Registration in Mobile IP is carried out in a smooth and flexible manner. When mobile node registers directly with its home agent, the registration procedure requires following two messages:

1. Registration Request is sent to home agent by a mobile node.
2. Registration Reply is sent to mobile node by home agent informing whether request is granted or denied.

3.1.3 Registration Extensions

This section deals with extensions present in Registration request and Registration reply messages.

3.1.3.1 Registration Request message

The Source address and Destination address of IP fields and UDP fields in Registration request message is given below

i. <u>IP fields:</u>

Source Address: It is interface address from which message is sent.

Destination Address: Address of foreign agent or home agent.

ii. <u>UDP fields:</u>

Source Port: variable

Destination Port: 434

The UDP header is followed by Mobile IP fields shown in table 3.1

Table 3.1 UDP Header followed by Mobile IP fields

Type	S	B	D	M	G	r	T	X	Lifetime
Home Address									
Home Agent									
Care-of Address									
Identification									
Extensions									

UDP header in a Registration Request message

i. Type1 (Registration Request)

ii. S stands for Simultaneous bindings. If 'S' bit is set to 1, it means mobile node is requesting home agent to retain its prior mobility bindings.

iii. B indicates Broadcast datagrams. If 'B' bit is set, then mobile node requests home agent to tunnel any broadcast datagrams that it receives on home network.

iv. D stands for Decapsulation by mobile node. If 'D' bit is set, it means mobile node is using a co-located care-of address, thus mobile nodes itself will decapsulate datagrams that are sent to care-of address.

v. M is Minimal encapsulation. If 'M' bit is set, it indicates that mobile node requests its home agent to use minimal encapsulation [49] for datagrams tunneled to mobile node.

vi. G stands for GRE encapsulation. If 'G' bit is set, mobile node requests that its home agent use Generic Routing Encapsulation [50] for datagrams tunneled to mobile node.

vii. r is sent as zero; r bit is always ignored on reception. And care is taken that r SHOULD NOT be allocated for any other uses.

viii. T means there is a request for Reverse Tunneling [51].

ix. x is always sent as zero and it is ignored on reception.

x. Lifetime: It can be defined as number of seconds remaining before registration expires. To request for deregistration Lifetime value must be zero and a value of 0xffff indicates that Lifetime is infinity.

xi. Home Address is the IP address of mobile node.

xii. Home Agent is the IP address of mobile node's home agent.

xiii. Care-of Address is the IP address for end of tunnel.

xiv. Identification is a 64-bit number constructed by mobile node. Identification is used for matching Registration Requests with Registration Replies and also for protecting against replay attacks of registration messages [52].

xv. Extensions are fixed portion of Registration Request. An authorization-enabling extension MUST be included in all Registration Requests [26].

3.1.3.2 Registration Reply message

The details about Source and Destination of IP fields and UDP fields in Registration Reply message is as mentioned below;

i. IP fields:

Source Address: Typically copied from Destination Address of Registration Request to which agent is replying.

Destination Address: Copied from source address of Registration Request to which agent is replying.

ii. UDP fields:

Source Port: Copied from UDP Destination Port of corresponding Registration Request.

Destination Port: Copied from source port of corresponding Registration Request.

The UDP header is followed by Mobile IP fields shown in table 3.2.

Table 3.2 UDP header in a Registration Reply message

Type	Code	Life Time
Home Address		
Home Agent		
Identification		

Type: 3 (Registration Reply)

Code is a value that shows the result of Registration Request.

The following values are defined for use within Code field.

For successful Registration:

0 indicates that registration is accepted

1 indicates that registration is accepted, but mobility bindings remain unsupported

For Registration denied by home agent, code field takes a value of 128 which the reason is unspecified for registration denial. Similarly 129 means Registration is denied due to administrative prohibition. 130 indicate insufficient resources leading to registration denial. The following table 3.3 gives the value of code field along with interpretation of each value.

Table 3.3 Code Field and its interpretation

131	mobile node failed authentication
132	foreign agent failed authentication
133	registration Identification mismatch
134	poorly formed Request
135	too many simultaneous mobility bindings
136	unknown home agent address
194	Invalid Home Agent Address

The following table 3.4 gives the value for code field along with interpretation of each value for Registration being denied by foreign agent:

Table 3.4 Code field along with interpretation of each value for Registration being denied by foreign agent.

64	reason unspecified
65	administratively prohibited
66	insufficient resources
67	mobile node failed authentication
68	home agent failed authentication
69	requested Lifetime too long
70	poorly formed Request
71	poorly formed Reply
72	requested encapsulation unavailable
73	reserved and unavailable
77	invalid care-of address
78	registration timeout
80	home network unreachable
81	home agent host unreachable (ICMP error received)
82	home agent port unreachable (ICMP error received)
88	home agent unreachable (other ICMP error received)

Mobile IPv4 is the most promising solution for mobility management in the current IPV4 network. Mobile IPv4 was designed as an extension to the baseIPv4 protocol to support mobility. Mobile IP resolves the issue of mobility by assigning the mobile node a temporary address at each new location, maintaining the MN’s original IP address[53]. MIPv4 uses foreign agent routers functionality. MIPv4 involves a Mobile Node (MN), Home Agent (HA) that exists in home network and keeps mobility information on a MN, and Foreign Agent (FA) that controls its network having a moving MN and acts as Care-of-Address (COA) for the MN. And Corresponding Node (CN) is a node that wants to connect or is communicating with the MN [54]. FA periodically sends out agent advertisement message, MN receives this message if it enters the range controlled by FA. After receiving the message, MN starts location registration procedure by sending the registration request message with COA to HA through FA. Then, HA updates mobility information on the MN and sends the registration response and binding message. Whenever MN moves from a network to another network, HA tracks the MN's location by registration request and response. If a CN wants to communicate with MN, the CN sends data to MN. Because of standard internet routing, data from CN is arrived at the home network of MN, specially HA. HA now controls the binding list with the data's destination address and then transfers the data to its FA if MN is not in the home network. FA controls the visiting list with the destination address of the data arrived from HA by tunneling. And then, FA directly sends the data to MN if MN is in the foreign network [55].

However since IPv4 was not built with mobility in mind, MIPv4 suffers from certain problems as mentioned in secton 3.4 of this chapter. All of these issues mentioned above prompted the Internet Engineering Task Force (IETF) to begin the development of a replacement protocol for IPv4 that would solve the problems of IPv4 and be extensible to solve additional problems in the future. The replacement for IPv4 is IPv6.

3.2 PROTOCOL VERIFICATION

The steps for protocol verification implemented in this research work are similar for all modules specified in the following chapters. First the protocol was modelled, then assertions were developed, and finally checked for proofs. Difficulties with assertions or proofs may require a change in the model (e.g. addition of variables to facilitate proofs, or reduction of states), making the process cyclic. Protocol Verification of Mobile IPv4 in this thesis was carried out by following the steps mentioned below,

- Developing the schema diagram to represent Mobile IPv4 Model.
- Formulating design steps for each module from schema diagram.
- Implementing each design step using Programming language C.
- Noting down experimental results using screen shots.

3.2.1 Modeling Mobile IPv4 using Schema diagram

The schema diagram was developed using the description of Mobile IPv4 and the respective schema diagram is given in figure 3.2.

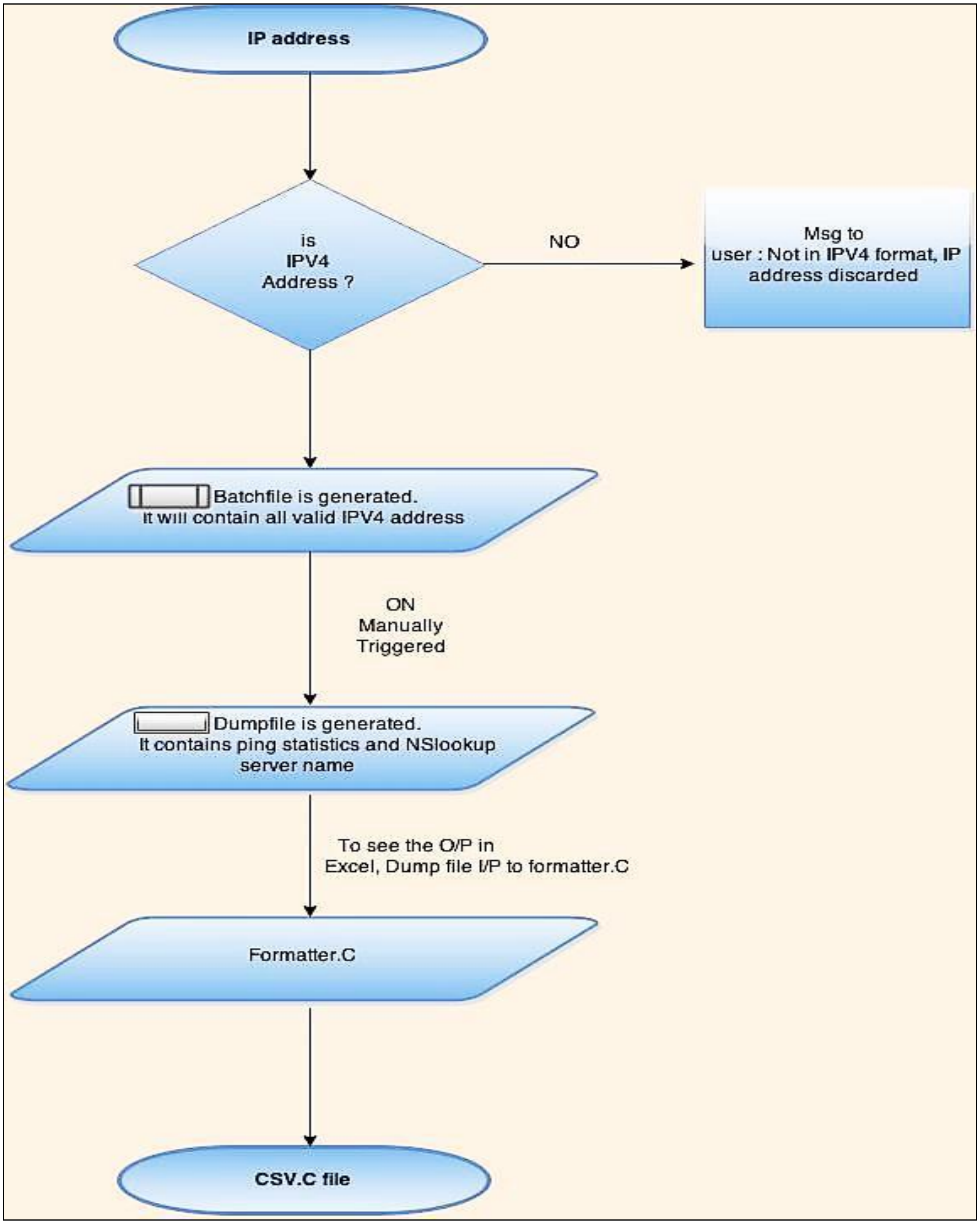

IP address
is IPV4 Address ?
NO
Msg to user : Not in IPV4 format, IP address discarded
Batchfile is generated. It will contain all valid IPV4 address
ON Manually Triggered
Dumpfile is generated. It contains ping statistics and NSlookup server name
To see the O/P in Excel, Dump file I/P to formatter.C
Formatter.C
CSV.C file

Figure 3.2 Schema diagram modeling MIPv4

3.2.2 Design for Protocol verification of Mobile IPv4

IP address are provided as Input.

Example: 1. 173. 194.113.145

2. 50.97.149.10
3. 157.166.239.102
4. 202.78.175.200

All valid IPv4 address are processed. Any input other than IPv4 will be discarded. After processing all valid IPv4 addresses, this section will be continued with the generation of a batch file containning all the valid input which is in IPv4 format. This batch file is present in the BIN folder with the name "IPv4_AddressChalange.bat", this file is overwritten for each execution. On executing this batch file, a dump file is generated. This dump file gives you the statistics of how many packets were sent, how many packets were recieved and how many packets were lost. The contents of Dump file can be viewed in Microsoft Excel by executing another program with name Formatter.c. Then the final .csv file is generated.

This function will be implemented using programming Language C. The inputs will be taken from command line arguments. The output will be displayed on a web browser

INPUTS

1. 173. 194.113.145
2. 50.97.149.10
3. 157.166.239.102
4. 202.78.175.200

Step wise Procedure:

Step 1.These inputs are checked if it is in the right format of IPv4.

Step 2. If the given input is not in the IPv4 format then a message is given to the users saying that this Module has been designed only to process IPv4 address and not any other IP Formats.

Step 3. If the input is in correct expected format then the user is asked for any more inputs.

Step 4. Once the user has given all the inputs then the code will generate a batch file to do the below mentioned steps

Step 6. Send nslookup command [NSLOOKUP] along with each input sequentially and perform ping opertion.

Step 7. The batch file must be manually triggered to genrate DUMP file which contains all the details of the command execution results.

Step 8. The same IPv4 should also be pinged to check if the packet has been received, In this check the user wants to test if the IPv4 is reachable and if the IPv4 is a valid IP address which can either be of an hardware which is in the network or can also be of a webserver. This is checked by pasting one of the valid IPv4 address on a web browser. At this juncture relevant website is opened to verify that the IPv4 address is a valid address.

Step 9. After processing all the input [IPv4] a message is given to the user saying that this Module has processed all the inputed IPv4 and the analysis has been written to another external file .

Program Formatter.c will take input as the DUMP file and generate a CSV file as an output. This is the final file and will be the output of the complete IPv4 cycle.

3.2.3 Implementation using programming language C

Function 1. Check_inputs_IPV4( ).

This function validates the format of the input IP address provided. If the input fails validation, this input was discarded and a corresponding entry was made in the log file. The inputs would be taken in loop and the user will be prompted for multiple inputs (depending upon the user)

INPUTS for the function int check_inputs_IPV4(char MNIP[ ])

1. 173. 194.113.145
2. 50.97.149.10
3. 157.166.239.102
4. 202.78.175.200

OUTPUT : All these inputs are validated and processed

Function 2. Generate_IPv4 batchfile( ).

This finction will be responsible to generate a batch file for all valid IPv4 addresses and will contain predefined commands for PING and NSLOOKUP. Once this file is generated it was manually executed to generate the output a DUMP file.

NOTE: This file will not be saved and it is regenerated after each run.

Once the dump file is generated this file will contain the results of the IPv4 address challenge.

INPUT for the function void Generate_IPv4 batchfile( ): All valid IPv4 address

OUTPUT : MIPv4.bat file is generated in BIN folder, on executing this batch file, Dump file is generated

Function 3. Program Formatter.c

This will take DUMP file as Input and generate a CSV file as an output. This is the final file and will be the output of the complete IPv4 cycle.

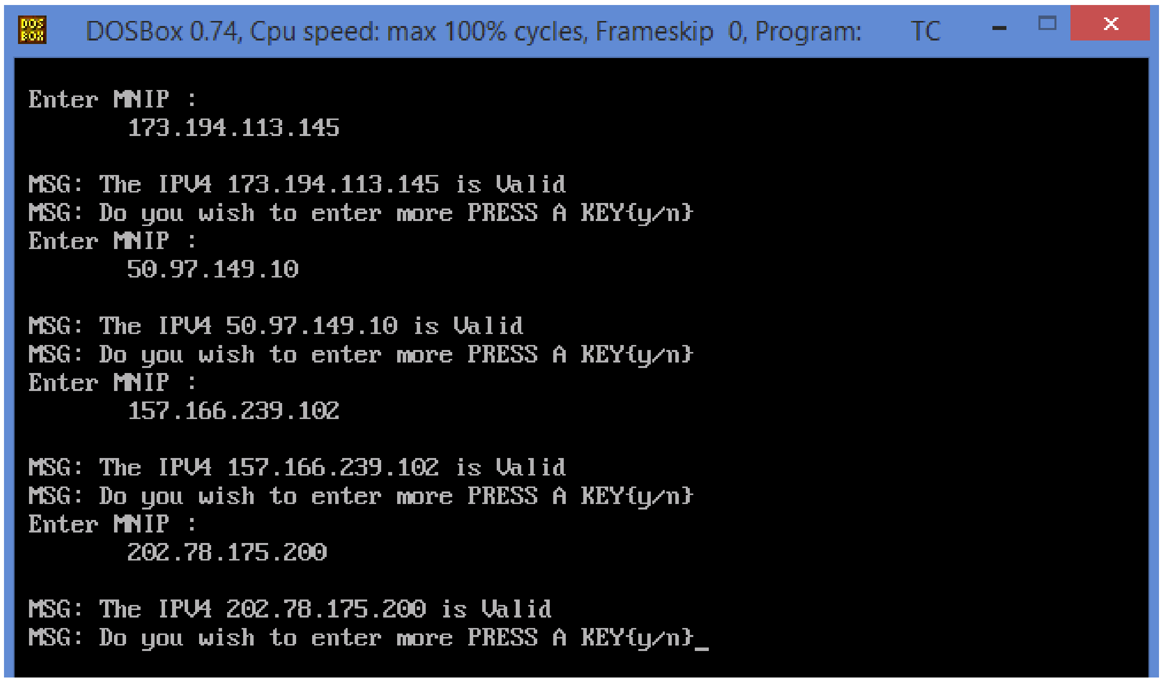
INPUT for the Program Formatter.c : Dump file

OUTPUT : CSV file is generated to show the contents of dump file in EXCEL.

3.2.4 Experimental Results

INPUTS:

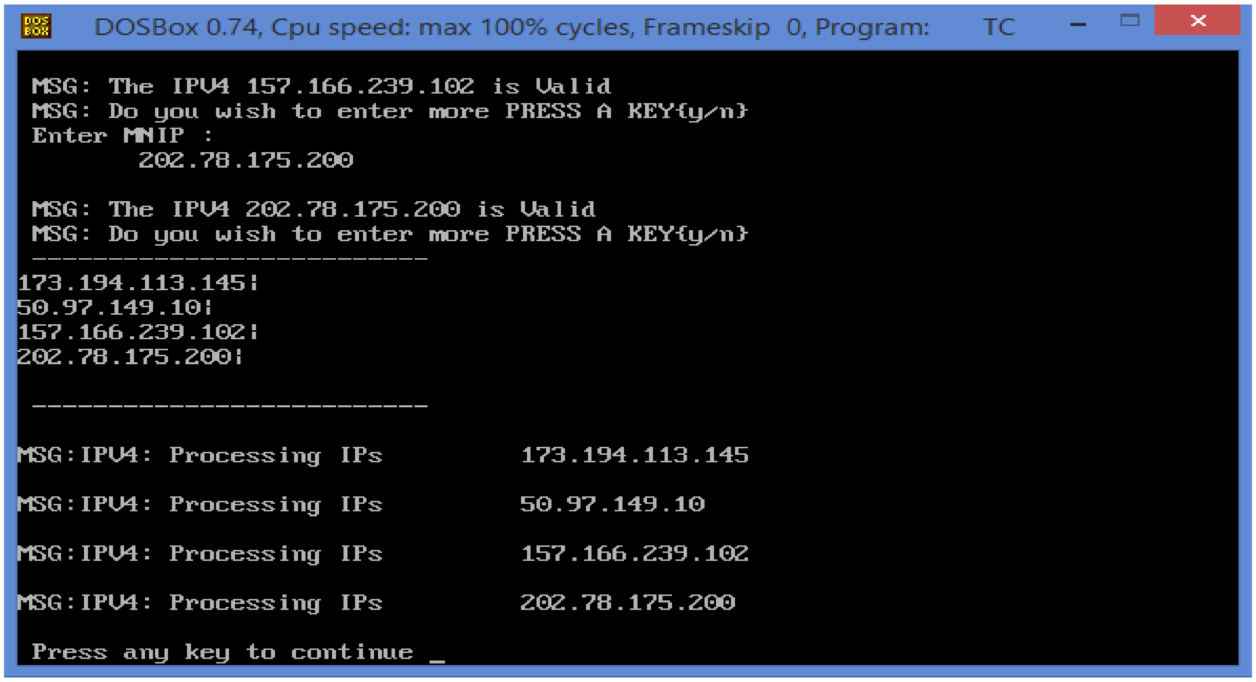
1. 173. 194.113.145
2. 50.97.149.10
3. 157.166.239.102
4. 202.78.175.200


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
Enter MNIP :
       173.194.113.145

MSG: The IPV4 173.194.113.145 is Valid
MSG: Do you wish to enter more PRESS A KEY{y/n}
Enter MNIP :
       50.97.149.10

MSG: The IPV4 50.97.149.10 is Valid
MSG: Do you wish to enter more PRESS A KEY{y/n}
Enter MNIP :
       157.166.239.102

MSG: The IPV4 157.166.239.102 is Valid
MSG: Do you wish to enter more PRESS A KEY{y/n}
Enter MNIP :
       202.78.175.200

MSG: The IPV4 202.78.175.200 is Valid
MSG: Do you wish to enter more PRESS A KEY{y/n}_
```

Figure 3.3 Different inputs provided by the user.

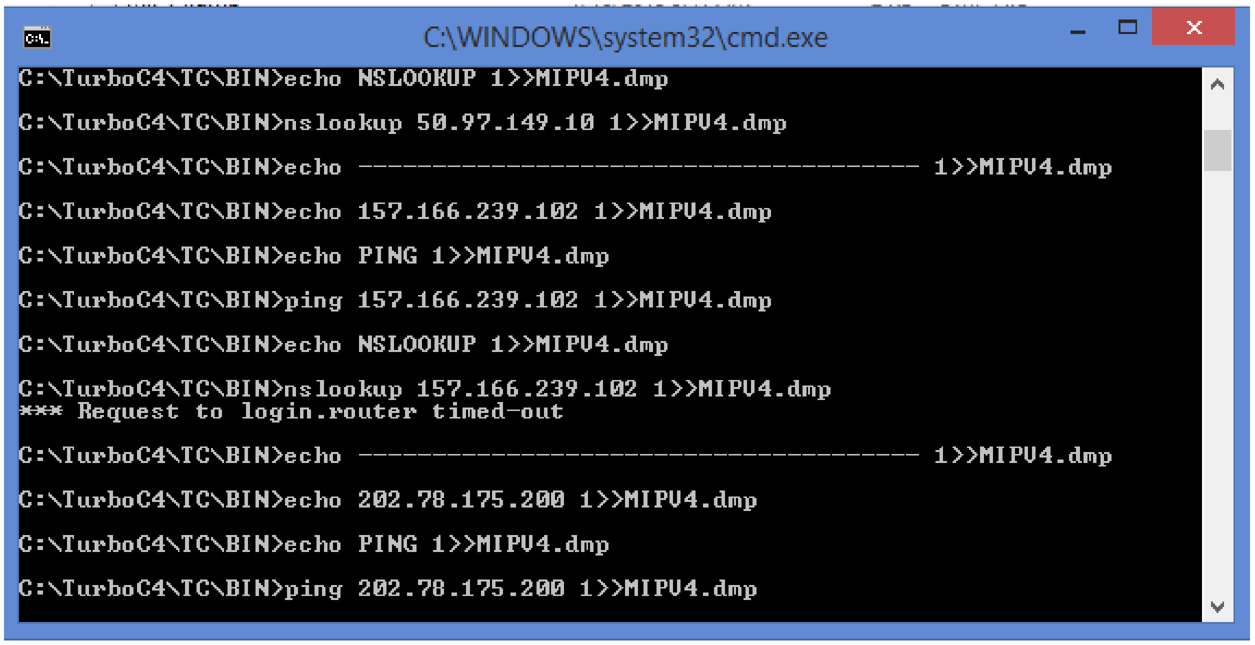
Processing all the IPv4 addresses



DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
 MSG: The IPV4 157.166.239.102 is Valid
 MSG: Do you wish to enter more PRESS A KEY(y/n)
 Enter MNIP :
        202.78.175.200

 MSG: The IPV4 202.78.175.200 is Valid
 MSG: Do you wish to enter more PRESS A KEY(y/n)
 --------------------------
173.194.113.145|
50.97.149.10|
157.166.239.102|
202.78.175.200|

 --------------------------

MSG:IPV4: Processing IPs         173.194.113.145

MSG:IPV4: Processing IPs         50.97.149.10

MSG:IPV4: Processing IPs         157.166.239.102

MSG:IPV4: Processing IPs         202.78.175.200

 Press any key to continue _
```

Figure 3.4 Processing all IPv4 address inputs.

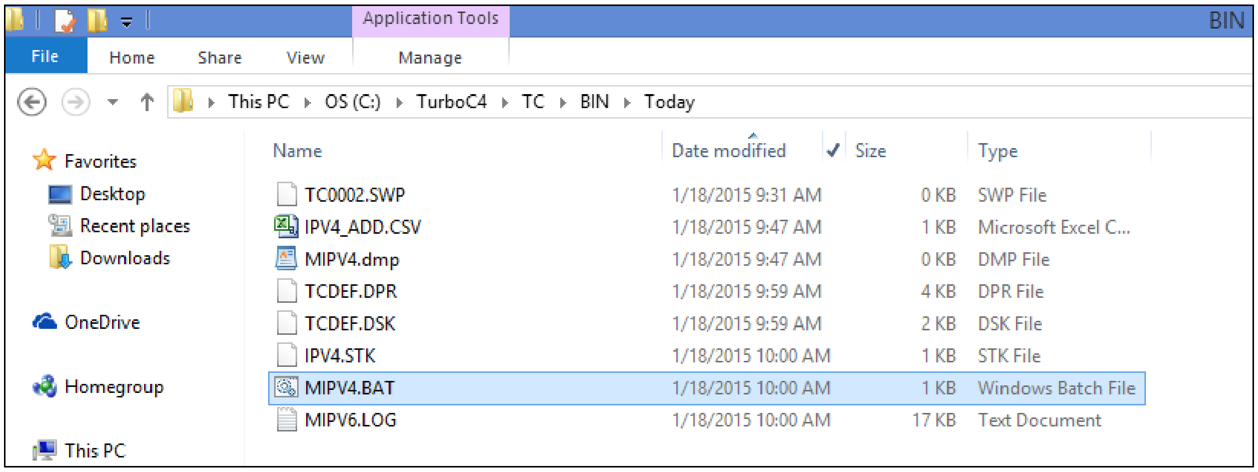


C:\WINDOWS\system32\cmd.exe

```
C:\TurboC4\TC\BIN>echo NSLOOKUP 1>>MIPV4.dmp

C:\TurboC4\TC\BIN>nslookup 50.97.149.10 1>>MIPV4.dmp

C:\TurboC4\TC\BIN>echo ---------------------------------------- 1>>MIPV4.dmp

C:\TurboC4\TC\BIN>echo 157.166.239.102 1>>MIPV4.dmp

C:\TurboC4\TC\BIN>echo PING 1>>MIPV4.dmp

C:\TurboC4\TC\BIN>ping 157.166.239.102 1>>MIPV4.dmp

C:\TurboC4\TC\BIN>echo NSLOOKUP 1>>MIPV4.dmp

C:\TurboC4\TC\BIN>nslookup 157.166.239.102 1>>MIPV4.dmp
*** Request to login.router timed-out

C:\TurboC4\TC\BIN>echo ---------------------------------------- 1>>MIPV4.dmp

C:\TurboC4\TC\BIN>echo 202.78.175.200 1>>MIPV4.dmp

C:\TurboC4\TC\BIN>echo PING 1>>MIPV4.dmp

C:\TurboC4\TC\BIN>ping 202.78.175.200 1>>MIPV4.dmp
```

Figure 3.5 Processing all IPv4 address inputs.

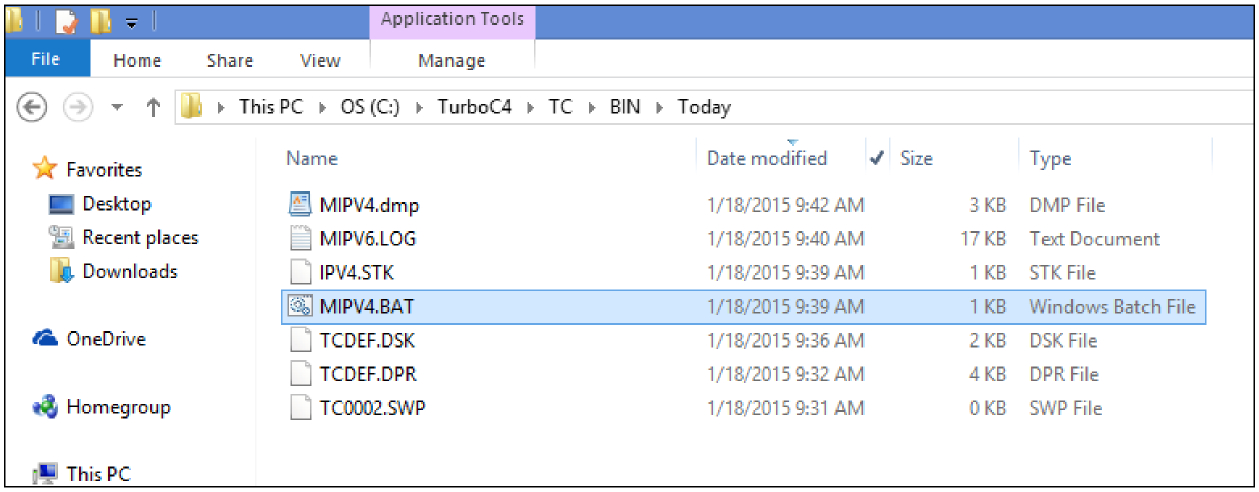
Batch file is generated



Name	Date modified	Size	Type
TC0002.SWP	1/18/2015 9:31 AM	0 KB	SWP File
IPV4_ADD.CSV	1/18/2015 9:47 AM	1 KB	Microsoft Excel C...
MIPV4.dmp	1/18/2015 9:47 AM	0 KB	DMP File
TCDEF.DPR	1/18/2015 9:59 AM	4 KB	DPR File
TCDEF.DSK	1/18/2015 9:59 AM	2 KB	DSK File
IPV4.STK	1/18/2015 10:00 AM	1 KB	STK File
MIPV4.BAT	1/18/2015 10:00 AM	1 KB	Windows Batch File
MIPV6.LOG	1/18/2015 10:00 AM	17 KB	Text Document

Figure 3.6 Batch file.

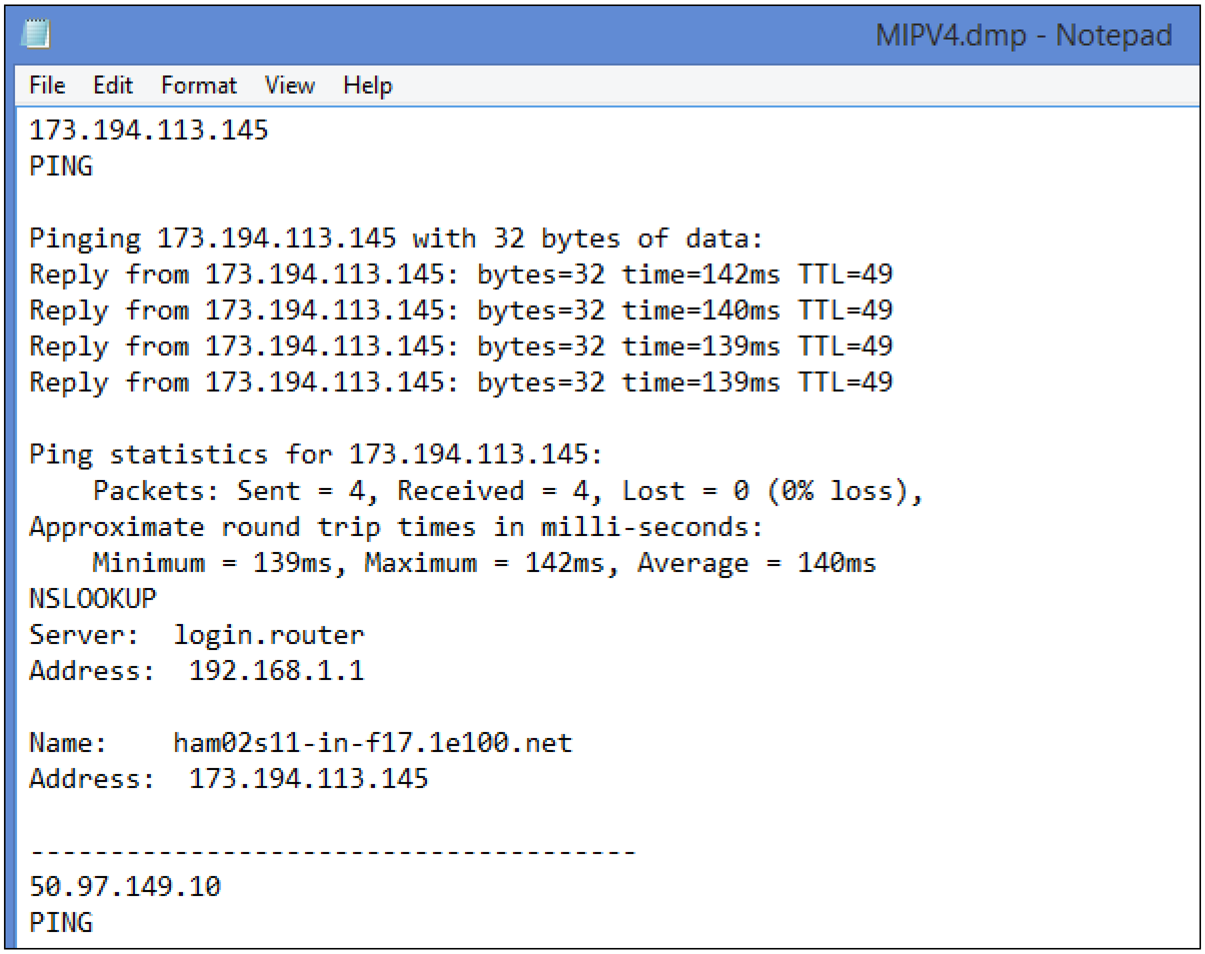
Executing the batch file (by double click on filename MIPV4.bat) Dump file is generated



Name	Date modified	Size	Type
MIPV4.dmp	1/18/2015 9:42 AM	3 KB	DMP File
MIPV6.LOG	1/18/2015 9:40 AM	17 KB	Text Document
IPV4.STK	1/18/2015 9:39 AM	1 KB	STK File
MIPV4.BAT	1/18/2015 9:39 AM	1 KB	Windows Batch File
TCDEF.DSK	1/18/2015 9:36 AM	2 KB	DSK File
TCDEF.DPR	1/18/2015 9:32 AM	4 KB	DPR File
TC0002.SWP	1/18/2015 9:31 AM	0 KB	SWP File

Figure 3.7 Executing the batch file to generate the Dump file.

