

TRANSITION OF CAMPUS NETWORK TO  
IP NEXT GENERATION

By

KHAKIM SULTANOV

Bachelor of Science

Rochester Institute of Technology

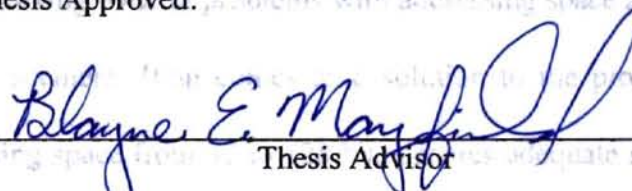
Rochester, New York

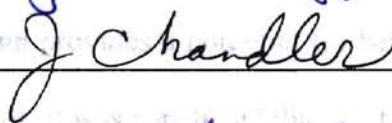
1994

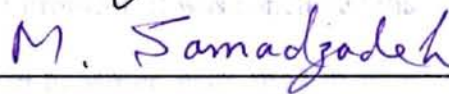
Submitted to The Faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
MASTER OF SCIENCE  
December 1997

TRANSITION OF CAMPUS NETWORK TO  
IP NEXT GENERATION

Thesis Approved:

  
Thesis Advisor

  
\_\_\_\_\_

  
\_\_\_\_\_

  
Dean of Graduate College

## PREFACE

This work was conducted to define the necessity and feasibility of transition of the campus network for Oklahoma State University to IP Next Generation. IP Next Generation, or IPng, is a new protocol designed as a successor to the current version of IPv4 used in networking. Due to problems with addressing space and routing tables, IPv4 is in need of replacement. IPng comes as a solution to the problems related to IPv4. Increased addressing space from 32 to 128 bits ensures adequate space for growth of the Internet, improved design provides a potential for better performance of routers working with the Internet Protocol. It was concluded that the tunneling procedure does not impose much overhead in transition from IPv4 to IPv6. A plan of introducing IPv6 on the OSU campus was presented.

## ACKNOWLEDGMENTS

Page

I wish to express my appreciation to my advisor, Dr. Mayfield for his time and constructive guidance throughout the course of this work. Many thanks go to my committee members: Drs. Samadzadeh and Chandler. Their support and suggestions proved to be invaluable.

I would like to thank my former supervisors Eddy Boujaoude and Salvador Paredes in Data Communications, Oklahoma State University, who made this research possible.

This graduate study was funded by REAP (Russian East European Award Program). Thanks are due to the Dean of the College of Arts and Sciences at OSU, Dr. Smith Holt and Dr. Keith Tribble, who offered me an opportunity to participate in this program.

A special appreciation goes to my friend, Jayson Murray, for his assistance with English in writing this work.

## TABLE OF CONTENTS

Chapter	Page
I	INTRODUCTION ..... 1
1.1	Networking of Computers ..... 1
1.2	A Brief History of the Internet ..... 1
1.3	Current Problems of the Internet..... 2
1.4	Purpose of the Research ..... 3
II	TCP/IP NETWORKS ..... 4
2.1	TCP/IP Network Components ..... 4
2.2	The Seven Layer Model ..... 6
2.3	Internet Protocol ..... 10
2.4	Internet Addressing ..... 11
2.5	Transmission Control Protocol ..... 14
2.6	Problems with IPv4 ..... 15
2.6.1	Problems with Class B Addresses..... 16
2.6.2	Routing Table Explosion ..... 17
2.7	Need for IPng ..... 18
III	SPECIFICATIONS OF IPV6 ..... 20
3.1	IPv6 Design ..... 20
3.1.1	Comparison Between IPv4 and IPv6 ..... 20
3.1.2	Simplifications ..... 21
3.1.3	Extension Headers ..... 24
3.2	Routing and Addressing ..... 27
3.2.1	IPv6 Address Format ..... 28
3.2.2	Provider Address ..... 30
3.2.3	Special Address Formats..... 32
IV	TRANSITION TECHNIQUES AND PROBLEMS CAUSED BY THE TRANSITION ..... 35
4.1	Introduction ..... 35

Chapter	Page
4.2	Dual Stack ..... 35
4.3	Tunneling ..... 36
4.4	Problems with IPv6 ..... 38
4.4.1	TTL in Tunnels ..... 38
4.4.2	Application Interface ..... 40
4.4.3	Bandwidth on Demand ..... 40
4.4.4	Industry's Opinion ..... 41
4.5	Current Projects and Implementation of IPv6 ..... 41
4.5.1	6Bone ..... 41
4.5.2	University Projects ..... 42
4.5.3	Industry Overview ..... 43
4.6	Bringing IPv6 on Campus ..... 44
V	LAB TESTS ..... 46
5.1	Building a Test Network ..... 46
5.2	Address Translation ..... 49
5.3	Configuring the Tunnel ..... 51
5.4	Results and Analysis ..... 52
VI	CONCLUSIONS AND SUGGESTED RESEARCH ..... 57
	REFERENCES ..... 59
	APPENDICES ..... 61
	APPENDIX A – OSI NETWORK ARCHITECTURE REFERENCE MODEL ..... 61
	APPENDIX B – FILE TRANSFER SESSIONS ..... 62
	APPENDIX C – RESULTS OF THE TESTS ..... 74
	APPENDIX D – TEST LABORATORY EQUIPMENT ..... 75
	APPENDIX E – TEST MACHINES' SETUP ..... 76
	APPENDIX F – GLOSSARY ..... 79

## LIST OF TABLES

Table	Page
1. Ranges of IP addresses within classes .....	11
2. An Example Conversion of 139.78.114.2 into IPv6 address .....	49
3. An Example Conversion of 139.78.114.154 into IPv6 address .....	50
4a. Averaged results of data transfer from PC1 to PC2 .....	53
4b. Averaged results of data transfer from PC2 to PC1 .....	53
C1. Results of File Transfer from PC1 to PC2 .....	74
C2. Results of File Transfer from PC2 to PC1 .....	74

## LIST OF GRAPHS

Graphs	Page
1 Assigned and Allocated Network Numbers .....	17



## LIST OF FIGURES

Figure	Page
1. Two PCs running serial line IP (SLIP) .....	5
2. A host A from Network A is communicating via the Router to Host F on Network B .....	5
3. OSI Seven Layer Model .....	6
4. Four layer Model .....	7
5. Encapsulation of data going down the protocol stack .....	9
6. IP datagram .....	10
7. Class A IP address .....	12
8. Class B IP address .....	12
9. Class C IP address .....	12
10. Class D IP address .....	13
11. Class E IP address .....	13
12. The IPv6 Header .....	20
13. The IPv4 Header .....	21
14. Examples of Chains of Headers .....	24
15. Routing Extension Header .....	25
16a. Routing Scheme. Host S1 is connected to H1 .....	26
16b. Routing Scheme. Host S1 is connected to B1 .....	26
17. Generic Format of Provider Addresses .....	30
18. The Initial Structure of a Provider-Based Address .....	31
19. Structure of a site local address .....	33
20. Structure for a link local address .....	33
21. Encapsulation of IPv6 over IPv4 .....	36
22. Tunneling .....	37
23. Test LAN Layout .....	47
24. The Ethernet frame with MTU less than 1500 .....	48

## CHAPTER I

### INTRODUCTION

#### 1.1 Networking of Computers

Computers used to be stand-alone systems. Every computer had a required software to do a particular job and peripheral resources to take an input from a user and to produce an output for a job. The situation changed when computer systems started to share resources and data. Nowadays computer share resources and information using networks. On a single Local Area Network (LAN), several computers can share a printer. People can send electronic messages to different countries using Global Networks. The complexity of a network can vary from a simple LAN to the Internet, a global conglomerate of networks.

#### 1.2 A Brief History of the Internet

The Internet appeared as a result of development of Advanced Research Project Agency Network (ARPANET), a research project sponsored by the Department of Defense (DOD).

The primary objective of this project was to create ways of connecting LANs and Wide Area Networks (WAN). In the late 1960s, ARPANET was connecting several military sites, universities, and research institutions. In the beginning of the 1980s, a set of communication protocols was developed to move data across the ARPANET. That protocol suite is referred to today as TCP/IP (Transmission Control Protocol, Internet Protocol). The name TCP/IP came from the names of the most popular protocols in the suite TCP and IP.

Using the TCP/IP suite, the National Science Foundation (NFS) backbone network got connected to the ARPANET. The NFS backbone was created under a project funded by the NFS to connect several national supercomputer centers. Also, NFS has funded several regional network projects that provide network access to every state.

### 1.3 Current Problems of the Internet

In the 1990s, the number of nodes on the Internet is doubling every year [6]. By 1996, more than 150 countries were connected to the Internet. With the current rate of growth, it is expected that the Internet will deploy all of its available addressing space by 2010 [19].

The Internet Engineering Task Force (IETF) has coordinated common efforts to overcome the problems. As a result of these directed efforts, new standards have been worked out to substitute the current version of the Internet Protocol (IP), a protocol responsible for addressing and routing on the Internet. Internet Protocol version 6 (IPv6),

or as some people call it, IP Next Generation (IPng), will replace its predecessor IPv4, the current version of IP [3].

Another problem the Internet faces is an explosion of routing tables. Each of the devices on a network has to have a routing table to determine how to get to the other devices (hosts). As the number of nodes on the network grows, so does the size of the routing tables. With the growth of the Internet, the backbone routers started to experience problems due to the increased size of the routing tables. The problems were caused by a lack of ability of the devices to keep up with the routing updates.

#### 1.4 Purpose of the Research

The purpose of this research was to investigate the differences between IPv6 and IPv4, and the ways of migrating a network from IPv4 base to IPv6. Within the framework of my research, a test IPv6 network was built. The idea of the test setup was to gain experience with IPv6 and the performance evaluation of a host running IPv6. As a result of the research, a set of procedures for introducing IPv6 to the Oklahoma State University campus network is presented.

Networks can be a LAN (one "Token Ring" network) or a WAN (a network in which the hosts exchange data)

## CHAPTER II

### TCP/IP NETWORKS

#### 2.1 TCP/IP Network Components

Physical networks are interconnected computer systems such as workstations, personal computers, and mainframes. Peripheral devices such as hard disks and line printers usually are connected to the network via a computer system. For example, a print-server (i.e., a computer controlling a printer on a network) allows "sharing" the printer between the users on the network.

The TCP/IP networks consist of three components. These are the hosts, the networks, and the routers.

**Host:** A computer system running TCP/IP protocols. The application programs allow the computers to communicate with each other. A host can be any PC, a graphics workstation, a mainframe, etc.

**Network:** A TCP/IP network is a collection of more than two hosts running TCP/IP protocols to communicate with each other. One of the advantages of TCP/IP is being independent of underlying physical networks. Two computers connected over a serial line can establish a TCP/IP network using the TCP/IP stack of protocols (Figure 1). At the same time a campus network using fiber optic connections can be a TCP/IP network running on top of the fiber.

The “Network A” represented in Figure 2 can be a LAN. The “Token Ring” network, on the same figure is a network in which the hosts exchange data by passing a *token* (see [21]).

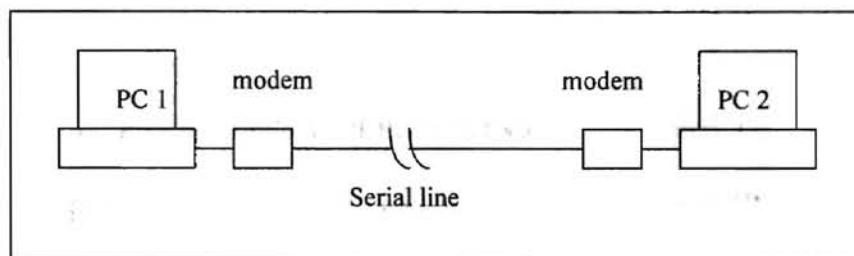


Figure 1. Two PCs running serial line IP (SLIP)

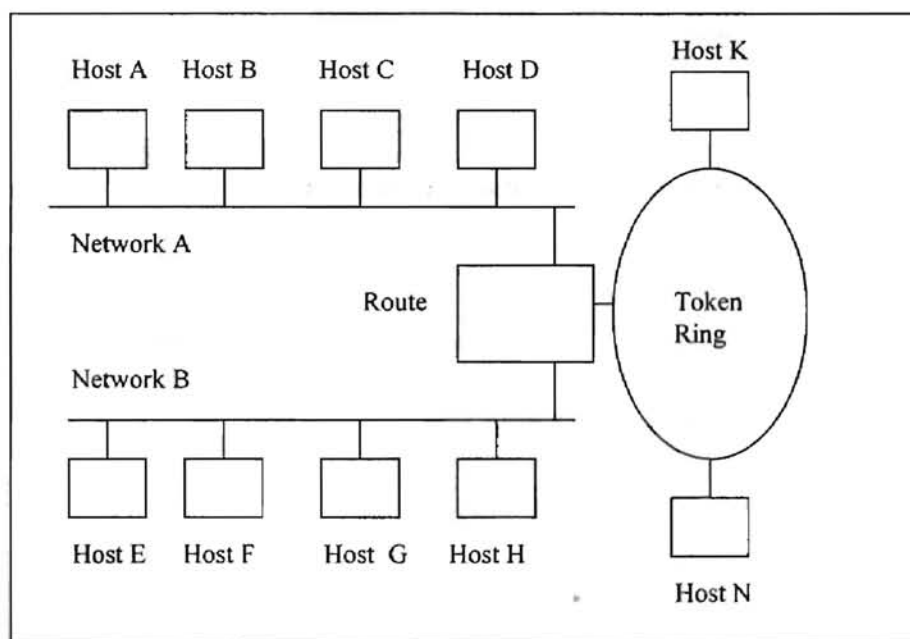


Figure 2. A host A from Network A is communicating via the Router to Host F on Network B

**Router:** A device that provides connectivity between the various individual networks. A router moves data from one physical network to another when a host

attached to one network is communicating with a host on some other physical network (Figure 2).

## 2.2 The Seven Layer Model

Although the protocol suite used in IP networking is called TCP/IP, there are other protocols comprising it besides the widely known TCP and IP. Other wellknown protocols in the TCP/IP suite are SNMP (Simple Network Management Protocol), ICMP (Internet Control Message Protocol), and SMTP (Simple Mail Transfer Protocol). A detailed description of the protocols constituting the TCP/IP suite can be found in a book by Stevens [21].

The communication process is divided into layers. Each layer is responsible for providing a particular service to make the communication process possible. Most systems may be described using the Open Systems Interconnection model (OSI) as proposed by International Standard Organization (ISO) [20]. There are seven layers present in the model (Figure 3):

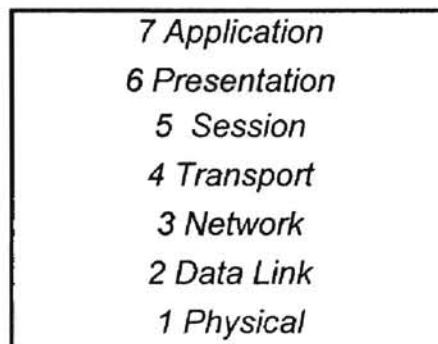


Figure 3. OSI Seven Layer model [20]

Attention in this research will be focused upon layer #3, which is responsible for addressing on the network and navigating data packets across the net. A description of the other layers is provided in Appendix A.

What follows is a simplified 4 layer model of the 7 layer presentation with correspondence to the OSI Levels.

