Evaluating an IPv4 and IPv6 Network

GILBERT LIDHOLM
and
MARCUS NETTERBERG

KTH Information and
Communication Technology

Degree project in
Communication Systems
First level, 15.0 HEC
Stockholm, Sweden

KTH Royal Institute of Technology

Evaluating an IPv4 and IPv6 Network

Gilbert Lidholm & Marcus Netterberg

2012-09-08

Bachelor's thesis

Examiner & supervisor:
Professor Gerald Q. Maguire Jr.

Abstract

This thesis is the result of the bachelor's thesis project "Evaluating an IPv4 and IPv6 network".

The IPv6 protocol was created with the main purpose of solving the problem of the depletion of IP-addresses that IPv4 is currently facing. This thesis gives an introduction to the differences between IPv4 and IPv6 and when one should use one protocol rather than the other. It describes the services that we will use in order to evaluate what kinds of problems IPv4 may experience and if these problems can be solved by using IPv6. We also show how to set up a network with both protocols for each service that we examine. We will subsequently evaluate the performance of these two protocols for each of these services. We found that there were no significant differences in the performance of any of the applications that we tested with both IPv4 and IPv6. Due to the depletion of IPv4 addresses and the continuing rapid growth of the Internet, this thesis describes a very current and a relevant issue for computer networks today.

Abstrakt

Denna avhandling är resultatet utav högskoleingenjörsexamensarbetet ”Utvärdera ett IPv4- och IPv6 nätverk”.

IPv6-protokollet skapades huvudsakligen för att lösa bristen på IP adresser som IPv4 står inför. Avhandlingen ger en introduktion till skillnaden mellan IPv4 och IPv6 och när det skulle vara mer lämpligt att använda det ena protokoll framför den andra. Den beskriver de tjänster som vi kommer att använda och utvärdera vilka typer av problem som IPv4 kan erfara och om dessa problem kan lösas med hjälp av IPv6. Vi förklarar också hur man sätter upp ett nätverk med de två protokollen för varje tjänst som vi utvärderar. Vi kommer sedermera utvärdera prestandan för båda protokollen för dessa tjänster. Vi kom fram till att det inte var några signifikanta skillnader i prestanda för någon av de applikationer som vi testade med både IPv4 och IPv6. På grund av utarmningen av IPv4-adresser och den snabba tillväxten av internet, så beskriver denna avhandling ett väldigt aktuellt och relevant problem i datornätverk idag.

The intended audience

This thesis is mainly written for people with average to advanced knowledge in computer networking that wishes to gain an insight in the difference between IPv4 and IPv6 in the aspect of structure, performance and implementation. There are a lot to be learned as a novice but many parts will be hard to follow.

Acknowledgments

We would like to extend our sincere thanks and appreciation to our supervisor and examiner Professor Gerald Q. Maguire Jr. for providing us with this topic and his extensive feedback throughout the entire course of this thesis.

Furthermore our thanks go out to our families and friends, both at KTH and at home, for all the support. These three years could never have been completed without you.

Table of contents

List of figures

List of tables

List of acronyms and abbreviations

AD	Active Directory
AH	Authentication Header
APNIC	Asia Pacific Network Information Centre
APT	Advanced Packaging Tool
ARIN	American Registry for Internet Numbers
ARP	Address Resolution Protocol
ARPA	Advanced Research Project Agency
AS	Autonomous System
AfriNIC	African Network Information Centre
BGP	Border Gateway Protocol
BIND	Berkeley Internet Name Domain
BIND9	Berkeley Internet Name Domain version 9
BITS	Bump In The Stack
BITW	Bump In The Wire
CD	Checking Disabled
CDN	Content Delivery Network
CERNET	China Education and Research Network
DAD	Duplicate Address Detection
DAV	Distributed Authoring and Versioning
DHCP	Dynamic Host Configuration Protocol
DHCPv4	Dynamic Host Configuration Protocol for Internet Protocol version 4
DHCPv6	Dynamic Host Configuration Protocol for Internet Protocol version 6
DNS	Domain Name System
DNSKEY	Domain Name System Public Key
DNSSEC	Domain Name System Security Extensions
DS	Delegation Signer
DoD	Department of Defense
DoS	Denial of Service
ESP	Encapsulation Security Payload
FEC	Forwarding Equivalence Class
FTP	File Transfer Protocol
GUI	Graphical User Interface
GWS	Google Web Server
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
ICMPv4	Internet Control Message Protocol for Internet Protocol version 4
ICMPv6	Internet Control Message Protocol for Internet Protocol version 6
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IIS	Internet Information Server
IKE	Internet Key Exchange
IP	Internet Protocol
IPIP	IP in IP
IPsec	Internet Protocol Security
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IS-IS	Intermediate System to Intermediate System
ISATAP	Intra-Site Automatic Tunnel Addressing Protocol
ISC	Internet System Consortium
ISP	Internet Service Provider
KB	Kilobyte
KSK	Key Signing Key