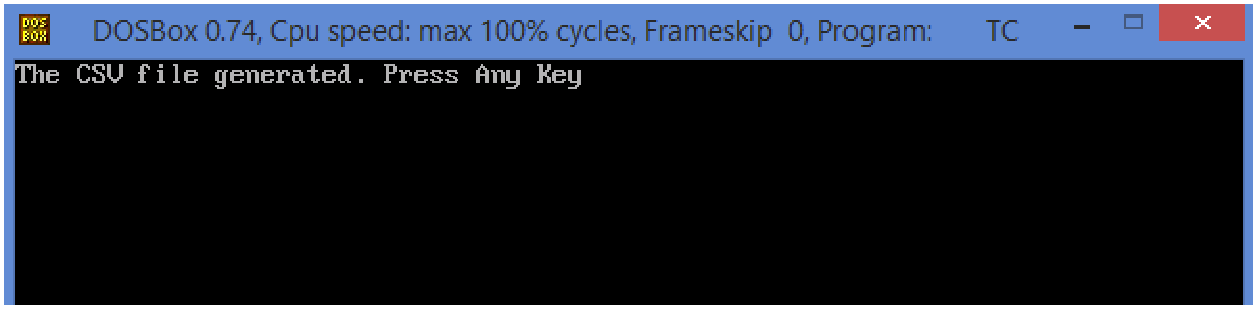
The contents of the Dump file is shown below



```
173.194.113.145
PING

Pinging 173.194.113.145 with 32 bytes of data:
Reply from 173.194.113.145: bytes=32 time=142ms TTL=49
Reply from 173.194.113.145: bytes=32 time=140ms TTL=49
Reply from 173.194.113.145: bytes=32 time=139ms TTL=49
Reply from 173.194.113.145: bytes=32 time=139ms TTL=49

Ping statistics for 173.194.113.145:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 139ms, Maximum = 142ms, Average = 140ms
NSLOOKUP
Server:  login.router
Address:  192.168.1.1

Name:    ham02s11-in-f17.1e100.net
Address:  173.194.113.145

--------------------------------------
50.97.149.10
PING
```

Figure 3.8 Dump file is generated.

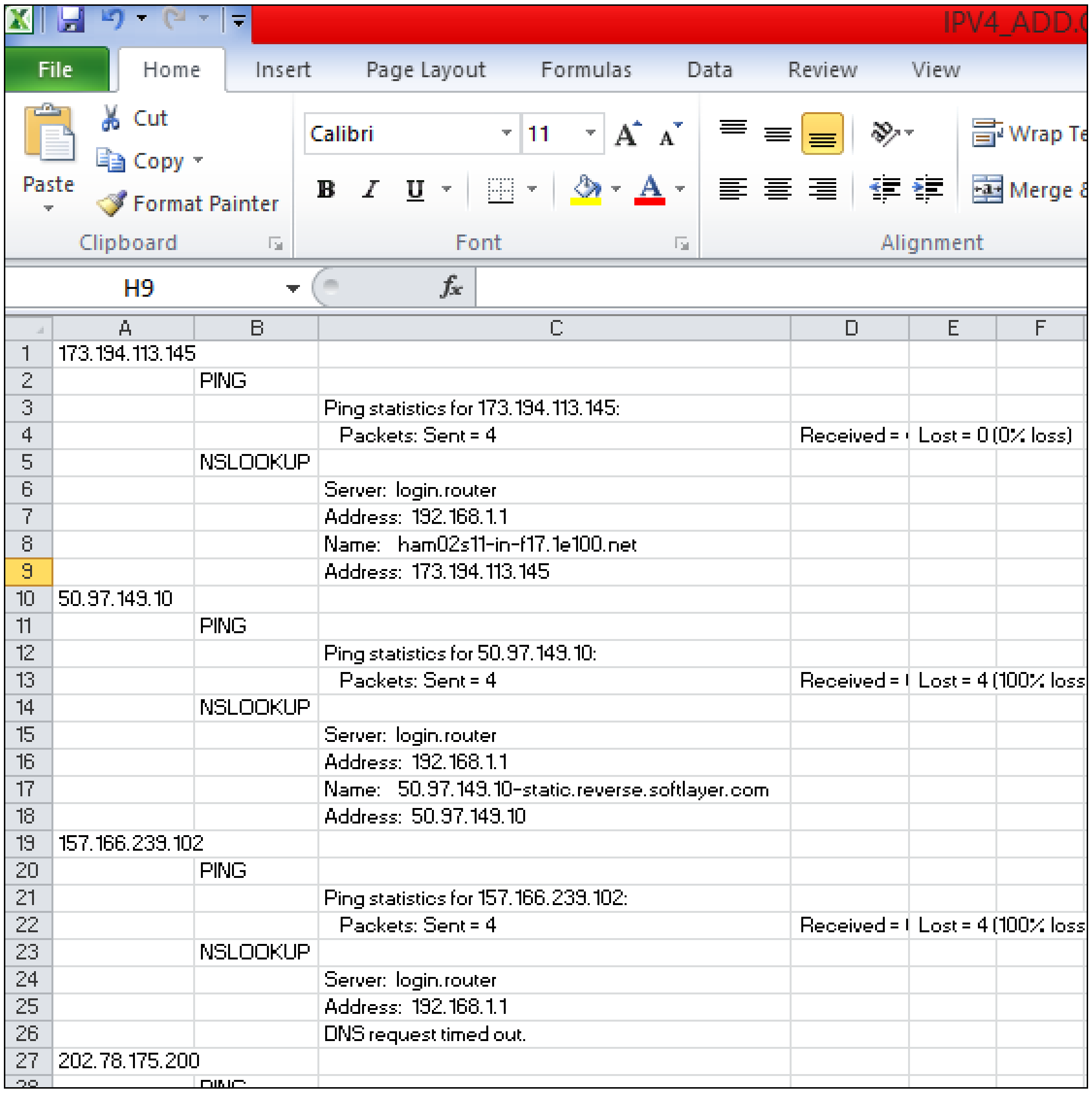
Executing formatter.c code to generate a csv file for final output


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
The CSV file generated. Press Any Key
```

Figure 3.9 CSV file is generated.

Excel file



	A	B	C	D	E	F
1	173.194.113.145					
2		PING				
3			Ping statistics for 173.194.113.145:			
4			Packets: Sent = 4	Received =	Lost = 0 (0% loss)	
5		NSLOOKUP				
6			Server: login.router			
7			Address: 192.168.1.1			
8			Name: ham02s11-in-f17.1e100.net			
9			Address: 173.194.113.145			
10	50.97.149.10					
11		PING				
12			Ping statistics for 50.97.149.10:			
13			Packets: Sent = 4	Received =	Lost = 4 (100% loss	
14		NSLOOKUP				
15			Server: login.router			
16			Address: 192.168.1.1			
17			Name: 50.97.149.10-static.reverse.softlayer.com			
18			Address: 50.97.149.10			
19	157.166.239.102					
20		PING				
21			Ping statistics for 157.166.239.102:			
22			Packets: Sent = 4	Received =	Lost = 4 (100% loss	
23		NSLOOKUP				
24			Server: login.router			
25			Address: 192.168.1.1			
26			DNS request timed out.			
27	202.78.175.200					

Figure 3.10 Excel file to show ping statistics and NS lookup records.

<u>On pasting one of the IPv4 address on the web browser</u>

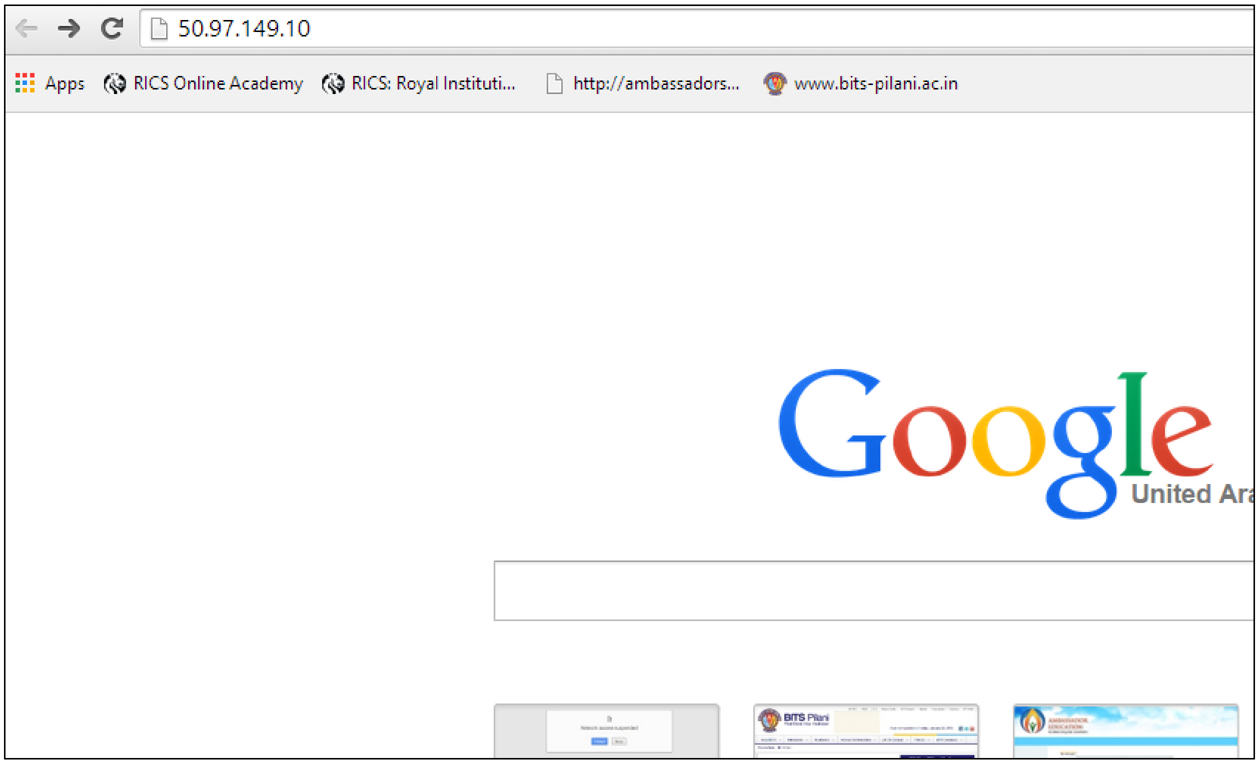


Figure 3.11 Web browser with input address as provided in Figure 3.2

<u>Respective Website is opened as shown below</u>

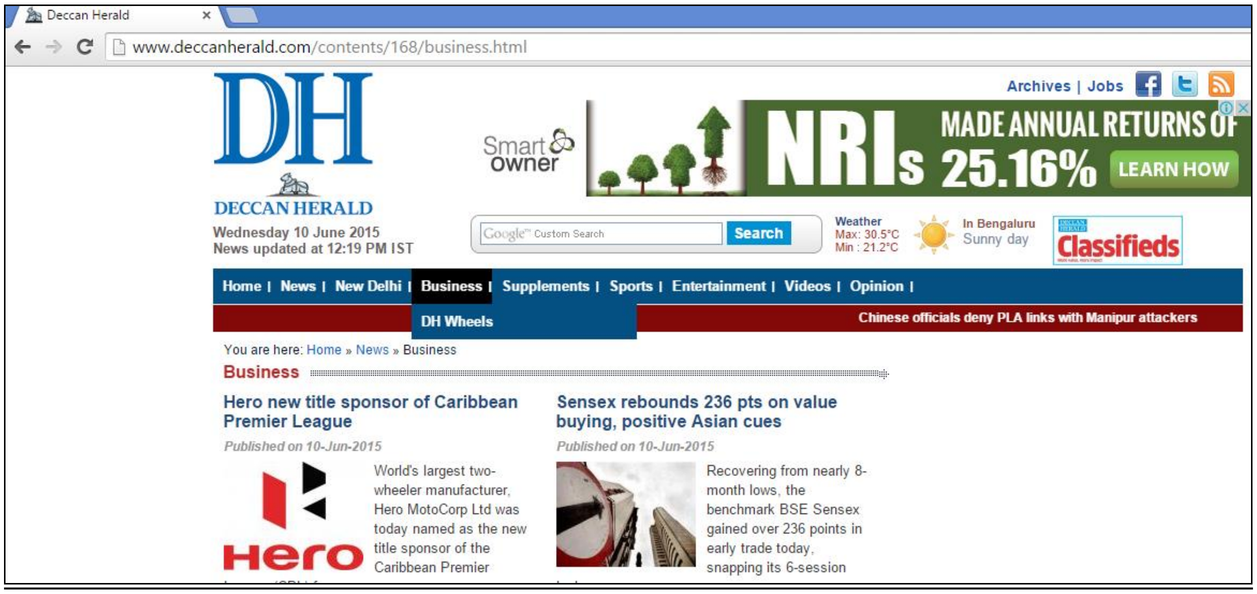
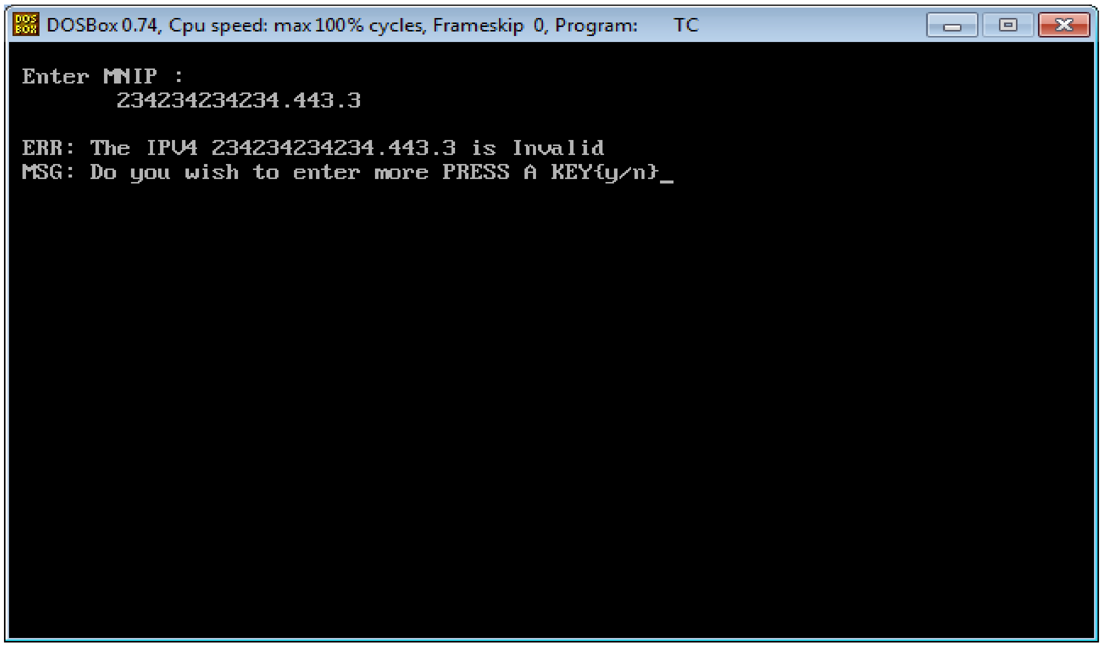

Deccan Herald
www.deccanherald.com/contents/168/business.html
Archives | Jobs
DH
DECCAN HERALD
Wednesday 10 June 2015
News updated at 12:19 PM IST
Smart owner
NRIs MADE ANNUAL RETURNS OF 25.16% LEARN HOW
Search
Weather Max: 30.5°C Min : 21.2°C
In Bengaluru Sunny day
Classifieds
Home | News | New Delhi | Business | Supplements | Sports | Entertainment | Videos | Opinion |
DH Wheels
Chinese officials deny PLA links with Manipur attackers
You are here: Home » News » Business
Business
Hero new title sponsor of Caribbean Premier League
Published on 10-Jun-2015
World's largest two-wheeler manufacturer, Hero MotoCorp Ltd was today named as the new title sponsor of the Caribbean Premier
Sensex rebounds 236 pts on value buying, positive Asian cues
Published on 10-Jun-2015
Recovering from nearly 8-month lows, the benchmark BSE Sensex gained over 236 points in early trade today, snapping its 6-session

Figure 3.12 Respective website is opened, in this example Deccan Herald.com.

IP Address Incorrect Format


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
Enter MNIP :
       234234234234.443.3

ERR: The IPV4 234234234234.443.3 is Invalid
MSG: Do you wish to enter more PRESS A KEY{y/n}_
```

Figure 3.13 Incorrect IP address is provided by the user.

3.2.5 Result Analysis

The function Check_inputs_IPV4( ) is responsible for the input IP address. It validates the format of the input IP address provided by the user. The inputs would be taken in loop and the user will be prompted for multiple inputs. Since Check_inputs_IPv4( ) loops through the function for every IP address and suppose there are n IP address as inputs, then the complexity of the function Check_inputs_IPv4( ) is of the order of n given by O(n).

3.3 DRAWBACKS OF MOBILE IPV4

i. Mobility. Mobility is a new requirement for Internet-connected devices, in which a node can change its address as it changes its physical attachment to the Internet and still maintain existing connections. Although there is a specification for IPv4 mobility, due to a lack of infrastructure, communications with an IPv4 mobile node are inefficient.

ii. Limited address space. The most noticeable and crucial challenge with using IPv4 on the modern Internet is the rapid exhaustion of public addresses. Due to the initial address class allocation practices of the early Internet, public IPv4 addresses are becoming scarce.

iii. Flat routing infrastructure. In the early Internet, address prefixes were not allocated to create a summarizable, hierarchical routing infrastructure. Instead, individual address prefixes were assigned and each address prefix became a new route in the routing tables of the Internet backbone routers. Today’s Internet is a mixture of flat and hierarchical routing, but there are still more than 85,000 routes in the routing tables of Internet backbone routers.

iv. Configuration. IPv4 must be configured, either manually or through the Dynamic Host Configuration Protocol (DHCP). DHCP allows IPv4 configuration administration to scale to large networks, but there is a overhead involved to configure and manage a DHCP infrastructure.

v. Security. Security for IPv4 is specified by the use of Internet Protocol security (IPsec). However, IPsec is optional for IPv4 implementations. Because an application cannot depend on on IPsec being present to secure traffic, an application might possibly chose other security standards or a proprietary security scheme. The prerequisite for built-in security is even more important today, when we face an increasingly aggressive environment on the Internet.

vi. Prioritized delivery. Special handling parameters such as low delay and low variance in delay for voice or video traffic, categorized under Prioritized packet delivery are possible with IPv4. However, it relies on a new interpretation of the IPv4 Type Of Service (TOS) field, which is not supported for all the devices on the network. Additionally, identification of the packet flow must be done using an upper layer protocol identifier such as a TCP or User Datagram Protocol (UDP) port. This additional processing of the packet by intermediate routers makes forwarding less efficient.

3.4 SUMMARY

In this chapter Protocol verification of Mobile IPv4 was carried out by first analysing the description of Mobile IPv4. A schema diagram was developed to represent Mobile IPv4 model. Then design steps were formualated to check all the IP addresses provided as input. These inputs were processed and only IPv4 addresses were validated and stored in a Batch file. The batch file was designed to send NSlookup command along with ping command to process all the valid IPv4 addresses. The ping statistics were listed in a dump file which gave information about the number of packets sent, recieved and lost. This dump file was treated as an input to another program called Formatter.c which provided the statistics of all IPv4 addresses in an Excel file.

CHAPTER 4

PROTOCOL VERIFICATION OF DUAL STACK

Dual-stack allows side-to-side implementation of IPv4 and IPv6 protocols while both protocols run on the same network infrastructure. Currently, two mobility management protocols defined are Mobile IPv4 and Mobile IPv6. Dual stack motivates to deploy both Mobile IPv4 (MIPv4) and Mobile IPv6 (MIPv6) protocol in such a way that MIPv4-capable node can use IPv4 stack and MIPv6-capable node can use IPv6 stack as they move between different subnets. Dual stack mechanism is explained by the following figure 4.1.

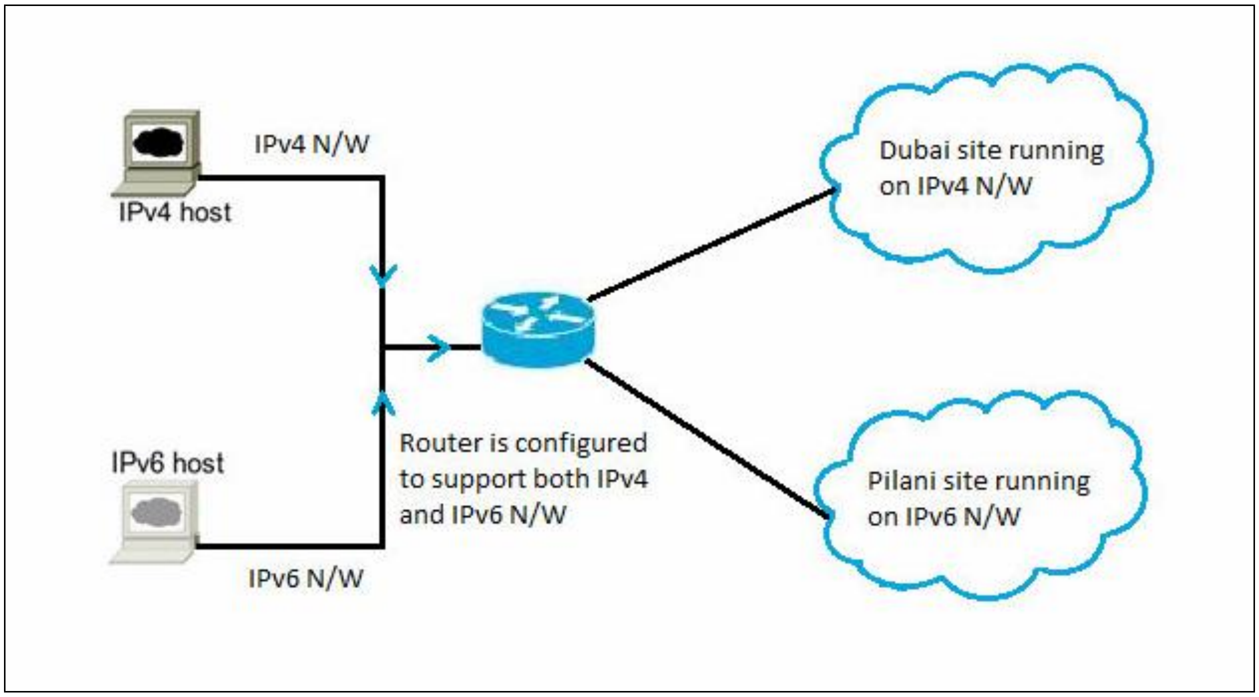

IPv4 N/W
IPv4 host
Dubai site running on IPv4 N/W
IPv6 host
Router is configured to support both IPv4 and IPv6 N/W
Pilani site running on IPv6 N/W
IPv6 N/W

Figure 4.1 Dual Stack showing a router that can support both IPv4 and IPv6 network.

4.1 INTRODUCTION

Mobile IP defines protocols and procedures by which packets can be routed to a mobile node, regardless of its current point-of-attachment to the Internet, and without changing its IP address. A MIPv4-capable node can use Mobile IPv4 protocol to maintain the connectivity as it moves between the subnets of IPv4. Similarly MIPv6-capable node uses Mobile IPv6 protocol in order to maintain the connectivity as it moves between the subnets of IPv6. In Dual Stack, nodes are deployed which are both IPv4 and IPv6 capable. These nodes make use of both IPv4 and IPv6 protocol to migrate between the two networks. Such nodes will be able to communicate when they get both IPv4 and IPv6 addresses. This communication occurs with the current IPv4 Internet, any IPv6 nodes and any available networks [56].

In dual-stack, a network node installs both IPv4 and IPv6 stacks in parallel. All IPv4 applications make use of IPv4 stack and similarly IPv6 applications make use of the IPv6 stack. Flow decisions are based on the IP header version field and destination address type. The IP header version field is responsible for receiving data packets and destination address type is responsible for sending data packets. DNS lookup table gives information about address types, while the DNS record times is used to choose the appropriate stack. Several commercial operating systems have already started providing dual IP protocol stacks, Hence Dual-stack is the most widely deployed transitioning solution [57]. This thesis deals with the IPv4-to-IPv6 Dual Stack Transition Mechanism (4to6 DSTM) which can operate even in the case that hosts in the IPv4 network initiate connections within hosts in the IPv6 network.

If Mobile IP is considered as a Dual stack setup protocol, then Mobile IP should be able to set-up IP in IPv4/IPv6 networks. This should be done independently, that is regardless of the IP version being used in the outer and the inner headers. Every IP address consists of ‘network number’. Routing, therefore, takes advantage of these numbers and hence the physical location of the computer is encoded by the IP address and the location is fixed by default. Whenever users need to carry their mobile devices across different LAN subnets, Mobile IP is the one in use and it is most often found in all wired and wireless devices. One of the key features of Mobile IP design is that all required functionalities for processing and managing mobility information are embedded in well-defined entities namely, The Home Agent (HA), The Foreign Agent (FA) and Mobile Node (MN). No change is required to the currently existing internet routers and hosts. Mobile IPv4 offers a transparent view of its transport layers and of its higher layers. The Mobile Node is allowed to retain its IP address by the Mobile IP. This is regardless of the attachment it has with the network. By using two IP addresses, this condition can be fulfilled. The higher layer connections are identified by the home address which is the first IP address, for example, TCP. The care of address is the second IP address that can be used by the mobile node. While the mobile is roaming among different networks, the Care-of Address changes. The reason behind this is, Care-of Address has to identify the mobile’s new point of attachment with respect to the network topology. In Mobile IPv4 the Care-of Address management is achieved by an entity called Foreign Agent.

Mobile IPv6 has added the roaming capabilities of mobile nodes in IPv6 network. The major benefit here is that the mobile nodes (such as IPv6 nodes) change their point of attachment to the IPv6 internet without changing their IP address. This allows mobile devices to move from one network to another and still maintain existing connections. Although Mobile IPv6 is mainly targeted for mobile devices, it is equally applicable for wired environments. The need for Mobile IPv6 is necessary because the mobile nodes in fixed IPv6 network cannot maintain the previously connected link (using the address assigned from the previously connected link) when changing location. To accomplish the need for mobility, connections to mobile IPv6 nodes are made (Without user interaction) with a specific address that is always assigned to the mobile node, and through which the mobile node is always reachable. Mobile IPv6 is expected to be used in IP over WLAN, WiMAX (Worldwide Interoperability for Microwave Access) or BWA (Broadband wireless association) [58].

A proposal is made that a work area should be taken by the IETF on this subject which also needs discussion on the requirements in order to reach an appropriate solution. Since the deployment of MIPV6 requires improvement in the protocol design of MIPV4, continuous research and development is required in MIPv4 protocol to address the problems which are widely stated above.

If it is assumed that Dual stack is deployed and whenever a mobile node which is both MIPv4 and MIPv6 capable roams within the internet, the mobile node is expected to maintain the following requirements.

- The security associations and the amount of configuration in the mobile node and the home agent must be doubled.
- The network operator needs to ensure that the home agent supports both protocols or that it has two separate Home Agents supporting the two protocols, each requiring its own management. There is a need to duplicate IP-layer local network optimizations for handovers.

4.2 PROTOCOL VERIFICATION

The methodology implemented in this research work for Protocol Verification of Dual Stack comprises of the following steps,

- Developing the schema diagram to represent Dual Stack Model
- Formulating design steps for each module from schema diagram

- Implementing each design step using Programming language C
- Noting down experimental results using screen shots.

4.2.1 Schema diagram to represent Dual stack

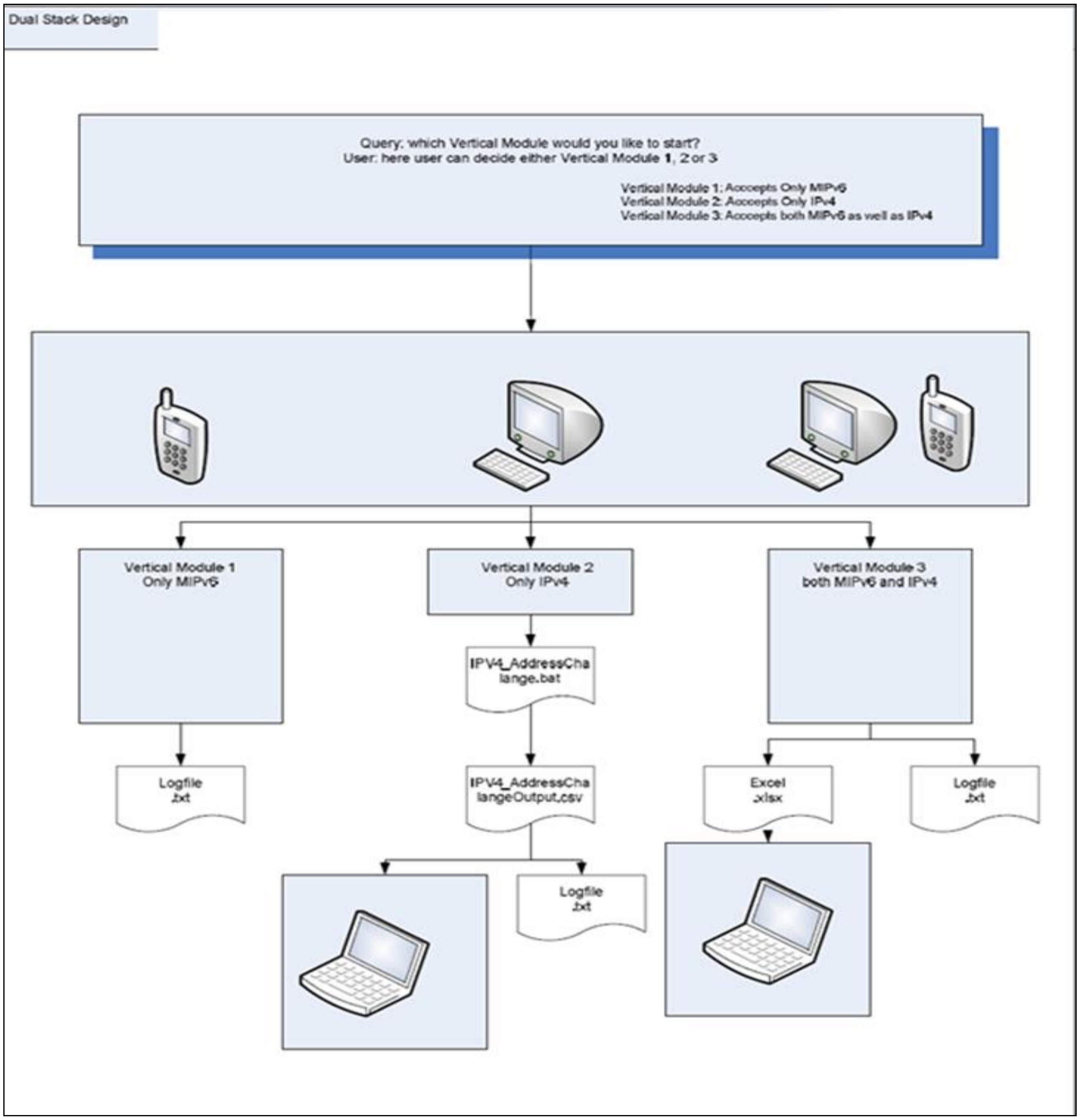

Dual Stack Design
Query: which Vertical Module would you like to start?
User: here user can decide either Vertical Module 1, 2 or 3
Vertical Module 1: Accepts Only MIPv6
Vertical Module 2: Accepts Only IPv4
Vertical Module 3: Accepts both MIPv6 as well as IPv4
Vertical Module 1 Only MIPv6
Vertical Module 2 Only IPv4
Vertical Module 3 both MIPv6 and IPv4
IPV4_AddressChalange.bat
Logfile .txt
IPV4_AddressChalangeOutput.csv
Excel .xlsx
Logfile .txt
Logfile .txt

Figure 4.2 Schema Diagram of Dual Stack

4.2.2 Design for Protocol verification of Dual Stack

The design has been divided into three vertical modules and one horizontal module, the horizontal module designed here provides the option to choose any one of the vertical module listed below. Below mentioned are the three vertical modules designed for Dual stack.

- Vertical Module 1: MIPv4
- Vertical Module 2: MIPv6
- Vertical Module 3: Dual Stack i.e. hybrid case

4.2.2.1 Vertical Module 1: MIPv4

Step 1: This module was designed in such a way that it will accept all valid IPv4 address and discard any address of different formats other than IPv4.