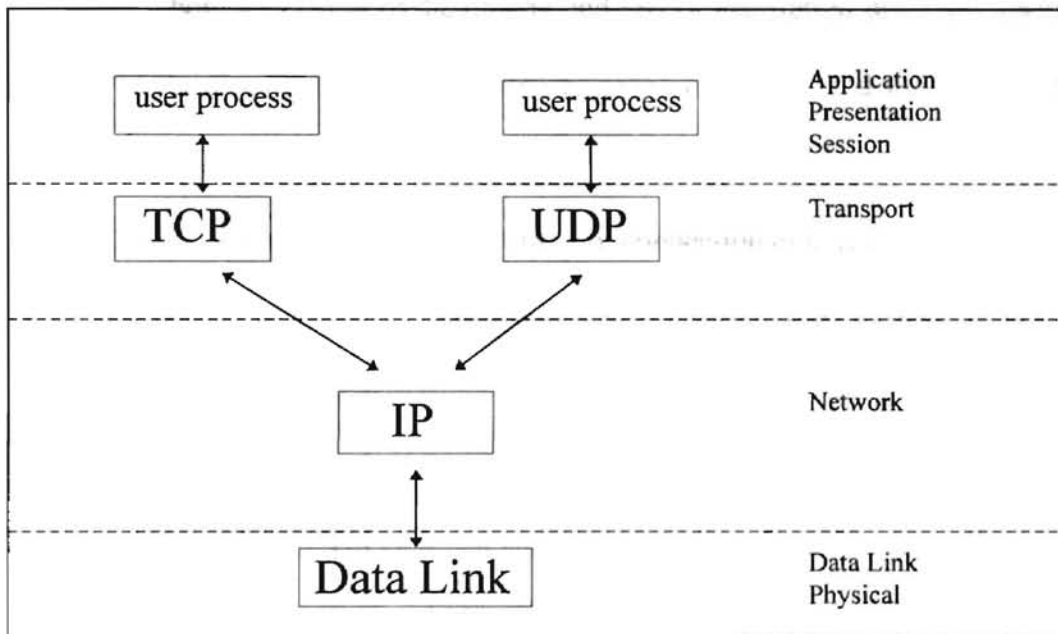


Figure 4. Four layer Model [3]

The components of Figure 4 are as follows:

**User Process** An application such as telnet or ftp running on a host.

**TCP** A connection-oriented protocol for user processes that provides a reliable, full-duplex (i.e., simultaneous communication in both directions) byte stream for a user process. Before remote hosts exchange data, a connection must be established. TCP provides service, which enables



processes on remote hosts to talk as if the processes had a point-to-point link. That is why TCP is connection-oriented. Reliability is achieved by maintaining the checksum of the TCP packet, and acknowledging every uncorrupted packet received. If a packet becomes corrupted during the transmission, the packet is discarded and no acknowledgment is sent. The byte stream service comes from the fact that no record markers (special symbols which signify beginning and end of transferred data) are inserted by TCP. One end puts a sequence of bytes into TCP, and the same sequence arrives at the other end.

**UDP** (User Datagram Protocol) A connectionless, unreliable protocol for user process. There is no warranty that a datagram (i.e., a packet of data) will reach its destination.

**IP** Protocol providing packet delivery for TCP and UDP. A further discussion of this protocol will be given later.

**Data Link** It is also called Link Layer, or Network Interface Layer. This layer consists of a network interface card and corresponding device driver. The Data Link layer takes care of physical interfacing with the wire and transmitting data in the form of bits over the wires.

When a user application on one host communicates with a user application on a remote host, the application of the sending user passes data down to the transport layer either to TCP or UDP. The transport layer passes the data down to the network layer. The network layer itself passes data down to the link layer. This process of passing the

data down the layers goes on until a stream of bits crosses the network, arriving at the remote host (Figure 5b). Each layer receiving data from the layer above puts a header in front of the data it receives. As is depicted in Figure 6, an IP datagram consists of two parts: header and body. Thus, the body of IP datagram is composed of the TCP header and the TCP data payload. This process is called encapsulation (Figure 5a). IP receives from TCP a *TCP segment*. IP itself passes an *IP datagram* down to the Link Layer. The stream of bits flowing across a network is called a *frame*.

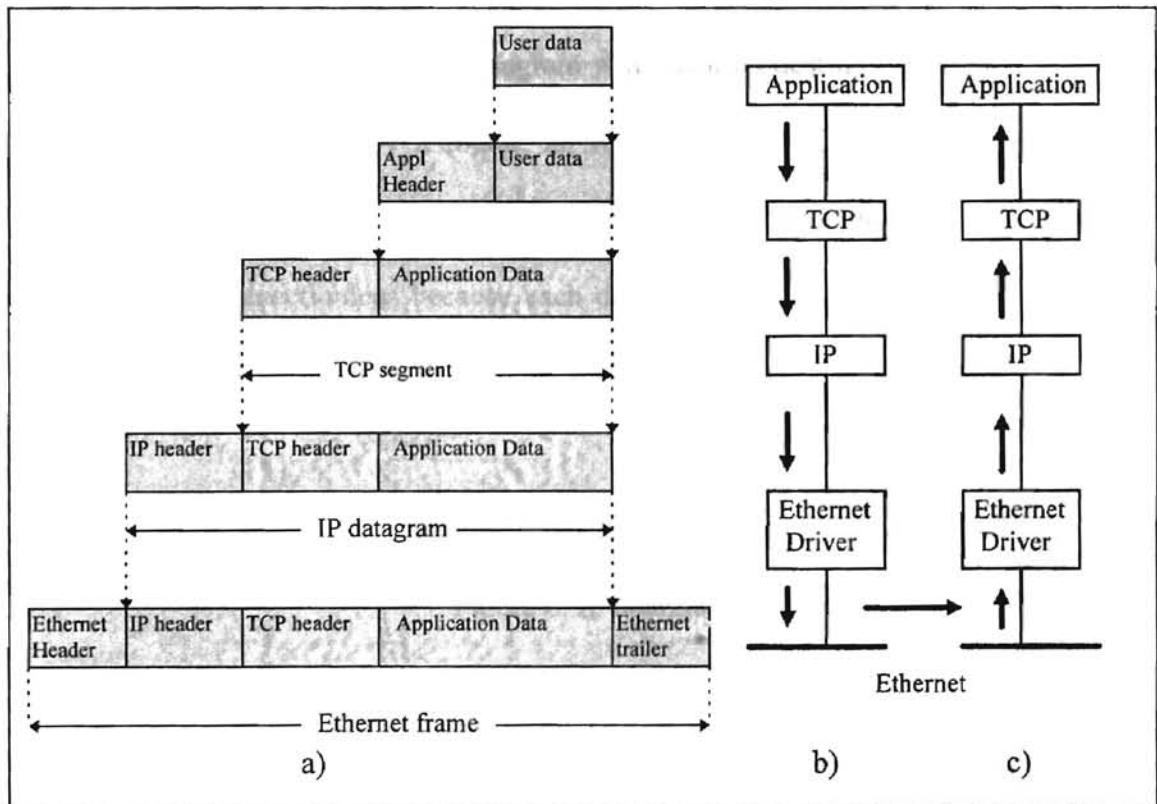


Figure 5. Encapsulation of data going down the protocol stack [21]

At the destination host, the Link Layer frame goes all the way up to the Application layer (Figure 5c). Headers are removed at each layer by a corresponding protocol, in an order reverse to which the headers were added. To determine which of the programs of upper layer the data should be forwarded to, at every layer, each protocol looks up the corresponding field of its header. This process is called *demultiplexing*.

### 2.3 Internet Protocol

IP, in its essence, is an unreliable connectionless protocol. Unreliable means that there are no guarantees that an IP datagram will reach its destination successfully. If an IP datagram is discarded by a router, an ICMP (Internet Control Message Protocol) message is sent back to the source. The layers above IP provide reliability of the data transfer. It is connectionless because each datagram is delivered independently. Each datagram consists of two parts: header and body (Figure 6).



Figure 6. IP datagram

The IP layer routes datagrams through the Internet. It also controls fragmentation, i.e., if a router receives an IP datagram that is too big to be transmitted, there is an option for the router to break the datagram into smaller datagrams and continue transmission [21].

## 2.4 Internet Addressing

An Internet address in IPv4 is 32 bits long. Every host on an IP network must have a unique host address consisting of a network part and a host ID. There are five types of Internet addresses. Each of the four bytes of an address represents a binary equivalent of a decimal number. These four decimal numbers are combined in dotted-decimal notation to represent an IP address (Table 1). By looking at the first number in an address, one can distinguish the class of the address [21]. There are five address classes as depicted in Table 1 and described below:

Ranges of IP addresses within classes		
A	0.0.0.0	127.255.255.255
B	128.0.0.0	191.255.255.255
C	192.0.0.0	223.255.255.255
D	224.0.0.0	239.255.255.255
E	240.0.0.0	247.255.255.255

Table 1. Ranges of IP addresses within classes [20]

**Class A:** Used by networks which have many hosts. In the Class A address only the first 8 bits are allocated for the network ID, with the first bit set to zero. This leaves 24 bits for the host ID. So the total number of hosts on a class A network can be  $2^{24}$ .

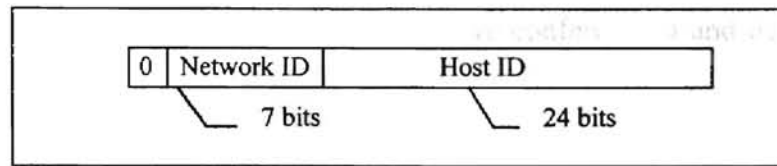


Figure 7. Class A IP address [20]

**Class B:** Used for networks smaller than Class A. The network part occupies 16 bits for ID, with the first two bits equal to '10', leaving 16 bits to be allocated for the host. Campus and Corporate networks are usually ideal candidates for Class B.

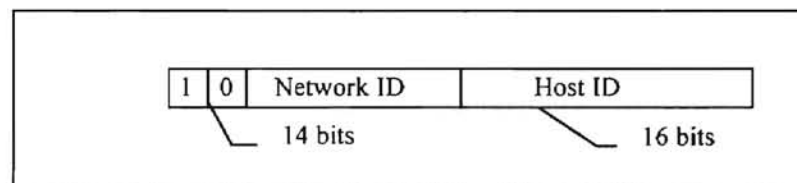


Figure 8. Class B IP address [20]

**Class C:** Used for small networks with a limited number of hosts. The network part is 24 bits, with the first 3 bits equal to '110', and the remaining 8 bits are allocated for the hosts on the network.

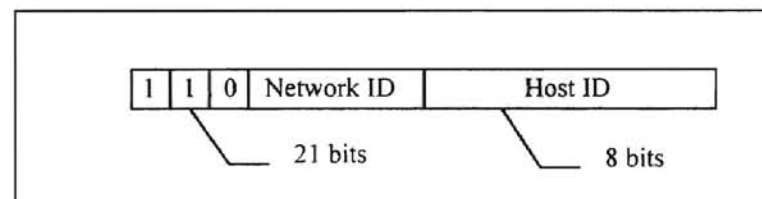


Figure 9. Class C IP address [20]

**Class D:** Multicast Addressing. Used for delivery of datagrams to multiple destinations. Example include interactive conferencing and delivery of news to multiple subscribers.

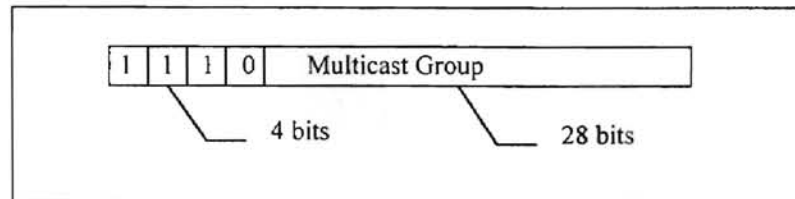


Figure 10. Class D IP address [20]

**Class E:** Reserved for future use.

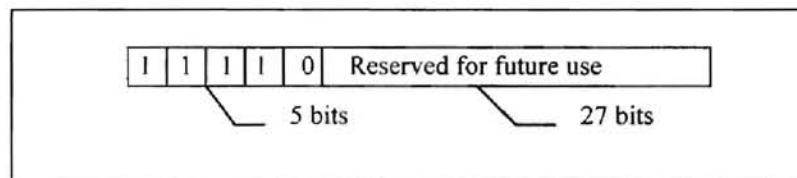


Fig 11. Class E IP address [20]

The information enclosed in a datagram header includes a source address and a destination address, so every datagram may be delivered independently. The datagram is similar to an envelope, on which one would write two addresses: the addressee and the return address. IP is unreliable because there is no mechanism in the IP itself to retransmit a particular datagram, and an acknowledgment is neither sent nor received on delivered datagrams. There is a checksum of the header to catch possible errors during the

transmission. When an error is detected in a datagram, that datagram is discarded. The body of an IP datagram provides information to the upper layer protocols. The data payload of IP is passed on so the transport layer protocols, TCP or UDP, may compose a whole message from incoming datagrams. When packets are arriving too fast from a host, IP sends ICMP (source quench) message to suspend sending packets. This is a primitive form of slow control in IP called “hop-by-hop control” [5].

## 2.5 Transmission Control Protocol

TCP provides a reliable and connection-oriented service to the application level. Due to the fact that IP provides unreliable, connectionless data flow for TCP, TCP itself has the mechanisms to ensure the reliability of delivering the datagrams. The mechanisms ensuring the reliability include the checksum of the TCP header and data, acknowledgment on the received datagrams, and the flow control.

The purpose of the checksum is to detect possible modifications of the TCP segment during transmission. If a TCP segment arrives at its point of destination with invalid checksum, the TCP discards it and the TCP segment is not acknowledged. The checksum ensures that a corrupted TCP segment gets detected by the remote host.

If the checksum is valid, an acknowledgment is sent by TCP to inform the remote end that data has been received. This way the remote end knows that the TCP segment arrived successfully and uncorrupted. In case the checksum is invalid, the datagram is discarded and no acknowledgment is sent. The TCP layer sends a segment and waits for the other end to acknowledge it. If no acknowledgment is received during a certain period of

time, then the segment is retransmitted. By doing this TCP guarantees that all segments reach the remote end. This makes TCP a reliable connection.

The flow control is provided by maintaining a buffer of a finite size. A sending TCP process sends as much data as the receiving end can accommodate in the buffer. This way remote hosts with a different speed of transmission can communicate without loss of data. If a buffer overflow occurs, the receiver informs the sender to suspend the data transmission. As soon as the buffer clears, the receiver requests to resume the transmission.

Each TCP header has source and destination port numbers, which signify the sending and receiving applications. A stream of bytes flows across the TCP connection. A sending application might send 100 bytes followed by 50 bytes, but the receiving application might read the 150 bytes in three reads of 50 bytes each. TCP does not interpret the content of the transmitted bytes. Only the application above the TCP interprets the byte stream coming from the TCP. A TCP header contains a sequence number identifying the order and number of bytes in the stream of data. This way TCP can reconstruct the original message in proper order. A detailed description of TCP can be found in the RFC 768 [14].