LACNIC	Latin America and Caribbean Network Information Centre
LAN	Local Area Network
LDP	Label Distribution Protocol
LSP	Label Switched Path
MAC	Media Access Control
MB	Megabyte
mDNS	Multicast Domain Name System
MPLS	Multiprotocol Label Switching
MTA	Mail Transfer Agent
MTU	Maximum Transmission Unit
MX	Mail Exchange Record
NA	Neighbor Advertisement
NAT	Network Address Translation
ND	Neighbor Discovery
NDIS	Network Driver Interface Specification
NFS	Network File System
NLRI	Network Layer Reachability Information
NS	Neighbor Solicitation
NSEC	Next Secure
NUD	Neighbor Unreachability Detection
OS	Operation System
OSPF	Open Shortest Path First
OSPFv2	Open Shortest Path First version 2
OSPFv3	Open Shortest Path First version 3
OpenLDAP	Lightweight Directory Access Protocol
P2P	Peer-to-Peer
PCATTCP	Printing Communications Association Port of Test Transmission Control Protocol
PI	Provider Independent
PTR	Domain Name Pointer
QoS	Quality of Service
RA	Router Advertisement
RFC	Request For Comments
RIP	Routing Information Protocol
RIPE NCC	Réseaux IP Européens Network Coordination Center
RIPng	Routing Information Protocol next generation
RIR	Regional Internet Registry
RR	Resource Record
RRSIG	Resource Record Signature
RS	Router Solicitation
RSA	Rivest Shamir Adleman
RSVP	Resource Reservation Protocol
SA	Security Association
SEND	Secure Neighbor Discovery
SFTP	Secure Shell File Transfer Protocol
SHA1	Secure Hash Algorithm 1
SIIT	Stateless IP/ICMP Translation
SIT	Simple Internet Transition
SMTP	Simple Mail Transfer Protocol
SOA	Start Of Authority
Juniper SSG	Juniper Secure Service Gateway
SSH	Secure Shell
ST-II	Internet Stream Protocol version 2
SVN	Subversion
TCP	Transmission Control Protocol
TLV	Type Length Value

TTL	Time To Live
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VLC	VideoLAN Client
VPN	Virtual Private Network
VSFTPD	Very Secure File Transfer Protocol Daemon
VoIP	Voice over IP
WAN	Wide Area Network
WINS	Windows Internet Name Service
XMPP	Extensible Messaging and Presence Protocol
ZSK	Zone Signing Key

1. Introduction

The Internet is a vast and continuously growing network of networks through which the entire world is interconnected and exchanging information. Large investments have been made by countries all over the world to ensure that as large of a portion as possible of each country is able to connect to a reliable and fast Internet connection. As this expansion progresses the individual computers, cell phones, and other devices that now can connect to the Internet gain in speed, functionality, and accessibility. This in turn leads to companies and private people developing their business, and even their life, around this connectivity. Even the low cost alternative of IP telephony is starting to be favored over regular telephone handsets connected to the public switched telephony network. The rapid development in recent decades has led to an ever-growing need for more IP addresses. Having an IP address is crucial to connect to the Internet. Additionally, more and more devices need (or desire) constant connectivity in order to provide the proper functionality to its user or users.