INPUTS

1. 173. 194.113.145
2. 50.97.149.10

Step 2: After accepting all valid IPv4 address, these addresses are further processed and sent to a separate stack called IPv4 stack.

Step 3: Each IPv4 address is processed, batch file and dump file are generated, and the final output is shown in Excel file as mentioned in section 3.3.3 of chapter 3.

4.2.2.2 Vertical Module 2: MIPv6

Step 1: In this module all IPv6 inputs were accepted after validation and any input other than IPv6 was discarded.

INPUTS :

1. Mobile Node IP: E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420, this is mandatory input & it is referred as MNIP

2. Home Agent IP: E3D7:0000:0000:0000:51F4:9BC8:C0A8:6422, this is optional & referred to as HAIP

Step 2: After accepting all valid IPv6 address, these addresses were further processed and sent to a separate stack called IPv6 stack.

Step 3: Each IPv6 address is processed and the final output was shown in log file.

4.2.2.3 Vertical Module 3: Dual stack, i.e. hybrid case

Step 1: This case is a hybrid case where all the IP formats (i.e. IPv4 and IPv6) are accepted. This module is a combination of Vertical Module 1 and Vertical Module 2, this Module will accept any type of input whether it is IPv6 or IPv4.

INPUTS

1. IPv4 address : 173. 194.113.145
2. IPv4 address : 50.97.149.10
3. IPv6 address (MNIP): E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420
4. IPv6 address (HAIP): E3D7:0000:0000:0000:51F4:9BC8:C0A8:6422.

Step 2: After accepting all valid IPv4 and IPv6 addresses, IPv4 addresses are processed and sent to a separate stack called IPv4 stack while IPv6 addresses are processed and sent to IPv6 stack.

Step 3: Each IPv4 address was processed and the final output is shown in Excel file while the output for IPv6 is shown in the log file.

4.2.3 Implementation using programming language C

Function : populate_hybrid_stack( )

This function is responsible to check whether MNIP is IPv4 or IPv6, the parameters used in this function are char array for MNIP , HAIP

INPUTS to Function : populate_hybrid_stack( )

1. IPv4 address : 173. 194.113.145
2. IPv4 address : 50.97.149.10
3. IPv6 address (MNIP): E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420
4. IPv6 address (HAIP): E3D7:0000:0000:0000:51F4:9BC8:C0A8:6422.

OUTPUT:

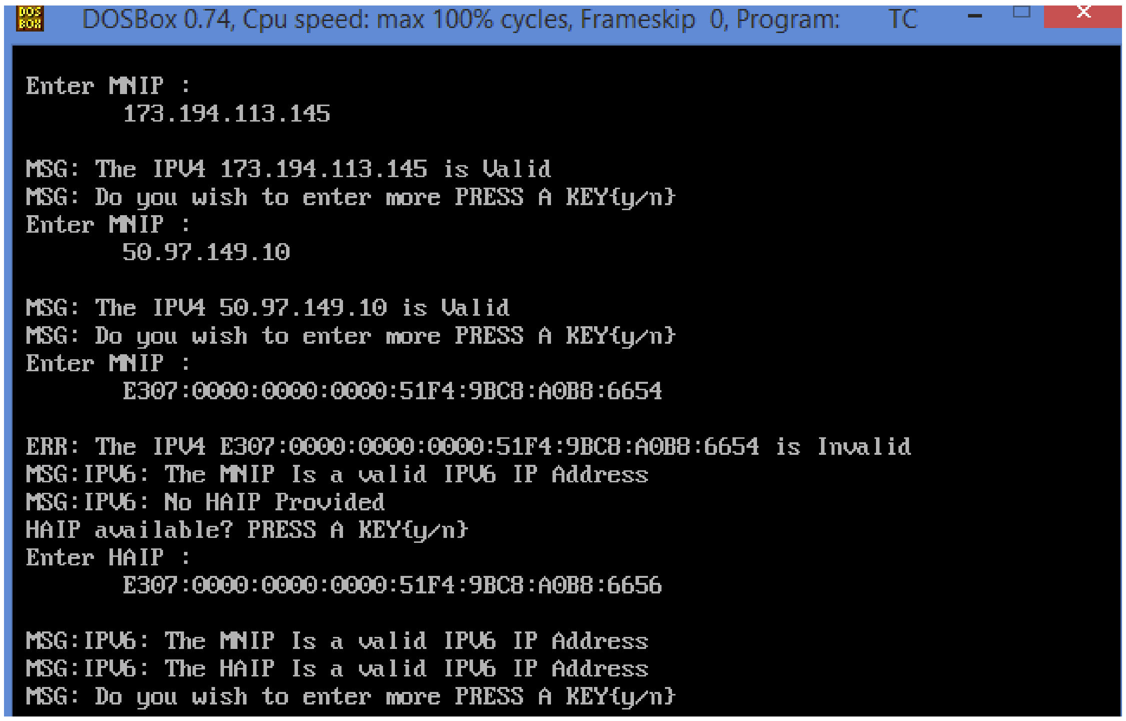
All IPv4 addresses are validated, processed and sent to IPv4 stack while IPv6 addresses are sent to IPv6 stack.

4.2.4 Experimental Results

INPUT:

2 IPv4 address and 1 IPv6 address is provided by the user

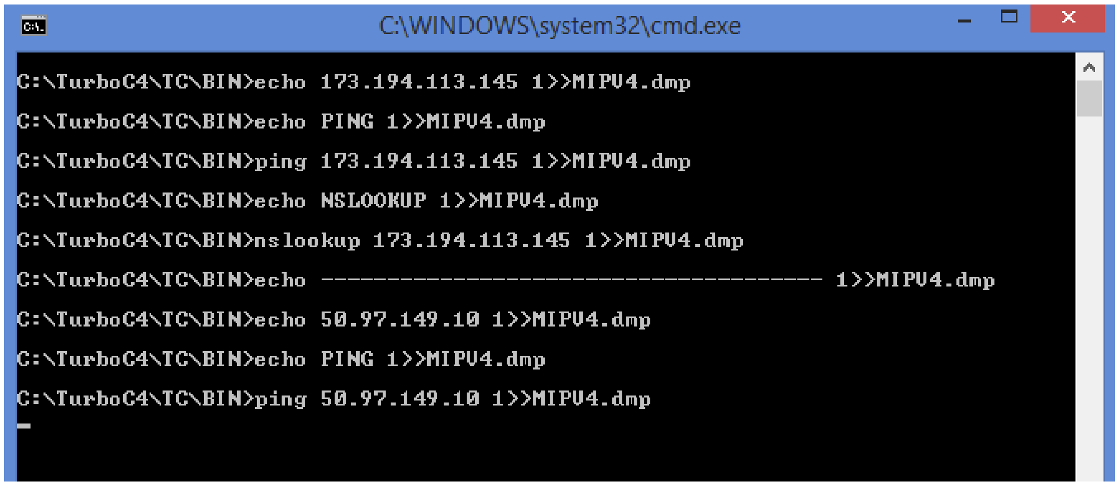
1. IPv4 address : 173. 194.113.145
2. IPv4 address : 50.97.149.10
3. IPv6 address (MNIP): E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420
4. IPv6 address (HAIP): E3D7:0000:0000:0000:51F4:9BC8:C0A8:6422.


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
Enter MNIP :
       173.194.113.145

MSG: The IPV4 173.194.113.145 is Valid
MSG: Do you wish to enter more PRESS A KEY{y/n}
Enter MNIP :
       50.97.149.10

MSG: The IPV4 50.97.149.10 is Valid
MSG: Do you wish to enter more PRESS A KEY{y/n}
Enter MNIP :
       E307:0000:0000:0000:51F4:9BC8:A0B8:6654

ERR: The IPV4 E307:0000:0000:0000:51F4:9BC8:A0B8:6654 is Invalid
MSG:IPV6: The MNIP Is a valid IPV6 IP Address
MSG:IPV6: No HAIP Provided
HAIP available? PRESS A KEY{y/n}
Enter HAIP :
       E307:0000:0000:0000:51F4:9BC8:A0B8:6656

MSG:IPV6: The MNIP Is a valid IPV6 IP Address
MSG:IPV6: The HAIP Is a valid IPV6 IP Address
MSG: Do you wish to enter more PRESS A KEY{y/n}
```

Figure 4.3 2 IPv4 address and 1 IPv6 address is provided by the user.

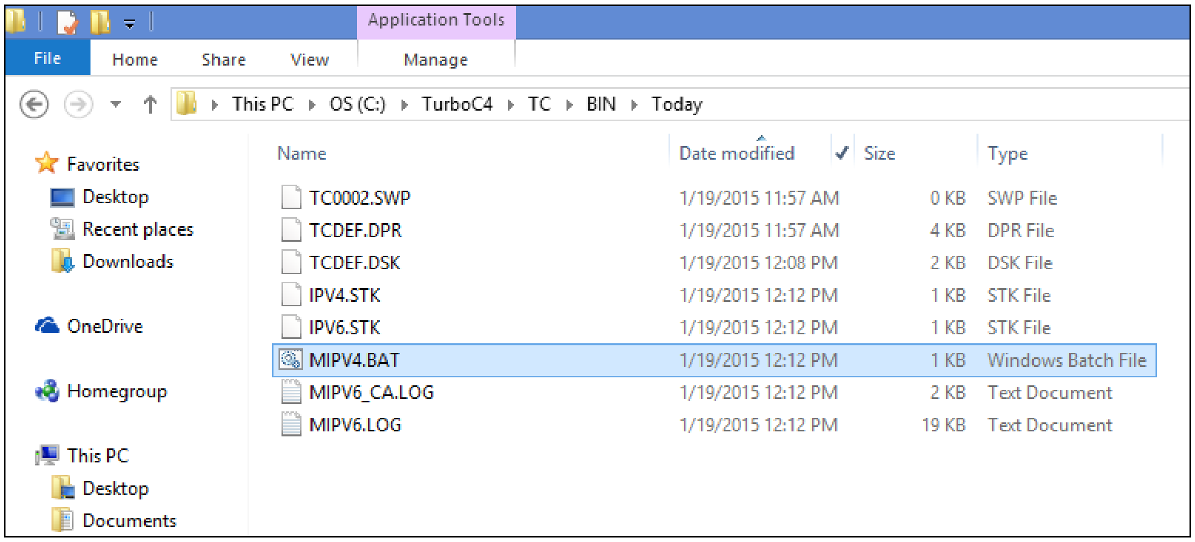
<u>Processing both the IP addresses</u>



```
C:\TurboC4\TC\BIN>echo 173.194.113.145 1>>MIPV4.dmp

C:\TurboC4\TC\BIN>echo PING 1>>MIPV4.dmp

C:\TurboC4\TC\BIN>ping 173.194.113.145 1>>MIPV4.dmp

C:\TurboC4\TC\BIN>echo NSLOOKUP 1>>MIPV4.dmp

C:\TurboC4\TC\BIN>nslookup 173.194.113.145 1>>MIPV4.dmp

C:\TurboC4\TC\BIN>echo -------------------------------------- 1>>MIPV4.dmp

C:\TurboC4\TC\BIN>echo 50.97.149.10 1>>MIPV4.dmp

C:\TurboC4\TC\BIN>echo PING 1>>MIPV4.dmp

C:\TurboC4\TC\BIN>ping 50.97.149.10 1>>MIPV4.dmp
_
```

Figure 4.4 Processing IPv4 and IPv6 addresses.

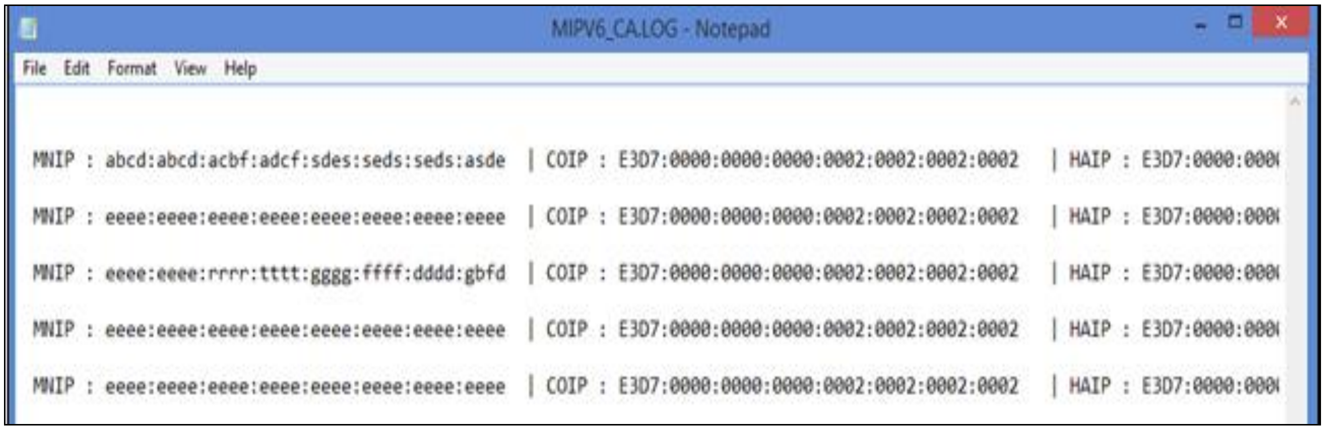
<u>Batch file and Log file generated simultaneously</u>



Name	Date modified	Size	Type
TC0002.SWP	1/19/2015 11:57 AM	0 KB	SWP File
TCDEF.DPR	1/19/2015 11:57 AM	4 KB	DPR File
TCDEF.DSK	1/19/2015 12:08 PM	2 KB	DSK File
IPV4.STK	1/19/2015 12:12 PM	1 KB	STK File
IPV6.STK	1/19/2015 12:12 PM	1 KB	STK File
MIPV4.BAT	1/19/2015 12:12 PM	1 KB	Windows Batch File
MIPV6_CA.LOG	1/19/2015 12:12 PM	2 KB	Text Document
MIPV6.LOG	1/19/2015 12:12 PM	19 KB	Text Document

Figure 4.5 Batch file generated for IPv4 address and Log file generated for IPv6 address.

Further processing of IPv4 address is carried out by Dump file being generated and the final output is shown in Excel file as mentioned in Section 3.3.3 of Chapter 3. IPv6 addresses are processed and sent to IPv6 stack which can be viewed using log file as shown below



```
MNIP : abcd:abcd:acbf:adcf:sdes:seds:seds:asde | COIP : E3D7:0000:0000:0000:0002:0002:0002:0002 | HAIP : E3D7:0000:000
MNIP : eeee:eeee:eeee:eeee:eeee:eeee:eeee:eeee | COIP : E3D7:0000:0000:0000:0002:0002:0002:0002 | HAIP : E3D7:0000:000
MNIP : eeee:eeee:rrrr:tttt:gggg:ffff:dddd:gbfd | COIP : E3D7:0000:0000:0000:0002:0002:0002:0002 | HAIP : E3D7:0000:000
MNIP : eeee:eeee:eeee:eeee:eeee:eeee:eeee:eeee | COIP : E3D7:0000:0000:0000:0002:0002:0002:0002 | HAIP : E3D7:0000:000
MNIP : eeee:eeee:eeee:eeee:eeee:eeee:eeee:eeee | COIP : E3D7:0000:0000:0000:0002:0002:0002:0002 | HAIP : E3D7:0000:000
```

Figure 4.6 Log file showing IPv6 addresses.

Note: Unlike IPv4 addresses where in batch file is overwritten for every input, it is not the same case for IPv6 addresses, here each input i.e. IPv6 address is recorded in the log file as shown in the figure 4.6.

4.2.5 Result Analysis

The function populate_hybrid_stack( ) is responsible to check whether the input provided is IPv4 or IPv6. For n number of IPv4 address and m number of IPv6 addresses as inputs and since populate_hybrid_stack( ) loops through the function for every IP address then the complexity of the function populate_hybrid_stack( ) is of the order of (n+m) given by O(n+m).

4.3 DRAWBACKS OF DUAL STACK

While implementing Dual Stack Transition mechanism, inefficiency of Mobile IPv4 and Mobile IPv6 must also be considered. Network Administrators are expected to maintain two sets of mobility management systems on the same network, with its own set of optimizations for each of its mobility management systems. Also it is required that the Mobile nodes must be both MIPv4 and MIPv6 capable while these mobile nodes must be capable to send two sets of signaling messages on every handoff. When both mobility management protocols MIPv4 and MIPv6 are running simultaneously, there are chances that certain issues like operational issues, Potential inefficiencies and IP connectivity problems may raise. To address these issues, we firstly understand their scope in the design and then according to their operation structure, find suitable solutions.

a. Operational issues: To set up tunnels, Mobile IPv4 and Mobile IPv6 use a signaling protocol. These signaling protocols are Registration Requests in Mobile IPv4 and Binding Updates in MIPv6 used to set up tunnels in two end points. At present, Mobile IP signaling is tightly coupled to the IP address family. No fundamental techniques are followed in order to process such couplings.

b. The Unfeasibility of Sustaining IP connectivity: Routing methods were built to maintain static networks. In these networks, the hosts were not allowed to travel from one subnet to another subnet. One of the most serious difficulties approached during Tunneling, is whenever a node tries to connect via IPv4, the connectivity of the IPv6 will not be able to be maintained by the network and vice versa. Despite the fact that a mobile node happens to be both, MIPv4 and MIPv6 capable, there is no guarantee in the connectivity across the different networks, since it also depends on the IPv4 or IPv6 capabilities of the networks the mobile is visiting.

c. Implementation Burdens: To roam within the Internet a node must be IPv4 and IPv6 capable and also MIPv4 and MIPv6 capable. The implementation of the protocol becomes possible only when there is a possibility to employ both IP versions from one mobility protocol.

This can be done after making the assumption that the protocol choice is known. However, in situations where the mobile node must be capable of working in any network, it may still need two protocols. The overheads involved to deploy two protocols are adequate enough thus giving rise to ample implementation burdens.

d. Operative Problems: On deploying the two protocols, another issue that comes into picture is the management of two protocols. This management is required for both the mobile node as well as the home agent. It raises significant issues for the network operator wherein any network operator is expected to have knowledge of both the protocols. Due to the lack of substantial gains, this is an operation that an operator may not be able to justify.

In order to deploy both the protocols all the security credentials must be duplicated. This duplication is applicable on mobile nodes as well as home agents. This includes keying material, IPSec security associations and new authentication protocols for Mobile IPv6 along with the associations required by Mobile IPv4. It's possible to rely on one set of common credentials. This depends on the security mechanism which is being used and some more work in this field needs to be put in. Considering everything else to remain constant, with no gain to the operator or to the mobile node, such duplication is a significant operational burden.

e. Mobility Management Inefficiencies: When mobile nodes move between two different IP networks, the mobility of these mobile nodes cannot be supported by the conventional Mobile IP protocols if the current network evolves gradually from IPv4 to IPv6. In this case it needs to acquire sufficient IP addresses and also guarantee the Quality of service (Qos). There are insufficient address spaces in the IPv4 which leads to asymmetric route for packets in MIPv4. These packets are transmitted by the Correspondent node and the Mobile Node. This phenomenon is called triangular routing. In most cases it will be quite different from the optimal routing path. Therefore as a solution to these problems, the MIPv6 protocol is proposed. This seems to be not only a solution but also an evolution of MIPV4 [59]. Consider that there is a mobile node which is moving within a network which is a dual stack network. The mobile node is expected to communicate information between two networks using mobile IP messages. These messages are sent to its home agent every time it moves to allow its IPv4 and IPv6 connections to survive. There is no particular reason for such duplication. If local mobility optimizations were deployed (e.g., Hierarchical Mobile IPv6 (HMIPv6), Fast handovers for Mobile IPv4), the mobile node will need to update the local agents running each protocol. Also ironically, one agent might be running both HMIPv6 and local MIPv4 home agent. Evidently, it is not desirable to have to send two messages along with the completion of these two sets of transactions for the same fundamental optimization. Thus the amount of bandwidth which is needed at the critical handover time will increase and also the complication of mobility management rises within the Internet when such parallel operations of Mobile IPv4 and Mobile IPv6 are run.

However due to the above mentioned issues, it becomes harder to deploy MIPv6 and any dual stack solutions. It is therefore expected to suggest or deploy a new protocol that will overcome the above mentioned drawbacks. Thus Tunneling is suggested and analyzed as the next topic in this research work.

4.4 SUMMARY

This chapter focused on Protocol Verification of Dual stack. First the operations involved in Dual stack were described and based on this description, a schema diagram was modeled to represent Dual stack. From this schema diagram, design steps was formulated which consists of 3 vertical modules along with one horizontal module. The horizontal module gave the user a choice to select one option among the 3 vertical modules. The 3 vertical modules considered here are as mentioned below

Vertical Module 1: Mobile IPv6

Vertical Module 2: Mobile IPv4

Vertical Module 3: Mobile IPv4 / Mobile IPv6

The vertical module 1 was designed to accept only IPv6 addresses whereas vertical module 2 was designed to accept IPv4 addresses only. The third module was the hybrid option which accepted Mobile IPv4 and Mobile IPv6 addresses. In the third module, all IPv6 addresses were sent to IPv6 stack and Ipv4 addresses were sent to IPv4 stack. Thus proving Dual stack.

CHAPTER 5
PROTOCOL VERIFICATION OF TUNNELING IN MIPV4/ MIPV6

The key to a successful IPv6 transition is compatibility with the large installed base of IPv4 hosts and routers. Maintaining compatibility with IPv4 while deploying IPv6 will streamline the task of transitioning the Internet to IPv6. The mechanisms discussed here are designed to be employed by IPv6 hosts and routers that need to interoperate with IPv4 hosts and utilize IPv4 routing infrastructures. A tunnel is used to provide a channel or a path through which any roaming mobile node with IPv6 data packet can travel over an IPv4 core network. When the mobile node is roaming, the IP address provided enables the IPv6 data packet to tunnel over IPv4 core network. Certain embodiments allow an IPv4 core network to support a mobile node that uses Simple IPv6 or MIPv6 addressing. This system and method can be applicable to situations where the mobile node uses 128 bits addressing, while the core network supports 32 bit addressing.

Encapsulation of IPv6 packets within IPv4 packets allows two IPv6 hosts/networks to be connected with each other while running on existing IPv4 networks. IPv6 packets are encapsulated within IPv4 packets and then are transmitted over IPv4 networks. At the destination, these packets are de-capsulated to the original IPv6 packets. It should be noted that during encapsulation of IPv6 packets within IPv4 packets, only IPv4 routing properties will be utilized and hence the IPv6 packet will lose any special IPv6 features until it is de-capsulated at the receiving host/router [60,78].

5.1 INTRODUCTION

The present development relates to a system and method of communicating information in a first protocol over a network that supports a second protocol. More particularly, a tunnel is used to communicate information in a second protocol over a network designed for a first protocol, while also providing address mobility.

Tunneling is specified by two mechanisms as mentioned below:

1. Encapsulation
2. Decapsulation

1. Encapsulation: IPv6 data packet is encapsulated within IPv4 packet as shown in figure 5.1 in a method

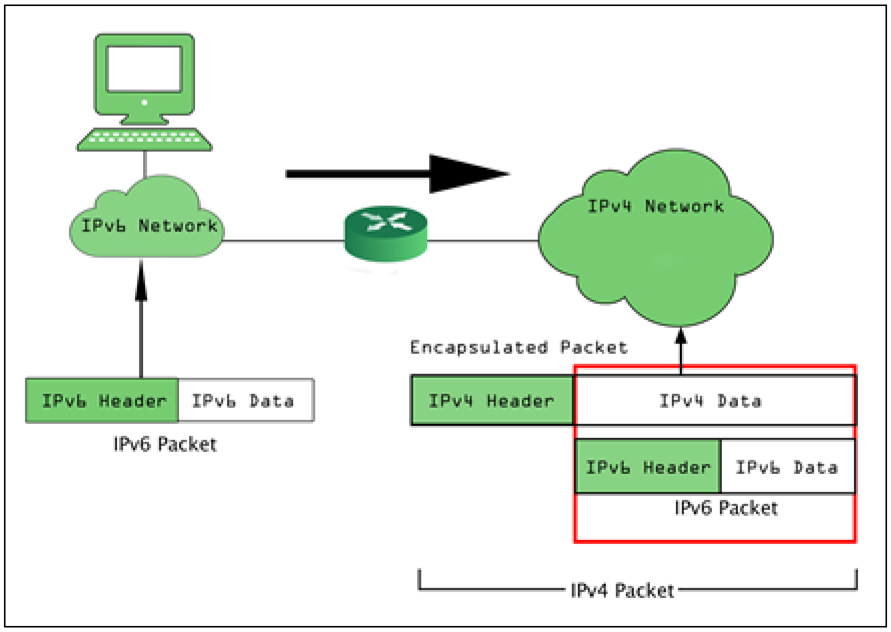


Figure 5.1 Encapsulation in Tunneling

The node at starting point of the tunnel acts as an encapsulator [61]. This encapsulator creates an encapsulating IPv4 header for an IPv6 data packet and transmits it into the tunnel [62]. When an IPv6 packet is transmitted over a tunnel, the destination and source addresses for the encapsulating IPv4 header are set as described in table 5.1.

Table 5.1 IPv4 Header construction

Version:	4
IP Header Length in 32-bit words:	5
Type of Service:	0 unless otherwise specified.
Fragment Offset:	Set as necessary if fragmenting.
Source Address:	An IPv4 address of the encapsulator: either configured by the administrator or an address of the outgoing interface.
Destination Address:	IPv4 address of the tunnel endpoint.
Header Checksum:	Calculate the checksum of the IPv4 header [63].
Protocol:	41 (Assigned payload type number for IPv6).
Time to Live:	Set in an implementation-specific manner [64].
Flags:	a. Set the Don't Fragment (DF) flag.

	b. Set the More Fragments (MF) bit as necessary if fragmenting.
Identification:	Generated uniquely as for any IPv4 packet transmitted by the system.
Total Length:	Payload length from IPv6 header plus length of IPv6 and IPv4 headers (i.e., IPv6 payload length plus a constant 60 bytes).

In order to process IPv6 packets forwarded into the tunnel the encapsulator is supposed to maintain and record information about maximum transmission unit (MTU) for each tunnel. During Encapsulation, care is taken that tunnel MTU is between 1280 and 1480 bytes [63, 65].

2. Decapsulation: As the original IPv6 packet is forwarded further into the tunnel, the IPv4 packet header is stripped and this process is termed as Decapsulation. The exit node of tunnel acts as a decapsulator. Each decapsulator will have more than one tunnel configured to it. The decapsulator will match the received data packets to the tunnels that are configured to it, and it allows only the packets in which IPv4 source addresses match the tunnels that are configured on the decapsulator. Thus it is the responsibility of operator to ensure that the tunnel's IPv4 address configuration is the same both at the encapsulator and the decapsulator. Decapsulation is well explained with a diagram as shown in figure 5.2

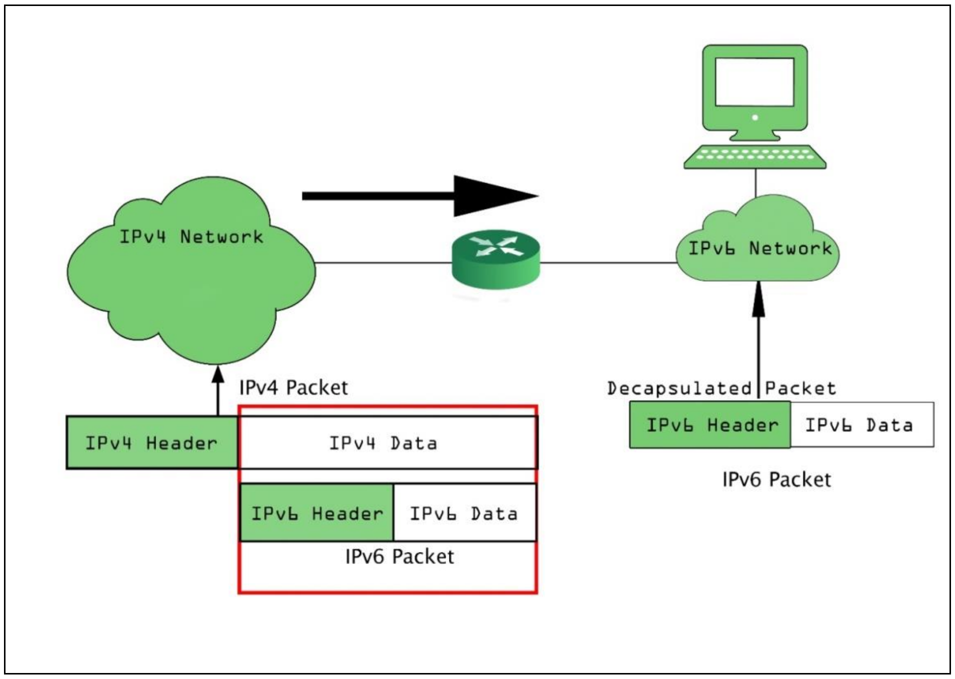


Figure 5.2 Decapsulation

When an IPv6/IPv4 host or a router receives an IPv4 datagram, the packet is potentially a tunnel packet and needs to be verified to belong to one of the configured tunnel interfaces (by checking source/destination addresses), reassembled (if fragmented at the IPv4 level), and have the IPv4 header removed such that the resulting IPv6 datagram is submitted to the IPv6 layer. The decapsulator must verify that the tunnel source address is correct before further processing packets. This is done by verifying that the source address is the IPv4 address at the encapsulator, as configured on the decapsulator. Packets for which the IPv4 source address does not match must be discarded. The decapsulator should be capable of having, on the tunnel interfaces, an IPv6 MRU (maximum receive unit) of at least the maximum of 1500 bytes [65].

Systems and methods are provided for transferring data packets with IPv6 protocol over a network core that supports IPv4 protocol. The data packet is transferred through a tunnel that also allows a mobile node to maintain the same address while roaming on the network. The mobile node communicates with a FA in IPv6 or MIPv6 while the network between the routing device and a home agent is an IPv4 network. A bi-directional tunnel provides communication in IPv6 or MIPv6 over the IPv4 network. This system features a data packet communication system in which a FA communicates with a mobile node using IPv6 protocol and a FA in a network core that uses IPv4 protocol. A home agent (HA) is coupled to the network core, wherein a tunnel is established between the FA and the HA to carry encapsulated data packets using the IPv6 protocol over the IPv4 network. It features a communication method where in an address is assigned to a mobile node; establishing a tunnel from a FA to a home agent to exchange data packet in IPv6 protocol over a network core that uses IPv4 protocol.

5.2 PROTOCOL VERIFICATION

The network environment in Tunneling consists of IPv6 / IPv4 routers interconnected by an IPv4 infrastructure that can tunnel IPv6 packets between themselves [19]. It is the co-existence phase in which IPv4 is not entirely replaced by IPv6 and IPv6 is not completely deployed. The first step carried out in this research work leading towards Protocol Verification of Tunneling in MIPv4 / MIPv6, was to model the description of Tunneling as provided in section 2.4.1 of this thesis. The basic step in Tunneling is to validate the given IP address. The input IP address is checked for the IPv4 and IPv6 address format. After IP address is validated, all IPv4 datapackets are processed as mentioned in section 3.3 while IPv6 datapackets are processed by Encapsulation and Decapsulation methods. In Encapsulation method, the IPv6 datapackets are encapsulated in IPv4 datapackets and sent through IPv4 network and later each IPv4 datapackets are decapsulated to its original form. Protocol Verification of Tunneling was carried out by following steps,

- Modeling of Tunneling in MIPv4 / MIPv6 using schema diagram.
- Formulating design steps for each module from schema diagram
- Implementing each design step using Programming language C
- Noting down experimental results using screen shots.

5.2.1 Modeling of Tunneling in MIPv4 / MIPv6 using schema diagram.

The schema diagram of Tunneling inMIPv4 / MIPv6 is shown in figure 5.3. In the flowchart we have an IP address given as input. If it is not a valid IP address then an error message is displayed, while on the other hand if the input is a valid IP address then the function check_inputs( ) is used to check if the IP address is a valid IPv4 address or IPv6 address. The function Generate_Packet_data( ) will generate an IPv4 data packet if the input is a valid IPv4 address while if the input is IPv6 address then an IPv6 data packet is generated which is shown in an XML file. Now all the IPv6 data packets are sent as input to the function Generate_encapsulate_packet_data( ) and this function will encapsulate IPv6 data packets in an IPv4 data packets. The function Decapsulate_packet_data( ) will strip off IPv4 header from the encapsulated IPv6 data packet to give the original IPv6 data packet as the final output. The above mentioned steps are utilized to develop the schema diagram for Tunneling in MIPv4 / MIPv6 as shown in figure 5.3.

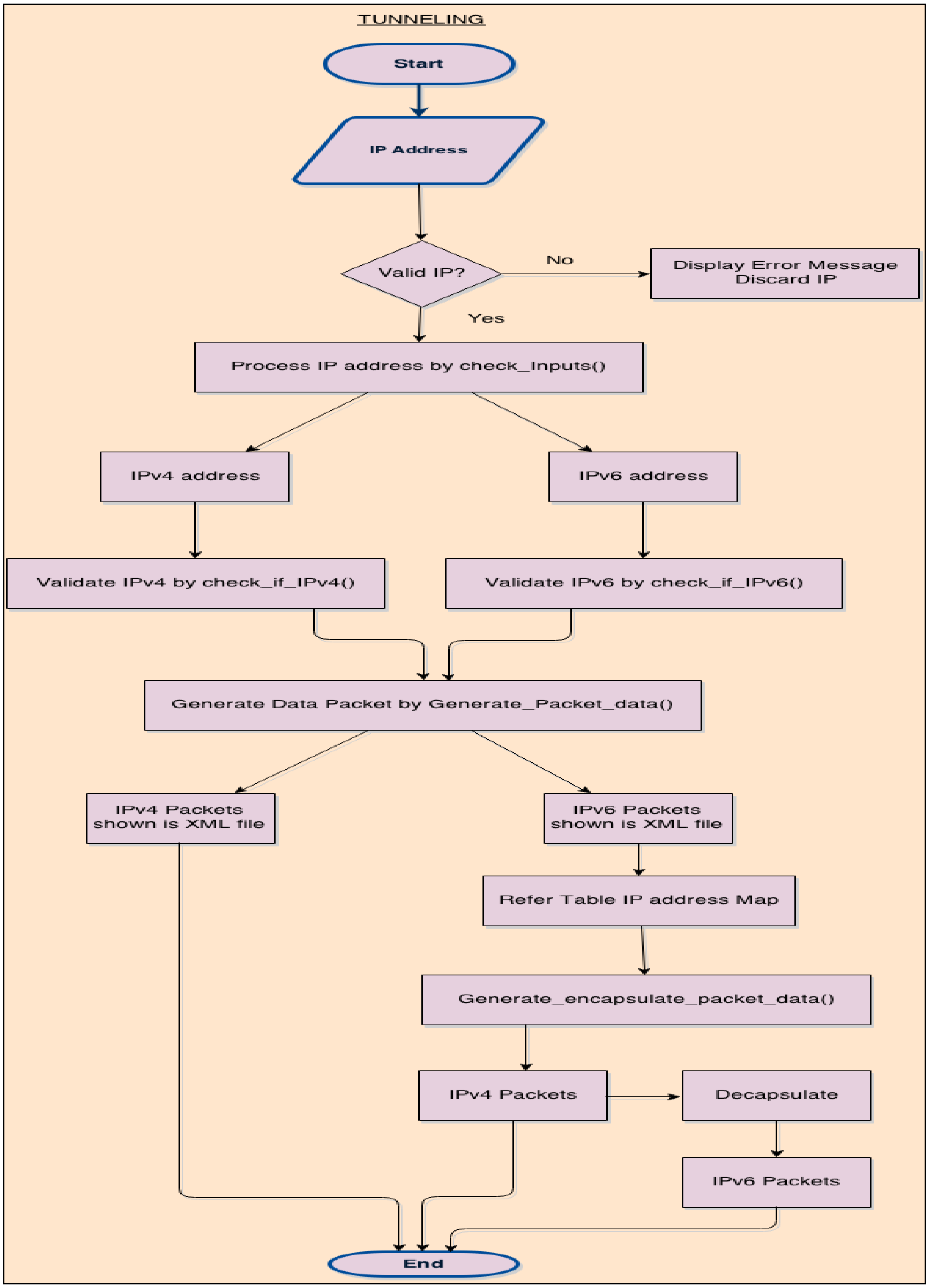

TUNNELING
Start
IP Address
Valid IP?
No
Display Error Message Discard IP
Yes
Process IP address by check_Inputs()
IPv4 address
IPv6 address
Validate IPv4 by check_if_IPv4()
Validate IPv6 by check_if_IPv6()
Generate Data Packet by Generate_Packet_data()
IPv4 Packets shown is XML file
IPv6 Packets shown is XML file
Refer Table IP address Map
Generate_encapsulate_packet_data()
IPv4 Packets
Decapsulate
IPv6 Packets
End

Figure 5.3 Schema diagram of Tunneling in MIPv4/MIPv6

5.2.2 Formulating design steps for each module from schema diagram

This section presents the design steps for the encapsulation of IPv6 datapackets with IPv4 datapackets and its transfer over IPv4 network using tunneling simulation. This design will be implemented using programming language **C**. The inputs will be taken from command line arguments. The output and logging will be done in separate log files. The scenario assumed here is a network with IPv6/IPv4 routers interconnected by an IPv4 infrastructure which can tunnel IPv6 packets between themselves [19].

The functions used for Tunneling in MIPv4 / MIPv6 is explained as below

5.2.2.1 Function Check_Inputs( )

This function will read the inputs from the user and check for the validity of the input IP address, whether the input is in valid IP address format.

5.2.2.2 Function Check_If_IPv4( )

This function will validate all IPv4 address entered in the input field and any address other than IPv4 format will be discarded

5.2.2.3 Function Check_If_IPv6( )

This function will validate IPv6 address only and discard all the inputs other than IPv6 format.

5.2.2.4 Function Generate_packet_data( )

This function will take all inputs provided by the user and generate a datapacket. Each datapacket will have the following,

i) Source address,

ii) Destination address and

iii) The body with one or two text messages as shown below,