## 2.6 Problems with IPv4

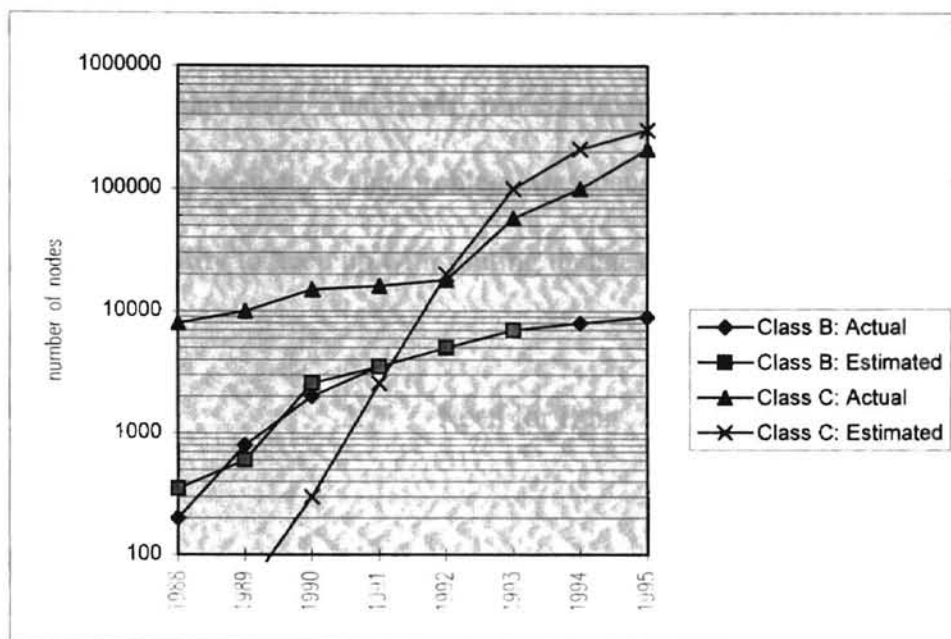
At the time IPv4 was developed, it was big enough to provide address space to all computers on connected networks. Over time, with the growth of the Internet, the available address space has been diminishing at alarming rate. IPv6 was introduced as a



solution to the existing problem. Besides the current features offered by IPv4, IPv6 contains other new features. Some of the new features of IPv6 include support for real time flows, provider selection, host mobility, end-to-end security, expanded routing, and addressing capabilities.

### 2.6.1 Problems with Class B addresses

In the beginning of 1990, some Internet experts became worried about the manner in which IPv4 addresses were assigned. At that time, about 20% of the available Class B network numbers were already allocated. The rate of growth of Class B networks was exponential already. The number of Class B networks was doubling every 14 months. At that rate, the Class B addresses could have been depleted by the end of 1994 [19]. An explanation for heavy demand for the Class B network numbers is presented by Frank Solensky in "IPv4 address Lifetime Expectations" [19]. Solensky says that most organizations estimating the future size of their network would assume that a network of 256 nodes is too small and, "a network over 16 million nodes would far exceed their most ambitious plans, so a Class B network number would be *just right*" [19].



Graph 1. Assigned and Allocated Network Numbers [19]

It was obvious that drastic measures had to be taken. To avoid depleting the Class B address space, NIC (Network Information Center) severely limited the assignment of these addresses to those organizations that could clearly prove the need for it; otherwise, an organization was given a block of Class C network addresses [19]. As a result of the action, the higher utilization of IPv4 addresses was achieved. As it shown on Graph 1, the growth of Class B network numbers was slowed down with an simultaneous increase of Class C network addresses.

### 2.6.2 Routing Tables Explosion

The routers are another source of the problems with IPv4. With the growth of the Internet many routers start to perform poorly and require memory upgrade [1]. The

resides on a router, is consulted to make a routing decision. Routers have more global view of the network than hosts. That is why routers run a special routing protocol to keep their routing tables up-to-date. The problem was caused by limited memory on the routers and lack of ability to keep up with routing updates.

## 2.7 Need for IPng

A wide use of PCs over past twenty years is the primary reason for the exponential growth of the Internet. The PCs have become the major consumers of Internet traffic. The endpoint computers of Internet communications range from PCs to Supercomputers.

Nowadays, most of the Internet connections are established via leased lines and dial-up connections. However, new technologies are being developed. Eventually these technologies will offer an alternative way of connecting to the Internet. It is reasonable to assume the growth of the Internet continues with the advancement of such technologies as wireless communications and high definition television. In their article "IPng, Internet Protocol Next Generation," Scott Bradner and Alison Mankin consider the development of several "new markets" where IPng can possibly find its use [1]. These markets are nomadic personal computing, networked entertainment, and device control, as described below.

Bradner and Mankin state, "nomadic personal computing devices seem certain to become ubiquitous as their prices drop and their capabilities increase" [1]. One of the key capabilities for such devices is the ability to be networked. These devices will use various forms of physical connections ranging from radio frequency to physical wires. It

will be imperative for the Internet protocol to support auto configuration, mobility, and of course, authentication. The protocol has to have very little overhead. This requirement comes from the fact that, unlike LANs with a very high speed, the wireless media is slower because of the constraints imposed by available frequency spectrum allocation, error rates and power consumption.

“Networked entertainment“ can utilize IPng too. With the advent of “high definition TV”, television will be taking over more and more functions of a PC and it is quite possible that a TV set will be a true Internet host in the future. A new TV market will need a routing protocol which supports routing and addressing on a large scale and auto configuration [1].

Device control is another area where IPng can be exploited. In the environment where sensors and device controls regulate household utilities, as well as industrial equipment, it will be useful to have a network protocol that can collect statistics and monitor the settings on control devices. The economic effect is obvious -- higher utilization of the equipment.

## CHAPTER III

### SPECIFICATION OF IPV6

#### 3.1 IPv6 Design

The design of IPv6 was based on the fact that IPv4 did not have any major flaws; the success of the Internet is proof of this. The designers of IPv6 decided to keep most of the characteristics of IPv4 [5].

##### 3.1.1 Comparison between IPv4 and IPv6

In addition to expanding the size of the IP address, the designers introduced several changes into the IP header. The new header is in fact much simpler than that of the classic IP.

Version	Priority	Flow Label	
Payload Length		Next Header	Hop Limit
Source Address			
Destination Address			

Figure 12. The IPv6 Header [5]

In comparison with IPv4 (Figure 13), the IPV6 header (Figure 12) has only six fields and two addresses, whereas the old IP has ten fields, two addresses, and some options. The presence of options makes the length of a IPv4 header variable.

Version	IHL	Type of Service	Total Length	
Identification		Flags	Fragment Offset	
Time to Live	Protocol	Header Checksum		
Source Address				
Destination Address				
Options			Padding	

Figure 13. The IPv4 Header [5]

For the transition period, IPv6 and IPv4 are planned to be used simultaneously on the same local networks. The network programs are expected to determine how a packet should be processed, by looking up the version field. Certain fields of the IPv4 header did not get transferred to the IPv6 header. These are header length (IHL), type of service, the identification, the flags, the fragmentation offset, and the header checksum.

### 3.1.2 Simplifications

Over a period of twenty years, since the time when IP was developed back in the 1970s, experience with IP has shown that the options field (since the options are rarely used) could be removed from the header. This does not mean that it is impossible to send the options field along with a packet on the network [5]. IPv6 “extension headers” are used instead. This way, it is possible to keep the length of the IP header fixed.

In IPv6, to speed up header processing of a router, the length of a header is kept fixed and the checksum is removed. As one of the developers of IPv6, Huitema, says, “removing the header checksum may seem a rather bold move. The main advantage is to diminish the cost of header processing, because there is no need to check and update the checksum at each relay” [5]. Nowadays, the connection lines have become more reliable, reducing the risk of a packet being corrupted during transmission. Also most of the encapsulation techniques have a packet checksum. Examples of such are IEEE standard 802.2, the Ethernet, and modem protocols like V.35.

Fragmentation is not an option in IPv6. In IPv4, large packets could be divided into smaller ones when needed in case a transient network is not able to handle big packets. The destination host waits for all the parts of the divided packet to arrive to proceed with the processing of the packet. For the successful transmission of a packet, all the parts of the divided packet must arrive at to their destination successfully. If any part of the divided packet is missing, the whole packet is retransmitted.

In IPv6, before a packet is put on a network, its maximal size is determined using “path MTU (Maximum Transmission Unit) discovery.” The MTU discovery allows the sender to determine the maximum packet size that can be sent from a source to a destination without a fragmentation of along the way. This feature also alleviates the load on routers, where packets can be forwarded without worrying about size. As a result of getting rid of the fragmentation, the packet identification, segmentation control flags, and the fragment offset become unnecessary.

In accordance with the standard, all IPv6 networks must be able to handle packets of “the minimum supported packet size of 576 octets” [5]. In case a source host is not

willing to discover the MTU path, the host simply can send a packet of 576 bytes. The payload length field in IPv6 is the replacement of the total length field in the IPv4 header. The difference is that payload length actually shows the length of the data carried in the packet. For a packet with a TCP payload of 20 bytes and 400 bytes of application data, the payload length will be 420. The length of the payload field is 16 bits, which limits the payload to 64 kilobytes.

The next header field in the IPv6 header provides a way to include an option header. If there are no option headers included, this field will be set to the value of either TCP or UDP. This way it is possible to include extension headers between the IP header and the TCP or UDP payload.

The “hop limit” field is the replacement of the TTL (time to live) field of the IPv4 header. The difference is that in IPv4, TTL is shown in seconds, indicating how long a packet can remain on a network. In the IPv4 specifications, TTL is decreased by one second by each router passing the packet, with disregard to how many seconds it took a routers to forward the packet. The research conducted by developer of IPv6 has shown that it is hard to calculate the actual time a packet remains on a network. That is why the “hop limit” field is supposed to reflect the travel distance (complexity) of a packet from source to destination.

The new fields introduced are the “flow label” and “priority.” The “flow label” may be used with a routing extension header. This will tell a router to treat the packet in a specific way. A flow is a collection of packets coming from the same source to the same destination. By setting a flow label, a sending host expresses special handling by an intermediate router.



### 3.1.3 Extension Headers

IPv6 gives the ability to insert extension headers between the payload and the IP header. Extension headers are linked in a “daisy chain.” Each of the extension headers appears at most once in the chain and carries information about header type, options, and the type of the next header or the type of the payload, if this turns out to be the last header in the sequence of extension headers.

Presented in Figure 14, are examples of the chains of Option Headers:

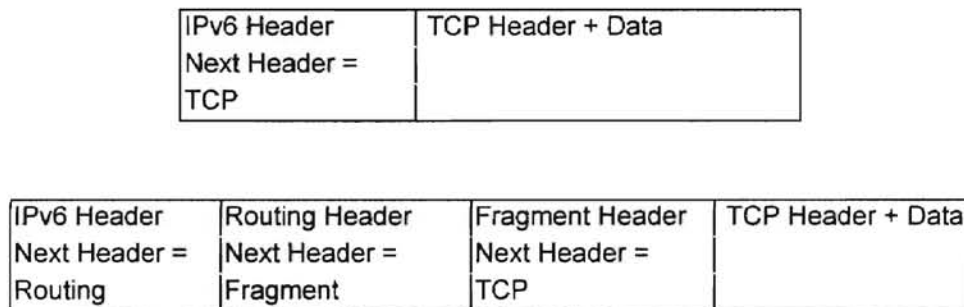


Figure 14. Examples of Chains of Headers [5]

By the specifications of IPv6, the order of extension headers is as follows:

1. *Hop-by-Hop options header*
2. *Destination options header*
3. *Routing header*
4. *Fragment header*
5. *Authentication header*
6. *Destination options header*
7. *Upper-layer header (for example , TCP or UDP)*

The Hop-by-Hop option header is designed to provide a mechanism to carry additional information. This information is checked by every node along a packet's deliver path. The information may be used for handling large packets, "jumbograms" [7].

In the Destination header option, some options provide additional information on the packets context or preferences. This information is examined by a packets destination node. This header may be used in conjunction with the "Fragment header" and the "Authentication header" to provide information of "fragment number" and origin of packet [5].

The Routing header (Figure 15) is the best example of using extension headers in IPv6. This header is a list of intermediate nodes that should be visited on a packet's way from source to destination.

Next Header	Routing Type=0	Num address	Next Address
Reserved	Strict / Loose Bit Mask		
Address[0]			
Address[1]			
.....			
Address[Num Addr -1]			
Address[Num Addr]			

Figure 15. Routing Extension Header [5]

Exploiting the routing header of IPv6 achieves higher performance results in comparison with the IPv4 source routing option. Unlike IPv4, where the IP header is checked by every intermediate node in the "source route," the IPv6 routing header is examined by a station if the destination address equals the station address. If this is the case, the station either continues forwarding the packet if the path complies with the

sequence indicated in the list of addresses or passes the payload to the upper layers if the station itself is the final destination of the packet. When the station continues forwarding the packet, it swaps the destination address with an address in the address list and increments the pointer "next address" in the list of addresses. The possible use of the extension header may be demonstrated by the following examples.

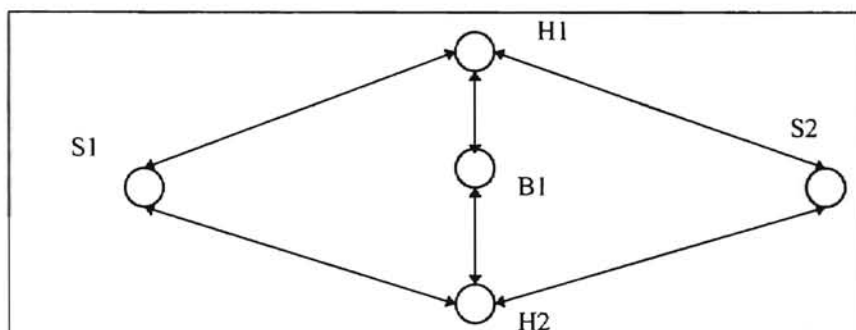


Figure 16a. Routing Scheme. Host S1 is connected to H1

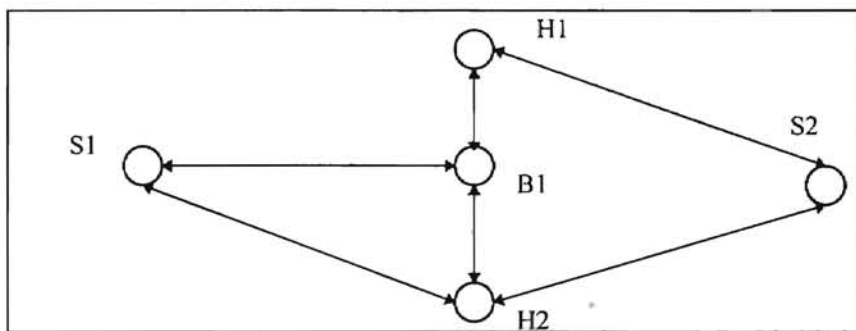


Figure 16b. Routing Scheme. Host S1 is connected to B1

By creating a sequence of IPv6 addresses, the new routing functionality gives directions on how a particular datagram should be routed. When sending a packet from S1 to S2 through H1 (Figure 16a), one would specify the sequence S1H1S2, which would

tell the routing module of IP to use H1 on the way to S2. In case S1 changes its location (mobility) from H1 to B1 (Figure 16b), and one would still like to impose the restriction of traveling through H1, the routing option sequence would become S1B1H1S2. To send a reply, S2 should reverse the sequence which then becomes - S2H1B1S1. This ensures the route will go through H1.