The Internet today relies on the Internet Protocol version 4 (IPv4) protocol. When originally developed in the late 1960s, the need for an enormous number of addresses that we see now was not anticipated. At that time computers had just started to appear but, just like now, they were much more useful if they were able to communicate with each other. A demand for a network that would interconnect and make computer resources available grew. The United States Department of Defense (DoD) needed to make a distributed set of computer recourses available to researchers that were working on contracts for them. A packet switching network was developed by the Advanced Research Project Agency (ARPA) of DoD in 1969, and it was called ARPAnet[1]. After further developments and trials this eventually grew into the modern Internet utilizing IPv4. Initially only universities, large companies with military contracts, and the military could utilize this network, hence only a small number of computers needed an IP address. The approximately 4.3 billion addresses that IPv4 provides seemed like an endless amount when IPv4 was introduced on January 1 1983[2], and even if only 3.7 billion addresses can be allocated to ordinary devices ($2^7*2^{24} + 2^{14}*2^{16} + 2^{21}*2^8 = 3{,}758{,}096{,}384$), it was considered enough to cover all future needs.

But in the early 1990s, with the increasing number of IP addresses being requested, it was clear that they would eventually run out. As of 31 January 2011, the pool of unallocated IPv4 addresses officially ran out[3]. The last two blocks of addresses were assigned by the Internet Assigned Numbers Authority (IANA) to the Asia Pacific Network Information Centre (APNIC)[4]. This does not mean that there are no more IPv4 addresses whatsoever, but it does mean that each regional Internet registry (the registry is responsible for allocating Internet number resources in its own region) cannot request a new block of addresses to allocate. This means that when a registry runs out of addresses that it cannot allocate any additional addresses within its region.

As a result of the realization that the addresses would eventually be depleted, the Internet Engineering Task Force (IETF) was assigned the task to develop a successor to IPv4. The 32-bit IP address space was simply not going to be sufficient as large numbers of devices each needed one or more unique IP address assigned to it. The decision on this successor took some time, but it was decided that a 128-bit address scheme would be adopted. Improvements, in addition to extending the address space, were made based upon the long experience with IPv4. These improvements include autoconfiguration of devices for easier administration and built-in security with IPsec. As a result the specifications of IP version 6 (IPv6) were established in RFC 1883[5] in December 1995.

What happened to IPv5 then? The original thought was that the Internet Stream Protocol version 2 (ST-II) protocol was to become IPv5. These packets were identified with Internet Protocol version number 5; however, the Resource Reservation Protocol (RSVP) was favored over ST-II[6].

2. IPv6 compared to IPv4

This chapter will discuss some differences between the two protocols and what is new in IPv6.

2.1 Address space

The most obvious difference between IPv4 and IPv6 is the size of the addresses. In the IPv4 protocol addresses are 32 bits long. This leads to a theoretical limit of 2^{32} = 4,294,967,296 addresses. In the IPv6 protocol the addresses is 128 bit long. This makes the total number of possible addresses to 2^{128} ~$3.4 * 10^{38}$ addresses.

As the set of available IPv4 addresses were being rapidly depleted there was a clear need to migrate to another Internet protocol. The very large number of addresses that would be available with IPv6 would hopefully last for quite a while. Additionally, these addresses were to be allocated in a hierarchic manner to minimize the size of the global routing tables[7]. However, there are exceptions where this hierarchical structure is not followed. An organization can be assigned Provider Independent (PI) addresses if they intend to use multihoming. These PI addresses are smaller blocks assigned separately directly from Regional Internet Registry (RIR)[8]. To be assigned PI addresses from the Réseaux IP Européens Network Coordination Center (RIPE NCC) the organization must demonstrate that it will be multihomed[9]. Another advantage is that the organization does not need to change all its IP addresses when changing Internet Service Provider (ISP).

2.2 Address notation

There are some differences in the notation between IPv4 and IPv6 addresses. IPv4 is represented in a dot-decimal notation where every byte in the address is represented by a decimal number. These numbers are demarcated with dots. In IPv6 two bytes are represented as a four digit hexadecimal number separated with colons. As the addresses are 128 bit, or 16 byte, long there can be up to seven colons. Leading zeros can be omitted in both IPv4 and IPv6. In IPv6 one or several fields of zeroes can be compressed and represented with two colons. However, this can only be done once.

Example:

IPv4 address: 192.168.10.5

IPv6 address: 2001:db8:0000:0102:0033:0000:0000:00ab

2001:db8:0:102:33:0:0:ab

2001:db8::102:33:0:0:ab

2001:db8:0:102:33::ab

Prefix length is represented by a slash and the length in number of bits in both IPv4 and IPv6.