```
< BODY >      (indicates beginning of body)
            Hello           (text message)
< / BODY >     (  / indicates end of body)
```

This function Generate_packet_data ( ) generates datapacket based on the input provided by the user, if the input is IPv4 address then IPv4 datapacket is generated as shown below,

```
<IPv4>
<SOURCE_IP> Source IPv4 Address </SOURCE_IP>
<DESTINATION_IP> Destination IPv4 Address </DESTINATION_IP>
<BODY >
MESSAGE
</BODY>
</IPv4>
```

And if the input is IPv6 address then IPv6 datapacket gets generated as shown below

```
<IPv6>
<SOURCE_IP> Source IPv6 Address </SOURCE_IP>
<DESTINATION_IP> Destination IPv6 Address </DESTINATION_IP>
<BODY >
MESSAGE
</BODY>
</IPv6>
```

The code makes use of XML file tags to show how an actual IPv4 and IPv6 datapackets will appear when provided with all fields.

5.2.2.5 Function Generate-encapsulate-packet-data ( )

This function is responsible for Encapsulation of IPv6 datapackets within an IPv4 datapacket.

Case 1: The IPv6 datapacket generated from the previous function is used as the input to generate Encapsulated IPv6 datapacket. The internal mapping table IPADDRESSMAP which is a simple flat file was used to find the corresponding IPv4 address. If no match is found within the mapping table, then a default IPv4 address was provided. This IPv4 address was used as the header that will be attached to the IPv6 datapacket to complete the process of Encapsulation.

The Function Generate-encapsulate-packet-data ( ) makes use of an Internal Table named IPADDRESSMAP.txt. Here IPv4 addresses are mapped to IPv6 addresses. This file contains the required information in format provided below

```
IPv6 ADDRESS1 | IPv4 ADDRESS1
IPv6 ADDRESS2 | IPv4 ADDRESS2
IPv6 ADDRESS3 | IPv4 ADDRESS3
IPv6 ADDRESS4 | IPv4 ADDRESS4 and so on.
```

Case 2: If the Input is IPv4 address, then IPv4 datapacket is generated as normal datapackets.

5.2.2.6 Function Decapsulate-packet-data ( )

This function will take as input datapacket generated in the previous section. The IPv4 header is now stripped off to generate the original IPv6 data packet. The design steps for Encapsulation and Decapsulation in Tunneling used in this thesis is summarized as follows;

The Function Generate_packet_data( ) generates IPv6 datapacket [intermediate file], which acts as an input to the function Generate-encapsulate-packet-data ( ) to generate encapsulated IPv6 datapackets. Then the Function Decapsulat_packet_data( ) will take as encapsulated datapacket [file] as input which will be decoded onto its underlying message and then provided as a flat file with message details.

5.2.3 Implementing each design step using Programming language C

Step 1: Encapsulation of IPv6 datapacket in IPv4 datapacket

This function generate_packet_data ( ) will generate IPv4 datapackets [intermediate file] which will be the input for the function Generate-encapsulate-packet-data ( ) which will generate Encapsulated IPv6 datapacket which is demonstrated using XML file.

Case 1:

INPUT:

Source address:	E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420
Destination address:	EEEE:E3D7:0000:0000:51F4:C048:FFFF:B9C8
Text message:	IPv6 datapacket

OUTPUT:

IPv6 encapsulated in IPv4 datapacket as shown below

```
<IPv4>
<SOURCE_IP> Source IPv4 Address </SOURCE_IP>
<DESTINATION_IP> Destination IPv4 Address </DESTINATION_IP>
<BODY >
        <IPv6>
        <SOURCE_IP> Source IPv6 Address </SOURCE_IP>
        <DESTINATION_IP> Destination IPv6 Address </DESTINATION_IP>
        <BODY >
        Message
        </BODY>
        </IPv6>
</BODY>
</IPv4>
```

Case 2:

INPUT:

Source address:	173. 194.113.145
Destination address:	50.97.149.10
Text Message:	IPv4 datapacket

OUTPUT :

Normal IPv4 datapacket

Step 2 : Decapsulation

The Function Decapsulate-packet-data ( ) will strip off IPv4 header to generate the original IPv6 datapacket.

INPUT: Encapsulated IPv6 datapacket

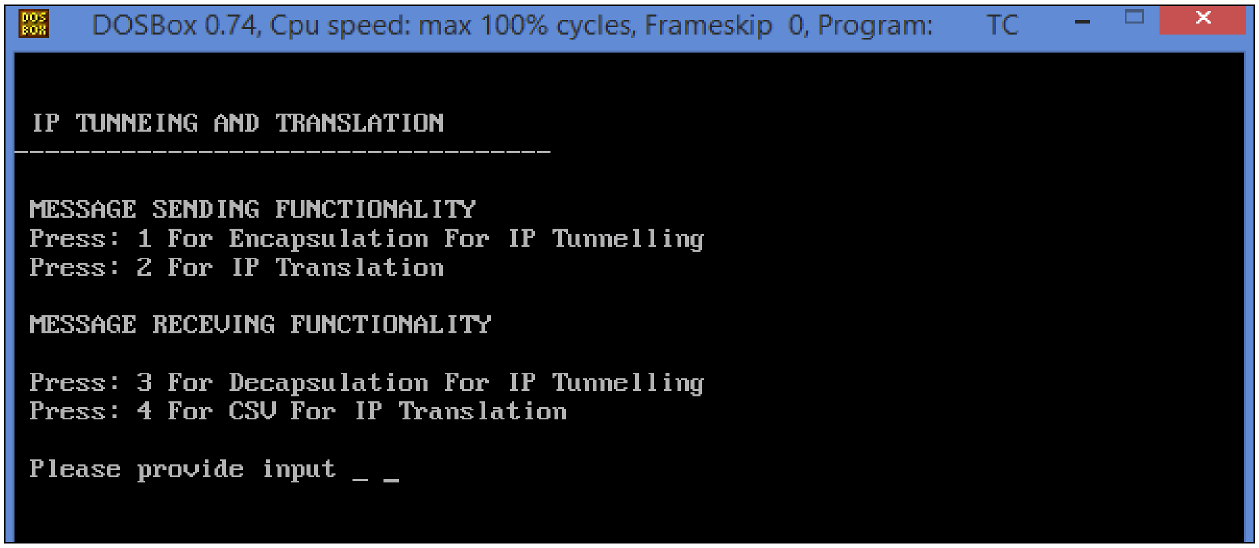
OUTPUT: Original IPv6 datapacket

Decapsulated packet will look like below.

```
<IPv6>
<SOURCE_IP> Source Ipv6 Address </SOURCE_IP>
<DESTINATION_IP> Destination Ipv6 Address </DESTINATION_IP>
<BODY >
Message
</BODY>
</IPv6>
```

5.2.4 Experimental Results

Option 1 is selected for Encapsulation

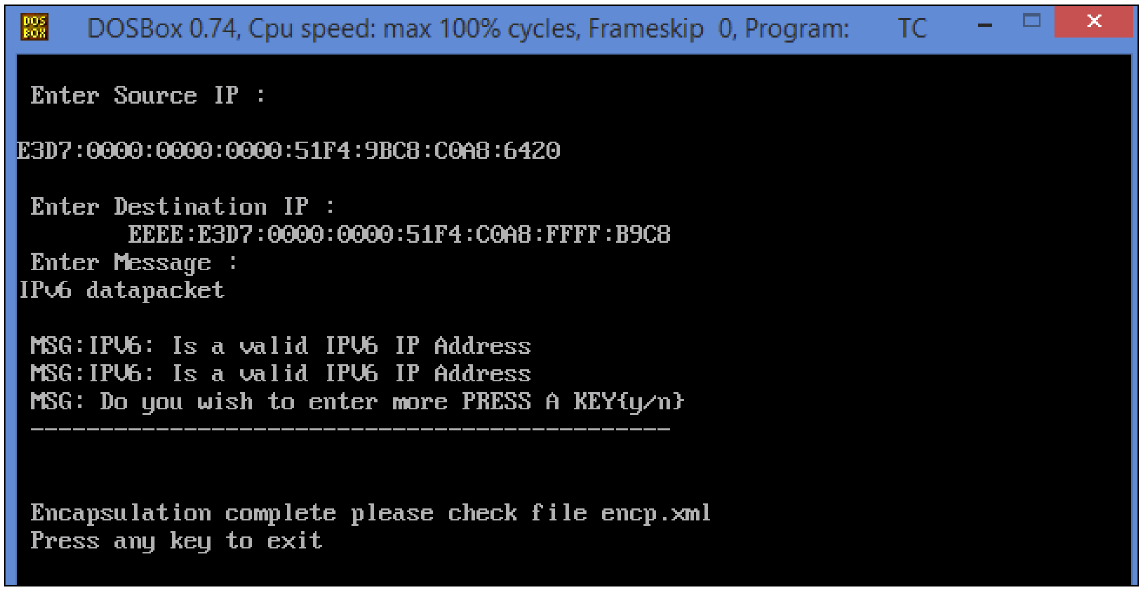

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
 IP TUNNEING AND TRANSLATION
-----------------------------------

 MESSAGE SENDING FUNCTIONALITY
 Press: 1 For Encapsulation For IP Tunnelling
 Press: 2 For IP Translation

 MESSAGE RECEVING FUNCTIONALITY

 Press: 3 For Decapsulation For IP Tunnelling
 Press: 4 For CSV For IP Translation

 Please provide input _ _
```

Figure 5.4 Options for Encapsulation, Decapsulation and Translation.

After Option 1 is selected, User provides the following INPUT

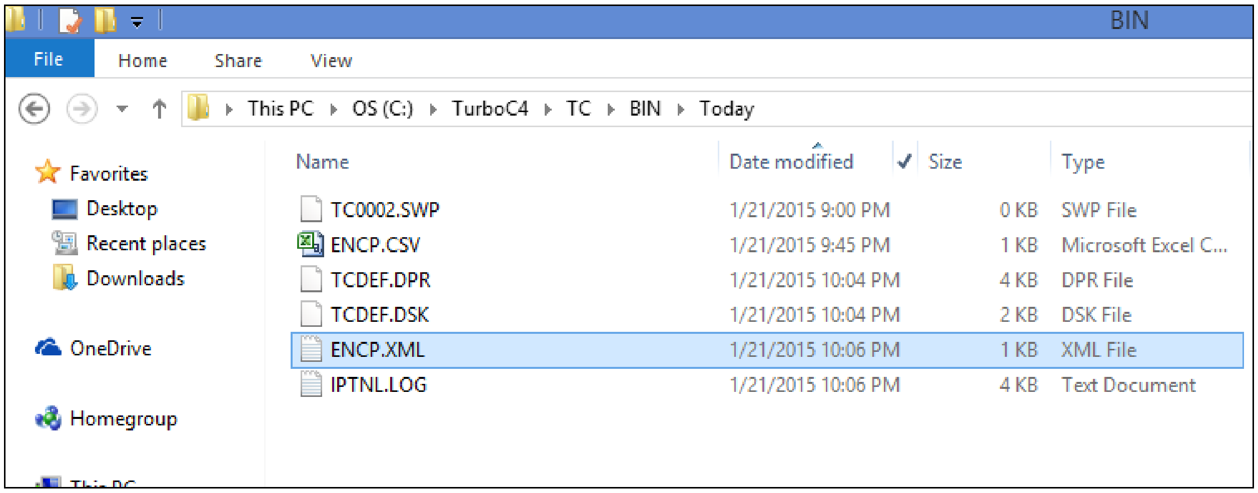
Source address:	E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420
Destination address:	EEEE:E3D7:0000:0000:51F4:C048:FFFF:B9C8
Text message:	IPv6 datapacket


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
 Enter Source IP :

E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420

 Enter Destination IP :
        EEEE:E3D7:0000:0000:51F4:C0A8:FFFF:B9C8
 Enter Message :
IPv6 datapacket

 MSG:IPV6: Is a valid IPV6 IP Address
 MSG:IPV6: Is a valid IPV6 IP Address
 MSG: Do you wish to enter more PRESS A KEY{y/n}
 ----------------------------------------------


 Encapsulation complete please check file encp.xml
 Press any key to exit
```

Figure 5.5 XML file is generated to show that Encapsulation is complete.

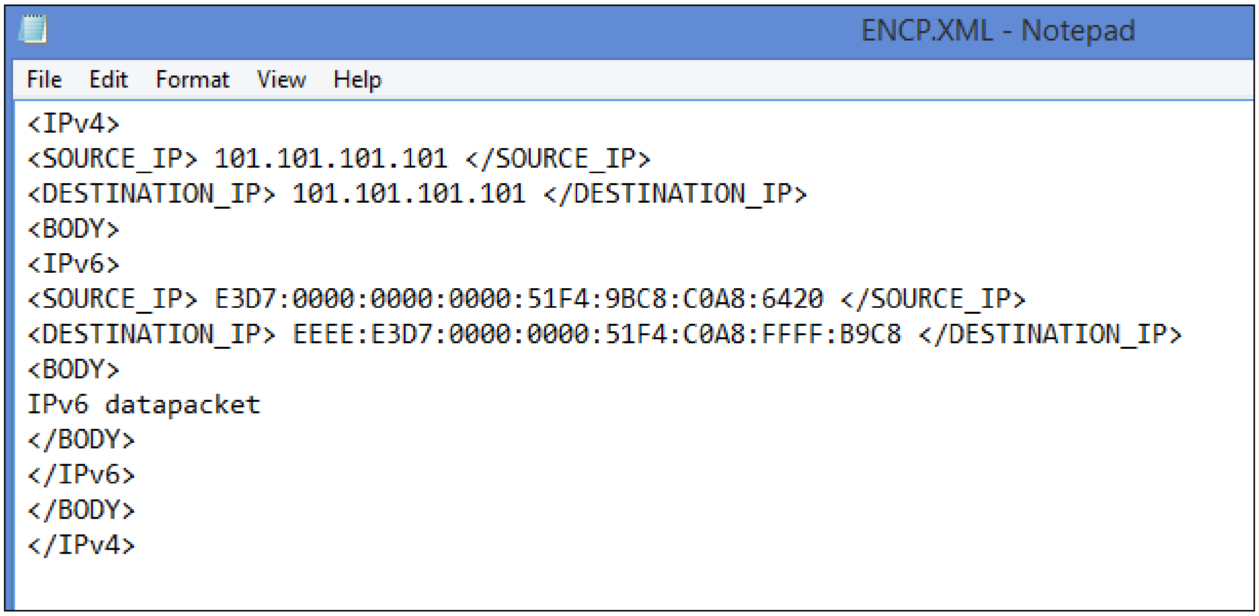
After Encapsulation XML file is generated as shown below



Name	Date modified	Size	Type
TC0002.SWP	1/21/2015 9:00 PM	0 KB	SWP File
ENCP.CSV	1/21/2015 9:45 PM	1 KB	Microsoft Excel C...
TCDEF.DPR	1/21/2015 10:04 PM	4 KB	DPR File
TCDEF.DSK	1/21/2015 10:04 PM	2 KB	DSK File
ENCP.XML	1/21/2015 10:06 PM	1 KB	XML File
IPTNL.LOG	1/21/2015 10:06 PM	4 KB	Text Document

Figure 5.6 XML file is generated after Encapsulation of IPv6 datapacket.

IPv6 datapacket encapsulated within an IPv4 datapacket

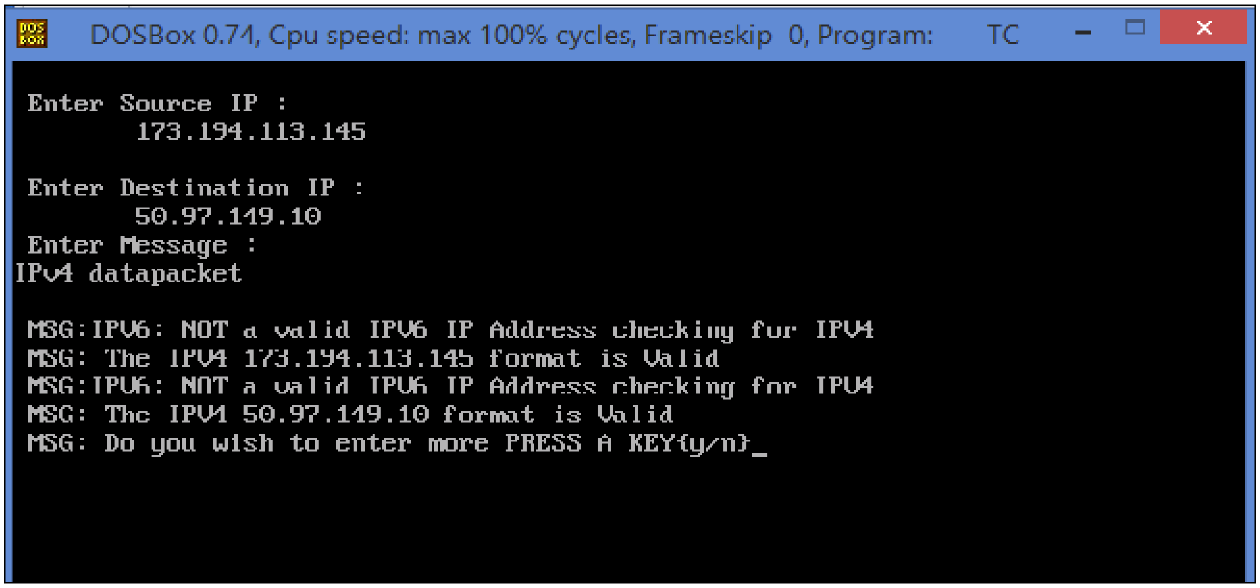


```
<IPv4>
<SOURCE_IP> 101.101.101.101 </SOURCE_IP>
<DESTINATION_IP> 101.101.101.101 </DESTINATION_IP>
<BODY>
<IPv6>
<SOURCE_IP> E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420 </SOURCE_IP>
<DESTINATION_IP> EEEE:E3D7:0000:0000:51F4:C0A8:FFFF:B9C8 </DESTINATION_IP>
<BODY>
IPv6 datapacket
</BODY>
</IPv6>
</BODY>
</IPv4>
```

Figure 5.7 IPv6 datapacket encapsulated in IPv4.

Case 2: User provides the following INPUT

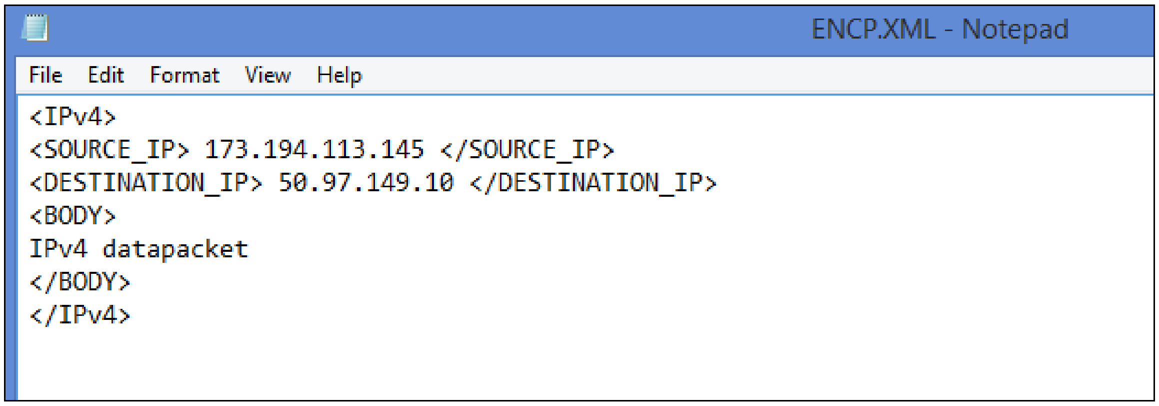
Source address:	173. 194.113.145
Destination address:	50.97.149.10
Text Message:	IPv4 datapacket


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
 Enter Source IP :
        173.194.113.145

 Enter Destination IP :
        50.97.149.10
 Enter Message :
IPv4 datapacket

 MSG:IPV6: NOT a valid IPV6 IP Address checking for IPV4
 MSG: The IPV4 173.194.113.145 format is Valid
 MSG:IPV6: NOT a valid IPV6 IP Address checking for IPV4
 MSG: The IPV4 50.97.149.10 format is Valid
 MSG: Do you wish to enter more PRESS A KEY{y/n}_
```

Figure 5.8 Tunneling of IPv4 datapacket.

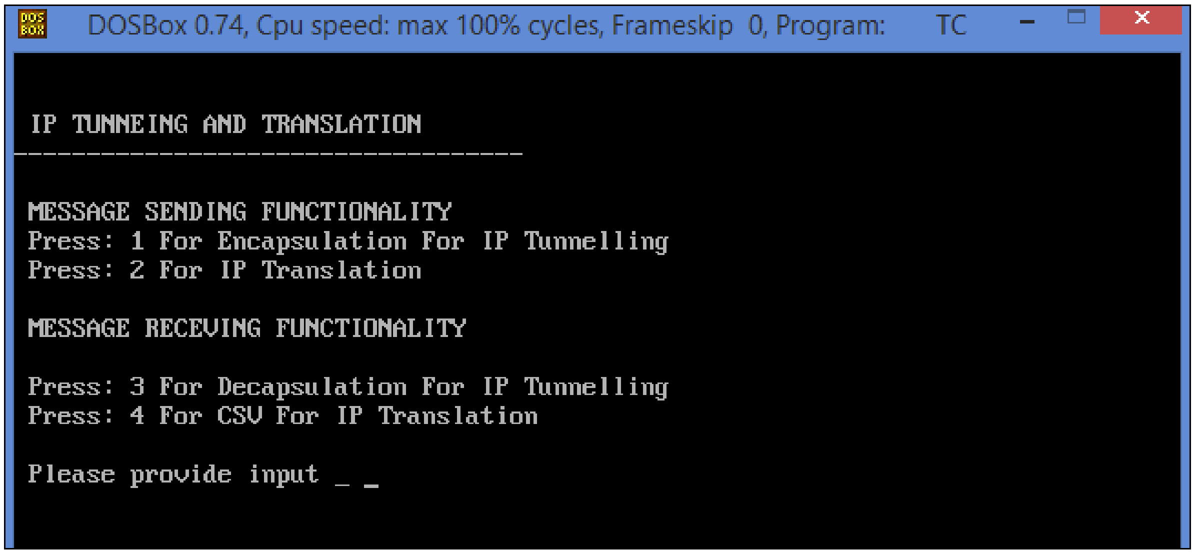
IPv4 datapacket with IPv4 Source address and IPv4 Destination address in the header field along with a text message in the body of the datapacket.



```
<IPv4>
<SOURCE_IP> 173.194.113.145 </SOURCE_IP>
<DESTINATION_IP> 50.97.149.10 </DESTINATION_IP>
<BODY>
IPv4 datapacket
</BODY>
</IPv4>
```

Figure 5.9 IPv4 datapacket.

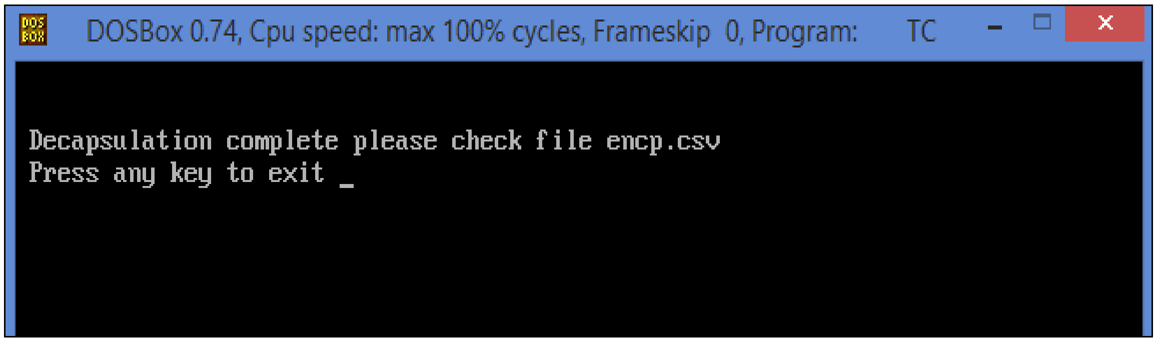
Option 3 is selected for Decapsulation


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
 IP TUNNEING AND TRANSLATION
-----------------------------------

MESSAGE SENDING FUNCTIONALITY
Press: 1 For Encapsulation For IP Tunnelling
Press: 2 For IP Translation

MESSAGE RECEVING FUNCTIONALITY

Press: 3 For Decapsulation For IP Tunnelling
Press: 4 For CSV For IP Translation

Please provide input _ _
```

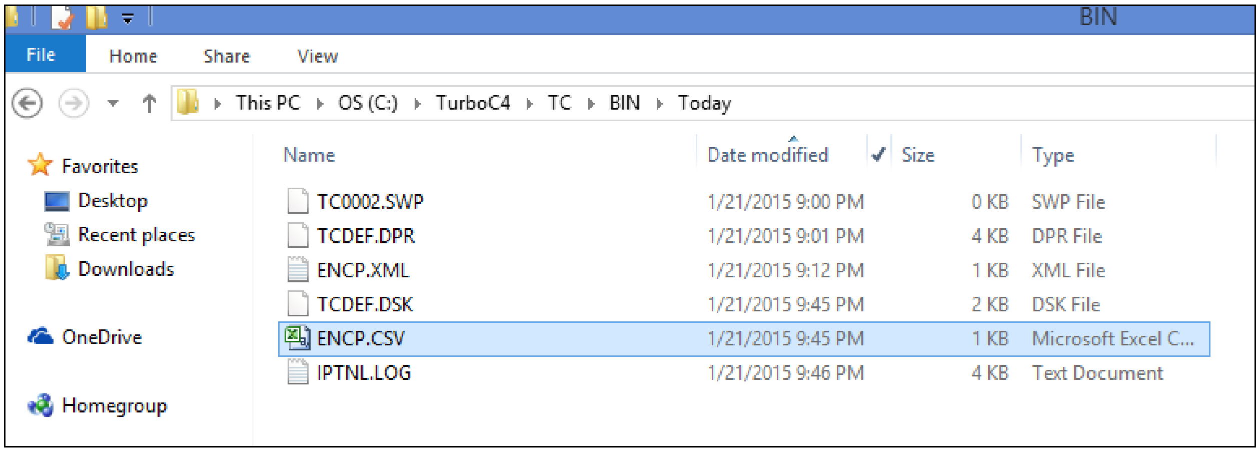
Figure 5.10 User selects Option 3 for Decapsulation

After decapsulation, CSV file is generated to show the original IPv6 datapacket.


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

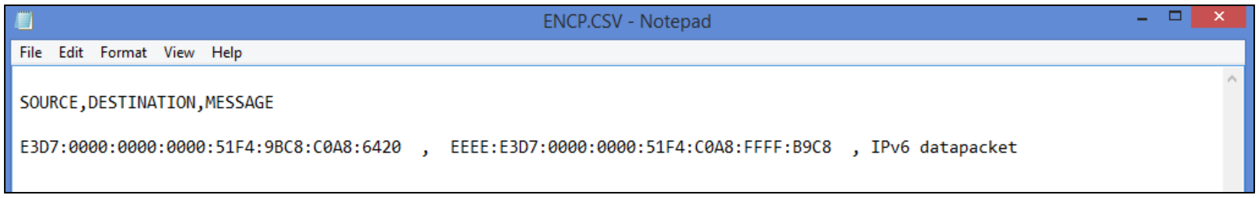
```
Decapsulation complete please check file encp.csv
Press any key to exit _
```

Figure 5.11 Decapsulation is complete.



Name	Date modified	Size	Type
TC0002.SWP	1/21/2015 9:00 PM	0 KB	SWP File
TCDEF.DPR	1/21/2015 9:01 PM	4 KB	DPR File
ENCP.XML	1/21/2015 9:12 PM	1 KB	XML File
TCDEF.DSK	1/21/2015 9:45 PM	2 KB	DSK File
ENCP.CSV	1/21/2015 9:45 PM	1 KB	Microsoft Excel C...
IPTNL.LOG	1/21/2015 9:46 PM	4 KB	Text Document

Figure 5.12 CSV file is generated.