Suppose a network has two connections to the Internet, one primary and the other secondary. The communication links are charged based on the amount of traffic moved. Assume that on a primary connection the rate in the morning is very low and increases in the afternoon. The opposite is true for the secondary link, the rate is high in the morning and low in the afternoon. A network manager may take an advantage of the tariffs and route. For example, a multimedia traffic may travel through the primary link in the mornings, and use the secondary link in the afternoons [1].

The Authentication header is a mechanism that ensures that a packet delivered to a recipient is genuine and has not been altered during the transmission.

The Fragment head is used as a provision in IPv6 to send a data packet on a network when the size of a packet is larger than the MTU of the path from source to destination. The Fragment header has enough information to reassemble the fragments at the destination node [5].

### 3.2 Routing and Addressing

With an expanded address space from 32 to 128 bits, IPv6 allows a more advanced degree of hierarchy. In general the IP address identifies an interface connected to a

network, but not a workstation. In IPv6, an interface can have several addresses to facilitate routing or management [17]. There are three categories of IPv6 addresses: Unicast, Anycast, and Multicast. In the RFC 1884 [16], these addresses are defined as follows:

*Unicast:* An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address.

*Anycast:* An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the "nearest" one, according to the routing protocols' measure of distance).

*Multicast:* An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address.

The difference between multicast and anycast is that a multicast packet is delivered to a single unicast address, which is the nearest member of the group, but not to all members of the group [16].

### 3.2.1 IPv6 Address Format

It is possible to represent an IPv6 address in one of the three possible forms. The most common way is x:x:x:x:x:x:x, where "x" is a four digit hexadecimal. So, there are eight 16-bit pieces. For example, the following are examples of addresses in this notation:

5F13:D600:8B4E:7200:0072:0060:8C39:B21A

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

For the sake of convenience, it is not necessary to write the leading zeros; thus, the first address may be reduced to :

5F13:D600:8B4E:7200:72:60:8C39:B21A

There may be addresses with long strings of zeros. The use of the “::” notation make the representation of such addresses easier. The “::” can appear only once in an address.

The following addresses [16]:

1080:0:0:0:8:800:200C:417A	a unicast address
FF01:0:0:0:0:0:0:43	a multicast address
0:0:0:0:0:0:0:1	the loopback address
0:0:0:0:0:0:0:0	the unspecified address

may be represented as:

1080::8:800:200C:417A	a unicast address
FF01::43	a multicast address
::1	the loopback address
::	the unspecified address

The third form of the addresses is used in an environment where both IPv4 and IPv6 nodes are present. The notation resembles the following:

x:x:x:x:x:d.d.d.d

With “x” being a hexadecimal value, and “d” being a decimal number in the range  $0 \leq d \leq 256$ .

The use of a *Format Prefix* identifies a specific type of IPv6 address. The Format Prefix (FP) is a variable length field, consisting of the leading bits of the IPv6 address. For example, link local addresses (used to address tunnels) start with

FE80

To setup a tunnel one would use:

FE80 :: (IPv4 endpoint address)

The complete table of initial allocation of the Format Prefixes can be found in RFC 1884 [16]. With the current scheme of the allocation of IPv6 addresses, only about 20% of the total address space has been allocated with the remainder of the address space left for future use. The use of Format Prefixes permits network devices such as routers and hosts to recognize and treat different groups of addresses differently. For example, the multicast address should be treated differently from a unicast address [5].

### 3.2.2 Provider Address

The first true IPv6 addresses will be allocated according to a provider based plan. In this plan, the provider based address consists of six fields, as shown in Figure 17:

3 bits	m bits	n bits	o bits	p bits	128-3-m-n-o-p bits
010	registry ID	provider ID	subscriber ID	subnetwork ID	interface ID

Figure 17. Generic Format of Provider Addresses [16]

At this time, a temporarily allocation plan is being used. Any IPv6 system using this plan will have to renumber its addresses some time in the future [18]. The address format for IPv6 test addresses shown in Figure 18.

3 bits	5 bits	16 bits	8 bits	24 bits	8 bits	16 bits	48 bits
010	11111	ASN	RES	IPv4 Net Addr	RES	Subnet Addr	Intrf ID

Figure 18. The Initial Structure of a Provider-Based Address [16]

The components of the Figure 18 are as follows:

**010** Format Prefix used to identify provider based unicast address

**11111** This is the registry ID given temporarily by IANA (Internet Addressing and Naming Authority) for testing.

**ASN** Autonomous System Number. The ASN numbers are assigned to the Internet Service Providers (ISP) that provide services for the IPv6 testers organizations. The values for the autonomous system numbers of an ISP can be found in the database maintained by internic.net at URL <http://www.internic.net>

**RES** Reserved and must be set to zero

**The IPv4 Network Address** is formed by taking the high 24 bits of the IPv4 address. For example, the address 139.78.114.2 in binary notation is:

10001011010011100111001000000010

the high 24 bits is



100010110100111001110010 The type of address

which in hexadecimal notation

8B4E72

**Subnet Address** - Assignment of this field is up to an individual subscriber. It is possible to use bits of an IPv4 address which identify an IPv4 subnet.

**Interface ID** - 48 bits IEEE 802 MAC Address. Hardware address of an Ethernet Attached Device.

### 3.2.3 Special Address Formats

The unicast addresses can be subdivided into five types: unspecified address, loopback address, IPv4-based address, site local address, and link local address, as described below.

**Unspecified address:** Consists of a string of zeros (to be more exact , “16 null bytes”). The notation “: :” is used for 0:0:0:0:0:0:0. This type of address can be used as a source address by a station that has not yet been configured with a regular address.

**Loopback address:** 0:0:0:0:0:0:1. May be used by a node to send a packet to itself. The loopback address and the unspecified address may never be assigned to an interface [18].

**IPv4 based address:** Constructed by preceding a prefix of 96 zero bits to the 32 bit IPv4 address. These addresses are represented by a “double colon” followed by a “dotted decimal” notation. For example, an IPv4-based IPv6 address for a

host with the address 139.78.114.2 will be ::139.78.114.2. This type of address is expected to be used during the transition period from IPv4 to IPv6.

**Site local address:** Constructed by using FEC0 prefix. These addresses are reserved for organizations using TCP/IP without actually being connected to the Internet. A site local address has the following format as shown in Figure 19:

8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	16 bits	48 bits
FE	C0	0	0	0	0	0	0	0	subnet	station ID

Figure 19. Structure of a site local address [16]

Site local addresses are not routed on the Internet. The uniqueness of a site local address has to be ensured within the network of an organization.

**Link local address:** Used for stations that are not yet configured with either a provider-based address or a site local address. These addresses also are the site local prefix. This type of address has the format shown in Figure 20:

8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	48 bits
FE	C0	0	0	0	0	0	0	0	0	0	station ID

Figure 20. Structure for a link local address [16]

The difference between site local address and link local address is that packets with site local destination addresses will be routed within a domain, but packets with link local destination addresses will not leave the boundaries of a LAN. The scope for these addresses is restricted by a common link to which the stations are connected.

## CHAPTER IV

### TRANSITION TECHNIQUES AND PROBLEMS CAUSED BY THE TRANSITION

#### 4.1 Introduction

Due to the heterogeneous structure of the Internet, it is impossible to think of simultaneous transitions of all the networks composing it. To ensure a smooth transition, the coexistence of IPv4 and IPv6 must be an essential part of the transition techniques. These techniques include address mapping from the old version to the new one, support of coexistence for both versions, and name service change [17]. There will be three types of addresses during the transition period:

**IPv4 host:** A host or router that is IPv4-capable only. It does not understand an IPv6 address.

**IPv6/IPv4 host:** A host supporting both IPv6 and IPv4

**IPv6 host:** A host or router that is IPv6-capable only. It does not understand an IPv4 address.

## 4.2 Dual Stack

IPv6 addresses are divided into two categories: IPv4-compatible and IPv6-only. An IPv4-compatible IPv6 address is an address constructed by appending a 96-bit string of zeros in front of a 32 bit IPv4 address, as described in Section 3.2.3.

This type of address is designed to be used for tunneling. An IPv6-only address has its higher 96 bits different from 0:0:0:0:0:0.

## 4.3 Tunneling

Tunneling becomes the primary technique used in the transition. Tunneling is a mechanism that allows the IPv6 packets to travel over IPv4 networks by encapsulating IPv6 packets within IPv4 packets. This way, IPv6 packets are carried as a data payload of IPv4 packet, as illustrated in Figure 21:

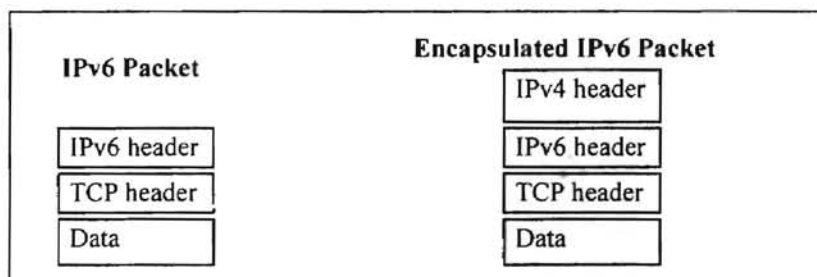


Figure 21. Encapsulation of IPv6 over IPv4

When an encapsulated packet gets to the end of a tunnel the IPv4 header is stripped off (Figure 22). This process is called decapsulation.

There are two types of tunneling: configured and automatic tunneling. Configured tunneling implies that the IPv4 address of the tunnel endpoint is determined from the configuration, i.e., the routing table, of the encapsulating node. In automatic tunneling, the tunnel endpoint is determined by an IPv4 address presented in an IPv4-compatible IPv6 address. In accordance with this, packets that are not addressed to IPv4-compatible nodes cannot be forwarded using automatic tunneling.

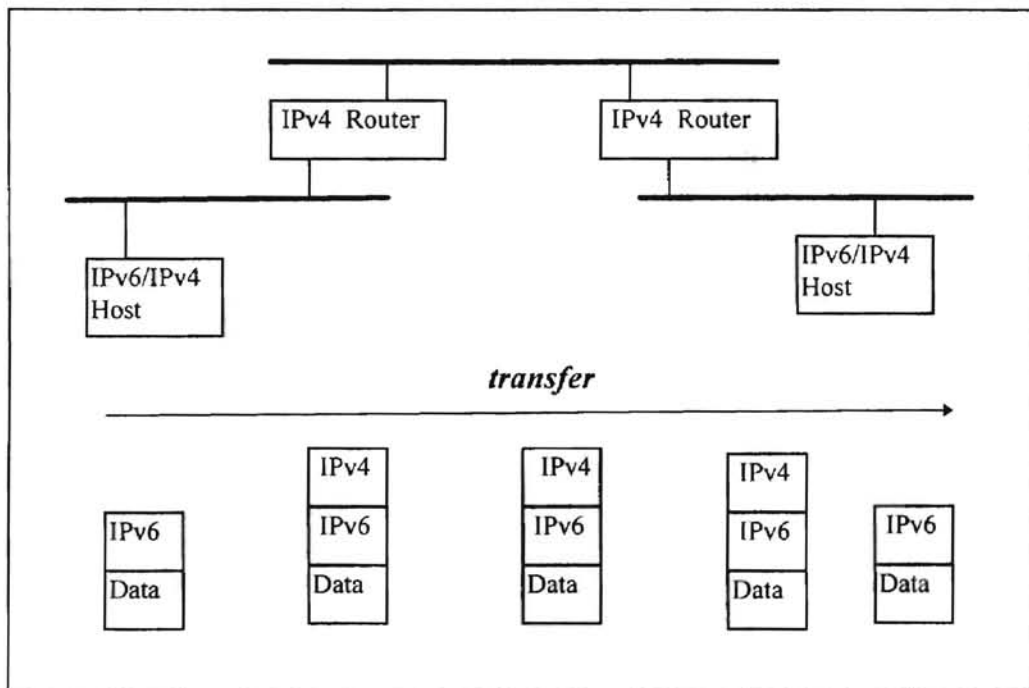


Figure 22. Tunneling [3]

For example, consider entries from the routing table of one of the test machines on which IPv6 over IPv4 tunneling was used:

FE80::8b4E:729A/10                   ::139.78.114.154

5F00::0/8                            ::139.78.114.154

The '/10' and '/8' suffixes in both lines signify the bit mask. The bit mask tells that all packets with the destination addresses matching the first n bits should be forwarded to the specified interface. The first line explains that to get to the IPv6 node FE80::8B4E:729A/10, a packet should follow a tunnel with the endpoint 139.78.114.154. The prefix 'FE80::' denotes a link-local address. The second line shows that any address starting with the 8-bit value as '5F' should be forwarded to the endpoint ::139.78.114.154.

IPv4/IPv6 nodes are assigned both IPv4 and IPv6 addresses. These addresses may be related, though it is not necessary. For the nodes that use automatic tunneling, the lower 32 bits of 128 bits of IPv6 addresses represent the IPv4 address.

#### 4.4 Problems with IPv6

Obviously, IPv6 is the protocol of the future. It supplies the very features that lack development in the current version of IP. These features include expanded addressing space, real time application support, security, etc. There are several factors that are must be taken into consideration, while migrating to an IPv6 base. These factors are TTL in tunnels, application interface, and bandwidth of demand.

##### 4.4.1 TTL in Tunnels

By setting up IPv6 software on local computers and installing an IPv6 router, it is possible to create an IPv6 network on a LAN. An IPv6 domain can be created by connecting several IPv6-capable LANs. Whenever this kind of "IPv6 island" needs to

communicate with another IPv6 LAN via an IPv4 segment, tunneling is used to establish the link.

It is certain that the ability of two remote IPv6 domains willing to talk to each other via a tunnel depends on the control parameters of the tunnel. The MTU becomes a key factor in this issue. According to the specifications of IPv4, the minimum packet size is 48 bytes. For IPv6 the smallest value is 576 bytes. In case the IPv6 packet size is bigger than the MTU of the tunnel, an IPv6 application will get an ICMP message that “a packet is too big.” There are three possible scenarios of transmitting an IPv6 packet over an IPv4 tunnel:

1. The IPv6 packet is not bigger than the MTU of the tunnel; the packet is sent without any fragmentation.
2. The IPv6 packet is bigger than the MTU, but less than 576 bytes. In this case, IPv4 is used.
3. The tunnel’s MTU is lower than 576 bytes. This means IPv6 packets of a size greater than 576 bytes will not be transmitted and consequently get dropped off. An error message will be sent to the IPV6 application generating the IPv6 packet.

To provide good performance, MTU discovery can be used. MTU discovery provides IPv6 users the best performance on IPv4 tunneling [5].



#### 4.4.2 Application Interface

Another possible problem with IPv6 is related to the application running on top of the IP layer. In applications, network addresses are handled at a low level. Some applications assume that the IP host address is a fixed 32 bits and store the addresses in a 32 bit variable. The current IPv4 socket structure in the API is 32 bits. It is clear that this structure should be different for IPv6. Modification to the programs could follow one of two possible ways. The easiest way is to introduce changes into the API header files and recompile the programs. The extreme scenario would involve heavy software modifications, requiring a great deal of code to be rewritten and debugged.