IPv4 prefix: 192.168.10.0/24

IPv6 prefix: 2001:db8:0:102::/64

2.3 Simpler header

The header of IPv6 was made a fixed size of 40 bytes, while the IPv4 header could be between 20 and 60 bytes depending on the options used. Some fields have been removed from the header, such the header length (which is unnecessary as it is constant), identification, flags, fragment offset, header checksum, specifically and the options field[10].

The identification field along with the fragment offset field has been moved to a fragment header extension header[11]. The third bit in the flag field that indicates if there are more fragments[12] or not is replaced by an M flag in the fragment header extension header[11]. In IPv4 fragmentation is done if needed by the routers along the way, whilst with IPv6 fragmentation is only allowed at the source.

The header checksum is removed, as IPv6 relies on upper level protocols, lower layer checksums and error correction schemes, or security extensions for data integrity[11]. This also means that recalculation of the checksum at every hop, as the Time To Live field is changed at every hop, is no longer needed.

Options are no longer defined in the IPv6 header, but rather there are extension headers that are equivalent to IPv4 options.

2.4 Version

The version field is a 4 bit field that indicates the version of the Internet Protocol. For IPv6 the version field is of course 6 and for IPv4 it is 4.

2.5 Traffic class

An 8 bit traffic class field can be used by hosts or routers to mark packets so theses packets can be distinguished and given special treatment[11]. It replaces the Type of Service field in IPv4. Nodes are allowed to change all of these bits. Nodes that do not support a specific use should ignore this field and leave it unchanged. There are proposed standards for using the bits in this field, see RFC 2474[13].

2.6 Flow Label

A 20 bit flow label field may be used to label packets from a source as belonging to a certain flow that all require the same treatment[11]. Nodes that do not support a specific use of this field should ignore this field and leave it unchanged. Nodes that do not support flow labels shall set the field to zero when sending any packets. RFC 6437[14] is a proposed standard specifying the use of this field. No equivalent field is present in the IPv4 header.

2.7 Payload length

A 16 bit payload length field specifies the length of the data carried, including any extension headers, in numbers of bytes[11]. This mean that up to 65,535 bytes of payload can be carried. However, there is a Jumbogram extension header that allows for even larger packets, for details see RFC 2675[15].

2.8 Next header

An 8 bit next header field identifies the type of the header directly after the IPv6 header. It replaces the protocol field in the IPv4 header[11]. The values corresponding to different protocols are specified in RFCs (the latest being RFC 1700), but have been replaced with an online database[16], [17].

2.9 Hop limit

An 8 bit hop limit field indicates how many hops are left before the packet should be dropped[11]. The value is decreased by one every time it passes through a router. The time to live field in the IPv4 header has the same functionality, but the field was renamed to reflect the actual use of the field.

2.10 Source and destination

The source and destination IP address fields simply indicate the source and destination addresses of the packet. The fields are 128 bits for IPv6. One difference from IPv4 is that in IPv6 the address in the destination field might not be the final destination if a Routing header extension header is used[13].

2.11 Extension headers

IPv4 allows for options that are carried inside the header. The minimum size of the IPv4 header is 20 bytes and the maximum 60 bytes. This limitation is due to the fact that the Internet Header Length field that specifies the total header length in 32 bit words, but is only 4 bits in size. The maximum value is 15, thus 15 * 4 bytes = 60 bytes. This poses restrictions on some of the options, such as the strict source and record route options. The record route option records the IP addresses of the routers the packet traversed. That means that only (60 - 20 - 4)/4 = 9 IP addresses can be recorded which is a serious limitation.