```
SOURCE,DESTINATION,MESSAGE
E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420  ,  EEEE:E3D7:0000:0000:51F4:C0A8:FFFF:B9C8  , IPv6 datapacket
```

Figure 5.13 IPv6 datapacket in CSV file.

This section presented the steps in which IPv4 header with source address and destination address being attached to IPv6 datapacket to demonstrate the process of Encapsulation, similarly Decapsulation was demonstrated by IPv4 header being stripped off to generate the original IPv6 datapacket, thus completing the Tunneling mechanism.

5.2.5 Result Analysis

The function generate_packet_data ( ) will generate IPv4 datapackets which will be the input for the function Generate-encapsulate-packet-data ( ) which will generate encapsulated IPv6 datapacket. Since generate_packet_data ( ) loops through the function for every IP address and for n number of inputs then the complexity of the function generate_packet_data ( ) is of the order of (n) given by O(n).

5.3 DRAWBACKS OF TUNNELING

There are two disadvantages in Tunneling:

1. Users of the new architecture are unable to use the services of the underlying infrastructure, and
2. It doesn't enable new protocol users to communicate with old protocol users without dual-stack hosts

Thus Tunneling features a system that provides a data packet communication system to communicate between a mobile node using IPv6 protocol and a network core that uses IPv4 protocol wherein a tunnel is established to carry encapsulated data packets using IPv6 protocol over IPv4 network. The specification mentioned here contains rules that apply tunnel source address verification in particular and ingress filtering [66, 67] in general to packets before they are decapsulated. Here next question that is raised is whether ingress filtering should be applied for IPv4 and IPv6, as without specific ingress filtering checks in the decapsulator, it would be possible for an attacker to send an IPv6-in-IPv4 packet and make the data packet a spoofed packet giving rise to several threats which make Tunnels more vulnerable. Thus there is a need to consider another transition mechanism called Translation.

5.4 SUMMARY

This chapter dealt with Protocol verification of Tunneling. Tunneling includes Encapsulation and Decapsulation which were described here in a detailed manner. As the next step, a schema diagram was developed to represent Tunneling. Then design steps were formulated such that data packet with IPv6 address was encapsulated inside an IPv4 data packet. The design steps were implemented in C to generate xml file to show IPv6 data packet encapsulated within an IPv4 data packet, similarly the code generated a CSV file to show decapsulation.

CHAPTER 6
PROTOCOL VERIFICATION OF TRANSLATION IN MIPv4 / MIPv6

If IPv6-to-IPv4 translation is satisfactory to address the systems to which a user desires access, then one wishes only to reallocate the current existing IPv4 address space to crack the problem. If it becomes compulsory to deploy IPv6 in order to attain large amounts of address space and a company fails to do so then it will obviously fail to propose connectivity to new markets. There is one way to reduce this risk. While implementing IPv6, deploy it in a network isolated from the one running user services for IPv4. By doing this, even the cost issue is solved as we know running IPv4 and IPv6 at the same time will cost more than running either one alone. Once IPv6 usage gets implemented in the network and IPv6 has been proven to be satisfactory for the purpose then we can start removing IPv4 Address and Mail Exchanger records (A and MX) from DNS. After following the procedures and algorithms properly, once the translation is executed and it is finally in operation it can be used either as a permanent solution or a temporary solution

6.1 INTRODUCTION

Translation is subdivided into following five elements

1) Address translation
2) IP & ICMP translation
3) Maintaining translation state
4) DNS64 & DNS46
5) ALGs

6.1.1 Address translation

Basically address translation is a method of translating IP address (of one network) to different IP address (under another network). Network Address Translation (NAT) allows a single device, such as a router, to act as an agent between the Internet (or "public network") and a local (or "private") network. This means that only a single, unique IP address is required to represent an entire group of computers.

NAT is like the receptionist in a large office. Let's say you have left instructions with the receptionist not to forward any calls to you unless you request it. Later on, you call a potential client and leave a message for that client to call you back. You tell the receptionist that you are expecting a call from this client and to put her through. The client calls the main number to your office, which is the only number the client knows. When the client tells the receptionist that she is looking for you, the receptionist checks a lookup table that matches your name with your extension. The receptionist knows that you requested this call, and therefore forwards the caller to your extension. Network Address Translation is used by a device (firewall, router or a computer) that sits between an internal network and the rest of the world.

NAT Configuration

NAT router is configured to translate unregistered (inside, local) IP addresses, that reside on the private (inside) network, to registered IP addresses. This happens whenever a device on the inside with an unregistered address needs to communicate with the public (outside) network. An ISP assigns a range of IP addresses to an organization. The assigned block of addresses are registered, unique IP addresses and are called inside global addresses. Unregistered, private IP addresses are split into two groups. One is a small group (outside local addresses) that will be used by the NAT routers. The other, much larger group, known as inside local addresses, will be used on the stub domain or LAN. The outside local addresses are used to translate the unique IP addresses, known as outside global addresses, of devices on the public network. Thus we can say that NAT is transparent. Figure 6.1 shows the mapping and NAT required for Translation.

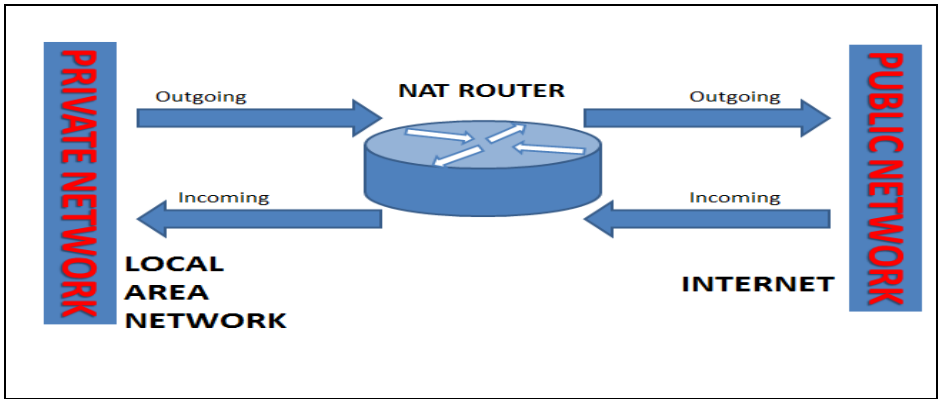


Figure 6.1 Mapping with NAT

1. Most computers on the LAN communicate with each other using the inside local addresses.
2. Some computers on the LAN communicate a lot outside the network. These computers have inside global addresses, which means that they do not require translation.
3. When a computer on the LAN that has an inside local address wants to communicate outside the network, the packet goes to one of the NAT routers.
4. The NAT router checks the routing table to see if it has an entry for the destination address. If it does, the NAT router then translates the packet. If the destination address is not in the routing table, the packet is dropped.
5. Using an inside global address, the router sends the packet on to its destination.
6. A computer on the public network sends a packet to the private network. The source address on the packet is an outside global address. The destination address is an inside global address.
7. The NAT router looks at the address translation table and determines that the destination address is in there, mapped to a computer on the LAN.
8. The NAT router translates the inside global address of the packet to the inside local address, and sends it to the destination computer.

Thus with the help of Mapping technique mentioned above, all the data packets with IPv4 addresses are translated to IPv6 addresses.

6.1.2 IP & ICMP translation

In order to understand IP and ICMP translation, first the relation between IP and ICMP is given below. The IP (Internet Protocol) depend on numerous other protocols to achieve essential control and routing tasks [68,69]. Whereas ICMP abbreviated as internet control message protocol is actually helper protocol which supports IP for error reporting and flow control. ICMP messages are encapsulated as IP datagrams. ICMP is considered a vital part of IP, even though it is architecturally layered upon IP [70]. Figure 6.2 shows an IP header attached to ICMP message.

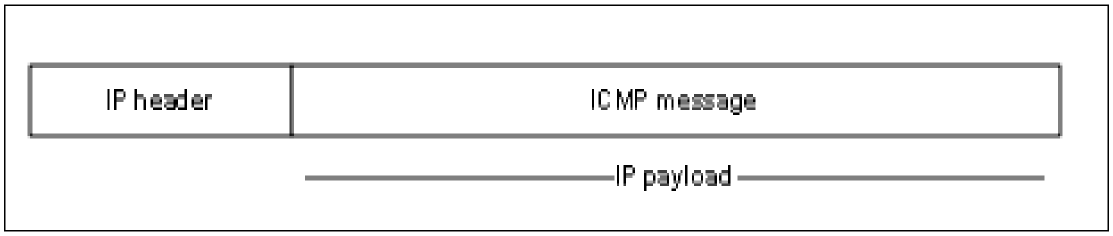


Figure 6.2 IP header is attached to ICMP message.

In IP/ICMP translation, all IPv4 headers are converted to IPv6 headers as follows, the following table 6.1 explains the translation or conversion of IPv4 header to IPv6 header [71].

Table 6.1 Translation of IPv4 header to IPv6 header

Version	6
Traffic Class (Tclass)	The bits from IP Type Of Service and Precedence field is copied at TClass of IPv6 packet with the same semantics as in IPv4 data packet. The role of translator is to provide the ability to ignore the IPv4 "TOS" and always set the IPv6 traffic class to zero.
Flow Label	All bits are set to 0
Payload Length	It is recalculated. Total length value of IPv4 packet minus the size of the IPv4 header. The Total Length field (16 bits) contains the total length of the packet, in bytes. The Header Length field (4 bits) indicates the length of IPv4 packet header [72].
Next Header	Protocol field copied from IPv4 header
Hop Limit	It is the TTL value copied from IPv4 header. As router plays the role of translator, when it forwards the packet it needs to decrement IPv4 TTL before the translation. To do so the translator (as any router) needs to check for zero and send the ICMPv4 or ICMPv6 "ttl exceeded" error.
Source Address	The low-order 32 bits is copied from IPv4 source address while the high-order 96 bits is the IPv4-mapped prefix (::ffff:0:0/96).
Destination Address	The low-order 32 bits is the IPv4 destination address. The high-order 96 bits is the IPv4-translated prefix (0::ffff:0:0:0/96).

In Address translation, IPv4 addresses are translated to IPv6 addresses using Mapping technique and in IP/ICMP translation, IPv4 headers are converted to IPv6 headers, thus figure 6.3 below shows the IPv4 data packet translated to IPv6 data packet.

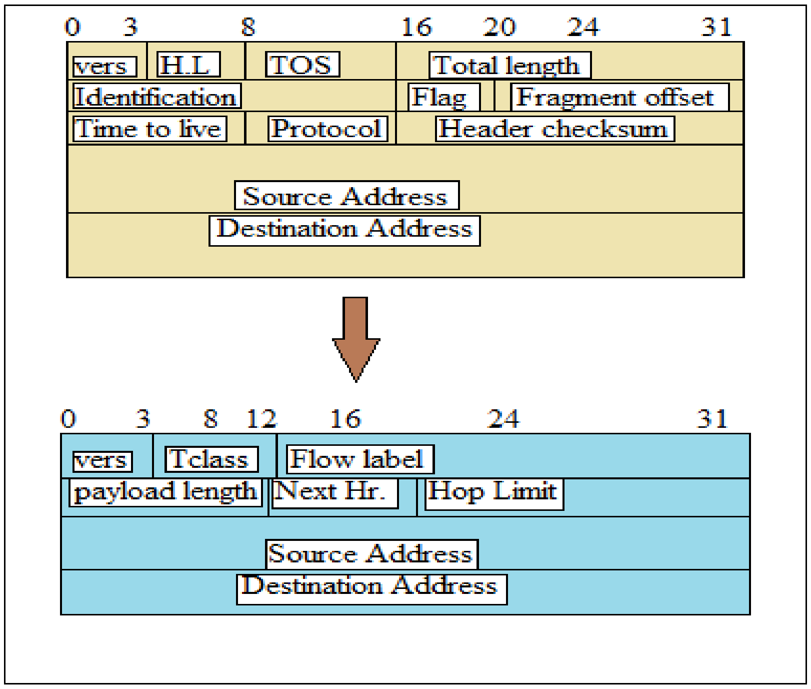


Figure 6.3 Conversion of IPv4 to IPv6

6.1.3 Maintaining the translation state

The mechanisms required to maintain these converted data packets are known as Maintaining the translation state. Every data packet contains header and data section, after converting all IPv4 address to IPv6 address and all IPv4 headers to respective IPv6 headers, there is a need to maintain these data packets. Maintaining the translation state is achieved by

a) DNS64 and

b) NAT64.

The DNS64 and NAT64 work together in an organized manner to let the evolving IPv6 clients to access present IPv4 services. On the other hand DNS46 and NAT46 work together to let current IPv4 clients to access evolving IPv6 services.

6.1.4 DNS 64 & NAT 64

The Domain Name System (DNS) is a hierarchical distributed naming system for computers, services, or any resource connected to the Internet or a private network. It translates easily memorized domain names to the numerical IP addresses needed for the purpose of locating computer services and devices worldwide. For example, the domain name www.example.com translates to the addresses 93.184.216.119 (IPv4) and 2606:2800:220:6d: 26bf:1447:1097:aa7 (IPv6).

DNS46 explains the translation of a DNS A record for IPv4 into AAAA record for IPv6 and also forms mapping records between the IPv6 addresses and IPv4 addresses whereas DNS64 is translation of the IPv6 AAAA record into the IPv4 A record. So, DNS46 is used for retrieving websites on the Internet that are IPv6 only where the core systems are IPv4 only on the other hand DNS64 is used for hosts that are IPv6 only to access IPv4 services.

NAT64 and NAT46 are terms used to refer to the mechanism that allows IPv6 addressed hosts to communicate with IPv4 addressed hosts and vice-versa. Without such a mechanism an IPv6 node on a network such as a corporate LAN would not be able to communicate with a web site that was still in an IPv4 only environment and IPv4 environments would not be able to connect to IPv6 networks. Because the IPv6 range of addresses is so much larger than the IPv4 range, a one to one mapping is not feasible. Therefore the NAT64 function is required to maintain any IPv6 to IPv4 mappings that it synthesizes. The NAT64 function converts IPv6 and IPv4 packets and handles the routing and forwarding of packets. The DNS64 function converts AAAA queries from the IPv6-only host into IPv4-only A record queries and maintains the mapping between the two[73]. The figure 6.4 shows that DNS64 and NAT64 functions are applicable to the "IPv6 to IPv4" scenario.

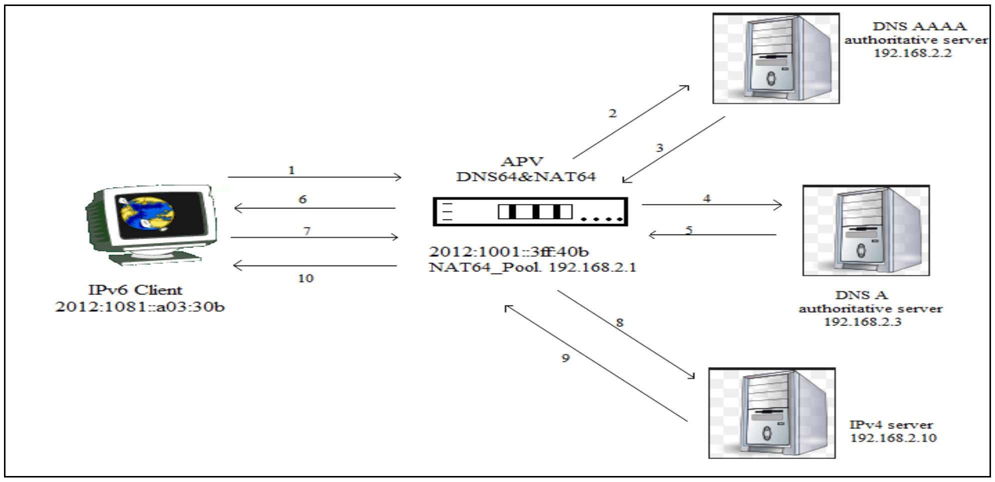


Figure 6.4 DNS64 and NAT64 functions

6.1.5 Application Layer Gateway (ALGs)

The combination of DNS64/46 and NAT 64/46 plays an important role while translating IPv6 datapackets to IPv4. But it has been noted that File transfer protocol (FTP's) does not work well across NATs without extra support. To solve this issue, a simple operational component was recommended: there should be a server in each domain, or an instance of the server should have an interface in each domain. Thus ALGs are considered as a good solution. ALG (Application Layer Gateway) - it is a security component, which is usually found in either a firewall device or a router. It basically enhances the capability for some specific protocols to traverse NAT.

6.2 PROTOCOL VERIFICATION

Protocol Verification of Translation in MIPv4 / MIPv6 was carried out by following steps,

- Modeling of Translation in MIPv4 / MIPv6 using schema diagram.
- Formulating design steps for each module from schema diagram
- Implementing each design step using Programming language C
- Noting down experimental results using screen shots.

6.2.1 Modeling of Translation in MIPv4 / MIPv6 using schema diagram.

Translation refers to the direct conversion of protocols. The network environment in Translation consists of translators that are responsible to convert AAAA records of IPv6 to A records of IPv4 and vice versa. In this thesis, the network scenario is assumed to be in IPv6 -> IPv4 direction which consists of IPv6 to IPv4 translators. The schema diagram of Translation model is as shown in figure 6.5

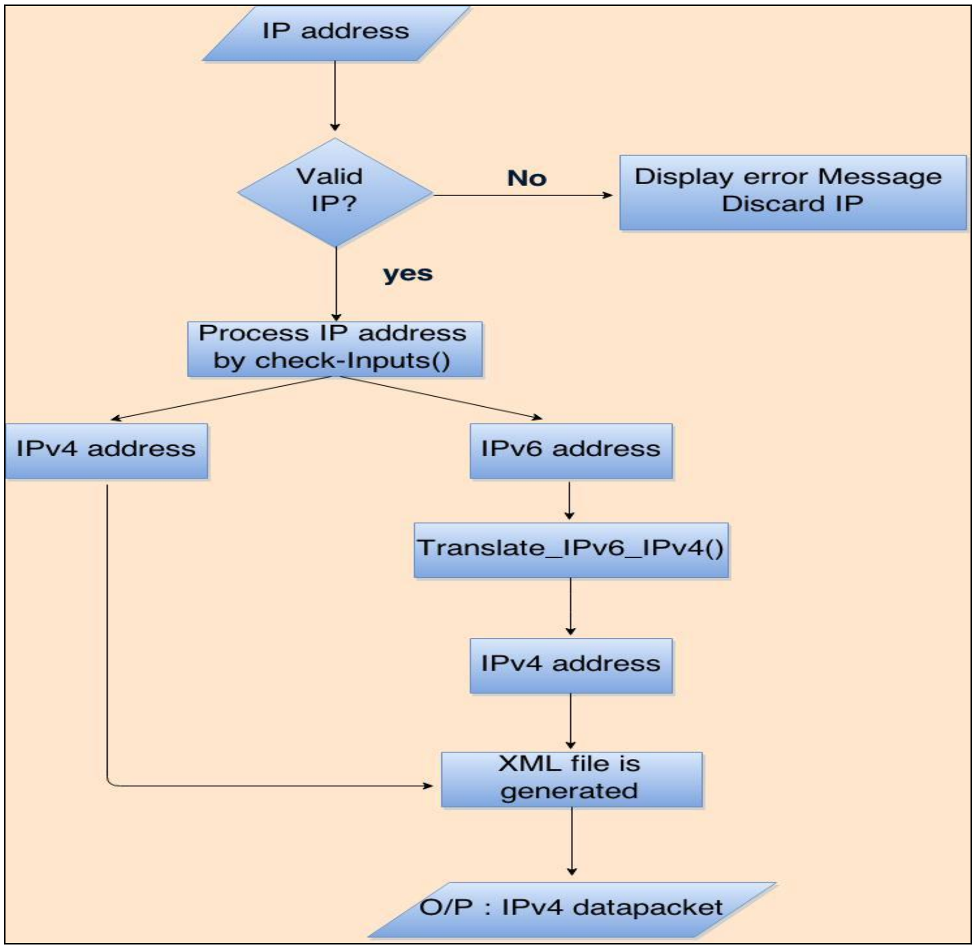


Figure 6.5 Schema diagram of Translation model

6.2.2 Formulating design steps for each module from schema diagram

This section presents the design steps to demonstrate translation of IPv6 address to IPv4 address. This design will be implemented using the function TRANSLATE_IPv6_IPv4( ). After the translation process is complete, xml file gets generated to show the IP datapacket with translated IP address [14]. Every datapacket can be classified into four categories as shown in table 6.2

Table 6.2 Datapacket with source address and destination address

Datapacket	Source address	Destination address
Case i.	IPv4	IPv4
Case ii.	IPv6	IPv6
Case iii.	IPv4	IPv6
Case iv.	IPv6	IPv4

During address translation, this design makes use of an internal table named IPADDRESSMAP.txt as mentioned in section 5.4.2.5. The translator in this design translates by comparing the IPv6 address provided by the user with the IPv6 address present in the internal table. If match is found, corresponding IPv4 address is used for translation else default address gets selected. The function TRANSLATE_IPv6_IPv4( ) is used to translate IPv6 address to IPv4. After translation, XML file is generated which contains the translated IP datapacket.

6.2.3 Implementing each design step using Programming language C

Function 1: TRANSLATE_IPv6_IPv4( )

Case i:

INPUT : Datapacket with source address 173.194.113.145 and destination address 50.97.149.10

OUTPUT: XML file with IPv4 datapacket as shown below

```
<IPv4>
<SOURCE_IP>173.194.113.145  </SOURCE_IP>
<DESTINATION_IP> 50.97.149.10 </DESTINATION_IP>
<BODY >
MESSAGE
</BODY>
</IPv4>
```

Case ii:

INPUT : Datapacket with source address E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420 and destination address EEEE:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:EEEE

OUTPUT: Default IPv4 address gets selected and respective IPv4 datapacket gets generated as shown in XML file

```
<IPv4>
<SOURCE_IP>101.101.101.101  </SOURCE_IP>
<DESTINATION_IP> 101.101.101.101 </DESTINATION_IP>
<BODY >
MESSAGE
</BODY>
</IPv4>
```

Case iii:

INPUT: Datapacket with source address 173.194.113.145 and destination address E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420

OUTPUT: Default IPv4 address gets selected and respective IPv4 datapacket gets generated as shown in XML file

```
<IPv4>
<SOURCE_IP>173.194.113.145 </SOURCE_IP>
<DESTINATION_IP> 101.101.101.101 </DESTINATION_IP>
<BODY >
MESSAGE
</BODY>
</IPv4>
```

Case iv:

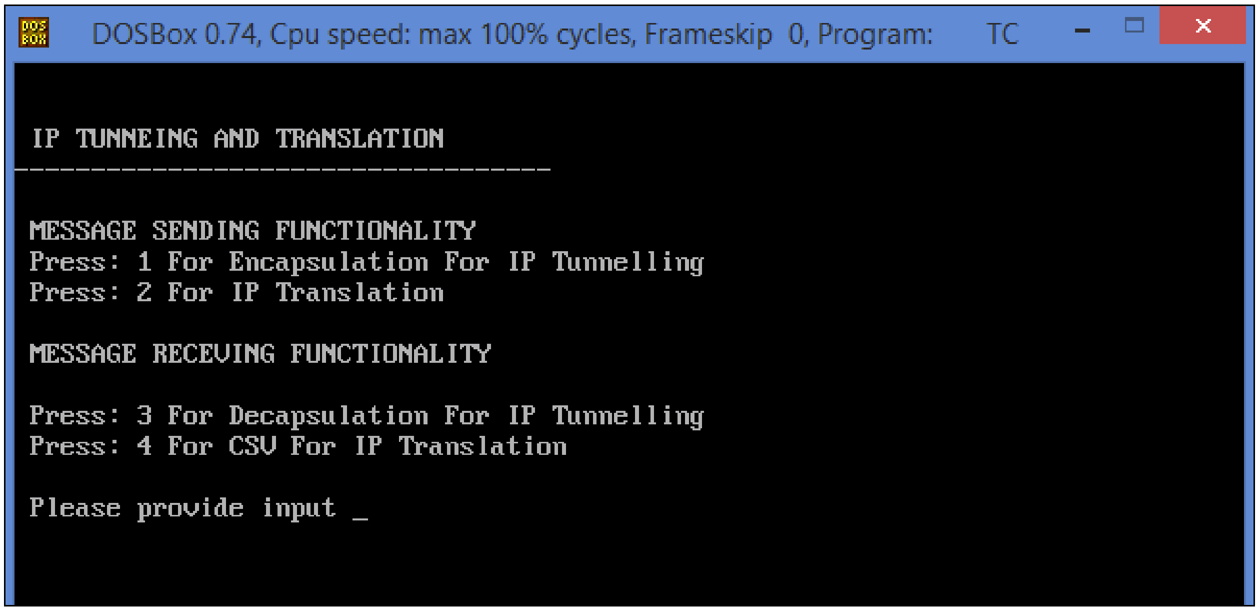
INPUT: Datapacket with source address E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420 and destination address 173.194.113.145

OUTPUT: Default IPv4 address gets selected and respective IPv4 datapacket gets generated as shown in XML file

```
<IPv4>
<SOURCE_IP>101.101.101.101</SOURCE_IP>
<DESTINATION_IP> 173.194.113.145  </DESTINATION_IP>
<BODY >
MESSAGE
</BODY>
</IPv4>
```

6.2.4 Experimental Results

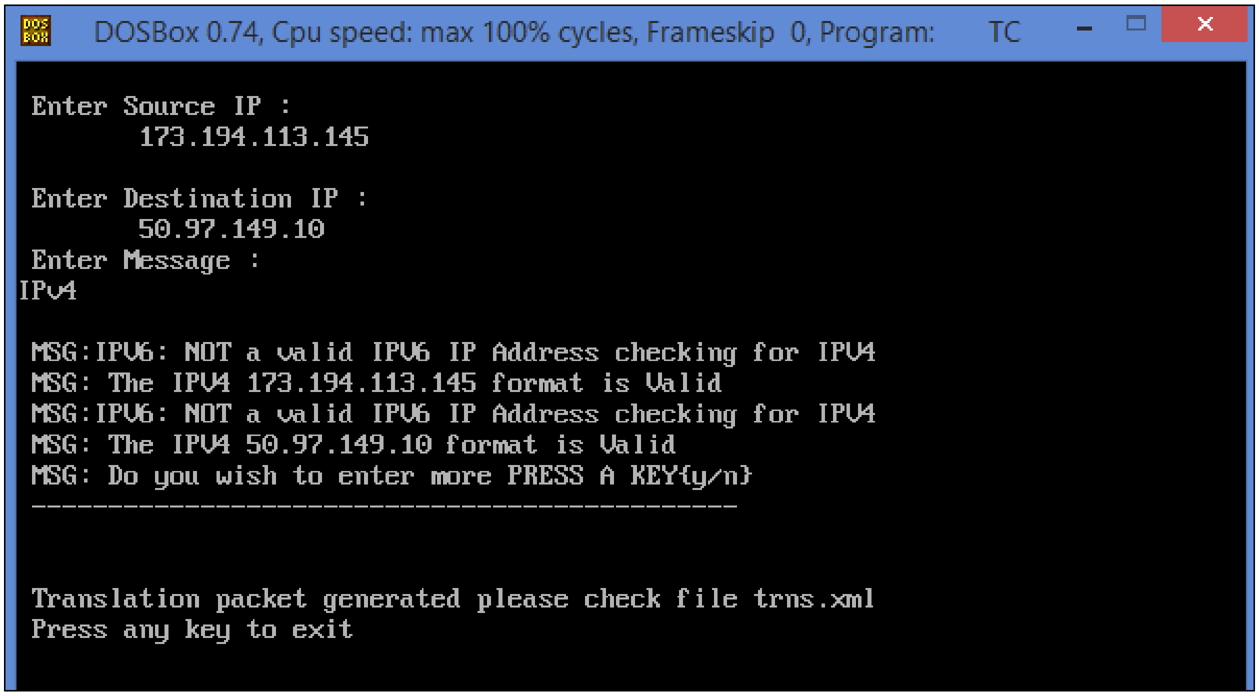
Select option 2 for Translation


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
 IP TUNNEING AND TRANSLATION
-----------------------------------

 MESSAGE SENDING FUNCTIONALITY
 Press: 1 For Encapsulation For IP Tunnelling
 Press: 2 For IP Translation

 MESSAGE RECEVING FUNCTIONALITY

 Press: 3 For Decapsulation For IP Tunnelling
 Press: 4 For CSV For IP Translation

 Please provide input _
```

Figure 6.6 Option 2 is selected by user to perform Translation.

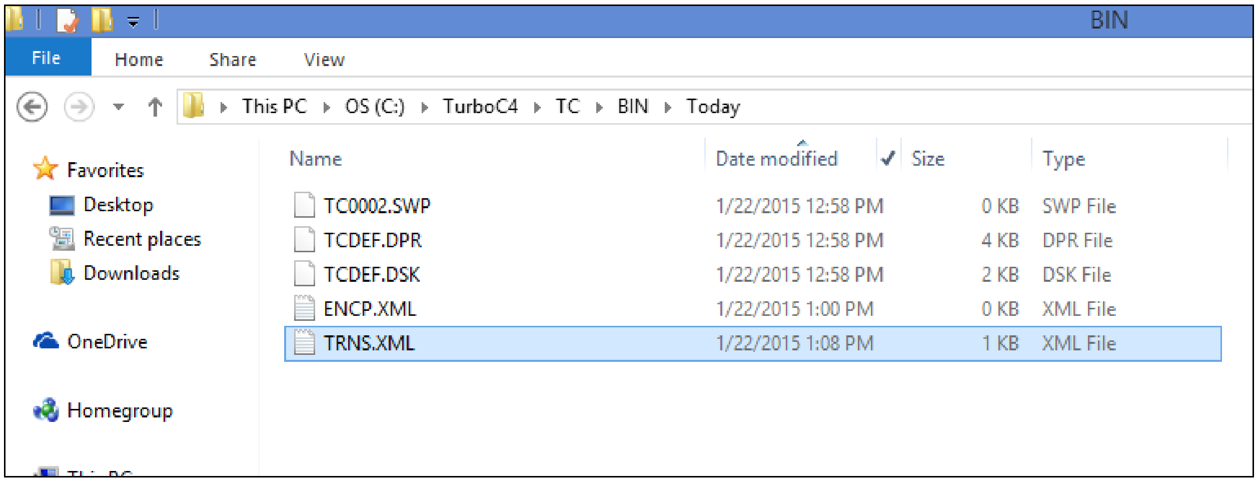
Case i: Source and Destination both are IPv4


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
 Enter Source IP :
        173.194.113.145

 Enter Destination IP :
        50.97.149.10
 Enter Message :
IPv4

 MSG:IPV6: NOT a valid IPV6 IP Address checking for IPV4
 MSG: The IPV4 173.194.113.145 format is Valid
 MSG:IPV6: NOT a valid IPV6 IP Address checking for IPV4
 MSG: The IPV4 50.97.149.10 format is Valid
 MSG: Do you wish to enter more PRESS A KEY{y/n}
 ----------------------------------------------


 Translation packet generated please check file trns.xml
 Press any key to exit
```

Figure 6.7 IPv4 datapacket is sent for Translation

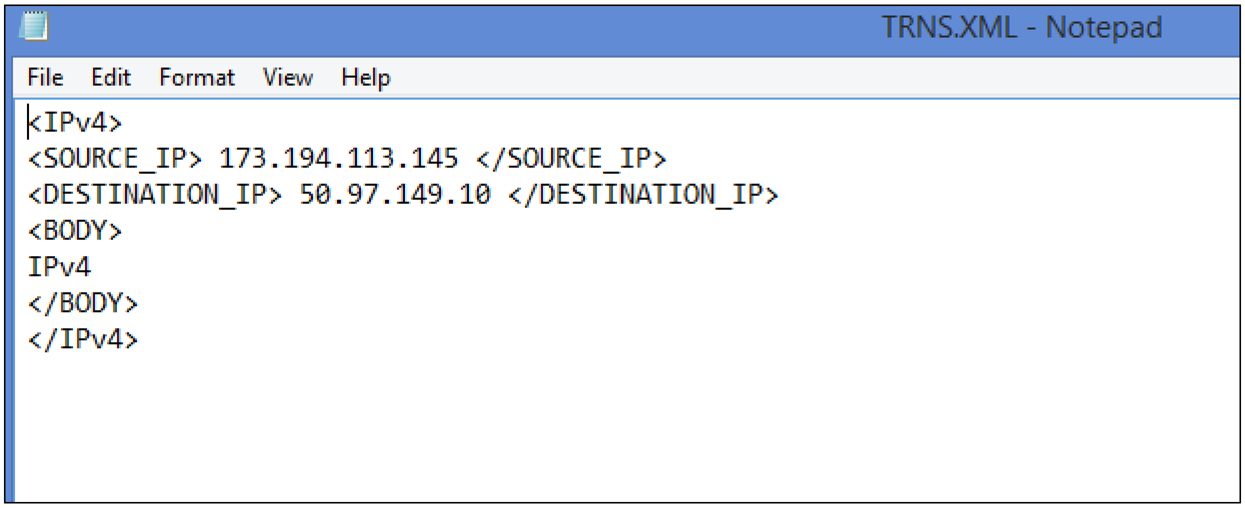
XML file is generated after Translation



Name	Date modified	Size	Type
TC0002.SWP	1/22/2015 12:58 PM	0 KB	SWP File
TCDEF.DPR	1/22/2015 12:58 PM	4 KB	DPR File
TCDEF.DSK	1/22/2015 12:58 PM	2 KB	DSK File
ENCP.XML	1/22/2015 1:00 PM	0 KB	XML File
TRNS.XML	1/22/2015 1:08 PM	1 KB	XML File

Figure 6.8 XML file is generated.

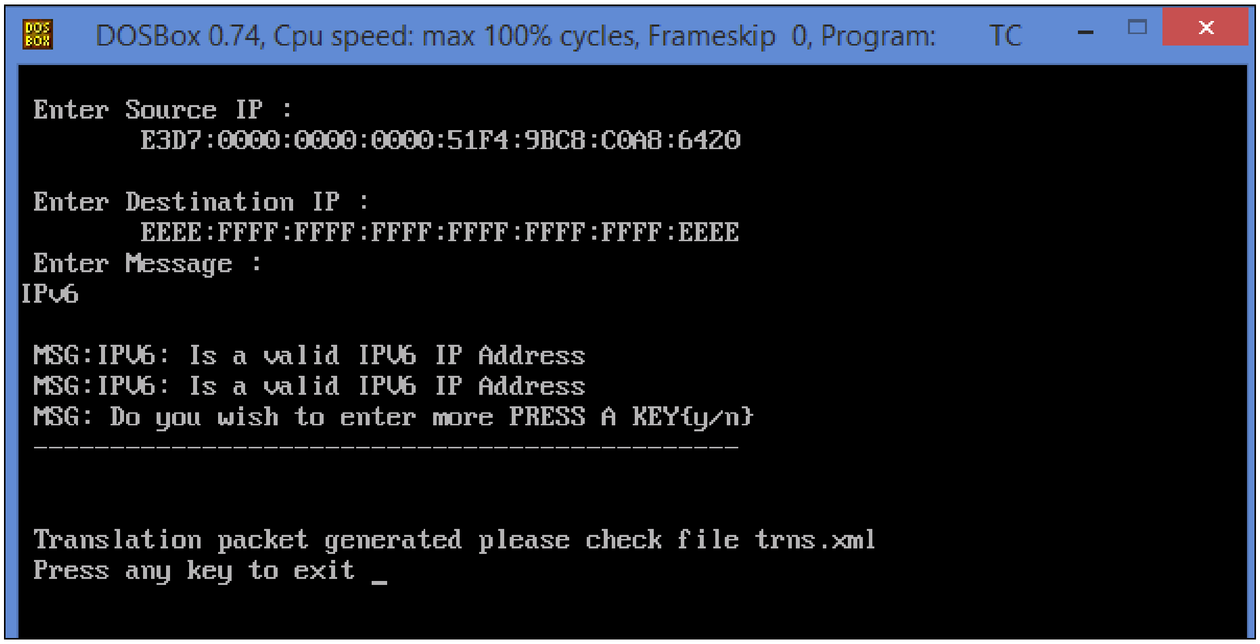
IPv4 datapacket shown in XML file



```
<IPv4>
<SOURCE_IP> 173.194.113.145 </SOURCE_IP>
<DESTINATION_IP> 50.97.149.10 </DESTINATION_IP>
<BODY>
IPv4
</BODY>
</IPv4>
```

Figure 6.9 Contents of XML file

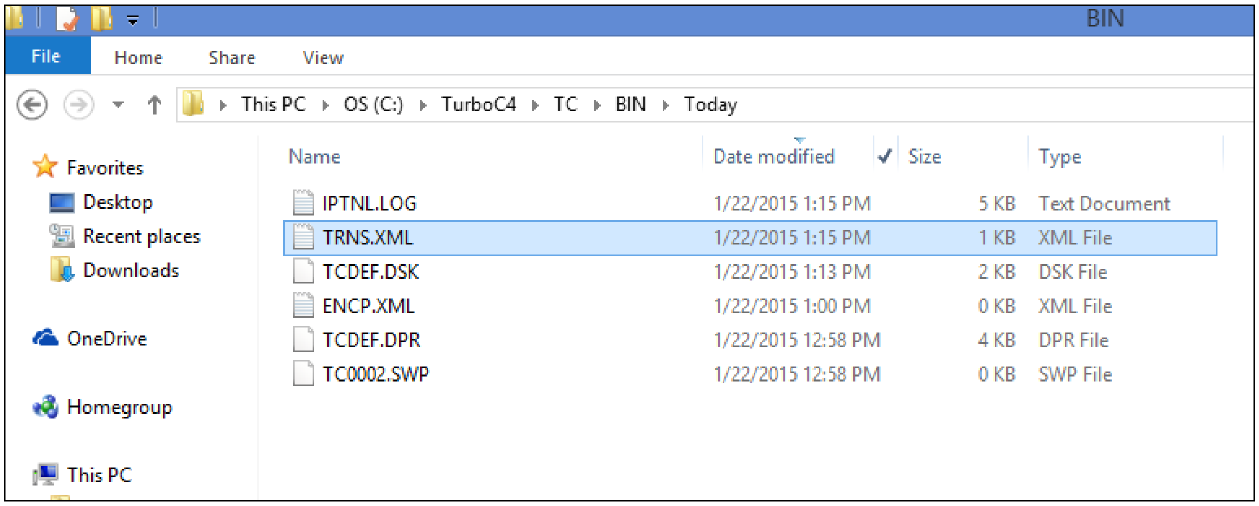
Case ii: Source and Destination is IPv6


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
 Enter Source IP :
        E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420

 Enter Destination IP :
        EEEE:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:EEEE
 Enter Message :
IPv6

 MSG:IPV6: Is a valid IPV6 IP Address
 MSG:IPV6: Is a valid IPV6 IP Address
 MSG: Do you wish to enter more PRESS A KEY{y/n}
 ----------------------------------------------


 Translation packet generated please check file trns.xml
 Press any key to exit _
```

Figure 6.10 IPv6 datapacket sent in IPv4 network.