#### 4.4.3 Bandwidth on Demand

Also a potential problem that may appear with the wide deployment of IPv6, is when the users get an ability to reserve as much bandwidth as they need, called “bandwidth on demand.” If each of the users can reserve as much as he or she needs, it may happen so that their requests will swamp the capacity of the network. Some economists suggest “usage-sensitive” pricing. MacKie-Mason and Varian propose a method of “priority queuing” with regard to a pricing mechanism [10]. This way the users will be charged more for traffic requiring higher priority. This simple method will help to avoid the situation in which everybody’s traffic becomes high priority.

#### 4.4.4 Industry's Opinion

As Eric Fleischman in "A User's View of IPng" notices, "Computing is merely a tool to help automate business processes in order to more efficiently accomplish the business goals of the corporation. Automation is accomplished via applications. The central lesson of this point is that IPng will be deployed according to the applications which use it and not because it is a better technology" [1]. According to Fleischman, a Senior Principal Scientist (Network Architect) with Boeing Computer Services' Delivery Systems Architecture and Technical Planning group, every protocol family deployed on the network requires a certain training for support personnel. It is natural for a corporate administration to reduce a diversity of Data Communications Protocols and network support cost. That is why the introduction of IPv6 is viewed as an entirely new set of protocols, and will not be in a positive light because the work caused by the introduction of IPv6 will produce more work to support the computing resources [1].

### 4.5 Current Projects and Implementation of IPv6

#### 4.5.1 6Bone

As a result of the IETF (Internet Engineering Task Force) IPng project, 6Bone was created to construct an infrastructure to support the transmission of IPv6 packets in an

IPv6 and IPv4 network environment. At the present time, a great variety of projects involved in IPv6 take part in 6Bone. The 6bone is an informal collaborative project spanning over North America, Europe, and Asia. In its essence, 6Bone is a network consisting of many ISPs and networks receiving IPv6 traffic. It is a virtual network working on top of the existing Internet. Various islands supporting IPv6 domains are separated by IPv4 routers and to communicate with each other they use IPv6 over IPv4 tunnels [2].

The primary idea in creating 6Bone was to gain experience with IPv6. It is expressed on the home page of 6Bone that “as confidence builds to allow production routers to carry native IPv6 . . . the 6Bone will disappear. The 6Bone is thus focused on providing the early policy and procedures necessary to provide IPv6 transport in a reasonable fashion so testing and experience can be carried out” [2].

#### 4.5.2 University Projects

Various projects conducted at Universities around the world cover a great range of problems and applications involved with IPv6. Lancaster University, UK, has several concurrent projects directly dealing with IPv6. Researchers at Lancaster University say that they are investigating the “ways of improving support for multimedia streams within networks such as Internet” [22]. They claim that bringing the load of multimedia applications from endstations into the network results in a reduced workload on the endstations, and improved multicast group management. For example, consider a group of computers participating in a video conference. Each of the participants can provide

audio and video input to one server. The server distributes the audio and video feed to all the participants of the video conference. Thus, the server takes a certain portion of workload from a client system by distributing the video stream to all participants of the video conference. The research group is developing IPv6 routers based on Java. Mobil IPv6 is another area of interest to the group. The project dealing with Mobil IPv6 is designed to investigate how IPv6 can be used in applications for adaptive mobil systems. Mobil systems are characterized by frequent changes in a network during customers' relocation within a wireless communications domain [22].

#### 4.5.3 Industry Overview

At the present time, many vendors of network equipment and software companies conduct active work to implement products supporting IPv6. *FTP Software Inc.* was one of the first software companies that released software supporting such applications as Telnet and FTP in the IPv6 stack of DOS and MS-Windows machines. A complete IPv6 and IP security implementation for 4.4 BSD is built by the *US Naval Research Laboratory (NRL)* [7]. Implementations developed are designed to run on two types of platforms: *SPARC* and Intel based computers. All standard network applications (telnet/telnetd, ftp/ftpd, netstat, bind, ping multicast, etc.) are included in the implementation.

*IBM* has also announced support for *ALX* version 4.2. The prototype of IPv6 for *ALX* is established on the 4.4 *BSD* source code developed at the *National Research Institute of France, INRIA* [23]. This is an indication that the efforts of developing IPv6

implementations are international. The *INRIA* release was included in *AIX* version 4. Most of the features of IPv6 are supported in this product (a full list of the features of IPv6 supported in the release can be found in [23]). Also, authors of the IBM AIX web page indicate that there was interoperability testing between *AIX* IPv6 and *Solaris* IPv6 implementations, which “are able to do Neighbor Discovery, telnet, ftp, WWW, and ping over both Ethernet and tunneled IPv4 interface” [23].

Network equipment vendors also are promising delivery of products supporting IPv6. *3Com Corporation* says, “[3Com] will deliver IPv6 functionality across all its relevant products and platforms beginning in the second half of 1997, and continuing in 1998 and beyond” [4]. *Bay Network* equipment already supports such features of IPv6 as Neighbor Discovery, and full support of IPv4 to IPv6 transition mechanisms such as tunneling [4]. These efforts are evidence that industry believes that IPv6 will become the next logical step in development of the Internet.

#### 4.6 Bringing IPv6 on Campus

For the OSU network to face the advent of IPv6, it would be practical to start a pilot IPv6 network on campus. Building such a network will give an essential expertise in dealing with various IPv6 issues such as addressing, tunneling, DHCP (Dynamic Host Configuration Protocol) upgrade, handling DNS (Domain Name Service) entries, etc. The IPv6 capable islands and domains should be created on the least loaded subnets of the campus network to reduce the risk of service disruption. These islands can be connected by IPv6 over IPv4 tunnels. The implementation of IPv6 routers can be made using UNIX

7. Set up IPv6 DNS domain with AAAA record (AAAA records are needed so that queries by IPv6 hosts can be satisfied) [16]. Set up reverse DNS service.

## CHAPTER V

### LAB TESTS

#### 5.1 Building a Test Network

Using a tunnel introduces a certain amount of overhead in the data transfer. It was interesting to see the differences in the behavior of IPv6 and IPv4 during the data transfer process. To compare the performance between the IPv4 and IPv6, a test network using Linux running on PCs as hosts was set up. The set of application packages was obtained from the Internet (<http://www.terra.net/ipv6>) [8]. The detailed description of the system configuration of the PCs and the software is provided in Appendix E.

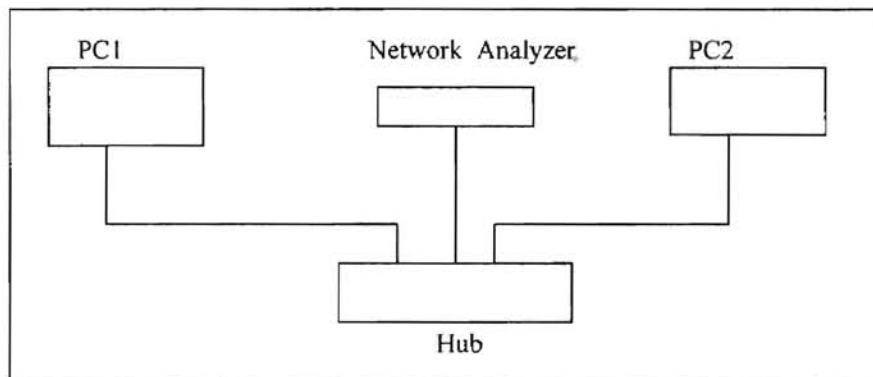


Figure 23. Test LAN Layout

## CHAPTER V

### LAB TESTS

#### 5.1 Building a Test Network

Using a tunnel introduces a certain amount of overhead in the data transfer. It was interesting to see the differences in the behavior of IPv6 and IPv4 during the data transfer process. To compare the performance between the IPv4 and IPv6, a test network using Linux running on PCs as hosts was set up. The set of application packages was obtained from the Internet (<http://www.terra.net/ipv6>) [8]. The detailed description of the system configuration of the PCs and the software is provided in Appendix E.

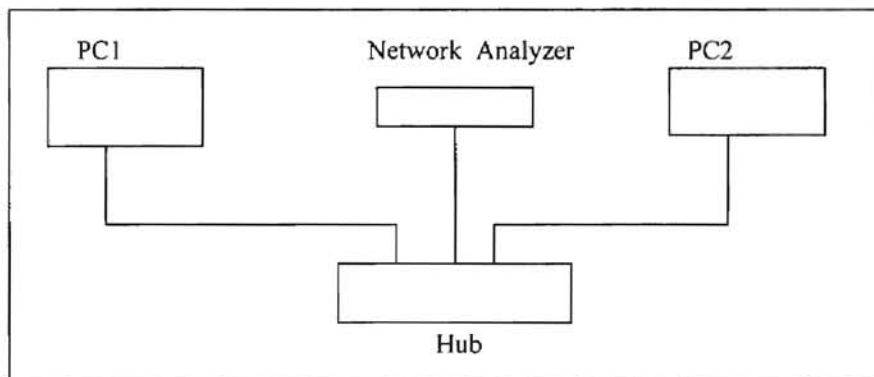


Figure 23. Test LAN Layout



As a measurement of performance, the amount of overhead imposed by IPv6 in comparison to IPv4 and the rate of the data transfer were chosen. The performance results are based on the statistic output produced by the FTP application, a program used to transfer files between two remote hosts.

For the subnet on which the test machines were located, the MTU was 1500 bytes, which means that the data payload of an Ethernet frame should not exceed 1500 bytes (Figure 24). This includes the IPv4 header plus the transport layer datagram.

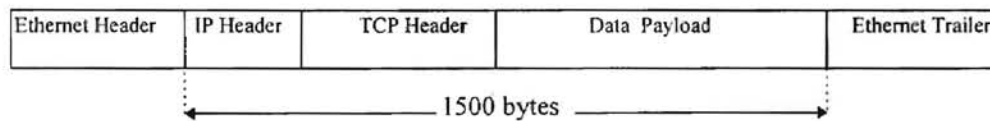


Figure 24. The Ethernet frame with MTU less than 1500 bytes

The testing was split into two parts. The first part was testing in a “back-to-back” configuration (Figure 23), and the second part was testing the connection on a shared network. “Back-to-back” means that the machines were located on an isolated LAN, and they were the only two hosts competing for the bandwidth on the segment. Being on a shared network means that besides the two machines there are other hosts “sharing” the 10 Mbits/s bandwidth. A list of the tests follows:

- Back-to-Back FTP using IPv4
- Back-to-Back FTP using IPv6
- Back-to-Back FTP using IPv6 tunneling over IPv4
- Back-to-Back FTP with one host being IPv4 only and the other IPv6/IPv4

- FTP on shared network using IPv4
- FTP on shared network using IPv6
- FTP on shared network using IPv6 tunneling over IPv4
- FTP on shared network with one host being IPv4 only and the other IPv6/IPv4

## 5.2 Address Translation

For IPv6-only addresses, construction of the address can be made in accordance with the procedures described in RFC 1887 (IPv6 Testing Address Allocation) [17]. The detailed description of how to “create an IPv6 address” can be found there. This research brings an example of how to create an IPv6 address that will be related to the addresses of the Oklahoma State University network. OSU uses Class B Network addresses.

**IPv4 Address:** 139.78.114.2  
**Hd Address:** 00:A0:24:95:D5:E4

Field Name	Value in Binary representation
Prefix	10111111
Reserved	00000000
ASN	0001001111010110
IP	100010110100111001110010
Subnet Address	0000000001110010
Reserved	00000000
MAC Address	00000000101000000010010010010101110101011100100

**Converting to hex:** 5f13d6008b4e7200007200a02495d5e4  
**IPv6 Address:** 5f13:000:d600:8b4e:7200:0072:00a0:2495:d5e4

Table 2. An Example Conversion of 139.78.114.2 into IPv6 address

The network number assigned to OSU is 139.78.0.0. The procedure of deriving IPv6 addresses for two machines on the OSU network, with the addresses 139.78.114.2 and 139.78.114.114, is shown in Tables 2 and 3. The IPv6 address format is 3 bits of Format Prefix followed by registry ID, provider ID, subscriber ID, subnetwork ID, and interface ID (refer to Figure 18, in Section 3.2.2). With this in mind, it is a fairly straightforward procedure to construct an IPv6 address. Given that the autonomous system number (i.e., the registry ID) of OSU's Internet Service Provider is 5078, the IPv4 address for one test machine is 139.78.114.2, and the interface address of the Ethernet card in that test machine is 00:A0:24:95:D5:E4, one can derive (see [17]) the values shown in Table 2. In a similar fashion, one can derive the values for the other test machine as shown in Table 3.

**IPv4 Address:** 139.78.114.154  
**Hd Address:** 00:60:8C:39:B2:1A

Field Name	Value in Binary representation
Prefix	10111111
Reserved	00000000
ASN	0001001111010110
IP	100010110100111001110010
Subnet Address	0000000001110010
Reserved	00000000
MAC Address	00000000110000010001100001110011011001000011010

**Converting to hex:** 5f13d6008b4e7200007200608C39B21A  
**IPv6 Address:** 5f13:000:d600:8b4e:7200:0072:0060:8C39:B21A

Table 3. An Example Conversion of 139.78.114.154 to IPv6 address

### 5.3 Configuring the Tunnel

Each of the test machines considers the other machine to be the end of an IPv6 over IPv4 tunnel. For example, the configuration file for machine 139.78.114.2 is the following:

```
# Your IPv6 prefix
PREFIX=5f13
# The host-part of the IPv6 address for this machine
ADDRESS=0:0:0:0:0:139.78.114.2      # IPv4 compatible IPv6 address
TUNNEL=139.78.114.154           # The IPv4 address of the far
# side of your tunnel
/sbin/ifconfig sit0 up           # Bringing up one end of the tunnel
/sbin/ifconfig eth0 add $PREFIX::$ADDRESS/80 # Assigning IPv6 address
                                     # to the Ethernet interface
/sbin/route -A inet6 add $PREFIX::0/80 dev eth0
/sbin/ifconfig sit0 tunnel 0::$TUNNEL # tunnel to outside
/sbin/ifconfig sit1 up
/sbin/route -A inet6 add 5f00::0/8 gw 0::$TUNNEL dev sit1
                                     #setting up default route
```

The IPv6 configuration file of the other machine is similar to the configuration of the first machine, with the exception that the address is set to:

```
ADDRESS=0:0:0:0:0:139.78.114.154  # IPv4 compatible IPv6 address
```

detailed configuration for both machines can be found in Appendix E.

## 5.4 Results and Analysis

The traffic was produced using an IPv6 compatible FTP application to move a file of 1 Megabyte. Below, in Tables 4a and 4b are the averaged results of the tests presented. The columns marked as "Mixed I" and "Mixed II" correspond to the "mixed environment." "Mixed I" corresponds to the configuration in which PC1 runs IPv4 stack of protocols only and PC2 runs dual stack (IPv6/IPv4). "Mixed II" is the opposite to "Mixed I": PC1 runs dual stack and PC2 is IPv4-only host.

Due to the fact that the test machines are not identical, the results of file transfer did not come out symmetrical. The test machine PC1 is based on an Intel 80486 Intel processor running at 66 MHz and PC2 is based on an Intel Pentium running at 75Mhz. Both machines used 3Com Ethernet Cards: model 3c509 for PC1 and model 3c59x Vortex for PC2. A detailed description of both machines can be found in Appendix D.