Instead of carrying options inside the header, IPv6 exploits extension headers that are placed between the IPv6 header and the next protocol header. Not only are options dealt in this way, but also fragmentation. As a result all the special fields in the IPv4 header used for fragment are no longer needed, making the header simpler and of a constant size. The next header type is indicated by the next header field. An IPv6 datagram can have an arbitrary number of extension headers. The extension headers are always a multiple of 8 octets long. If there is more than one extension header, then RFC 2460 states that the following order should be used:

1. IPv6 header
2. Hop-by-Hop Options header
3. Destination Options header (if the options are to be processed by the first router and succeeding)
4. Routing header
5. Fragment header
6. Authentication header
7. Encapsulating Security Payload header
8. Destination Options header (if the options are only to be processed by the final destination)
9. upper layer header

2.11.1 Fragmentation header

With IPv6 fragmentation is only allowed at the source and not by any router along the path, unlike IPv4 which permitted routers to fragment packets. Fragmentation is only to be done if the application cannot adjust the packet size to the measured path maximum transmission unit (path MTU). A next header value of 44 indicates that the next header is a fragment header[11]. Otherwise fragmentation functions much as in IPv4. There are six fields in the header: *next header*, *fragment offset*, *M*, *identification*, and two reserved fields. The *next header* field is 8 bits and indicates of what type the next header is. *Fragment offset* is 13 bits and indicates the fragment's offset in units of 8 octets to the start of the fragmented packet, just as the field with same name in the IPv4 header does[12]. The *M* field is 1 bit in size and indicates if there are more fragments or if this was the last fragment[11]. The last fragment is indicated by the third bit in the flags field in the IPv4 header. The *identification* field is 32 bits as opposed to the IPv4 16 bit field. An identification value is generated for all packets that need to be fragmented. This helps with the reassembly at the end node and has the same function as in the IPv4 header. The increased size of the identification field is to accommodate more simultaneously outstanding packets, due to the might higher link data rates today than when IPv4 was defined.

2.11.2 Hop-by-Hop Options header

The Hop-by-Hop Options header is used to carry options that all nodes the packets traverse must examine[11]. A next header value of 0 identifies the next header as a Hop-by-Hop Options header. There are three fields in the Hop-by-Hop Options header: *next header*, *hdr ext len*, and *options*. The *next header* field is 8 bits and identifies the immediately following header. The *hdr ext len* field is 8 bits and indicates the length of the whole header in units of 8 octets excluding the first 8 octets. The *options* field is of variable length containing Type Length Value (TLV) encoded options.

One of the options defined is the Jumbo Payload option that allows a source to send packets with payloads ranging from 65,536 octets to 4,294,967,295 octets (4 Gigabyte)[15]. There is also a Tunnel Encapsulation Limit option that specify how many times the packet is allowed to be encapsulated[18].

2.11.3 Routing header

The routing header contains a list of nodes that the packet should traverse[11]. This option is very similar to the IPv4 Loose Source Route option. A next header value of 43 identifies the next header as a Routing header. The routing header consists of the fields: *next header*, *hdr ext len*, *routing type*, *segments left*, and *type-specific data*. The next header field as usual indicates what type the following header is and the field is 8 bits long. The value of the *hdr ext len* field is the size of the header in 8 octet units, excluding the first 8 octets, and this field is 8 bits long. The *routing type* is an 8 bit field and identifies a routing header variant. The *segments left* field is the number of nodes left to visit or the number of segments left in the type-specific data field. The *segments left* field is 8 bits long. The *type-specific data* is of variable length and the format depends on the routing type that is used.

2.11.4 Destination Options header

This header carries options that only need to be examined by the destination node(s)[11]. A *next header* value of 60 means that the immediately following header is a Destination Options header. The Destination Options header is made up of the fields: *next header*, *hdr ext len*, and *options*. As with the previous extension headers the next header field is an 8 bit filed that indicates the type of the immediate following header. The 8 bit *hdr ext len* field that indicates the length of the header, expressed in 8 octet units excluding the first 8 octets. The options field is of variable length and contains TLV encoded options.

2.12 Multicast, unicast, and anycast

Multicast, unicast, and anycast addresses are types of addresses that are used for different purposes. Each will be described below. Broadcast, multicast, and unicast addresses are used with IPv4. Anycast is a new type, and the functionality that broadcast addresses served in IPv4 has been replaced by multicast addresses in IPv6.

2.12.1 Multicast

One way of transferring, and replicating, a packet to multiple destination addresses is to multicast the packet. Duplicates of the packet will be created as the packet traverses the network, thus distributing the load over the nodes (and as a byproduct of distributing the load over the physical network itself). The replication can be performed by routers and/or switches and the source sends each packet only once.