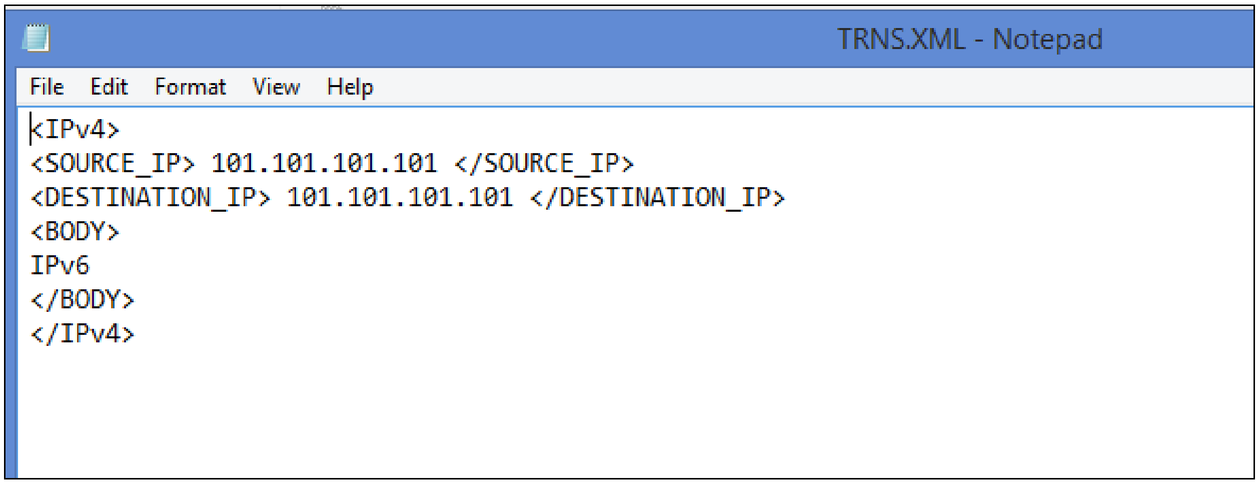
Translation is successful.



Name	Date modified	Size	Type
IPTNL.LOG	1/22/2015 1:15 PM	5 KB	Text Document
TRNS.XML	1/22/2015 1:15 PM	1 KB	XML File
TCDEF.DSK	1/22/2015 1:13 PM	2 KB	DSK File
ENCP.XML	1/22/2015 1:00 PM	0 KB	XML File
TCDEF.DPR	1/22/2015 12:58 PM	4 KB	DPR File
TC0002.SWP	1/22/2015 12:58 PM	0 KB	SWP File

Figure 6.11 XML file is generated.

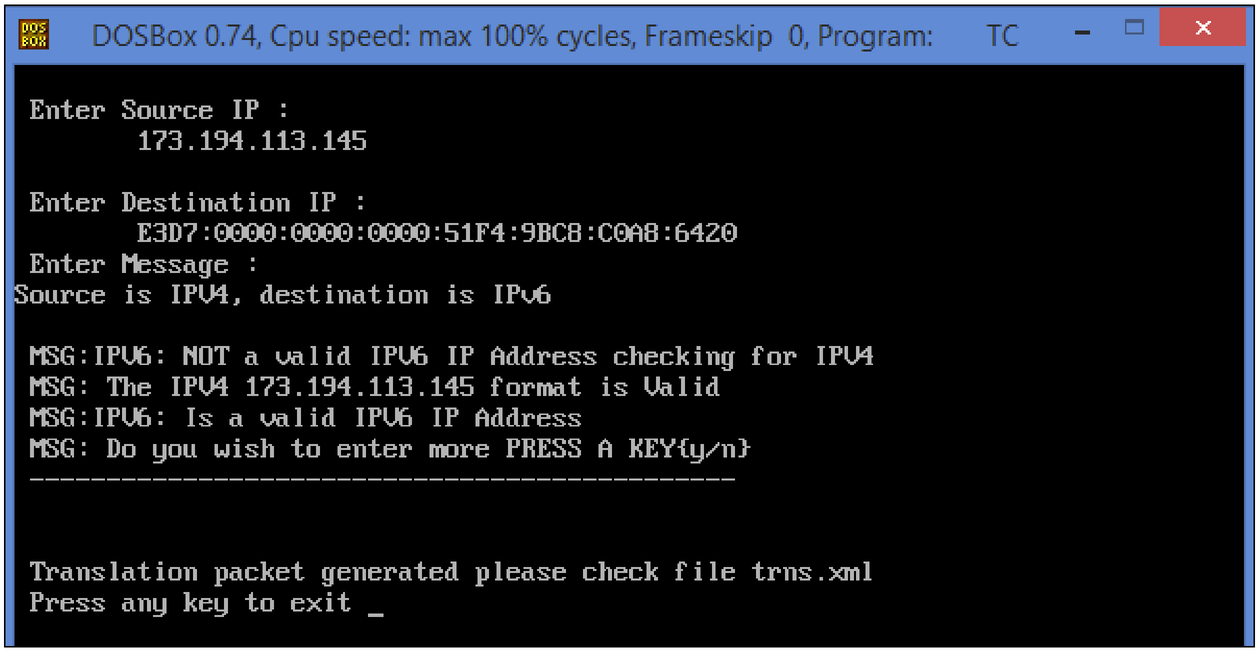
Default IPv4 address gets selected and IPv4 datapacket is generated as shown below



```
<IPv4>
<SOURCE_IP> 101.101.101.101 </SOURCE_IP>
<DESTINATION_IP> 101.101.101.101 </DESTINATION_IP>
<BODY>
IPv6
</BODY>
</IPv4>
```

Figure 6.12 IPv6 datapacket is translated to IPv4 datapacket.

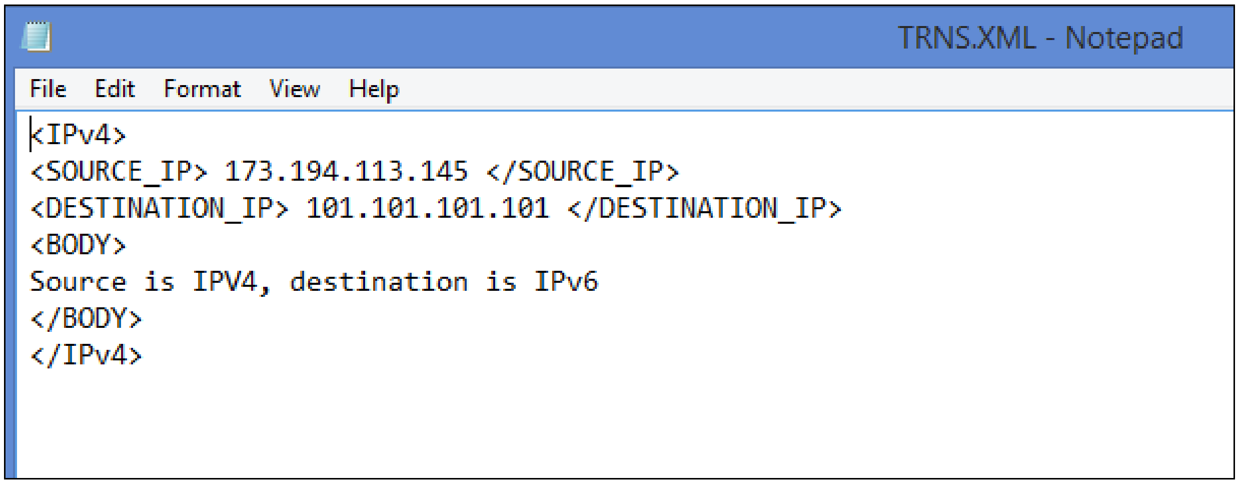
Case iii: Datapacket with IPv4 Source Address and IPv6 Destination Address.


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
 Enter Source IP :
        173.194.113.145

 Enter Destination IP :
        E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420
 Enter Message :
Source is IPV4, destination is IPv6

 MSG:IPV6: NOT a valid IPV6 IP Address checking for IPV4
 MSG: The IPV4 173.194.113.145 format is Valid
 MSG:IPV6: Is a valid IPV6 IP Address
 MSG: Do you wish to enter more PRESS A KEY{y/n}
 ----------------------------------------------


 Translation packet generated please check file trns.xml
 Press any key to exit _
```

Figure 6.13 Translation is complete.

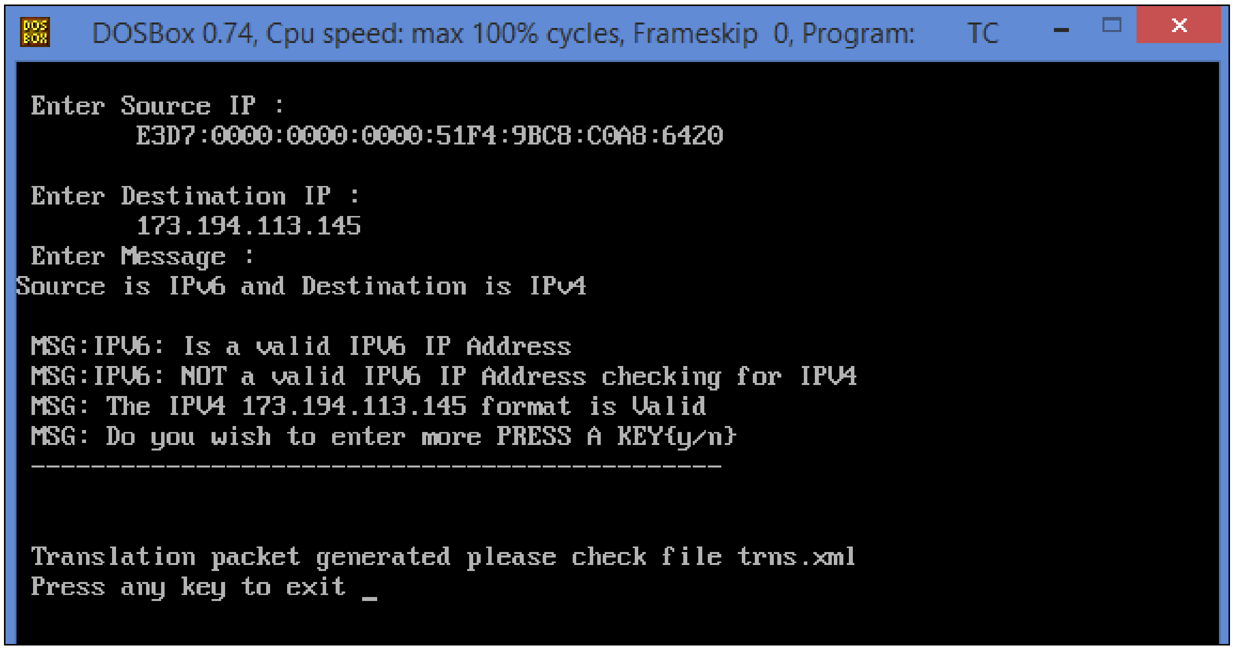
After translation, XML file is generated, and the source address remains same while destination address is translated to IPv4. The translated datapacket is shown below



```
<IPv4>
<SOURCE_IP> 173.194.113.145 </SOURCE_IP>
<DESTINATION_IP> 101.101.101.101 </DESTINATION_IP>
<BODY>
Source is IPV4, destination is IPv6
</BODY>
</IPv4>
```

Figure 6.14 Default IPv4 address is selected as destination address to generate IPv4 datapacket.

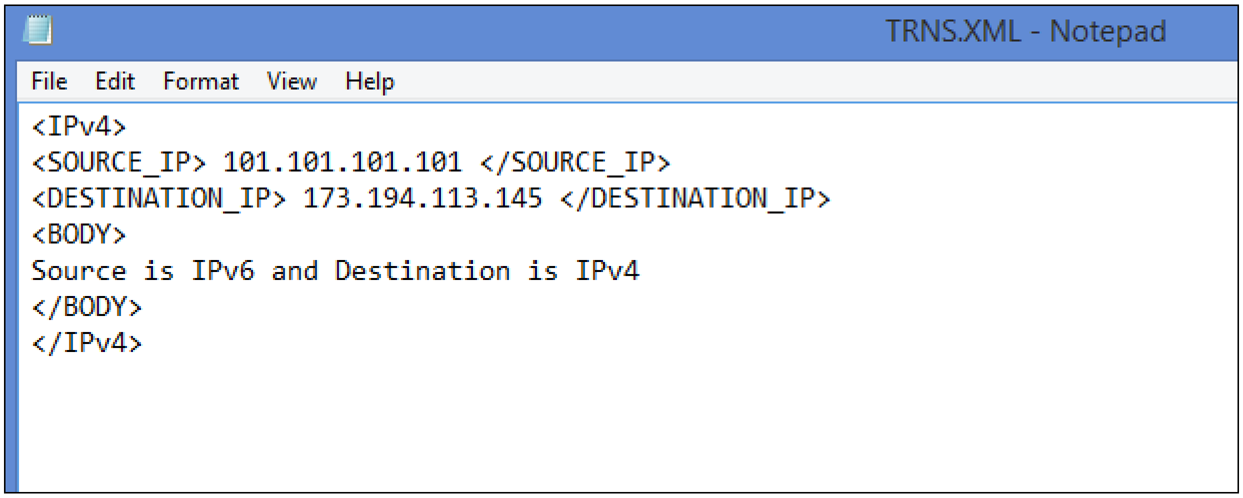
<u>Case iv: Similarly when source address provided is IPv6 and Destination address is IPv4.</u>


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
 Enter Source IP :
        E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420

 Enter Destination IP :
        173.194.113.145
 Enter Message :
Source is IPv6 and Destination is IPv4

 MSG:IPV6: Is a valid IPV6 IP Address
 MSG:IPV6: NOT a valid IPV6 IP Address checking for IPV4
 MSG: The IPV4 173.194.113.145 format is Valid
 MSG: Do you wish to enter more PRESS A KEY{y/n}
 ---------------------------------------------


 Translation packet generated please check file trns.xml
 Press any key to exit _
```

Figure 6.15 Translation is complete.

After translation, XML file is generated in which the source address is translated to IPv4 while destination address i,e IPv4 remains the same.



```
<IPv4>
<SOURCE_IP> 101.101.101.101 </SOURCE_IP>
<DESTINATION_IP> 173.194.113.145 </DESTINATION_IP>
<BODY>
Source is IPv6 and Destination is IPv4
</BODY>
</IPv4>
```

Figure 6.16 Default IPv4 address is selected as source address to generate IPv4 datapacket.

All four cases were tested and in each case, the translator function did perform its task as per the design.

6.2.5 Result Analysis

The function TRANSLATE_IPv6_IPv4( ) is used to translate IPv6 address to IPv4. Since TRANSLATE_IPv6_IPv4( ) loops through the function for every IP address and for n number of inputs then the complexity of the function TRANSLATE_IPv6_IPv4( ) is of the order of (n) given by O(n).

6.3 DRAWBACKS OF TRANSLATION

1) It directly translates only the IPv4 options that have direct IPv6 corresponding items.
2) It does not translate any IPv6 extension headers beyond the fragmentation extension header.
3) Tie up with ALG functionality causes hindrances to deployment of NAT-PT.

Translation is, however, generally documented as a necessity in certain cases to provide connectivity between IPv6-only and IPv4-only systems or networks. IETF strongly suggests that people use Network Address Translators to meet various needs, many of which are met as well by routing or protocol mechanisms that preserve the end-to-end addressability of the Internet [74]. But it did not consider the case in which there is an ongoing requirement to communicate with IPv4 systems. For example, configuring IPv4 routing is not the most desirable strategy in the network operator's view or is infeasible due to a shortage of global address space. Thus it can be concluded that IPv4/IPv6 translation under this preface is not a long-term support strategy, but it is a medium-term coexistence strategy that can be used to facilitate a long-term program of transition. Thus Translation between IPv4 and IPv6 is not considered as a practical long-term strategy.

6.4 SUMMARY

This chapter dealt with Protocol verification of Translation in Mobile IPv4 / Mobile IPv6. Various types of Translation was described in detail and a model was developed to represent Translation. Design steps were formulated to translate IPv6 address to IPv4 address. The datapackets were viewed using a xml file. Four different cases for each data packet.were identified as follows

Case 1:	Source address :IPv4	Destination address: IPv4
Case 2:	Source address :IPv6	Destination address: IPv6
Case 3:	Source address :IPv4	Destination address: IPv6
Case 4:	Source address :IPv6	Destination address: IPv4

In each case, the data packets with IPv6 address was successfully translated to IPv4 address.

CHAPTER 7

PROTOCOL VERIFICATION OF MOBILE IPv6

IPv6 addresses are 128 bits long, or four times the size of IPv4 addresses. The theoretical number of IPv6 addresses, about 3×10^{38}, is almost unimaginably large [75,77].

7.1 INTRODUCTION

There are basically three forms for representing IPv6 addresses:

1. The preferred form is x:x:x:x:x:x:x:x Here the 128-bit address is divided into 16-bit of 8 pieces. These are converted to their equivalent hexadecimal values which are written as groups of four hexadecimal digits separated by colons.

2. IPv6 addresses may contain long strings of zero bits due to the address allocation methods used. In order to facilitate writing of addresses containing long zero strings a special syntax is used to compress the zeros. The use of **"::"** indicates one or more groups of 16-bits of zeros. The **"::"** may be used to compress leading or trailing zeros, but it can appear only once in an address.

3. When dealing with a mixed environment of IPv4 and IPv6 addresses it may be more convenient to represent the address as x:x:x:x:x:x:d.d.d.d Here the 'x's are the hexadecimal values of the six high-order 16-bit pieces of the address and d.d.d.d is the dotted decimal (i.e. standard IPv4) representation of the four low-order bytes of the address.

Examples of IPv6 addresses:

IPv6 addresses:

FFEC: BA98: 7654: 3210: FEDC: BA98: 7654: 3210

1080: 0: 0: 0: 8: 8000: 200C: 417A

Address with zero compression:

1080**: :** 8**:** 8000**:** 200C**:** 417A

7.1.1 Address Notation

In full form, IPv6 addresses are written as 16-bit hexadecimal of 8 blocks separated by colons:

FE80:0000:0000:0000:1232:E4BF:FE1A:8324

IPv6 addresses can be interpreted as having two variable-length fields: an IPv6 prefix and an IPv6 interface identifier.

- The IPv6 prefix varies from 0 to 128 bits in length and forms the routable network number portion of the IPv6 address. The trailing Classless Inter-Domain Routing (CIDR) notation that may appear after human-readable IPv6 addresses (for example, /64) indicates the bit length of the IPv6 prefix.

- The IPv6 interface identifier consists of the non-prefix portion of the IPv6 address, if any, and identifies the host interface portion of the address, which identifies an IPv6 interface on the local network. The IPv6 interface identifier is typically generated automatically by the host as a function of a unique hardware identifier, such as an Ethernet MAC address. IPv6 hosts can automatically configure interface addresses by combining the IPv6 prefixes obtained from router advertisements with the IPv6 interface IDs that are determined locally.

For example:

IPv6 Prefix:	FEC0:0000:0000:0000:0000:0000:0000:0000/64
IPv6 Interface ID:	0260:08FF:FEFF:FFFF
IPv6 Address:	FEC0:0000:0000:0000:0260:08FF:FEFF:FFFF

To simplify address notation, IPv6 accepts abbreviations in address notation. For example, leading zeros in a 16-bit block may be omitted:

FEC0:0:0:0:0260:08FF:FFFF:FFFF

A double colon (::) can replace a series of consecutive zeros within an address:

FEC0::0260:08FF:FFFF:FFFF

Only one double colon can be used to compress an IPv6 address. If more than one double colon was included in an address, networking devices would not know how many zeros to insert for each double colon when expanding a compressed address to its full 128-bit representation.

In networks that support IPv4 and IPv6 nodes, IPv4 addresses can be embedded in the last four bytes of the address. An IPv4 address of 192.168.1.12 can be represented in IPv6 format as :: 192.168.1.12, where :: represents a string of zeroes to pad the address to 128 bits.

7.1.2 Addressing Modes

There are three types of addressing modes:

1. Unicast: This mode uses a one-to-one association between destination address and network endpoint: each destination address uniquely identifies a single receiver endpoint. It is an identifier for a single interface.

2. Anycast: This mode of addressing routes datagrams to a single member of a group of potential receivers that are all identified by the same destination address. This is a one-to-nearest association. It is an identifier for a set of interfaces.
3. Multicast: This mode of addressing uses a one-to-unique many association, datagrams are routed from a single sender to multiple selected endpoints simultaneously in a single transmission.

There is no foreign agent functionality in MIPv6 as mobile node can acquire its new IPv6 address within a foreign network by address auto-configuration method. In MIPv6, the mobility of mobile node is publicized by router advertisement message. That is, a mobile node can cause a router to send its advertisement message by solicitation if necessary. After publicizing its mobility, the mobile node gets its Care-Of Address and sends binding update message to home agent and correspondent node, the home agent and correspondent node will update its binding list and send their respective acknowledgement messages to correspondent node. Now if the correspondent node wants to communicate with mobile node it sends data to mobile node’s original IP address as it is unaware of mobile node's care-of address. As a result, data from correspondent node reaches the home network through standard IP routing. The home agent controls its binding list with the data's destination address and then tunnels the data to the mobile node's Care-Of Address. The data is now received and mobile node sends the binding update message to correspondent node. After receiving the update message, the correspondent node sends only data to mobile node's Care-Of Address [76, 11].

7.2 PROTOCOL VERIFICATION

Protocol Verification of Mobile IPv6 was carried out by following steps,

- Modeling of MIPv6 using schema diagram.
- Formulating design steps for each module from schema diagram
- Implementing each design step using Programming language C
- Noting down experimental results using screen shots.

7.2.1 Modeling of MIPv6 using schema diagram.

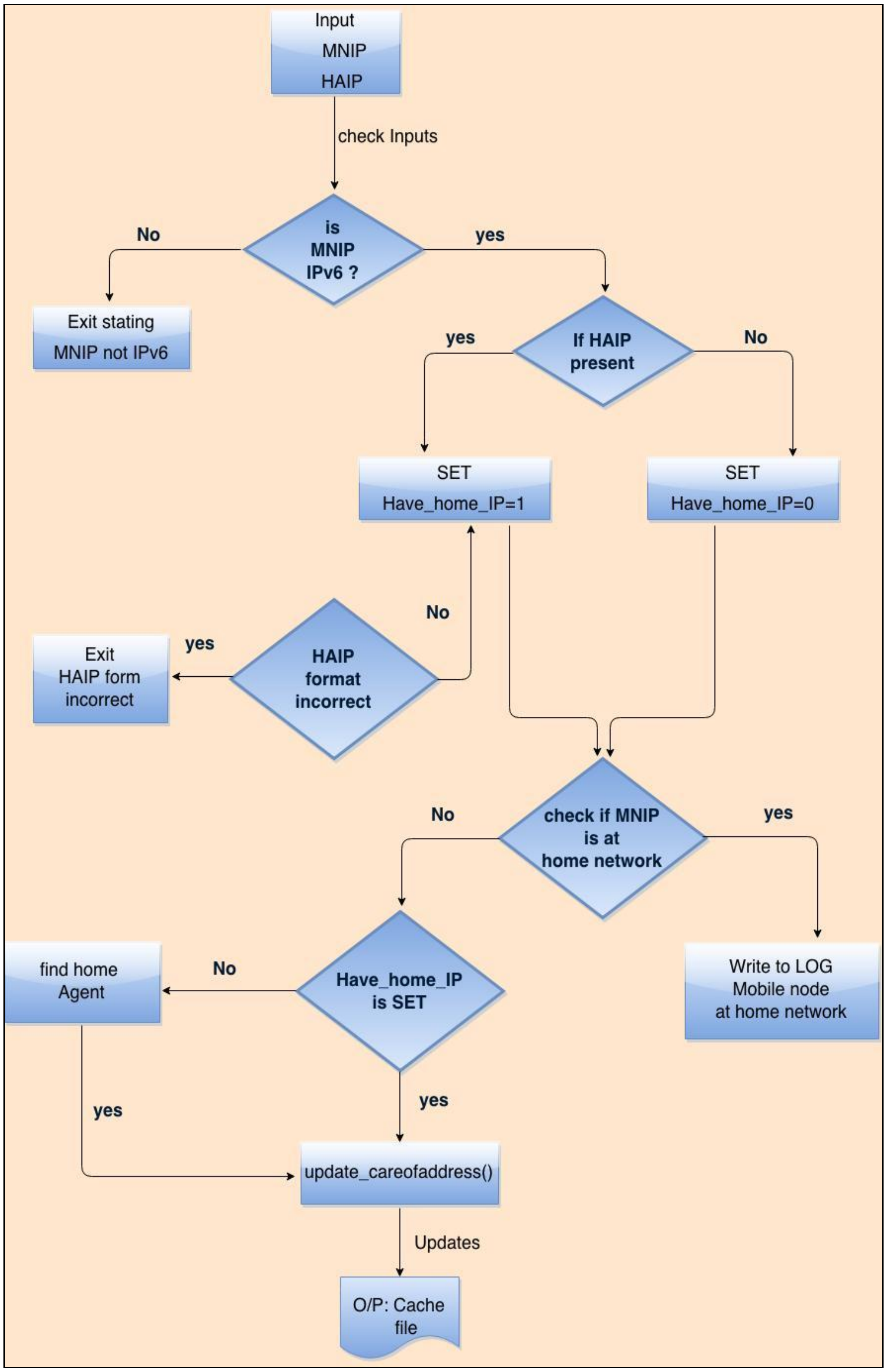


Figure 7.1 Modeling of MIPv6 using schema diagram

7.2.2 Formulating design steps for each module from schema diagram

IP address provided should all be in IPv6 format. All IPv6 inputs will be accepted after validation and placed in a separate stack called IPv6 stack. Any input other than IPv6 will be discarded.

INPUTS:

1. Mobile Node IP (Mandatory) - Referred to as MNIP:

E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420

2.Home Agent IP (Optional) - Referred to as HAIP:

E3D7:0000:0000:0000:51F4:9BC8:C0A8:6422

Function 1: Design for Function Check_IPv6_Inputs ( )

This function will read the inputs MNIP which is a mandatory input and HAIP (if provided). The function will check if the format of the MNIP Address matches IPV6 format. If MNIP does not match IPv6 format then exit the code stating IP Address is not a valid IPv6 address. And if MNIP matches with IPv6 format then check the availability HAIP and verify its format. Next task is to set Have_home_IP flag.

Function 2: Check_If_MNIP_Home ( )

This function will take the mobile node IP (MNIP) address and match it with the range of IP address which will be considered belonging to the present home agent. If the MNIP belongs to the home then no action is taken and a message will be updated in LOG file as Mobile node is at Home network. However If the MNIP is not present in the given range it will be treated as a foreign node.

Now the flow of the program depends on the value of variable Have_home_IP.

1. If Have_home_IP is SET call Function Update_CareOfAddress ( )
2. Else call Function Find_Home_Agent ( )

Function 3: Update_CareOfAddress ( ):This function will update the care-of address. One cache file is maintained for all foreign nodes and this request will be logged in the LOG file.

Function 4: Function Find_Home_Agent ( )

This function will also respond with a DUMMY value of the home agent IP and then this function will call Update_CareOfAddress ( ) to complete this functionality.