The performance results of file transfer using IPv4 in the "Back-to-Back" mode were chosen as the basis to which the results of the other tests are compared. Comparing columns one through three in Table 4a, one can observe that the rate of transfer in IPv6 and tunneling is about 25% less than in IPv4 ( $7.80E+05$  bytes/sec in vs.  $5.72E+05$  bytes/sec in tunneling and  $5.95E+05$  bytes/sec in IPv6 only). The results of tests obtained on a shared LAN (columns 6 through 8 Table 4a) demonstrated the performance degradation 41% and 38% for tunneling and IPv6-only respectively. The overhead imposed by tunneling itself was only 3% (the difference between 41% and 38%).

<b>FROM PC1 to PC2</b>	<i>Back to Back using IPv4</i>	<i>Back to Back using IPv6 over IPv4</i>	<i>Back to Back IPv6 only</i>	<i>Mixed I (Back to Back)</i>	<i>Mixed II (Back to Back)</i>	<i>IPv4 on Shared LAN</i>	<i>IPv6 over IPv4 Shared LAN</i>	<i>On Shared LAN IPv6 only</i>	<i>Mixed I (On Shared LAN)</i>	<i>Mixed II (On Shared LAN)</i>
Speed of Transfer (bytes/sec)	7.80E+05	5.72E+05	5.95E+05	7.58E+05	7.85E+05	7.72E+05	4.60E+05	4.77E+05	7.69E+05	7.73E+05
Difference in performance in comparison to the base case (IPv4 Back-to-Back)	0.00	-26.67%	-23.71%	-2.82%	0.64%	-1.03%	-41.03%	-38.85%	-1.41%	-0.89%

Table 4a. Averaged results of data transfer from PC1 to PC2

<b>FROM PC2 to PC1</b>	<i>Back to Back using IPv4</i>	<i>Back to Back using IPv6 over IPv4</i>	<i>Back to Back IPv6 only</i>	<i>Mixed I (Back to Back)</i>	<i>Mixed II (Back to Back)</i>	<i>IPv4 on Shared LAN</i>	<i>IPv6 over IPv4 Shared LAN</i>	<i>On Shared LAN IPv6 only</i>	<i>Mixed I (On Shared LAN)</i>	<i>Mixed II (On Shared LAN)</i>
Speed of Transfer (bytes/sec)	9.72E+05	2.92E+05	2.97E+05	8.37E+05	9.42E+05	8.82E+05	2.97E+05	5.19E+05	9.08E+05	8.58E+05
Difference in performance in comparison to the base case (IPv4 Back-to-Back)	0.00%	-69.96%	-69.44%	-13.89%	-3.09%	-9.26%	-69.44%	-46.60%	-6.58%	-11.73%

Table 4b. Averaged results of data transfer from PC2 to PC1

The results of file transfers from PC2 to PC1 demonstrate an even more drastic drop in performance with the switch to IPv6 (both tunneling and IPv6 only) from IPv4. One can see 69% reduction of data transfer rate:  $9.72\text{E}+05$  bytes/sec in Back-to-Back mode using IPv4 vs.  $2.92\text{E}+05$  bytes/sec in tunneling and  $2.97\text{E}+05$  bytes/sec in IPv6 only (Table 4b). The results for tests run on a shared network exhibit a similar pattern of reduction of data transfer ( $8.82\text{E}+05$  bytes/sec for IPv4 vs.  $2.97\text{E}+05$  bytes/sec in tunneling and  $5.19\text{E}+05$  bytes/sec in IPv6 only).

The results obtained in a “mixed” environments stand between results obtained using IPv4-only and using IPv6 ( $8.37\text{E}+05$  bytes/sec Mixed I in Back-to-Back mode and  $9.42\text{E}+05$  bytes/sec Mixed II in Back-to-Back mode vs.  $9.72\text{E}+05$  bytes/sec using IPv4 in Back-to-Back and  $2.97\text{E}+05$  bytes/sec using IPv6 in Back-to-Back mode).

The overall degradation of performance in switching from IPv4 to IPv6 could be attributed to two possible reasons: encapsulation taking place in tunneling and the current development of IPv6.

The encapsulation procedure puts an IPv6 packet inside IPv4, which takes CPU time. The percentage of data transferred is less in IPv6 than in IPv4. The MTU of an Ethernet packet, equal to 1500 bytes in the tests, leaves at most 1480 bytes for the IPv6 packet, assuming no options, such as routing or destination header, are included into the IPv4 packet. With the fixed size header in IPv6 equal to 40 bytes, the payload part of the packet is only 1440 bytes. For each encapsulated packet, there are 60 bytes of overhead in transferring data using tunneling. For IPv4-only, this is only 20 bytes.

The results for the transfer using IPv6-only infer that the tunneling procedure itself is not imposing too much overhead. The numbers in the IPv6-only column are a little bit higher than the numbers in the IPv6 over IPv4 using Back-to-Back mode ( $5.95E+05$  bytes/sec vs.  $5.72E+05$  bytes/sec columns 2 and 3 Table 4a).

The lower performance results for IPv6 in general could be due to the fact that IPv6 is only in the developmental stage, and most of the software for the IPv6 is in either Alpha or Beta version releases. This fact supports the observation that the overall degradation of performance from IPv4 to the tunneling procedure shown in the tests ranged from 30% to 70%. In every test involving IPv6, the performance is lower than in tests involving IPv4-only. For example, the speed of transfer from PC1 to PC2 in the "Back-to-Back" mode using IPv4-only is  $7.80E+05$  bytes/sec, whereas the speed on transfer from PC1 to PC2 in the same mode using IPv6 over IPv4 tunneling is  $5.72E+05$  bytes/sec (columns 2 and 3 of Table 4a).

Also it should be noted that differences between test systems contributed to a variance in results obtained during the tests. The kernels running on both test machines were compiled specifically to match the configuration of the machines. PC1 is a 486 based computer with IDE interface on the motherboard. PC2 is a Pentium based machine with a PCI motherboard.

To make the comparison more complete, it would be desirable to run the same sequence of tests in a configuration involving a router. The presence of a router can provide further insight into how the performance of IPv6 may differ from IPv4. The differences should be due to the fact that the headers in both versions are different, and thus the processing of the headers themselves should be different too; IPv4 has a



checksum and IPv6 does not have a checksum included in the header. Tests using a router are being set up currently, so it would be rather bold to make any predictions on a performance issue without obtaining the actual results.

## CHAPTER VI

### CONCLUSIONS AND SUGGESTED FUTURE RESEARCH

*"Transition is always a matter of carrots and sticks"*

*Huitema. IPv6. The New Internet Protocol.*

Proponents of IPv6 claim that the new protocol will expand many features that lacked development in IPv4. The fixed size header will speed up the processing of a packet on routers. Omitting the options and removing the checksum from the header are also viewed as a contributing factor to the speeding up of packet processing on the routers. The expanded address space will prevent the Internet from running "out of addressing space." Another great advantage IPv6 offers is the support of mobil hosts and security options.

Opponents of IPv6 believe that security for payloads already is present in IPv4, and ATM technology provides better real-time support than does IPv6. However, to integrate security into IPv4, a major system release is required, which will cost as much as the transition to IPv6 itself [5]. It should be easier to manage security in IPv6 because it is incorporated into IPv6, but is only an option in IPv4. For the claims that ATM technology is better fit for multimedia and other real-time applications, it should be noted that current Ethernet cards are much simpler and more price effective in comparison to ATM cards for PCs. Besides, ATM technology does not support adaptive multimedia

applications. ATM technology can be used in conjunction with IP by exploiting labeling of flows within IPv6 [5].

The results of the tests conducted within the framework of this research suggest that the tunneling procedure does not impose much overhead in the transition to IPv6 from IPv4. Even though with deployment of IPv6 a degradation in performance was observed, it should be noted that the tests were run in configurations involving no routers. In theory, IPv6 should outperform IPv4 because of simplifications introduced in IPv6. The simplifications aim to speed up the processing of a packet on a router. For example, a checksum is removed from the IPv6 header. This should reduce the overall time it takes a router to process a packet.

Further research on the impact of IPv6 performance is possible. A desirable set of tests may involve an IPv6 capable router and an update of DHCP (Dynamic Host Configuration Protocol) supporting both IPv4 and IPv6 addresses. According to the IPv6 specifications, IPv6 provides a better routing performance in comparison to IPv4 [5]. The IPv6 capable routers will be the key elements in testing this claim. Updating DHCP can provide some insight into the way IPv6 supports mobile hosts.

It is advisable to start planning for the transition to IPv6 now. The dual stack strategy will ensure the functionality of production networks, and piecemeal introduction of the new protocol is supposed to make it as painless as possible. It will take several years before the Internet will be fully IPv6 operational. This period should allow ample time to gain experience in dealing with IPv6 and conducting a successful transition.

## REFERENCES

- [1] Bradner, S. and Allison Mankin, *IPng, Internet Protocol Next Generation*, Addison-Wesley Publishing Company, Reading, MA, 1995.
- [2] Fink, B. *6Bone*. <http://www-6bone.lbl.gov/6bone>, created 1996. (accessed June 1997).
- [3] Hinden, R. *IP Next Generation Overview*, Internet Draft. <http://playground.sun.com/pub/ipng/html/INET-IPng-Paper.html>, created 1995. (accessed November 1996).
- [4] Hinden, R. *IPng Implementations*. <http://playground.sun.com/pub/ipng/html/ipng-implementations.html>, created 1997. (accessed May 1997).
- [5] Huitema, C. *IPv6 The New Internet Protocol*, Prentice Hall PTR, Upper Saddle River, NJ, 1996.
- [6] *Internet: Global Penetration 1996 and Forecast for the Year 2000*. <http://www.killen.com/ipf.htm>, created 1996. (accessed June 1997).
- [7] *JOIN IPv6 Information*, <http://www.join.uni-muenster.de/JOIN/ipv6/texte-englisch/informationsquellen.html>, created 1996. (accessed June 1997).
- [8] Osborne, E. *Linux IPv6 Howto/FAQ*, <http://www.terra.net/ipv6>, created 1997. (accessed May 1997).
- [9] *Linux Netdev mailing list*, <http://www.wcug.wvu.edu/lists/netdev>, created 1997. (accessed May 1997).
- [10] MacKie-Mason, J. and Hal R. Varian, *Pricing the Internet*, Prentice-Hall, Englewood Cliffs, NJ, 1995.
- [11] Mann, C. "Regulating Cyberspace", *SCIENCE*. Vol. 268, May 1995.
- [12] *OSI Reference Model*. [http://tfbbs.tvinet.com/amitcp/OSI\\_Reference\\_Model.html](http://tfbbs.tvinet.com/amitcp/OSI_Reference_Model.html), created 1997. (accessed April 1997)

- [13] RFC 751 "Internet Protocol", Postel, <http://info.internet.isi.edu/in-notes/rfc/files/rfc751.txt>, 1981.
- [14] RFC 768 "Transmission Control Protocol", Postel, <http://info.internet.isi.edu/in-notes/rfc/files/rfc751.txt>, 1981.
- [15] RFC 1883 "Internet Protocol, Version 6 (IPv6) Specification", Ipsilon Networks, <http://info.internet.isi.edu/in-notes/rfc/files/rfc1883.txt>, December 1996.
- [16] RFC 1884 "IP Version 6 Addressing Architecture", Xerox Parc, <http://info.internet.isi.edu/in-notes/rfc/files/rfc1883.txt>, December 1995.
- [17] RFC 1887 "An Architecture for IPv6 Unicast Address Allocation", Cisco Systems, <http://info.internet.isi.edu/in-notes/rfc/files/rfc1887.txt>, December 1995.
- [18] RFC 1933 "Transition Mechanisms for IPv6 Hosts and Routers", Sun Microsystems Inc., <http://info.internet.isi.edu/in-notes/rfc/files/rfc1933.txt>, April 1996.
- [19] Solensky, M. "Special Issue IP - The Next Generation", *ConneXions*, Vol. 8 no. 5. pp. 14-16, May 1994.
- [20] Stevens, R. *UNIX Network Programming*,. Prentice Hall, Englewood Cliffs, NJ. 1990
- [21] Stevens, R. *TCP/IP Illustrated*, Volume 1, Addison-Wesley Publishing Company, Reading, MA, 1994.
- [22] *UK IPv6 Resource Center*, <http://www.cs-ipv6.lancs.au.uk/ipv6/projects>, created 1997. (accessed May 1997).
- [23] Wise, S. *IBM AIX IPv6 Web Page*. <http://www.rs6000.ibm.com/ipv6>, created 1997. (access May 1997).

## APPENDICES

### APPENDIX A: OSI NETWORK ARCHITECTURE REFERENCE MODEL

The OSI Network Architecture Reference Model as taken from a Web Page on the Internet at URL [http://tfbbs.tvinet.com/amitcp/OSI\\_Reference\\_Model.html](http://tfbbs.tvinet.com/amitcp/OSI_Reference_Model.html) [12] follows.

1. The **physical layer** defines electrical signaling on the transmission channel; how bits are converted into electrical current, light pulses or any other physical form. Serial line is an example of the physical layer. A network device functioning at this layer only is called a repeater.
2. The **data link layer** defines how the network layer packets are transmitted as bits. Examples of data link layer protocols are PPP (Point to Point Protocol) and Ethernet framing protocol. Bridges work at the data link layer only.
3. The **network layer** defines how information from the transport layer is sent over networks and how different hosts are addressed. An example of a network layer protocol is the Internet Protocol. A device that takes care of the network level functions is called a router or sometimes a gateway in the Internet terminology.
4. The **transport layer** takes care of data transfer, ensuring the integrity of data if desired by the upper layers. TCP and UDP are operating at this layer.
5. The **session layer** establishes and terminates connections and arranges sessions to logical parts. TCP and RPC provide some functions at this layer.
6. The **presentation layer** takes care of data type conversion. An example of protocol residing at this layer is XDR (External Data Representation), which is used by RPC applications to provide interoperability between heterogeneous computer systems.
7. The **application layer** defines the protocols to be used between the application programs. Examples of protocols at this layer are protocols for electronic mail (e.g., SMTP), file transfer (e.g., FTP) and remote login.

## APPENDIX B: FILE TRANSFER SESSIONS

The following are the file transfer sessions between two PCs. The output obtained from FTP application was used for analysis of performance results of IPv4 and IPv6 stacks. PC1 was assigned an IP address 139.78.114.2. PC2 was assigned 139.78.114.154.

A file transfer session From PC1 to PC2 using IPv4 in “Back to Back” mode.

```

Connected to 139.78.114.154.
220 hakim FTP server (Version 6.00) ready.
Name (139.78.114.154:root). 331 Password required for hakim.
230 User hakim logged in.
ftp> 200 PORT command successful.
150 Opening ASCII mode data connection for '/bin/ls'.
total 2
-rw-r--r-- 1 root root 34 Nov 24 1993 .less
-rw-r--r-- 1 root root 114 Nov 24 1993 .lessrc
226 Transfer complete.
ftp> 250 CWD command successful.
ftp> Local directory now /tmp
ftp> 200 PORT command successful.
150 Opening ASCII mode data connection for '/bin/ls'
total 1916
drwxr-xr-x 2 root root 1024 May 4 1994 .X11-unix
-rw-r--r-- 1 root root 677 Jun 8 12:51 k
-rw-r--r-- 1 root root 1010376 Jun 8 22:13 test_file
-r-xr-xr-x 1 root root 505188 Jun 8 12:38 vmlinuz.1.2.42.old
-rwxr-xr-x 1 root root 431933 Jun 8 12:38 vmlinuz.1.2.42_486withV6
226 Transfer complete.
ftp> 200 Type set to I.
ftp> 200 Type set to I.
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (1010376 bytes).
226 Transfer complete.
1010376 bytes received in 1.33 secs (7.4e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.33 secs (7.4e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.12 secs (8.8e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.19 secs (8.3e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.3 secs (7.6e+02 Kbytes/sec)
ftp> ?Invalid command
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.36 secs (7.3e+02 Kbytes/sec)
ftp> 221 Goodbye.