A multicast address identifies a set of, usually different nodes', IPv6 interfaces. A packet sent to such an address is delivered to all the interfaces belonging to that set[19].

In IPv6, multicast has been made a mandatory part of the protocol[20] (unlike IPv4 where it is optional). Along with improvements to widen the support for multicast addressing, multicasting has replaced broadcast addressing in IPv4 – as broadcasts cause problems in most networks[13].

2.12.2 Unicast

A unicast address identifies a single IPv6 interface. A packet destined to such an address is delivered to the interface that is identified by this address[19].

2.12.3 Anycast

One of the new concepts introduced in IPv6 is anycast addresses. The definition of multicast is to send to all the interfaces in a group and unicast sends to a specific interface, while anycast packets are routed to any interface in the group. This routing of an anycast packet should be done as efficiently as possible, thus the packet will be routed to the nearest interface (the distance is calculated according to the routing protocol that is being used). The key concept is that the anycast group consists of any interface that can respond to a request sent to a single anycast IP address[20].

2.13 ICMPv6

Just as in IPv4, the Internet Control Message Protocol (ICMP) in IPv6 provides very useful information about the network. For example, *Traceroute*[21] makes use of control messages. ICMP error messages for *destination network/host/port unreachable* are well known. Probably one of the most fundamental diagnostic functions is to test the connectivity between nodes in a network via *ping* using ICMP Echo Request/Reply.

ICMPv6 is a requirement for every node that is to run IPv6[13]. ICMPv6 has a set of new features not in ICMPv4. An important new feature is Neighbor Discovery (ND). ND handles a variety of operations such as address autoconfiguration, determining the link layer address of nodes on the local network, and detecting routers and any alteration of link-layer addresses. ND provides resolution of network layer addresses into link layer addresses, similar to the Address Resolution Protocol (ARP) of IPv4. Further details of ND are given in the following subsection.

2.13.1 Neighbor Discovery

Neighbor Discovery (ND)[22] comes with modifications, improvements, and new features when compared to the related IPv4 protocols. The ND protocol performs functions similar to ARP, ICMP Router Discovery, and Router Redirect, but with improvements. The function of Neighbor Unreachability Detection (NUD) has been implemented which serves the purpose its name suggests: it is a mechanism for detecting if a neighbor is reachable or not. This could have been done with ICMP Echo Request and Reply in IPv4. Another function that has been introduced is Duplicate IP Address Detection (DAD) which will be described in section 2.12.3.

In order for nodes to be able to communicate over a local network they must discover each other on the local link. To do this ND[13] provides the following services:

- Resolves layer 2 addresses of nodes on the same link,
- Discovers adjacent routers that can forward packets, and
- Monitors the neighbor's reachability and changes in link-layer addresses.

Improvements that have been made over the IPv4 version of the similar functions include[13]:

- No need to get Router Discovery information from the routing table since router discovery is now a part of the base ICMP protocol.
- No need to send an additional ARP request (in IPv4) for a node that has received a Router Advertisement in order to get the router's link-layer address since it is included in the packet. The same is true for an ICMPv6 redirect message, as this message contains the link-layer address of the new next-hop router interface.
- No need to configure subnet masks since that information (the prefix of a link) is carried by Router Advertisements.
- Easy renumbering of a network by using ND's functionalities. Ability to set up new prefixes and addresses, with the old ones automatically deprecated and removed.
- Router Advertisements are used in stateless autoconfiguration and can notify hosts when to use stateful address configuration (e.g., DHCPv6).
- The MTU of the link can be advertised by routers.
- NUD is implemented to detect failed connectivity (i.e., that a neighbor is unreachable). Detects any alteration of link-layer addresses on interfaces and traffic will not be sent to a neighbor that is unreachable. It will also detect if a router is down and switch to an active router. This eliminates the problems that arise with old entries in ARP caches.
- Routers are identified by its link-local addresses sent in router advertisements and ICMP redirects. Hosts will therefore be able to keep their associations even if renumbering or use of a new global prefixes occurs.
- ND messages have a hop limit set to the maximum permitted value of 255. ND datagrams routed over one (or several) hop(s) are not valid, hence datagrams with a hop limit that differs from 255 are discarded. Since it is not possible to set a higher value than 255, packets will be ignored following a decrement of the hop limit. ND is thus immune to a denial-of-service attack coming from the outside the local link.
- DAD is implemented to detect IP address conflicts on a link.
- Application of standard IP authentication and security mechanisms.