OUTPUT: Following messages are displayed

Success Messages:

1) Mobile node at home network

2) Mobile Node at foreign network

3) Care-of address updated

In case of IP address mismatch, Error messages will be displayed

7.2.3 Implementing each design step using Programming language C

FUNCTION 1:

check_inputs( ) is responsible to check the format for MNIP and HAIP(if provided). The parameters are char array for MNIP and HAIP

FUNCTION 2:

check_if_mnip_home( ): The parameters are char array for MNIP

FUNCTION 3 :

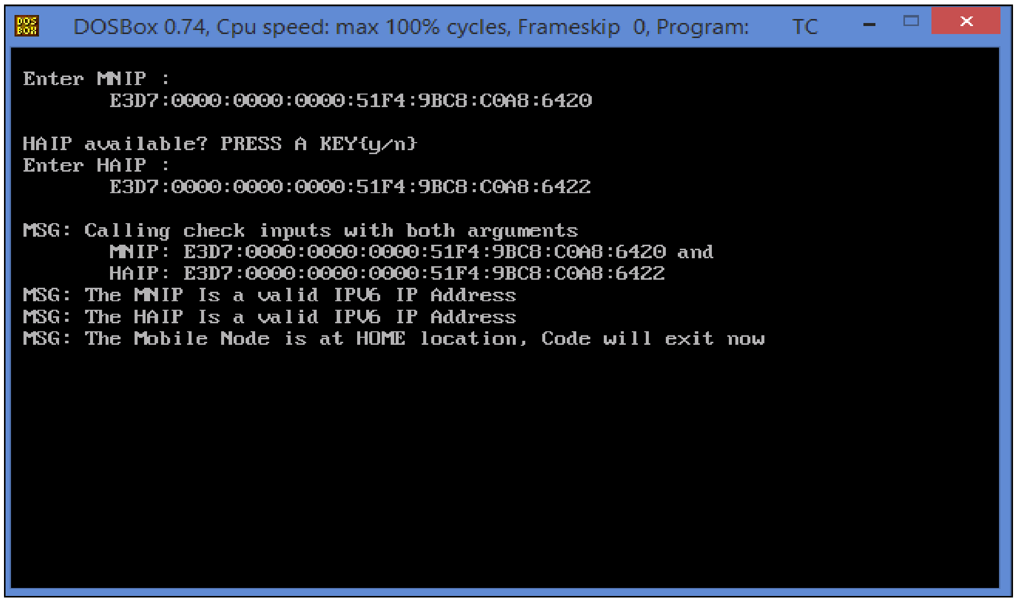
update_CareOfAddress( ):The parameters are char array for MNIP, COIP, HAIP

FUNCTION 4 :

find_home_agent( ):The parameters are char array for MNIP, COIP

7.2.4 Experimental Results

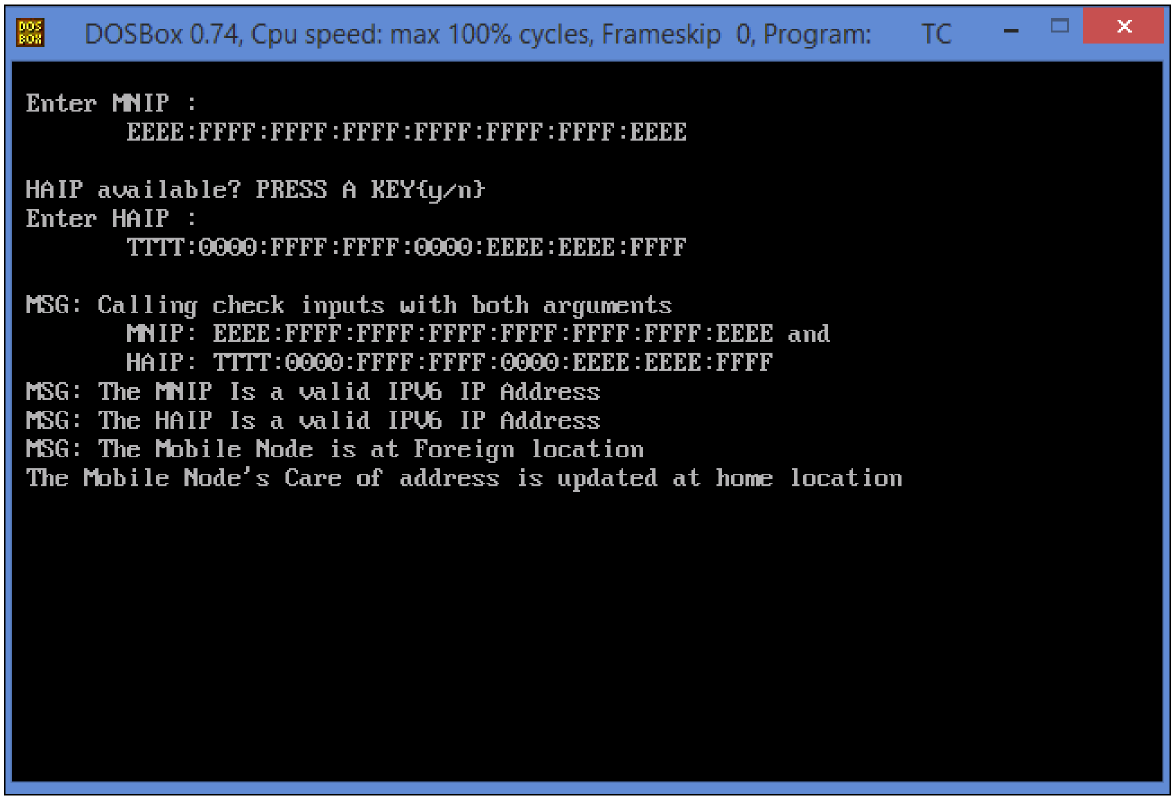
Enter Mobile Node IP and Home agent IP, on comparing the home agent IP address with the Mobile node IP address, the design verifies that the mobile node is in the home network.


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
Enter MNIP :
       E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420

HAIP available? PRESS A KEY{y/n}
Enter HAIP :
       E3D7:0000:0000:0000:51F4:9BC8:C0A8:6422

MSG: Calling check inputs with both arguments
       MNIP: E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420 and
       HAIP: E3D7:0000:0000:0000:51F4:9BC8:C0A8:6422
MSG: The MNIP Is a valid IPV6 IP Address
MSG: The HAIP Is a valid IPV6 IP Address
MSG: The Mobile Node is at HOME location, Code will exit now
```

Figure 7.2 Mobile node is in the home network

In this example, different inputs are provided for Mobile node IP and Home agent IP, The design verifies that the mobile node is at foreign location.

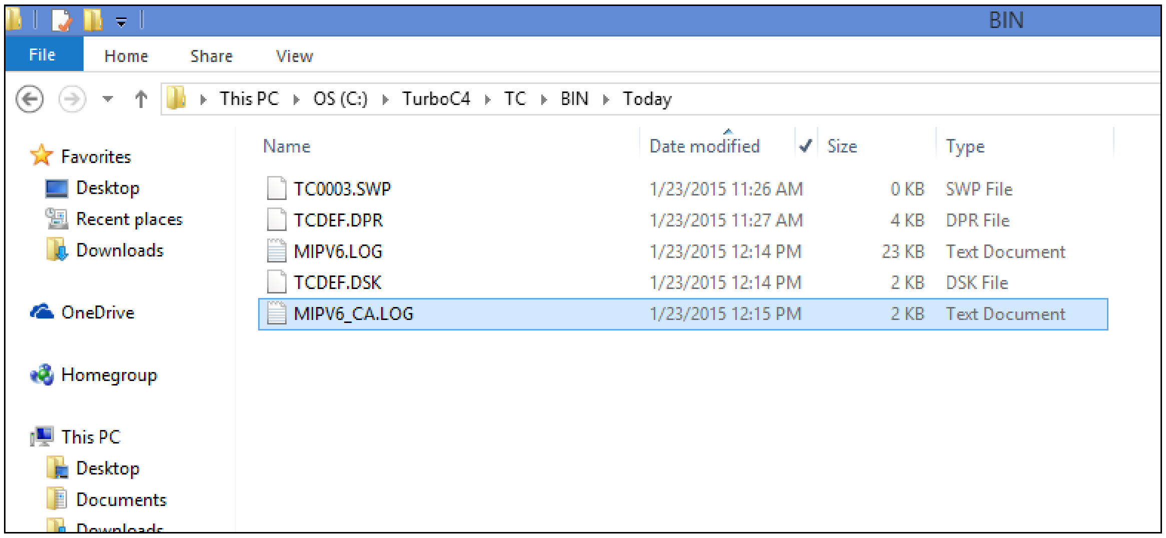


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
Enter MNIP :
       EEEE:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:EEEE

HAIP available? PRESS A KEY{y/n}
Enter HAIP :
       TTTT:0000:FFFF:FFFF:0000:EEEE:EEEE:FFFF

MSG: Calling check inputs with both arguments
       MNIP: EEEE:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:EEEE and
       HAIP: TTTT:0000:FFFF:FFFF:0000:EEEE:EEEE:FFFF
MSG: The MNIP Is a valid IPV6 IP Address
MSG: The HAIP Is a valid IPV6 IP Address
MSG: The Mobile Node is at Foreign location
The Mobile Node's Care of address is updated at home location
```

Figure 7.3 Mobile node is foreign location

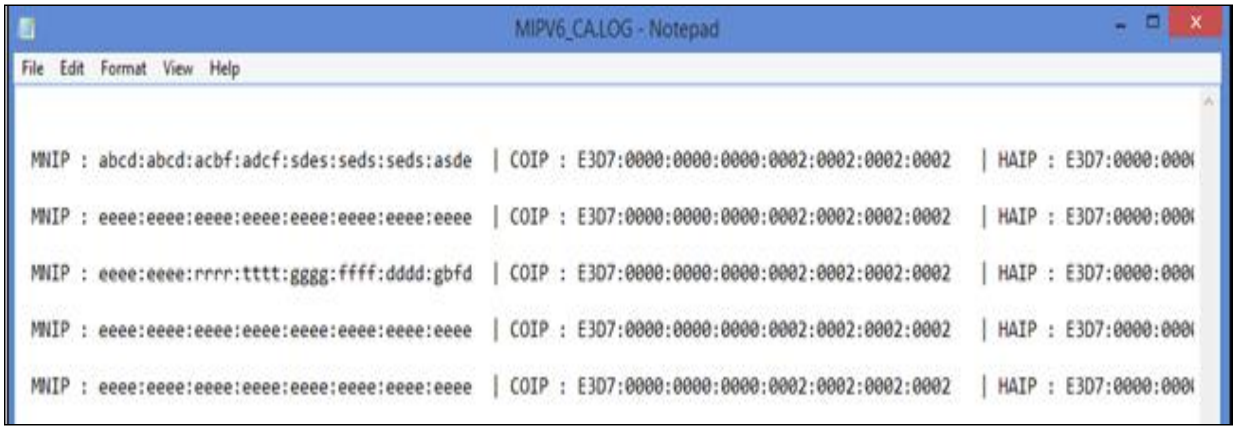
After it is confirmed that mobile node is in foreign location, the design now updates the care-of address as shown in the log file.



Name	Date modified	Size	Type
TC0003.SWP	1/23/2015 11:26 AM	0 KB	SWP File
TCDEF.DPR	1/23/2015 11:27 AM	4 KB	DPR File
MIPV6.LOG	1/23/2015 12:14 PM	23 KB	Text Document
TCDEF.DSK	1/23/2015 12:14 PM	2 KB	DSK File
MIPV6_CA.LOG	1/23/2015 12:15 PM	2 KB	Text Document

Figure 7.4 Care-of Address is updated

The log file generated contains the updated care-of address as shown below



```
MNIP : abcd:abcd:acbf:adcf:sdes:seds:seds:asde  | COIP : E3D7:0000:0000:0000:0002:0002:0002:0002   | HAIP : E3D7:0000:000
MNIP : eeee:eeee:eeee:eeee:eeee:eeee:eeee:eeee  | COIP : E3D7:0000:0000:0000:0002:0002:0002:0002   | HAIP : E3D7:0000:000
MNIP : eeee:eeee:rrrr:tttt:gggg:ffff:dddd:gbfd  | COIP : E3D7:0000:0000:0000:0002:0002:0002:0002   | HAIP : E3D7:0000:000
MNIP : eeee:eeee:eeee:eeee:eeee:eeee:eeee:eeee  | COIP : E3D7:0000:0000:0000:0002:0002:0002:0002   | HAIP : E3D7:0000:000
MNIP : eeee:eeee:eeee:eeee:eeee:eeee:eeee:eeee  | COIP : E3D7:0000:0000:0000:0002:0002:0002:0002   | HAIP : E3D7:0000:000
```

Figure 7.5 Log file with updated Care-of Address

All different cases were tested and in each case, the design function did perform its task by giving relevant messages about whether the mobile node was in home network or in foreign location, and care-of address was updated every time the mobile node moves to foreign location.

7.2.5 Result Analysis

The function check_inputs( ) is responsible to check the format for inputs mobile node IP address and home agent IP address. The complexity of the function check_inputs( ) is of the order of 1 given by O(1).

7.3 DRAWBACKS OF MIPV6

However IPv6 cannot solve all security problems. Basically it cannot prevent attacks on layers above the network layer in the network protocol stack. Few attacks that IPv6 cannot resolve are mentioned below:

1. Application layer attacks: Attacks performed at the application layer (OSI Layer 7) such as buffer overflow, viruses and malicious codes, web application attacks, and so on.

2. Brute-force attacks and password guessing attacks on authentication modules.

3. Rogue devices: Devices introduced into the network that is not authorized. A device may be a single PC, but it could be a switch, router, DNS server, DHCP server or even a wireless access point.

4. Denial of Service: The problem of denial of service attacks is still present with IPv6.

5. Attacks using social networking techniques such as email spamming, phishing, etc.

Below are some best practices for reference in building and maintaining secure IPv6 networks:

i. Use standard, non-obvious static addresses for critical systems;
ii. Ensure adequate filtering capabilities for IPv6;
iii. Filter internal-use IPv6 addresses at border routers;
iv. Block all IPv6 traffic on IPv4-only networks;
v. Filter unnecessary services at the firewall;
vi. Develop a granular ICMPv6 filtering policy and filter all unnecessary ICMP message types;
vii. Maintain host and application security with a consistent security policy for both IPv4 and IPv6;
viii. Use IPsec to authenticate and provide confidentiality to assets;
ix. Document the procedures for last-hop trace back; and
x. Pay close attention to the security aspects of transition mechanisms.

7.4 SUMMARY

This chapter presented the protocol verification of Mobile IPv6. The Text representation and notations of IPv6 addresses were described in detail. Next step was to develop a schema diagram to represent Mobile IPv6 model. Later design steps were formulated which was implemented using C and respective experimental results were noted down with the help of screen shots. The design steps included an option for the user to enter Mobile node IP address and this address was checked whether the input was an valid IPv6 address, once the mobile node IP address was validated, the user was prompted to enter Home agent IP address. This address was used to decide whether the mobile node is in home network or foreign network. The design was formulated to update care-of address if the mobile node was identified to be in foreign network. All valid mobile node IP address, Home agent IP address and care-of address was recorded in a log file.

CHAPTER 8
CONCLUSION, CONTRIBUTIONS and FUTURE WORK

The present work has been carried out to meet the objectives and scope as mentioned in first chapter. Firstly, the different phases in Mobile IP were identified in accordance with the Internet Transition plan. The identified phases of Mobile IP are Mobile IPv4, coexistence phase and Mobile IPv6. Each phase has been analyzed in detail and a schema diagram has been developed to model Mobile IPv4, coexistence phase and Mobile IPv6. From the schema diagram, design steps were formulated. Finally the input parameters for each module are given. The output from the modules is stored in Batch File, Dump File, Log file and standard output for subsequent viewing and interpretation.

8.1 CONCLUSIONS

Chapter 2 presented an overview of existing literature on mobile IP protocols and co-existent phase which includes Dual stack, tunneling in MIPv4 / MIPv6, Translation. Various mechanisms like Agent Discovery, Registration, Routing involved in Mobile IPv4 were described in detail. Similarly the description of Dual stack Transition mechanism was carried out followed by the benefits of Dual stack. The Tunneling and Translation in MIPv4/MIPv6 were described in detail. Finally the description and analysis of the basic operations involved in Mobile IPv6 was carried out.

Chapter 3 dealt with Protocol verification of Mobile IPv4. Here the design was formulated to process all the valid IPv4 addresses. These valid IPv4 addresses were stored in a Batch file, this batch file was designed to send nslookup command along with Ping operation. Now a Dump file was generated that consists of the ping statistics. The design of the code was extended such that all the valid IPv4 addresses along with the ping statistics were listed in an Excel file. This file was the final output.

Chapter 4 focused on Protocol Verification of Dual stack. First the operations involved in Dual stack were described and based on this description, a schema diagram was modeled to represent Dual stack. From this schema diagram, design steps was formulated to accept Mobile IPv4 and Mobile IPv6 addresses in such a way that all IPv6 addresses were sent to separate IPv6 stack and IPv4 addresses were sent to IPv4 stack. Finally respective results were noted down to prove Dual stack.

Chapter 5 presented Protocol verification of Tunneling. Encapsulation and Decapsulation are the two important mechanisms of Tunneling and these topics were described here in a detailed manner. A model was constructed to represent Encapsulation and Decapsulation in Tunneling. Then design steps were outlined such that code generated xml file to show an IPv6 data packet encapsulated within an IPv4 data packet, similarly the code generated a CSV file to show the IPv4 header stripped off to get back IPv6 datapacket which completes Decapsulation. The design steps were implemented in C.

Chapter 6 dealt with Protocol verification of Translation in Mobile IPv4 / Mobile IPv6. Different types of Translation were described in detail and schema diagram was constructed to represent Translation model. Design steps were framed to translate IPv6 address to IPv4 address. The design was implemented in C. This code was designed to generate xml file to view IPv6 and IPv4 datapackets. Four different cases for each data packet were identified and in each case, the data packets with IPv6 address was successfully translated to IPv4 address.

Chapter 7 presented the protocol verification of Mobile IPv6. The addressing notations used of IPv6 were described in detail. A schema diagram was constructed to represent Mobile IPv6 model. Design steps were formulated to give an option for the user to enter Mobile node IP address. This address was checked for validity of IPv6 address and then the user was prompted to enter Home agent IP address. This address was used to identify whether the mobile node was in home network or in foreign network. If the mobile node was identified to be in foreign network then a care-of address was updated. All valid mobile node IP address, Home agent IP address and care-of address was recorded in a log file. The design was implemented using C and respective experimental results were noted down.

The primary purpose of this thesis is to verify protocols running on end hosts in MIPv4-MIPv6 coexistence phase. Finally, to substantiate the idea used in this thesis, the conclusion can be summarized by analyzing the results of each module. Each module gives a Yes Output for all inputs thus proving the suggested proposals for each protocol.

8.2 SPECIFIC CONTRIBUTIONS

The specific contributions relating to the present research work are highlighted here.

- The design presented here generates a batch file that is used to store all valid IPv4 addresses. Executing this batch file generates a dump file that contains the status of datapackets sent, received and lost during transmission. The design was formulated to generate an excel file that shows the final result.The analysis of this result gives a “Yes“ as the final output, thus verifying the proposed protocol design. The design approach benefits by verifying the protocol in a feasible and less time consuming manner.

- The schema diagram modelled for Dual stack accepts a set of IPv4 as well as IPv6 addresses as Inputs. All IPv4 addresses were validated, processed and sent to IPv4 stack while IPv6 addresses were sent to IPv6 stack, thus verifying Dual stack proposal to be indeed the input protocol. This design approach provides a simplified level of verifying protocols by eliminating the requirement of any mathematical proofs to verify protocols.
- The source code developed from the design steps proposed for tunneling generates an xml file to view IPv4 and IPv6 data packet, each data packet consists of a header and body. The header of the datapacket holds a valid IPv6 source address and a destination address while the body holds a text message. On receiving this IPv6 datapacket as input, the design proposed here would give IPv6 datapacket encapsulated in IPv4 datapacket as its Output which is a “Yes” output thus proving the proposal. The xml file generated here was useful to demonstrate the datpackets.
- IPv4/IPv6 Translation is a vast quantitative aspect due to its eight different types of classifications. The internal mapping table used here helps to translate IPv6 addresses to IPv4 address. Thus, the design steps formulated provides an easy technique to model Translation.
- The design for Mobile IPv6 model identifies if the mobile node is at home network or in foreign network. The log file generated is beneficial to demonstrate the care-of address updated whenever the mobile node is in foreign network.

8.3 FUTURE SCOPE

This thesis presents a new set of approach that will enable to verify any given protocols. The presence of logical errors due to alterations in the assumptions and environment of the given protocol may lead to one or more statements or entire routines that will never be executed. This causes unnecessary work for the network designers and may result in discarding well developed protocol after its implementation. This thesis presents an approach that allows analyzing important properties of the protocol under consideration. The results obtained eventually help in synthesis of specific systems. It allows improving the correctness and the performance of a system.

Despite the existing wide variety of communication protocols, many new technologies are evolving. Because of this, the protocols, which define the network communication, are highly inter-related. Many protocols rely on others for operation. For example, many routing protocols use other network protocols to exchange information. There is a need to verify these protocols. Protocol Verification would be valuable for verifying the intent of the network designer, troubleshooting, and performing “what-if” analysis of failure scenarios. However verification can never guarantee total correctness but it can increase or decrease confidence in the model by showing presence of bugs. It is always good enough to find bugs at the initial stage rather than to discard a well-developed protocol post its implementation. It is necessary for protocol designers to have techniques and tools to detect errors in the early phase of design, because the later a fault is discovered in any process, the greater the cost of rectifying it.

REFERENCES

[1]. Andrew S Tanenbaum, David J Wetherall. Computer networks. Pearson Education, Limited. Jul 23, 2013.

[2]. Thomas J Walsh, Joyce S Kerr, Dr. Ashok K Jain. IPv6 Experimental Results Portend Operational benefits for Military Systems. IEEE. 6/07/2007. 1-4244-1513-0.

[3]. C Perkins. IP Mobility Support for IPv4. RFC 3344. Aug. 2002.

[4]. D Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6, RFC 3775. June 2004.

[5]. Jean E Martina. Verification of security protocols based on multicast communication. March 2012. [Online]. Available: http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-816.pdf.

[6]. F J Lin, P M Chu, M T Liu. Protocol verification using reachability analysis: the state space explosion problem and relief strategies. ACM SIGCOMM Computer Communication Review. 10/1987. 17(5):126-135.

[7]. Krishna Chakraborty, Nitul Dutta, S R Biradar. Simulation of IPv4-to-IPv6 Dual Stack Transition Mechanism (DSTM) between IPv4 hosts in integrated IPv6/IPv4 network. IEEE Conference on Computers and Devices for Communication. 2009. 978-1-4244-5073-2.

[8]. C Perkins. IP Mobility Support. RFC 2002. October 1996.

[9]. T Narten. Best Current Practice. RFC 4879. April 2007.

[10]. F Baker, X Li, K. Yin, C Bao. Framework for IPv4/IPv6 Translation. RFC 6144. April 2011.

[11]. Seyedeh Masoumeh Ahmadi. Analysis towards Mobile IPV4 and Mobile IPV6 in Computer Networks. I.J. Intelligent Systems and Applications. 2012. 33-39.

[12]. Lee, TT Lai. A relational algebraic approach to protocol verification Software Engineering. IEEE Transactions on Volume 14Feb. 1988 Page(s):184 – 193.

[13]. Ajin Jirachiefpattana, Richard Lai. An Estelle-NPN Based System for Protocol Verification. IEEE. 1995. 0-7803-2680-6.

[14]. Chung-Ming Huang, Duen-Tay Huang. A backward protocol verification method. TENCON'93 Proceedings. Computer, Communication, Control and Power Engineering.1993 IEEE Region 10 Conference on, 19-21 Oct. 1993 Page(s):515 - 518 vol.1.

[15]. Chung-Ming Huang, Jenq-Muh Hsu, Huei-Yang Lai, Jao-Chiang Pong, Duen-Tay Huang. An incremental protocol verification method for ECFSM-based protocols. Proceedings of the Eighth Annual Conference on 14-17 June 1993 Page(s):87 – 97.

[16]. Bhaskar Sardar, Debashis Saha. A Survey Of TCP Enhancements For Last-Hop Wireless Networks. 3rd Quarter 2006, Volume 8, No. 3. [Online]. Available: www.comsoc.org/pubs/surveys.

[17]. Wen-Chien Liu; Chyan-Goei Chung. Protocol verification using concurrent paths. Communications, Fifth Asia-Pacific Conference on Optoelectronics and Communications Conference, Volume 2, 18-22 Oct. 1999 Page(s):1188 -1191.

[18]. Wen-Chien Liu, Chyan-Goei Chung. Symbolic path-based protocol verification. Information and Software Technology 42. 2000. 245–255. [Online]. Available: www.elsevier.nl/locate/infsof.

[19]. Meng Xiao-jing, Wang Li-Ji. Analysis and Verification of Colored Petri Net in PPPoE Protocol.3rd International Conference on Advanced Computer Theory and Engineering(ICACTE). 2010.

[20]. C.A. Petri. Kommunikation mit Automaten, Second Edition, Technical Report RADC-TR-65--377, Vol.1, 1966, Pages: Suppl. 1.

[21]. K. Jensen. Coloured Petri Nets: A High Level Language for System Design and Analysis. InG. Rozenberg, editor, Advances in Petri Nets 1990, volume 483 of Lecture Notes in Computer Science, 1990. pages 342–416. Springer-Verlag, Berlin.

[22]. K. Jensen. Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. EATCS monographs on Theoretical Computer Science. 1992. Springer-Verlag, Berlin.

[23]. M. Ajmone Marsan, G. Balbo, and G. Conte. A Class of Generalised Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems. ACM Transactions on Computer Systems. May 1984. 2(2):93–122,

[24]. M. Ajmone Marsan, G. Balbo, and G. Conte et al. Modelling with Generalized Stochastic Petri Nets. Wiley series in parallel computing. Wiley, New York, 1995.

[25]. Brent T Hailpern, Susan S Owicki. Modular Verification of Computer Communication Protocols. IEEE Transactions on Communications. January 1983. Vol. COM-31, No. 1.

[26]. Deering, S. Ed, ICMP Router Discovery Messages. RFC 1256. September 1991.

[27]. J. Postel. User Datagram Protocol. RFC 768. 1980.

[28]. Calhoun, P. and C. Perkins. Mobile IP Network Access Identifier Extension for IPv4. RFC 2794. March 2000.

[29]. C. Perkins, Ed. IP Mobility Support for IPv4, Revised. RFC 5944. November 2010.

[30]. H. Krawczyk, M. Bellare, R. Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104. February 1997.

[31]. R. Rivest, The MD5 Message-Digest Algorithm,MIT Laboratory for Computer Science and RSA Data Security, Inc. RFC 1321. April 1992.

[32]. Postel, J. Internet Protocol. Darpa Internet Program, Protocol Specification. RFC 791. September 1981.

[33]. Plummer, D. Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. RFC 826. November 1982.

[34]. TechNet Library, IPv4 Routing, [Online]. Available:http://technet.microsoft.com/en-us/library/dd379495(v=ws.10).aspx.

[35]. P. Srisuresh, G. Tsirtsis, ,P. Akkiraju, A. Heffernan. DNS extensions to Network Address Translators (DNS_ALG). RFC 2694. September 1999.

[36]. P. Vixie, S. Thomson, Y. Rkhter, J. Bound. Dynamic updates in the domaon name system (DNS UPDATE). 1997.

[37]. Eun-Young Park, Jae-Hwoon Lee , Byoung-Gu Choe, An IPv4-to-IPv6 dual stack transition mechanism supporting transparent connections between IPv6 hosts and IPv4 hosts in integrated IPv6/IPv4 network. IEEE Communications, 2004. 0-7803-8533-0. 1024 - 1027 Vol.2.

[38]. Ra'ed AlJa'afreh, John Mellor, and Irfan Awan, A Comparison between the Tunneling process and Mapping schemes for IPv4/IPv6 Transition, International Conference on Advanced Information Networking and Applications Workshops, UK, 2009.

[39]. Kent S. IP Authentication Header. RFC 4302. December 2005.

[40]. www.cisco.com.[Online].Available: http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/enterprise-ipv6-solution/white_paper_c11-676277.html

[41]. Glass S, M Chandra. Registration Revocation in Mobile IPv4. RFC 3543. August 2003.

[42]. G Tsirtsis, P Srisuresh. Network Address Translation - Protocol Translation (NAT-PT). RFC 2766. February 2000.

[43]. Levkowetz H, S Vaarala. Mobile IP Traversal of NetworK Address Translation (NAT) Devices. RFC 3519. April 2003.

[44]. Conta A, S Deering. Generic packet Tunneling in IPv6 Specification. RFC 2473. December 1998.

[45]. Iljitsch van Beijnum. Running IPv6. A practical guide to configuiring IPv6 to Windows XP. Apress publishers. 2005.

[46]. S Kent. IP Authentication Header. RFC 4302. December 2005.

[47]. A Conta, S Deering, M Gupta. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 4443. March 2006.

[48]. R. Gunasundari and S. Shanmugavel, "Performance Comparison of Mobile IPv4 and Mobile IPv6 protocols in wireless systems," in Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International, 2009, pp. 1-8.

[49]. Plummer, D. An Ethernet Address Resolution Protocol. Symbolics Cambridge Research Center. RFC 826. November 1982.

[50]. Farinacci, D, Li T, Hanks S, Meyer D, and P. Traina. Generic Routing Encapsulation (GRE). RFC 2784. March 2000.

[51]. Montenegro G, Ed. Reverse Tunneling for Mobile IP, revised. RFC 3024. January 2001.

[52]. D. Eastlake 3rd. IP over Multi-Purpose Internet Mail Extension, Motorola Laboratories. November 2005.

[53]. Gunasundari and S. Shanmugavel, Performance Comparison of Mobile IPv4 and Mobile IPv6 protocols in wireless systems, in Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International, 2009, pp. 1-8.

[54]. C. Sujeong, P. Jaehyung, W. Yonggwan, Y. Mijeong, and C. Min Young, Performance Comparison of TCP Traffic over Mobile IPv4 and IPv6 Networks and a Mobile Network Deployment Approach, in Computer and Information Technology, 2005. CIT 2005. The Fifth International Conference on, 2005, pp. 469-473.

[55]. Seyedeh Masoumeh Ahmadi. Analysis towards Mobile IPV4 and Mobile IPV6 in Computer Networks. I.J. Intelligent Systems and Applications. 2012. 33-39.

[56]. G. Tsirtsis, Qualcom, H. Soliman, Problem Statement: Dual Stack Mobility, August 2007, 4977.

[57]. Daniel G. Waddington and Fangzhe Chang, Realizing the Transition to IPv6, Bell Research Laboratories. IEEE Communications Magazine. June 2002. 0163-6804.

[58]. Adrian Stoica, A Review on Mobile IP Connectivity and its QoS, International Journal of Multimedia and Ubiquitous Engineering. April, 2007. Vol. 2, No. 2.

[59]. Chuck Sellers. IPv6 Transition Mechanisms and Strategies. NTT Communications.[Online].Available:http://www.rmv6tf.org/wpcontent/uploads/2012/11/Chuck-Sellers.

[60]. McCann J, Deering S, and J. Mogul. Path MTU Discovery for IP version 6. RFC 1981. August 1996.

[61]. M.Samad,F.Yusuf, Habibah Hashim, Md Mahhdz Md Zan. Deploying Internet Protocol Version 6 (IPv6) Over Internet Protocol Version 4 (IPv4) Tunnel.StudentConference on Research and Development Proceedings, Malaysia. 2002.

[62]. Priyanka Rawat, Jean Marie Bonnin, and Laurent Toutain. Designing a Tunneling Header Compression (TuCP) for Tunneling over IP. Institute TELECOM/TELECOM Bretagne, Cesson S´evign ´e cedex, France.

[64]. Nordmark & Gilligan. Basic IPv6 Transition Mechanisms. RFC 4213. October 2005.

[63]. J. Mogul, S. Deering. Path MTU Discovery. RFC 1191. November 1990.

[65]. Lorenzo Colitti, Giuseppe Di Battista, and Maurizio Patrignani. Discovering IPv6-in-IPv4 Tunnels in the Internet. European Commission - 6NET - Large-scale International IPv6Testbed-IST-2001-32603.[Online].Available: http://www.dia.uniroma3.it/~compunet/www/docs/titto/noms-04.pdf.

[66]. P. Ferguson, D. Senie, Amaranth. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827. May 2000.

[67]. F. Baker. Ingress Filtering for Multihomed Networks. RFC 3704. March 2004.

[68]. Jyh-Cheng Chen, Tao Zhang. IP-Based Next-Generation Wireless Networks: Systems, Architectures, and Protocols. John Wiley & Sons. Feb 17, 2004.

[69]. Perkins C. Minimal Encapsulation within IP. RFC 2004. October 1996.

[70]. LauraChappell. Routing Sequences for ICMP. Novell Certified Professional. 2001.

[71]. X Li, C Bao , F Baker. IP/ICMP Translation Algorithm. RFC 6145. April 2011.

[72]. Lawrence E Hughes. Network Projects. [Online] Available: www.sixscape.com

[73]. Simon Perreault. IPv6 Migration workshop for IETF and 3GPP. DNS64 and NAT64. [Online]. Available: http://www.viagenie.ca/publications/2009-11-06-3gpp-ietf-ipv6-shanghai-nat64.pdf.

[74]. G Van de Velde, T Hain, R Droms, B.Carpenter, E Klein. Local Network Protection for IPv6. RFC 4684. May 2007.

[75]. R Hinden, S Deering. IP Version 6 Addressing Architecture. Request for Comments: 2373. July 1998.

[76]. J Younho, P Jaehyung, W Yonggwan, L Bae-Ho, N Seung Yoo, C Min Young. Comparative Evaluation of TCP Performances on MIPv4 and MIPv6 Protocols in Mobile Mesh Networks. 2nd IEEE/IFIP International Workshop on Broadband Convergence Networks. 2007. pp. 1-9.

[77]. Ki-sik kong, Wonjun lee, Youn-hee han, Myung-ki shin, Heungryeol you. Mobility Management for All-IP Mobile Networks: Mobile IPv6 vs. Proxy Mobile IPv6, IEEE Wireless Communications, April 2008, 1536-1284/08.

[78]. Xin Lin, Xing Li. Packet Fragmentation in IPv6 over IPv4 Tunnels. [Online]. Available: https://www.cs.duke.edu/~xinl/apan-mtu.pdf.

[79]. Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch. Network Time Protocol Version 4. Protocol and Algorithms Specification. RFC 5905. June 2010.

[80]. Postel J. Internet Protocol, Darpa Internet Program, Protocol Specification, Information Sciences Institute. RFC 791. September 1981.

[81]. Ms. Veena Bharti, Dr. Sachin Kumar. Survey of Network Protocol Verification Techniques. International Journal of Scientific and Research Publications. Volume 2, Issue 4. April 2012. 1 ISSN 2250-3153.