```

A file transfer session from PC1 to PC2 using IPv6 over IPv4 tunneling

```

Trying ::139.78.114.154.21...
Connected to ::139.78.114.154.
220 hakim FTP server (Version 6.00) ready.

```

```

Name (::139.78.114.154:root): 331 Password required for hakim.
230 User hakim logged in.
ftp> 250 CWD command successful.
ftp> Local directory now /tmp
ftp> 200 Type set to I.
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (578234711236635336 bytes).
226 Transfer complete.
1010376 bytes received in 2.05 seconds (491835 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.56 seconds (648555 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.58 seconds (641471 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.56 seconds (648268 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 2.02 seconds (500692 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.58 seconds (641471 bytes/s)
ftp> 221 Goodbye.

```

A file transfer session from PC1 to PC2 using IPv6 only.

```

Trying fe80::8b4e:729a:21...
Connected to fe80::8b4e:729a.
220 hakim FTP server (Version 6.00) ready.
Name (fe80::8b4e:729a:root): 331 Password required for hakim.
230 User hakim logged in.
ftp> 250 CWD command successful.
ftp> Local directory now /tmp
ftp> 200 Type set to I.
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (578234711236635336 bytes).
226 Transfer complete.
1010376 bytes received in 2.11 seconds (479311 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.61 seconds (627867 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 2.04 seconds (495237 bytes/s)
ftp> local: test_file remote: test_file

```



```

200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.91 seconds (529509 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.56 seconds (645882 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.54 seconds (654031 bytes/s)
ftp> 221 Goodbye.

```

A file transfer session from PC1 to PC2 using IPv4 on shared network.

```

Connected to 139.78.114.154.
220 hakim FTP server (Version 6.00) ready.
Name (139.78.114.154:root): 331 Password required for hakim.
230 User hakim logged in.
ftp> 250 CWD command successful.
ftp> Local directory now /tmp
ftp> 200 PORT command successful.
150 Opening ASCII mode data connection for '/bin/ls'.
total 1916
drwxr-xr-x 2 root root 1024 May 4 1994 .X11-unix
-rw-r--r-- 1 root root 677 Jun 8 12:51 k
-rw-r--r-- 1 root root 1010376 Jun 8 22:13 test_file
-r-xr-xr-x 1 root root 505188 Jun 8 12:38 vmlinuz.1.2.42.old
-rwxr-xr-x 1 root root 431933 Jun 8 12:38 vmlinuz.1.2.42_486withV6
226 Transfer complete.
ftp> 200 Type set to I.
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4611690931870984904 bytes).
226 Transfer complete.
1010376 bytes received in 1.35 secs (7.3e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.13 secs (8.8e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.34 secs (7.4e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.24 secs (8e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.34 secs (7.4e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.34 secs (7.4e+02 Kbytes/sec)
ftp> 221 Goodbye.

```

A file transfer session from PC1 to PC2 using IPv6 over IP4 tunneling on shared network.

```
Trying ::139.78.114.154.21...
Connected to hakim.cis.okstate.edu.
220 hakim FTP server (Version 6.00) ready.
Name (::139.78.114.154:root): 331 Password required for hakim.
230 User hakim logged in.
ftp> 200 SPORT command successful.
150 Opening ASCII mode data connection for '/bin/ls'.
total 2
-rw-r--r-- 1 root root 34 Nov 24 1993 .less
-rw-r--r-- 1 root root 114 Nov 24 1993 .lessrc
226 Transfer complete.
ftp> 250 CWD command successful.
ftp> Local directory now /tmp
ftp> 200 SPORT command successful.
150 Opening ASCII mode data connection for '/bin/ls'.
total 1916
drwxr-xr-x 2 root root 1024 May 4 1994 .X11-unix
-rw-r--r-- 1 root root 677 Jun 8 12:51 k
-rw-r--r-- 1 root root 1010376 Jun 8 22:13 test_file
-r-xr-xr-x 1 root root 505188 Jun 8 12:38 vmlinuz.1.2.42.old
-rwxr-xr-x 1 root root 431933 Jun 8 12:38 vmlinuz.1.2.42_486withV6
226 Transfer complete.
ftp> 200 Type set to I.
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (1010376 bytes).
226 Transfer complete.
1010376 bytes received in 2.23 seconds (452992 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 2.17 seconds (464855 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 2.15 seconds (470250 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 2.23 seconds (453174 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
ftp> 221 Goodbye.
```

A file transfer session from PC2 to PC1 using IPv4 in “Back-to-Back” mode.

```
Trying 139.78.114.2.21...
Connected to 139.78.114.2.
220 hakim2 FTP server (Version 6.00) ready.
Name (139.78.114.2:root): 331 Password required for hakim
230 User hakim logged in.
ftp> 250 CWD command successful.
```

```

ftp> Local directory now /tmp
ftp> 200 Type set to I.
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (578234711236635336 bytes).
226 Transfer complete.
1010376 bytes received in 1.16 seconds (867603 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.01 seconds (998632 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1 seconds (1006379 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.09 seconds (929173 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 0.998 seconds (1012572 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 0.995 seconds (1015881 bytes/s)
ftp> 221 Goodbye.

```

A file transfer session using IPv6 over IPv4 in “Back-to-Back” mode.

```

Trying ::139.78.114.2.21...
Connected to ::139.78.114.2.
220 hakim2 FTP server (Version 6.00) ready.
Name (::139.78.114.2:root): 331 Password required for hakim.
230 User hakim logged in.
ftp> 250 CWD command successful.
ftp> Local directory now /tmp
ftp> 200 Type set to I.
ftp> 200 SPORT command successful.
150 Opening ASCII mode data connection for '/bin/ls'.
-rw-r--r-- 1 root root 1010376 Jun 9 21:24 test_file
226 Transfer complete.
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4611690931870984904 bytes)
226 Transfer complete.
1010376 bytes received in 3.06 seconds (330618 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 3.83 seconds (264062 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 3.19 seconds (316961 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).

```

```

226 Transfer complete.
1010376 bytes received in 3.2 seconds (315641 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 3.94 seconds (256124 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 3.4 seconds (296934 bytes/s)
ftp> 221 Goodbye.

```

A file transfer session using IPv6 only in Back-to-Back mode.

```

Trying fe80::8b4e:7202.21...
Connected to fe80::8b4e:7202.
220 hakim2 FTP server (Version 6.00) ready.
Name (fe80::8b4e:7202:root): 331 Password required for hakim.
230 User hakim logged in.
ftp> 250 CWD command successful.
ftp> Local directory now /tmp
ftp> 200 Type set to I.
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (578234711236635336 bytes).
226 Transfer complete.
1010376 bytes received in 3.8 seconds (266087 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 3.6 seconds (280937 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 3.03 seconds (333145 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 3.04 seconds (331887 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 3.72 seconds (271789 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 3.8 seconds (266016 bytes/s)
ftp> 221 Goodbye

```

A file transfer session in “Mixed I” mode in Back-to-Back configuration.

```

Connected to 139.78.114.2.
220 hakim2 FTP server (Version 6.00) ready.
Name (139.78.114.2:root): 331 Password required for hakim.
230 User hakim logged in
ftp> 250 CWD command successful
ftp> Local directory now /tmp
ftp> 200 Type set to I.
ftp> 200 PORT command successful.
150 Opening ASCII mode data connection for '/bin/ls'.

```

```

-rw-r----- 1 hakim users 1010376 Jun 25 21:23 test_file
226 Transfer complete.
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4611690931870984904 bytes).
226 Transfer complete.
1010376 bytes received in 2.67 secs (3.7e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.02 secs (9.7e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.02 secs (9.7e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.26 secs (7.8e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.02 secs (9.7e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.01 secs (9.7e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.03 secs (9.6e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.04 secs (9.5e+02 Kbytes/sec)
ftp> 221 Goodbye.

```

A file transfer session in “*Mixed IP*” mode in Back-to-Back configuration.

```

Trying 139.78.114.2.21...
Connected to 139.78.114.2.
220 hakim2 FTP server (Version wu-2.4(1) Tue Dec 5 20:51:15 CST 1995) ready.
Name (139.78.114.2:root): 331 Password required for hakim.
230 User hakim logged in.
ftp> 250-Please read the file README
250- it was last modified on Fri May 30 10:56:06 1997 - 26 days ago
250 CWD command successful.
ftp> Local directory now /tmp
ftp> 200 Type set to I.
ftp> local: test_file remote: test_file
500 'SPORT 12,94': command not understood.
ftp> local: test_file remote: test_file
500 'SPORT 12,103': command not understood.
200 PORT command successful.
150 Opening BINARY mode data connection for test_file (1010376 bytes).
226 Transfer complete.
1010376 bytes received in 1.19 seconds (848002 bytes/s)
ftp> local: test_file remote: test_file
200 PORT command successful.
150 Opening BINARY mode data connection for test_file (1010376 bytes).
226 Transfer complete.
150 Opening BINARY mode data connection for test_file (1010376 bytes).
226 Transfer complete.
1010376 bytes received in 1.08 seconds (935521 bytes/s)
ftp> local: test_file remote: test_file
200 PORT command successful.

```

```

150 Opening BINARY mode data connection for test_file (1010376 bytes).
226 Transfer complete.
1010376 bytes received in 0.985 seconds (1025677 bytes/s)
ftp> local: test_file remote: test_file
200 PORT command successful.
150 Opening BINARY mode data connection for test_file (1010376 bytes).
226 Transfer complete.
1010376 bytes received in 1.01 seconds (1002359 bytes/s)
ftp> local: test_file remote: test_file
200 PORT command successful.
150 Opening BINARY mode data connection for test_file (1010376 bytes).
226 Transfer complete.
1010376 bytes received in 1.22 seconds (826298 bytes/s)
ftp> local: test_file remote: test_file
200 PORT command successful.
150 Opening BINARY mode data connection for test_file (1010376 bytes).
226 Transfer complete.
1010376 bytes received in 0.9/
seconds (1014094 bytes/s)
ftp> local: test_file remote: test_file
200 PORT command successful.
150 Opening BINARY mode data connection for test_file (1010376 bytes).
226 Transfer complete.
1010376 bytes received in 1.12 seconds (899394 bytes/s)
ftp> local: test_file remote: test_file
200 PORT command successful.
150 Opening BINARY mode data connection for test_file (1010376 bytes).
226 Transfer complete.
1010376 bytes received in 1.05 seconds (965296 bytes/s)
ftp> 221 Goodbye.

```

#### A file transfer session using IPv4 on shared LAN.

```

Connected to 139.78.114.2.
220 hakim2 FTP server (Version 6.00) ready.
Name (139.78.114.2:root): 331 Password required for hakim.
230 User hakim logged in.
ftp> 200 PORT command successful.
150 Opening ASCII mode data connection for '/bin/ls'
total 3
-rw-r--r-- 1 hakim users 186 Jun 7 20:54 .bash_history
-rw-r--r-- 1 hakim users 34 Nov 23 1993 .less
-rw-r--r-- 1 hakim users 114 Nov 23 1993 .lessrc
-rw-rw-r-- 1 hakim users 0 Jun 3 00:21 k
226 Transfer complete.
ftp> 250 CWD command successful.
ftp> Local directory now /tmp
ftp> 200 Type set to I.
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.41 secs (7e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.18 secs (8.4e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.09 secs (9.1e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.13 secs (8.7e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.

```

```

1010376 bytes received in 0.982 secs (1e+03 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.02 secs (9.7e+02 Kbytes/sec)
ftp> 221 Goodbye.

```

### A file transfer session using IPv6 over IPv4 tunneling on shared LAN.

```

Trying ::139.78.114.2.21...
Connected to hakim2.cis.okstate.edu.
220 hakim2 FTP server (Version 6.00) ready.
Name (::139.78.114.2:root): 331 Password required for hakim.
230 User hakim logged in.
ftp> 250 CWD command successful.
ftp> Local directory now /tmp
ftp> 200 Type set to I.
ftp> 200 SPORT command successful.
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4611690931870984904 bytes).
226 Transfer complete.
1010376 bytes received in 3.81 seconds (265084 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 3.22 seconds (314165 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 3.21 seconds (315043 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 3.2 seconds (315512 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 3.32 seconds (304537 bytes/s)
ftp> local: test_file remote: test_file
200 SPORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 3.22 seconds (314225 bytes/s)
ftp> 221 Goodbye.

```

### A file transfer session using IPv6 on shared LAN.

```

Trying fe80::8b4e:7202.21...
Connected to fe80::8b4e:7202.
220 hakim2 FTP server (Version 6.00) ready
Name (fe80::8b4e:7202:root): 331 Password required for hakim.
230 User hakim logged in.
ftp> Local directory now /tmp
ftp> 250 CWD command successful.
ftp> 200 Type set to I.
ftp> 200 SPORT command successful.
150 Opening ASCII mode data connection for '/bin/ls'
-rwxr-xr-x 1 root root 99600 Jun 2 21:52 telnetd
-rw-r----- 1 hakim users 1010376 Jun 25 14:45 test_file
226 Transfer complete.

```

```

1010376 bytes received in 1.19 secs (8.3e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.13 secs (8.7e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.01 secs (9.8e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.01 secs (9.8e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.01 secs (9.8e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.01 secs (9.8e+02 Kbytes/sec)
ftp> 200 PORT command successful.
150 Opening BINARY mode data connection for 'test_file' (4295977672 bytes).
226 Transfer complete.
1010376 bytes received in 1.22 secs (8.1e+02 Kbytes/sec)
ftp> 221 Goodbye.