Neighbor Solicitation

A Neighbor Solicitation (NS) message is sent when a host connects to the network. It is sent to the multicast address of other hosts, asking for their link-layer (MAC) address. This replaces ARP in IPv4. When performing the NUD function, the NS is sent to a unicast address, a response verifies reachability.

Neighbor Advertisement

Upon receiving a NS, the host responds with a Neighbor Advertisement (NA) message containing its link-layer (MAC) address.

NSs and NAs are also used in the DAD mechanism.

2.13.2 Router discovery

The router discovery process discovers active routers on the local link[23]. A router sends out Router Advertisement (RA) messages periodically to inform nodes that it is active. The waiting time between the advertisements can be skipped by the host by sending a Router Solicitation (RS), which will trigger the router to send a RA regardless of the interval between the regular RAs.

Router Solicitation

A Router Solicitation (RS) message is sent to the "all routers" multicast address of FF02::2 which all routers have to listen to. Hosts will ignore these messages, as they do not belong to this multicast group.

Router Advertisement

The Router Advertisement (RA) message contains the link-layer address of the router (source) and a value for the Maximum Transmission Unit (MTU). A RA is sent regularly (unsolicitated) to the "all nodes" multicast address of FF02::1 for the local network as its destination. It can also be sent as a unicast response (towards the requestor) when a RS arrives.

2.13.3 Duplicate Address Detection

Duplicate (IP) Address Detection (DAD) is used to ensure that the temporary address that a host has chosen is indeed a unique IPv6 address[24]. The validation is performed by the host multicasting NSs to the temporary address it has chosen for itself. If the host then receives a NA with the temporary address as a source address, then the address is not unique. In that case another node has received the NS, detected that the temporary address is used by itself and sent a NA. If the NA response does not come, then the temporary address is unique and can be assigned to the interface. In the case of a Cisco router the number of DAD attempts before the address is established as unique can be specified with the command *ipv6 nd dad attempts* <*value*>[25].

2.13.4 Autoconfiguration

Hosts (that are not manually configured) need a Dynamic Host Configuration Protocol (DHCP) server in IPv4 to provide an automated mean to assign an IP address to the host and for the host to get the other information needed to communicate via the network. The IP address, subnet mask, and default gateway are the most fundamental information that is usually provided by DHCP. The address of a Domain Name Service (DNS) sever is another example of information that the host may need. It is up to the network administrator to decide what is the best (and maybe the most convenient) solution to implement.

Autoconfiguration in IPv6 was defined so that there is no need for a DHCP server and the hosts will still be automatically configured. This simplifies administration, therefore hosts will be less time consuming to configure and hosts can communicate via a link local IPv6 address even in the absence of any infrastructure. ISPs use DHCP servers in order to dynamically allocate addresses. Eliminating the need for DHCP servers improves reliability, as only the router infrastructure is necessary and it is located nearer the host and has better fault tolerance[26]. However, autoconfiguration in IPv6 does not provide DNS information. This is a severe drawback since a lot of commonly used applications rely on DNS. Fortunately they are multiple ways to bootstrap DNS operations (for example, using public DNS server, anycast discovery of authoritative DNS servers – see RFC3258 [27], Multicast DNS (mDNS),…).

The host (or any other "network-aware" device) creates a temporary ("tentative") address that is to be used as its final address if the DAD process is successful[24]. When the DAD mechanism is completed and no NA has come in response, then the address is assigned to the node's interface (now called a "preferred address"). The address is valid during its lifetime and should not be used in new connections if it has expired (i.e., if its state is "deprecated"). The lifetime of an address will usually not expire since new RAs update the lifetime. However, lack of RAs will cause the lifetime to eventually expire.

2.14 IPv6 and DNS

The Domain Name System (DNS) maps domain names to IP addresses. These mappings are stored in resource records. A new record was needed for storing IPv6 addresses mapped to domain names[28]. The type of record mapping IPv4 addresses is called an A record so, naturally IPv6 addresses being four times as long as an IPv4 address, the records for IPv6 are called AAAA or quad-A records. The type value for AAAA records is 28.