APPENDIX A:

Source Code - Protocol Verification of Mobile IPv4

Function 1. Check_inputs_IPV4( ).

```
int check_inputs_IPV4(char MNIP[ ])
{
//        char MNIP[] = MNIP[];
          int count = 0;
          int length = 0;
          int var_ret = 0;
   while (MNIP[length] != '\0' )
{         //MNIP2++;
                    if (MNIP[length] == '.' && length == 0)       return(0);
                    else
                              {
                                        if (MNIP[length] == '.')
                                        {         count++;

                                                  if   (MNIP[length+1]   ==   '.'   ||   MNIP[length+1]   ==   '\0')
          return(0);
                                                  else
                                                  {
                                                            var_ret = 1;
                                                            fprintf(const_fp,"\n MSG: All good till now");
                                                  }
                                        }
                              }
                    length ++;
}
          if (count != 3 || length > 15)    {
                    printf("\n ERR: The IPV4 %s is Invalid",MNIP);
                    return (0);
          }
          printf("\n MSG: The IPV4 %s is Valid",MNIP);
          fprintf(const_fp,"\n MSG: The IPV4 is %s",MNIP);
          return(var_ret);
}
```

Function 2.	Generate_IPv4 batchfile( ).

```
void Generate_IPv4 batchfile( )
{
        FILE *Bat_fp = fopen( "MIPV4.BAT", "w+" );
   FILE *fw = fopen( "ipv4.stk", "r" );
        char line [ 128 ];
        char MNIP[40];
        char *token = NULL;
        const char s[2] = "|";
   if(fw!=NULL)
   {
                while( fgets ( line, sizeof line, fw ) != NULL ) /* read a line */
     {
                        token = strtok(line, s);
                        sprintf(MNIP,"%s\0",token);
                        printf("\nMSG:IPV4: Processing IPs\t %s\n",MNIP);
                        fprintf(Bat_fp, "echo %s>>MIPV4.dmp\n",MNIP);
                        fprintf(Bat_fp, "echo PING>>MIPV4.dmp\n");
                        fprintf(Bat_fp, "ping %s>>MIPV4.dmp\n",MNIP);
                        fprintf(Bat_fp, "echo NSLOOKUP>>MIPV4.dmp\n",MNIP);
                        fprintf(Bat_fp, "nslookup %s>>MIPV4.dmp\n",MNIP);
                        fprintf(Bat_fp, "echo ------------------------------------>>MIPV4.dmp\n");
                }
         }
   else
        {
                printf("\nMSG:IPV4:Stack is Empty");
        }
        fclose(Bat_fp);
        fclose(fw);
}
```

Function 3. Program Formatter.c

```
#include <stdio.h>
void clearfile();
int main ( void )
{
  static const char filename[] = "MIPV4.dmp";
  FILE *file = fopen ( filename, "r" );
  FILE *outfile = fopen ( "IPV4_AddressChalangeOutput.CSV", "w" );
  char *token;
  const char s[2] = ":";
  clrscr();
  if ( file != NULL )
  {
    char line [ 128 ]; /* or other suitable maximum line size */
        char tmpline [ 128 ];
    int linenu = 1;
        while ( fgets ( line, sizeof line, file ) != NULL ) /* read a line */
    {
      //printf("%d %s",linenu,line ); /* write the line */
                  if (line[0] == '-' && line[1] == '-' && line[2] == '-' && line[4] == '-')
                  {
                        linenu = 1;
                  }
                 else
                 {
                        if(linenu == 1)
                        {
                              fprintf(outfile,"%s",line);
                              fprintf(outfile,",PING\n");
                        }

                        if(linenu == 10)
                        {
                              fprintf(outfile,",,%s",line);
                        }
                        if(linenu == 11)
                        {
                              fprintf(outfile,",,%s",line);
                              fprintf(outfile,",NSLOOKUP\n");
                        }
                        if (linenu > 12)
                        {
                              strcpy(tmpline,line);
                              token = strtok(tmpline, s);
                              if     (!strcmp(token,"Server")     ||     !strcmp(token,"Name")     ||
!strcmp(token,"Address"))
                              {
                                    fprintf(outfile,",,%s",line);
                              }
                              if (line[0] == 'D' && line[1] == 'N' && line[2] == 'S' )
                              {
                                    fprintf(outfile,",,%s",line);
                              }
                        }
                        linenu++;
                 }
                 //if(linenu == 22) linenu = 1;
        //getch();
        }
      fclose ( file );
              fclose ( outfile );
   }
   else
   {
      perror ( filename ); /* why didn't the file open? */
   }
            //To truncate the input file
   clearfile();

   fclose ( outfile );
   printf("The CSV file generated. Press Any Key");
   getch();
   return 0;
}

void clearfile()
{
            static const char filename[] = "MIPV4.dmp";
            FILE *file = fopen ( filename, "w+" );
            fclose ( file );
}
```

APPENDIX B:

Source Code - Protocol Verification of Dual Stack

Function : populate_hybrid_stack( )

```
void populate_hybrid_stack(char MNIP[ ], char HAIP[ ] )
{
	char var_temp;
	char var_moreInputs = 'n';
	int var_is_ip_valid = 0;
	FILE *f6 = fopen ( "ipv6.stk", "w+" );
	FILE *f4 = fopen ( "ipv4.stk", "w+" );
	do
	{
		printf("\n Enter MNIP : \n\t");
		scanf("%s",MNIP);
		var_is_ip_valid=check_inputs_IPV4(MNIP);
		if (var_is_ip_valid)	//IPV4 input found
		{
			fprintf(f4,"%s|\n",MNIP);
		}
		else					//IPV6 input provided (probably)
		{
			var_is_ip_valid=check_inputs_IPV6(MNIP,NULL);
			if(var_is_ip_valid)
			{
				printf("\n HAIP available? PRESS A KEY{y/n} ");
				var_temp=getch(); // to check if user will provide HAIP
				if (var_temp == 'y' || var_temp == 'Y')
				{
					printf("\n Enter HAIP : \n\t");
					scanf("%s",HAIP);
					var_is_ip_valid=check_inputs_IPV6(MNIP,HAIP);
					if (var_is_ip_valid)
						{fprintf(f6,"%s|%s\n",MNIP,HAIP);}
				}
				else
				{
					if (debug)
					{printf("\n MSG: Calling check inputs with one argument
\n\tMNIP: %s and \n\tHAIP: _",MNIP);}
					fprintf(const_fp,"\n MSG: Calling check inputs with one
argument \n\tMNIP: %s and \n\tHAIP: _",MNIP);
					var_is_ip_valid=check_inputs_IPV6(MNIP,NULL);
					if(var_is_ip_valid)
```

```
                                        {fprintf(f6,"%s|NULL\n",MNIP);}
                              }
                         }
                    }
                    printf("\n MSG: Do you wish to enter more PRESS A KEY{y/n}");
                    var_moreInputs = getch();
               } while (var_moreInputs == 'y' || var_moreInputs == 'Y');
               fclose ( f4 );
               fclose ( f6 );
               process_ipv4();
               process_ipv6();
          }
```

APPENDIX C:

Source Code - Protocol Verification of Tunneling in MIPv4/ MIPv6

Step 1: Encapsulation of IPv6 datapacket in IPv4 datapacket

This function generate_packet_data ( ) will generate IPv4 datapackets [intermediate file] which will be the input for the function Generate-encapsulate-packet-data ( ) which will generate Encapsulated IPv6 datapacket which is demonstrated using XML file.

```
void generate_packet_data (int istunneling)
{
 do
 {
        clrscr();
        printf("\n Enter Source IP : \n\t");
        scanf("%s",source);
        printf("\n Enter Destination IP : \n\t");
        scanf("%s",destination);
        getchar();
        printf(" Enter Message : \n");
        fgets(message, sizeof message, stdin);

        var_is_sip6_valid=check_inputs_IPV6(source);
        if (var_is_sip6_valid == 0)
        {
                var_is_sip4_valid=check_inputs_IPV4(source);
                if(var_is_sip4_valid)
                {
                        srctype = 0;
                }
        }
        else
        {
                srctype = 1;
        }
        var_is_dip6_valid=check_inputs_IPV6(destination);
        if (var_is_dip6_valid == 0)
        {
                var_is_dip4_valid=check_inputs_IPV4(destination);
                if(var_is_dip4_valid)
                {
                        desttype = 0;
                }
```

```
            }
            else
            {
                        desttype = 1;
            }

            if(srctype != -1 && desttype  != -1)
            {
                        if(istunneling == 1)                        // The request is for tunnelling
                        {
                                    if(srctype == 1)                              // The packet is IP6
                                    {
/*
<IPv4>
<SOURCE_IP> Source IPv4 Address </SOURCE_IP>
<DESTINATION_IP> Destination IPv4 Address </DESTINATION_IP>
<BODY >
<IPv6>
<SOURCE_IP> Source IPv6 Address </SOURCE_IP>
<DESTINATION_IP> Destination IPv6 Address </DESTINATION_IP>
<BODY>
Message
</BODY>
</IPv6>
</BODY>
</IPv4>
*/
                                                fprintf(efile,"<IPv4>\n");
                                                mapsrc = getipv4addr(source);
                                                fprintf(efile,"<SOURCE_IP> %s </SOURCE_IP>\n",mapsrc);
                                                fprintf(const_fp,"<SOURCE_IP> %s </SOURCE_IP>\n",mapsrc);
                                                if (desttype == 1)
                                                {
                                                            mapdest = getipv4addr(destination);
                        fprintf(efile,"<DESTINATION_IP> %s </DESTINATION_IP>\n",mapdest);

                                                }
                                                else
                                                {
                        fprintf(efile,"<DESTINATION_IP> %s </DESTINATION_IP>\n",destination);
                                                }
                                                fprintf(efile,"<BODY>\n");
                                                fprintf(efile,"<IPv6>\n");
                                                fprintf(efile,"<SOURCE_IP> %s </SOURCE_IP>\n",source);
                        fprintf(efile,"<DESTINATION_IP> %s </DESTINATION_IP>\n",destination);
                                                fprintf(efile,"<BODY>\n");
                                                fprintf(efile,"%s",message);
                                                fprintf(efile,"</BODY>\n");
                                                fprintf(efile,"</IPv6>\n");
                                                fprintf(efile,"</BODY>\n");
                                                fprintf(efile,"</IPv4>\n");
                                    }
                                    else                                    // The packet is IPV4
                                    {
                                                fprintf(efile,"<IPv4>\n");
                                                fprintf(efile,"<SOURCE_IP> %s </SOURCE_IP>\n",source);
                                                if (desttype == 1)
                                                {
                                                            mapdest = getipv4addr(destination);
                        fprintf(efile,"<DESTINATION_IP> %s </DESTINATION_IP>\n",mapdest);
                                                }
                                                else
                                                {
                        fprintf(efile,"<DESTINATION_IP> %s </DESTINATION_IP>\n",destination);
                                                }
                                                fprintf(efile,"<BODY>\n");
                                                fprintf(efile,"%s",message);
                                                fprintf(efile,"</BODY>\n");
                                                fprintf(efile,"</IPv4>\n");
                                    }
                        }

            mapsrc = NULL;
            mapdest = NULL;

            printf("\n MSG: Do you wish to enter more PRESS A KEY{y/n}");
            var_moreInputs = getch();
         } while (var_moreInputs == 'y' || var_moreInputs == 'Y');

         fclose ( efile );
         fclose ( tfile );
         printf("\n -------------------------------------------\n");
        }
```

Step 2 : Decapsulation

The Function Decapsulate-packet-data ( ) will strip off IPv4 header to generate the original IPv6 datapacket.

```
void decapsulate_packet_data(char filename[], char outfilename[])
{
while ( fgets ( line, sizeof line, fw ) != NULL ) /* read a line */
{
if(flag == 1)
          {
                    if(!strncmp(line,"<IPv6>",6))
                    {
                              flag = 0;
                              ip = 0;
                              memset ( source, 0, 50 );
                              memset ( destination, 0, 50 );
                    }
                    else
                    {
                              fprintf(fw1,"%s , ",source);
                              fprintf(fw1,"%s , ",destination);
                              fprintf(fw1,"%s",line);

                              fprintf(const_fp,"\n SOURCE2 %s",source);
                              fprintf(const_fp,"\n DESTINATION1 %s",destination);
                              fprintf(const_fp,"\n MESSAGE %s",line);

                              token2=NULL;
                              token1=NULL;
                              memset ( source, 0, 50 );
                              memset ( destination, 0, 50 );
                              flag = 0;
                              ip = 0;
                    }
          }
                 // Get the BODY
          if (!strncmp(line,"<BODY>",6))
          {
                    if (ip == 2)
                    { flag = 1; } }  }
 fprintf(fw1,"\n\n");
   fclose ( fw );
   fclose ( fw1 );
}
```

APPENDIX D:

Source Code - Protocol Verification of Translation in MIPv4 / MIPv6

```
 void TRANSLATE_IPv6_IPv4();
{
 do
 {
        clrscr();
        printf("\n Enter Source IP : \n\t");
        scanf("%s",source);
        printf("\n Enter Destination IP : \n\t");
        scanf("%s",destination);
        getchar();
        printf(" Enter Message : \n");
        fgets(message, sizeof message, stdin);

        var_is_sip6_valid=check_inputs_IPV6(source);
        if (var_is_sip6_valid == 0)
        {
                var_is_sip4_valid=check_inputs_IPV4(source);
                if(var_is_sip4_valid)
                {
                        srctype = 0;
                }
        }
        else
        {
                srctype = 1;
        }
        var_is_dip6_valid=check_inputs_IPV6(destination);
        if (var_is_dip6_valid == 0)
        {
                var_is_dip4_valid=check_inputs_IPV4(destination);
                if(var_is_dip4_valid)
                {
                        desttype = 0;
                }
        }
        else
        {
                desttype = 1;
        }
// The packet is for translation
                    if(srctype == 1)                            // The packet is IP6
                              {
                                        mapsrc = getipv4addr(source);
                                        //mapdest = getipv4addr(destination);
                                        fprintf(tfile,"<IPv4>\n");
                                        fprintf(tfile,"<SOURCE_IP> %s </SOURCE_IP>\n",mapsrc);
                                        if (desttype == 1)
                                        {
                                                  mapdest = getipv4addr(destination);
                    fprintf(tfile,"<DESTINATION_IP> %s </DESTINATION_IP>\n",mapdest);

                                        }
                                        else
                                        {
                    fprintf(tfile,"<DESTINATION_IP> %s </DESTINATION_IP>\n",destination);
                                        }
                                        fprintf(tfile,"<BODY>\n");
                                        fprintf(tfile,"%s",message);
                                        fprintf(tfile,"</BODY>\n");
                                        fprintf(tfile,"</IPv4>\n");
                              }
                              else
                              {
                                        fprintf(tfile,"<IPv4>\n");
                                        fprintf(tfile,"<SOURCE_IP> %s </SOURCE_IP>\n",source);
                                        if (desttype == 1)
                                        {
                                                  mapdest = getipv4addr(destination);
                    fprintf(tfile,"<DESTINATION_IP> %s </DESTINATION_IP>\n",mapdest);
                                        }
                                        else
                                        {
                    fprintf(tfile,"<DESTINATION_IP> %s </DESTINATION_IP>\n",destination);
                                        }
                                        fprintf(tfile,"<BODY>\n");
                                        fprintf(tfile,"%s",message);
                                        fprintf(tfile,"</BODY>\n");
                                        fprintf(tfile,"</IPv4>\n"); } }
          mapsrc = NULL; mapdest = NULL;
           printf("\n MSG: Do you wish to enter more PRESS A KEY{y/n}");
          var_moreInputs = getch();
 } while (var_moreInputs == 'y' || var_moreInputs == 'Y');
  fclose ( efile );
 fclose ( tfile );
 printf("\n ---------------------------------------------\n"); }
```

APPENDIX E:
Source Code - Protocol Verification of Mobile IPv6

Function 1:

```
 int check_inputs(char MNIP[], char HAIP[])
{
	int var_ret = 0;
	if(MNIP[4] == ':' && MNIP[9] == ':' && MNIP[14] == ':' && MNIP[19] == ':' &&
MNIP[24] == ':' && MNIP[29] == ':' && MNIP[34] == ':')
	{
		if (debug)
			{printf("\n MSG: The MNIP Is a valid IPV6 IP Address");}
		fprintf(const_fp,"\n MSG: The MNIP Is a valid IPV6 IP Address");
		var_ret = 1;
	}
	else
	{	if (debug)
			{printf("\n MSG: The MNIP Is NOT a valid IPV6 IP Address");}
		fprintf(const_fp,"\n MSG: The MNIP Is NOT a valid IPV6 IP Address");
		var_ret = 0;
		return(var_ret);
	}
	if(HAIP == '\0' )
	{	if (debug)
			{printf("\n MSG: No HAIP Provided, will search for home");}
		var_ret = 1;
	}
	else
	{
		if(HAIP[4] == ':' && HAIP[9] == ':' && HAIP[14] == ':' && HAIP[19] == ':' &&
HAIP[24] == ':' && HAIP[29] == ':' && HAIP[34] == ':')
		{
```

```
                    if (debug)
                              {printf("\n MSG: The HAIP Is a valid IPV6 IP Address");}
                    fprintf(const_fp,"\n MSG: The HAIP Is a valid IPV6 IP Address");
                    var_ret = 1;
          }
          else
          {
                    if (debug)
                              {printf("\n MSG: The HAIP Is NOT a valid IPV6 IP Address");}
                    fprintf(const_fp,"\n MSG: The HAIP Is NOT a valid IPV6 IP Address");
                    var_ret = 0;
          }
     }
     return(var_ret);
}
```

FUNCTION 2:

check_if_mnip_home( ): The parameters are char array for MNIP

```
int check_if_mnip_home(char MNIP[])
{
     int is_ip_home = 0;
     if (debug)
          {
fprintf(const_fp,"\n MSG: Checking for the home IP Range");
}
if (MNIP[5] == '0' && MNIP[6] == '0' && MNIP[7] == '0' && MNIP[8] == '0' && MNIP[10]
== '0' && MNIP[11] == '0' && MNIP[12] == '0' && MNIP[13] == '0' && MNIP[15] == '0' &&
MNIP[16] == '0' && MNIP[17] == '0' && MNIP[18] == '0')
{
if (debug)
{
fprintf(const_fp,"\n MSG: IP Address section 2,3 & 4 Matched to 0");}
if (MNIP[0] == 'E' && MNIP[1] == '3' && MNIP[2] == 'D' && MNIP[3] == '7')
{
if (debug)
                        {fprintf(const_fp,"\n MSG: IP Address section 1 Matched");}
      if (MNIP[20] == '5' && MNIP[21] == '1' && MNIP[22] == 'F' && MNIP[23] == '4')
                  {
                        if (debug)
                  {fprintf(const_fp,"\n MSG: IP Address section 5 Matched");}
      if (MNIP[25] == '9' && MNIP[26] == 'B' && MNIP[27] == 'C' && MNIP[28] == '8')
                        {
                              if (debug)
                        {fprintf(const_fp,"\n MSG: IP Address section 6 Matched");}
      if (MNIP[30] == 'C' && MNIP[31] == '0' && MNIP[32] == 'A' && MNIP[33] == '8')
                              {
                                    if (debug)
                        {fprintf(const_fp,"\n MSG: IP Address section 7 Matched");}
                                    if (MNIP[35] == '6' && MNIP[36] == '4')
                                    {
                                          int a = atoi(&MNIP[37]);
                                          if (debug)
      {fprintf(const_fp,"\n MSG: IP Address section 8 Matching with last 2 digits as %i",a);}
                        if (atoi(&MNIP[37]) >= 20 && atoi(&MNIP[37]) <= 48)
                                          {
                                                if (debug)
                              {fprintf(const_fp,"\n MSG: MNIP is in home range");}
                                                is_ip_home = 1;
                                          }
                                    }}}}}}
      return(is_ip_home);
}
```

FUNCTION 3 :

update_CareOfAddress():The parameters are char array for MNIP, COIP, HAIP

```
int update_CareOfAddress(char MNIP[], char HAIP[], char COIP[])
{
        var_cache = fopen( "MIPV6_CA.log", "a+" );
        fprintf(var_cache, "\n\n MNIP : %s  | COIP : %s   | HAIP : %s ", MNIP,COIP,HAIP);
        fclose(var_cache);
        return(1);
}
```

FUNCTION 4 :

find_home_agent():The parameters are char array for MNIP, COIP

```
int find_home_agent(char MNIP[],char COIP[])
{
//Function is dummy returns a dummy IP and sends it to update cache function
        char
HAIP[40]={'E','3','D','7',':','0','0','0','0',':','0','0','0','0',':','0','0','0','0',':','0','0','0','1',':','0','0','0','1',':','0','
0','0','1',':','0','0','0','1','\0'};
        if (debug)
                {printf("\n MSG: No HAIP Provided, this function will return the HAIP
for the provided MNIP");}
        return (update_CareOfAddress(MNIP, HAIP,COIP));
}
```

APPENDIX F:

Cl Test

C1 Test is used for testing comprehensiveness and it is the first truly structure-based methodology adapted by many researchers in the field of protocol verification. The C1 measure has its origin in research work aimed at developing a strong measure for testing effectiveness that takes into account most, if not all, of the elementary features of a software system. C1 measures the total number of segments exercised in each test, or when computed cumulatively, the total fraction of segments that are exercised in one or more tests in a series.

The C1 goal is to have a set of tests that in aggregate exercise a high percentage of all of the segments it is possible to exercise. Current thinking is that a value of C1 of 85% or greater is a practical level of testing coverage. Experience suggests that this level of C1 coverage is adequate to discover perhaps 90% of the available errors. The methodology for getting C1 = 85% is quite well understood in general terms, and involves structured test planning, dynamic operation of the program, and a lot of study of the source text.

Conclusion: C1 Test is a good starting point for a systematic methodology for testing.

<u>C1 Test performed on Function: populate_IPV4_stack()</u>

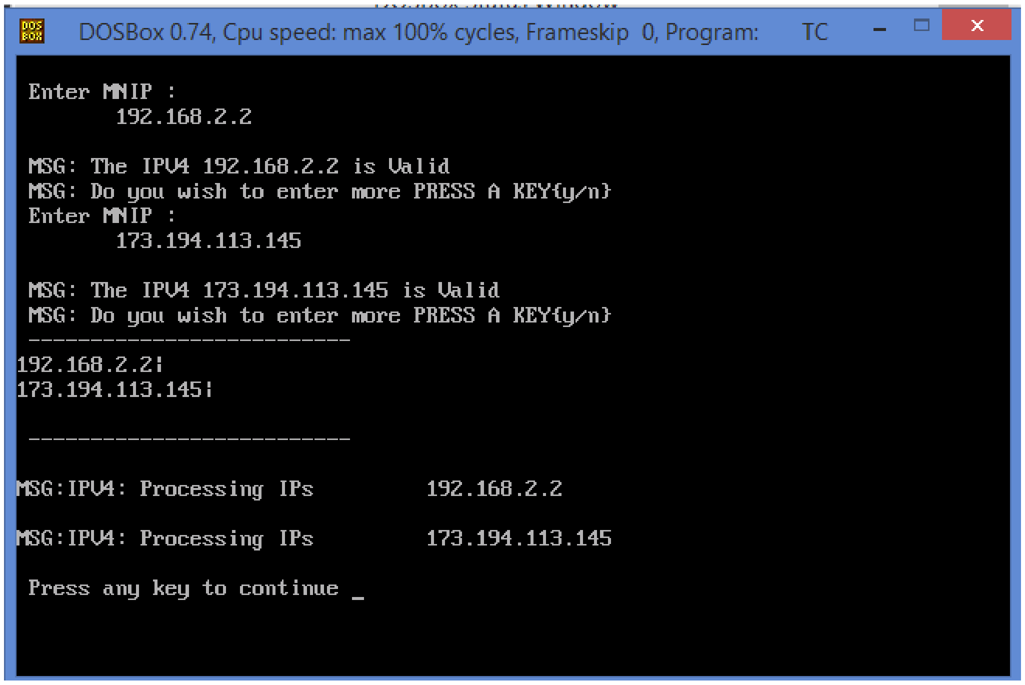
This function was responsible to populate IPV4 in the design. The parameters are char array for MNIP

```
void populate_IPV4_stack(char MNIP[])
{
        int var_is_ip_valid = 0;
        char var_moreInputs = 'n';
        FILE *file = fopen ( "ipv4.stk", "w+" );
        do
                {
                        printf("\n Enter MNIP : \n\t");
                        scanf("%s",MNIP);
fprintf(const_fp,"\n MSG: Calling check inputs with one argument \n\tMNIP: %s",MNIP);
                        var_is_ip_valid=check_inputs_IPV4(MNIP);
                        if(var_is_ip_valid)
                        {
```

```
                        fprintf(file,"%s|\n",MNIP);
                }
                else
                {
                        fprintf(const_fp,"\n ERR: The IP Address is not for IPV4 format");
                }

                printf("\n MSG: Do you wish to enter more PRESS A KEY{y/n}");
                var_moreInputs = getch();
        } while (var_moreInputs == 'y' || var_moreInputs == 'Y');
    fclose ( file );
    printf("\n --------------------------\n");
    display("ipv4.stk");
    printf("\n --------------------------\n");
    process_ipv4();
}
```

C1 TEST Results on function populate-IPV4-stack:

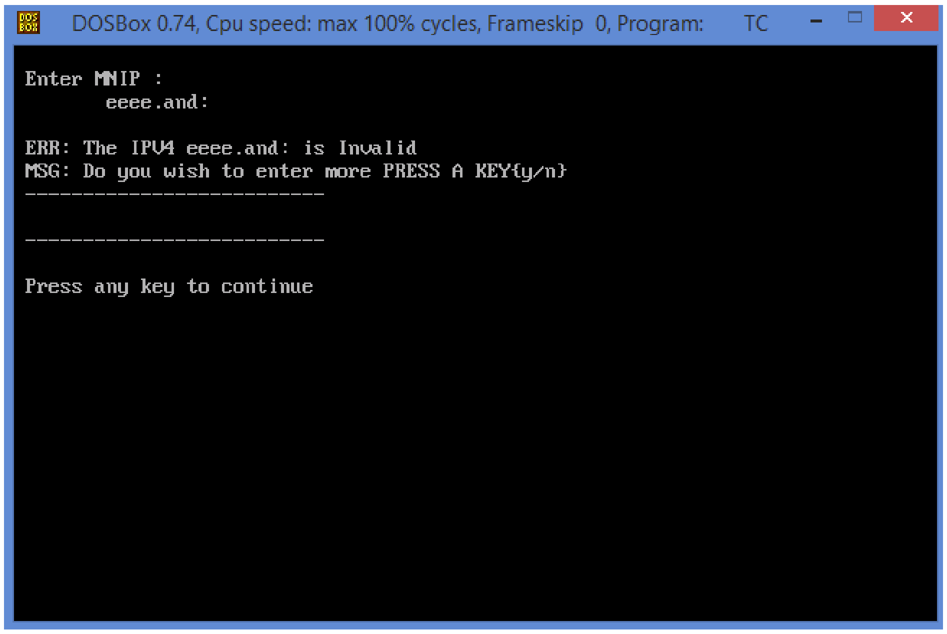
1. Enter correct IPv4 address, true part of IF gets executed



```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip  0, Program:    TC

 Enter MNIP :
        192.168.2.2

 MSG: The IPV4 192.168.2.2 is Valid
 MSG: Do you wish to enter more PRESS A KEY{y/n}
 Enter MNIP :
        173.194.113.145

 MSG: The IPV4 173.194.113.145 is Valid
 MSG: Do you wish to enter more PRESS A KEY{y/n}
 --------------------------
192.168.2.2|
173.194.113.145|

 --------------------------

MSG:IPV4: Processing IPs         192.168.2.2

MSG:IPV4: Processing IPs         173.194.113.145

 Press any key to continue _
```

Figure F.1 True part of *if* gets selected.

2. Enter any input other than IPV4 address, this will be go to else part


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
Enter MNIP :
       eeee.and:

ERR: The IPV4 eeee.and: is Invalid
MSG: Do you wish to enter more PRESS A KEY{y/n}
--------------------------

--------------------------

Press any key to continue
```

Figure F.2 Else part of *if* gets selected.

Note: There are two test cases identified, with each case contributing to 50% thus summing up each of the test case would give a C1 result of 100 %.

IF:

Table F.1 Tabular Column to show Code Coverage using C1 Test methodology

Test Case	*THEN*	*ELSE*	*COVERAGE*
Test Case1	1	0	50%
Test Case2	0	1	50%
Test Case3	0	0	0%

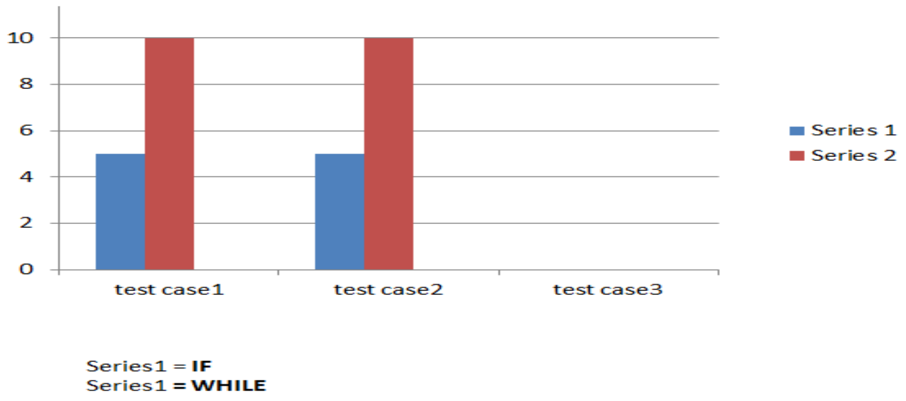


Figure F.3 Graph showing code coverage for *if* and *while*.

LIST OF PUBLICATIONS

International Journals

[1]. Susanna S Henry, V. Santhosh Kumar. Current Status of Mobile Internet Protocol version 4 and its Security Issues. IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 2, No 2, March 2012. ISSN (Online): 1694-0814

[2]. Susanna S Henry, V. Santhosh Kumar. Mobile Devices on IP Internetworks. IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 3, No 1, May 2012. ISSN (Online): 1694-0814.

[3]. Susanna S Henry, Arshdeep Singh. Website development for Course Management of an Academic Institution. International Journal of Computer Science Issues. January 2014. IJCSI-2015-12-1-10031.

[4]. Susanna S Henry, Hanna Zehara, V. Santhosh Kumar, B. Vijayakumar. Testing, Analysis and Implementation of Care-of address (CoA) and Home Agent (HA) leading to Verification of Mobile IPv6 Protocols. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA). (communicated)

[5]. Susanna S Henry, Mohammed Muzammil Kaleemulla, V. Santhosh Kumar, B. Vijayakumar. Design Consideration for setting up a Tunnel for IPv6 over IPv4. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA). (communicated)

[6]. Susanna S Henry, Indrani Sen, Niranjan Gopinath, V. Santhosh Kumar, B. Vijayakumar. Quote Quiz under Gaming Application Development for iOS . Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA). (communicated)

[7]. Susanna S Henry, Divya Henry, V. Santhosh Kumar, B. Vijayakumar. Design Analysis and Implementation of Dual Stack in Mobile IPv4 /IPv6. Journal of Wireless Mobile

Networks, Ubiquitous Computing, and Dependable Applications (JoWUA). (communicated)

International Conference

[8]. Susanna S Henry, Santhosh Kumar. A Review on Protocol Verification in Mobile Internet Protocol Version 4 and 6. Proceedings of the 2nd Conference on TechnologyManagement (MCTM - 2011), Dubai. 24th March, 2011. ISSN 2222-4696. Page No:61-64.

[9] Susanna S Henry, V. Santhosh Kumar. On Reachability Analysis in MIPv4-MIPv6 coexistence phase. 2013 International Conference on Current Trends in Information Technology (CTIT). Dubai. 11-12 December 2013. INSPEC Accession Number: 14131075.

[10]. Susanna S Henry, Vijay Kumar, V. Santhosh Kumar, Gurwinder Singh. Protocol Verification of Translation in Mobile Internet Protocol version 4 and 6. 2015 8th International Conference on Advanced Computer Theory and Engineering (ICACTE 2015). Web site:http://www.icacte.org. Germany, Berlin, 13-14 August 2015 (Communicated)

BRIEF BIOGRAPHY OF THE CANDIDATE

Ms. Susanna S Henry graduated in Bachelor of Engineering Electronics & Communication from Visvesvaraya Technological University, India in 2002. In 2005 she completed her Master of Technology in Computer Science from Visvesvaraya Technological University and joined Acharya Institute of Technology, India as a Lecturer. In 2007 she joined Department of Computer Science & Engineering at BITS-Pilani, Dubai Campus as lecturer and was promoted to senior lecturer in 2010.

Ms. Susanna S Henry is a professional member for Association of Computing Machinery and is a Life Member of the Indian Society for Technical Education (ISTE). She is the faculty incharge of BPDC ACM Student Chapter.

BRIEF BIOGRAPHY OF THE SUPERVISOR

Dr. V. SANTHOSH KUMAR graduated as Bachelors in Computer Engineering from Mangalore University, India in 1990. He then completed his Masters in Computer Science from Birla Institute of Technology and Science, Pilani, India in 1997. In 2007, he completed his Ph. D in Computer Science from Indian Institute of Science, Bangalore, India. His professional experiences include both academic and industry. He had worked as Staff Software Engineer in IBM India Software Labs, Bangalore, India during 2006 – 2009. Since 2009, he is working as an Assistant Professor in the Computer Science Department of BITS Pilani, Dubai Campus. His research interests include computer networks and performance evaluation of computer systems.