```

A file transfer session in “*Mixed IP*” mode on shared LAN.

```

Trying 139.78.114.2.21...
Connected to hakim2.cis.okstate.edu.
220 hakim2 FTP server (Version wu-2.4(1) Tue Dec 5 20:51:15 CST 1995) ready.
Name (139.78.114.2:root): 331 Password required for hakim.
230 User hakim logged in.
ftp> 250-Please read the file README
250- it was last modified on Fri May 30 10:56:06 1997 - 26 days ago
250 CWD command successful.
ftp> Local directory now /tmp
ftp> 200 Type set to I.
ftp> local: test_file remote: test_file
500 'SPORT 9,236': command not understood.
ftp> local: test_file remote: test_file
500 'SPORT 10,43': command not understood.
200 PORT command successful.
150 Opening BINARY mode data connection for test_file (1010376 bytes).
226 Transfer complete.
1010376 bytes received in 1.28 seconds (787436 bytes/s)
ftp> local: test_file remote: test_file
200 PORT command successful.
150 Opening BINARY mode data connection for test_file (1010376 bytes).
226 Transfer complete.
150 Opening BINARY mode data connection for test_file (;010376 bytes).
226 Transfer complete.
1010376 bytes received in 1.28 seconds (787270 bytes/s)
ftp> local: test_file remote: test_file
200 PORT command successful.
150 Opening BINARY mode data connection for test_file (1010376 bytes).
226 Transfer complete.
1010376 bytes received in 1.29 seconds (782339 bytes/s)
ftp> local: test_file remote: test_file
200 PORT command successful.
150 Opening BINARY mode data connection for test_file (1010376 bytes).
226 Transfer complete.
1010376 bytes received in 1.01 seconds (1004334 bytes/s)
ftp> local: test_file remote: test_file
200 PORT command successful.
150 Opening BINARY mode data connection for test_file (1010376 bytes).
226 Transfer complete.
1010376 bytes received in 1.3 seconds (776297 bytes/s)

```



```
ftp> local: test_file remote: test_file
200 PORT command successful.
150 Opening BINARY mode data connection for test_file (1010376 bytes).
226 Transfer complete.
1010376 bytes received in 0.999 seconds (1011234 bytes/s)
ftp> local: test_file remote: test_file
1010376 bytes received in 1.18 seconds (854337 bytes/s)
ftp> local: test_file remote: test_file
200 PORT command successful.
150 Opening BINARY mode data connection for test_file (1010376 bytes).
226 Transfer complete.
1010376 bytes received in 1.01 seconds (998673 bytes/s)
ftp> local: test_file remote: test_file
200 PORT command successful.
150 Opening BINARY mode data connection for test_file (1010376 bytes).
226 Transfer complete.
1010376 bytes received in 1.01 seconds (1003334 bytes/s)
ftp> 221 Goodbye.
```

## APPENDIX C: RESULTS OF THE TESTS

Below are the results of the individual tests. The tests involved transferring a file of size 1 megabyte between two PCs.

<b>Transfer Speed from PC1 to PC2 (bytes/sec)</b>										
Test #	<i>Back to Back</i>					<i>Shared LAN</i>				
	V4	V6overV4	V6	Mixed I	Mixed II	V4	V6overV4	V6	Mixed I	Mixed II
1	7.40E+05	4.92E+05	4.79E+05	7.54E+05	7.60E+05	7.30E+05	4.53E+05	4.70E+05	4.19E+05	7.50E+05
2	7.40E+05	6.48E+05	6.28E+05	7.59E+05	8.00E+05	8.80E+05	4.65E+05	4.54E+05	9.14E+05	7.70E+05
3	8.80E+05	6.41E+05	4.95E+05	7.60E+05	7.80E+05	7.40E+05	4.70E+05	4.59E+05	7.65E+05	8.10E+05
4	8.30E+05	6.48E+05	5.29E+05	7.59E+05	8.00E+05	8.00E+05	4.53E+05	5.09E+05	8.30E+05	7.70E+05
5	7.60E+05	5.00E+05	6.46E+05	7.5E+05	8.00E+05	7.40E+05	4.53E+05	4.68E+05	7.70E+05	8.00E+05
6	7.30E+05	6.41E+05	6.54E+05	7.62E+05	7.70E+05	7.40E+05	4.65E+05	5.02E+05	9.13E+05	7.40E+05

Table C1. Results of File Transfer from PC1 to PC2

<b>Transfer Speed from PC2 to PC1 (bytes/sec)</b>										
Test #	<i>Back to Back</i>					<i>Shared LAN</i>				
	V4	V6overV4	V6	Mixed I	Mixed II	V4	V6overV4	V6	Mixed I	Mixed II
1	8.68E+05	3.31E+05	2.66E+05	3.70E+05	8.48E+05	7.00E+05	3.31E+05	4.67E+05	8.30E+05	7.87E+05
2	9.99E+05	2.64E+05	2.81E+05	9.70E+05	9.35E+05	8.40E+05	2.64E+05	4.17E+05	8.70E+05	7.87E+05
3	1.01E+06	3.17E+05	3.33E+05	7.80E+05	1.02E+06	9.10E+05	3.17E+05	4.72E+05	9.80E+05	7.82E+05
4	9.29E+05	3.16E+05	3.32E+05	9.70E+05	1.00E+06	8.70E+05	3.16E+05	4.72E+05	9.80E+05	1.04E+05
5	1.01E+06	2.56E+05	2.72E+05	9.70E+05	8.26E+05	1.00E+06	2.56E+05	6.43E+05	9.80E+05	7.76E+05
6	1.02E+06	2.97E+05	2.66E+05	9.60E+05	1.01E+06	9.70E+05	2.97E+05	6.42E+05	8.10E+05	1.01E+05

Table C2. Results of File Transfer from: PC2 to PC1

## APPENDIX D: TEST LABORATORY EQUIPMENT

## Test Equipment:

- PC1 486, DX2. 66 MHz. ISA Motherboard. 16M RAM, 100M HD  
Ethernet Card - 3Com 3c509 10 Mbits/sec
- PC2 Pentium, 75MHz. PCI Motherboard. 16M RAM, 150M HD  
Ethernet Card - 3Com 3c59x Vortex 10 Mbits/sec

Network Analyzer "Network General", based on the 486 Intel Chip, 16M RAM,  
Ethernet Card - 3Com 3c505 10 Mbits/sec.

Repeater: 12 port 3Com FMS TP Hub.

## APPENDIX E: TEST MACHINES' SETUP

Below are the configurations of lab machines. Both PCs were running LINUX, which is UNIX for PCs based on an i386 architecture. The software is readily available on the Internet from various sites. Distribution used for the tests were "Slackware 3.1", Kernel Version 1.2.42, and network applications "net-tools.1.32." The software was obtained from ftp://ftp.wcug.wvu.edu/pub/ipv6. "Net-tools.1.32" included IPv6 capable network applications such as ping, ftp, ftpd, telnet, telnetd, and inted.

### Network configuration of PC1:

```
lo    Link encap:Local Loopback
      inet addr:127.0.0.1 Bcast:127.255.255.255 Mask:255.0.0.0
      inet6 addr: ::1/0 Scope:Host
      UP BROADCAST LOOPBACK RUNNING MTU:3584 Metric:1
      RX packets:24 errors:0 dropped:0 overruns:0 frame:0
      TX packets:24 errors:0 dropped:0 overruns:0 carrier:0 coll:0

eth0  Link encap:Ethernet HWaddr 00:60:8C:39:B2:1A
      inet addr:139.78.114.154 Bcast:139.78.114.255 Mask:255.255.255.0
      inet6 addr: fe80::60:8c39:b21a/10 Scope:Link
      inet6 addr: ::139.78.114.154/80 Scope:Compat
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:64954 errors:0 dropped:0 overruns:0 frame:0
      TX packets:35670 errors:0 dropped:0 overruns:0 carrier:0 coll:2446
      Interrupt:10 Base address:0x300

sit0  Link encap:IPv6-in-IPv4
      inet6 addr: ::127.0.0.1/0 Scope:Unknown
      inet6 addr: ::139.78.114.154/0 Scope:Compat
      UP RUNNING NOARP MTU:1480 Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:19631 errors:0 dropped:0 overruns:0 carrier:0 coll:0

sit1  Link encap:IPv6-in-IPv4
      inet6 addr: fe80::8b4e:729a/0 Scope:Link
      UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1480 Metric:1
      RX packets:26042 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 coll:0
```

### IPv6 configuration file:

```
#!/bin/bash
exec >/dev/console
exec </dev/console
exec 2>/dev/console
PATH=/usr/local/bin:/usr/local/etc:/usr/bin:/bin:/etc: export PATH

# Your IPv6 prefix
PREFIX=5f13
# The host-part of the IPv6 address for this machine
#ADDRESS=5f13:d600:8b4e:7200:72:60:8c39:b21a
ADDRESS=:0:0:0:0:0:139.78.114.154
# The IPv4 address of the far side of your tunnel
TUNNEL=139.78.114.2

echo 0 > /proc/sys/net/ipv6/accept_ra
echo 0 > /proc/sys/net/ipv6/accept_redirects
echo 1 > /proc/sys/net/ipv6/forwarding

/sbin/ifconfig sit0 up
```

```

/sbin/ifconfig eth0 add $PREFIX::$ADDRESS/80
/sbin/route -A inet6 add $PREFIX::0/80 dev eth0

# tunnel to outside
/sbin/ifconfig sit0 tunnel ::$TUNNEL
/sbin/ifconfig sit1 up
/sbin/route -A inet6 add $ff00::0/8 gw 0::$TUNNEL dev sit1

#/etc/radvd &

```

### Routing table:

```

Kernel IPv6 routing table
Destination      Next Hop          Flags Metric Ref  Use Iface
::1/128          ::0               0 0 0 00000000
::127.0.0.1/128 ::0               0 0 0 00000000
::139.78.114.154/128 ::0             0 0 0 00000000
::0/96           ::0               0 1 0 00004ca0
5f13::0/80      ::0               U 0 0 0 00000000
5f00::0/8       ::139.78.114.2   U 0 0 0 00000000
fe80::8b4e:729a/128 ::0             0 0 0 00000000
fe80::60:8c39:b21a/128 ::0             0 0 0 00000000
fe80::60:8c39:b21a/10 ::0             0 1 0 00000000
fe80::8b4e:7202/10 ::139.78.114.2  0 1 0 00000000
ff00::0/8       ff00::0          0 1 0 00000000
ff00::0/8       ff00::0          0 1 0 00000000
::0/128         ::0               GH 255 255 255 00000000

```

### Network configuration of PC2:

```

lo Link encap:Local Loopback
inet addr:127.0.0.1 Bcast:127.255.255.255 Mask:255.0.0.0
inet6 addr: ::1/0 Scope:Host
UP BROADCAST LOOPBACK RUNNING MTU:3584 Metric:1
RX packets:20 errors:0 dropped:0 overruns:0 frame:0
TX packets:20 errors:0 dropped:0 overruns:0 carrier:0 coll:0

eth0 Link encap:Ethernet HWaddr 00:A0:24:95:D5:E4
inet addr:139.78.114.2 Bcast:139.78.114.255 Mask:255.255.255.0
inet6 addr: fe80::a0:2495:d5e4/10 Scope:Link
inet6 addr: ::139.78.114.2/80 Scope:Compat
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:242054 errors:0 dropped:0 overruns:0 frame:0
TX packets:51689 errors:0 dropped:0 overruns:0 carrier:0 coll:2834
Interrupt:10 Base address:0xfce0

sit0 Link encap:IPv6-in-IPv4
inet6 addr: ::127.0.0.1/0 Scope:Unknown
inet6 addr: ::139.78.114.2/0 Scope:Compat
UP RUNNING NOARP MTU:1480 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:26042 errors:0 dropped:0 overruns:0 carrier:0 coll:0

sit1 Link encap:IPv6-in-IPv4
inet6 addr: fe80::8b4e:7202/0 Scope:Link
UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1480 Metric:1
RX packets:19631 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 coll:0

```

### IPv6 configuration file:

```
#!/bin/bash
```

```

exec >/dev/console
exec </dev/console
exec 2>/dev/console
PATH=/usr/local/bin:/usr/local/etc:/usr/bin:/bin:/etc; export PATH

# Your IPv6 prefix
PREFIX=5f13
# The host-part of the IPv6 address for this machine
#ADDRESS=5f13:d600:8b4e:7200:72:a0:2495:d5e4
ADDRESS=0:0:0:0:0:139.78.114.2
# The IPv4 address of the far side of your tunnel
TUNNEL=139.78.114.154

echo 0 > /proc/sys/net/ipv6/accept_ra
echo 0 > /proc/sys/net/ipv6/accept_redirects
echo 1 > /proc/sys/net/ipv6/forwarding

/sbin/ifconfig sit0 up

/sbin/ifconfig eth0 add $PREFIX::$ADDRESS/80
/sbin/route -A inet6 add $PREFIX::0/80 dev eth0

# tunnel to outside
/sbin/ifconfig sit0 tunnel 0::$TUNNEL
/sbin/ifconfig sit1 up
/sbin/route -A inet6 add 5f00::0/8 gw 0::$TUNNEL dev sit1

#/etc/radvd &

```

## Routing Table:

```

Kernel IPv6 routing table
Destination          Next Hop              Flags Metric Ref  Use Iface
::1/128              ::0                   0  0  0 00000000
::127.0.0.1/128     ::0                   0  0  0 00000000
::139.78.114.2/128  ::0                   0  0  0 00000000
::0/96                ::0                   0  1  0 000065b5
5f13::0/80           ::0                   U  0  0  0 00000000
5f00::0/8            ::139.78.114.154    U  0  0  0 00000000
fe80::8b4e:7202/128  ::0                   0  0  0 00000000
fe80::60:8c39:b21a/128 fe80::60:8c39:b21  0  1  0 00003414
fe80::a0:2495:d5e4/128 ::0                   0  0  0 00000000
fe80::8b4e:729a/10  ::139.78.114.154    0  1  0 00000000
fe80::a0:2495:d5e4/10 ::0                   0  1  0 00000000
ff00::0/8           ff00::0              0  1  0 00000000
ff00::0/8           ff00::0              0  1  0 00000000
::0/128             ::0                   UGH 255 255 255 00000000

```

## APPENDIX F: GLOSSARY

AAAA	A record in DNS table to answer queries by IPv6 hosts
API	Application Programming Interface
ASN	Autonomous System Number
ATM	Asynchronous Transfer Mode
Datagram	Internet Protocol network data packet
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Server
FTP	File Transfer Protocol
IANA	Internet Addressing and Naming Authority
IETF	Internet Engineering Task Force
ICMP	Internet Control Message Protocol
IP	Internet Protocol
ISP	Internet Service Provider
LAN	Local Area Network
MAC	Medium Access Control
MTU	Maximum Transmit Unit
OSI	Open System Interconnect
RFC	Request for Comments
SLIP	Serial Line IP
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
TTL	Time To Live
Tunneling	(IPv6-over-IPv4 tunneling) A method of transporting of IPv6 datagrams over IPv4 networks, by encapsulating IPv6 datagrams within IPv4 datagrams.

2

VITA

Khakim Sultanov

Candidate for the Degree of  
Master of Science

Thesis: TRANSITION OF CAMPUS NETWORK TO IP NEXT GENERATION

Major Field: Computer Science

Biographical:

Education: Graduate of the Rochester Institute of Technology, Bachelor of Science in Applied Mathematics, August 1994. Graduate of Tashkent State University, Uzbekistan. Diploma, Applied Mathematics and Theoretical Mechanics, 1995. Completed the requirements for the Master of Science degree in Computer Science at Oklahoma State University in July 1997.

Experience: System Administrator of Email facilities, Uzbekistan, USSR. 1994-1995.  
Part-time job with Data Communications OSU, May 1996 - August 1997



2

VITA

Khakim Sultanov

Candidate for the Degree of  
Master of Science

Thesis: TRANSITION OF CAMPUS NETWORK TO IP NEXT GENERATION  
Major Field: Computer Science

Biographical:

Education: Graduate of the Rochester Institute of Technology, Bachelor of Science in Applied Mathematics, August 1994. Graduate of Tashkent State University, Uzbekistan. Diploma, Applied Mathematics and Theoretical Mechanics, 1995. Completed the requirements for the Master of Science degree in Computer Science at Oklahoma State University in July 1997.

Experience: System Administrator of Email facilities, Uzbekistan, USSR. 1994-1995.  
Part-time job with Data Communications OSU, May 1996 - August 1997