An example AAAA record is:

example.com. IN AAAA 2001:db8:0:1:2:3:45:6789

An AAAA query has also been defined for fetching AAAA records from DNS servers[29]. When making a query such as MX type queries, this means that you want the canonical name of a mail server with a certain alias, then the DNS server sends in the additional section of the answer an A record providing the IP address for the mail server[6]. These types of queries are redefined to add both relevant A and AAAA records[29]. Returning both answers when possible is done for efficiency reasons[30].

For reverse lookups the special domain in-addr.arpa is defined for IPv4[31]. The domain name is suffixed to the IPv4 address represented in dotted-decimal form in reversed order in a PTR record type. For example the domain example.com with the IPv4 address 10.15.20.25 would have the following PTR record[28]:

25.20.15.10.in-addr.arpa. IN PTR example.com

For IPv6, the corresponding special domain is IP6.ARPA[29]. The domain name is suffixed to the IPv6 address in hexadecimal form with every digit in reverse order in a PTR record type. Worth noting is that no zeroes are compressed. The domain example.com with the IPv6 address 2001:db8:0:1:2:3:45:6789 would be represented like this:

9.8.7.6.5.4.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.IP6.ARPA. IN PTR example.com.

There was a record called A6 that also was used for representing IPv6 addresses, but RCF 2874 defining A6 records has been moved to historical status as of March 2012 by RFC 6563[32]. This means that A6 records should **not** be implemented **nor** deployed by operators. The code for A6 record (38) has been updated from experimental to obsolete in the parameters registry for DNS. Some of the reasons to deprecate A6 records are that it is confusing when deploying IPv6 to have two types of records to choose from and having two types of records leads to greater security risks, increased difficulty with respect to maintenance, and increased resolution latency.

One important fact is that DNS servers do not need to be addressable by an IPv6 address to retrieve an AAAA record and vice versa[29]. This is very useful during the transition from IPv4 to IPv6.

2.15 Avoiding NATs

Network Address Translation (NAT) provides the ability to hide a realm of private IP addresses behind a single public IP address[6][33].

Given a private network behind a NAT-enabled router, the IP address within the private network serves only a local purpose and cannot be used outside of it. The router appears as single device with its public IP address. The packets leaving the network all have the router's IP address as their source address, and all packets destined towards the network will have the router's IP address as their

destination address. Since all traffic has the same destination address arriving at the router, the router must use a NAT translation table to be able to forward the data to the correct host within the private network. The NAT translation table consists of a pair of internal and external IP addresses and port numbers. It is the port number that is used as the key to translation. As a result if there are a large number of hosts behind the NAT there can be problems due to the limited port number address space (2^{16}) for a given protocol.

As an example:

NAT translation table	
WAN side	LAN side
85.255.31.209, 5555	192.168.0.2, 3333
…	…

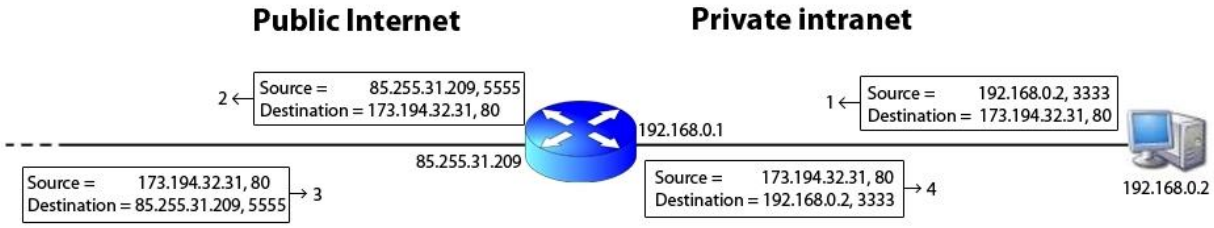


Figure 2.15.1 Example NAT translation table for a simple network configuration

If a host with private IP address 192.168.0.2 sends a request to a web server with the public IP address 173.194.32.31 and port number 80. The host sets the source port number of the datagram to some local TCP port number, such as 3333 and sends it. When the router receives the datagram, it replaces the original source IP address with its own public IP address 85.255.31.209 and allocates a new entry in its NAT translation table, perhaps with a new source port number 5555, in which case it replaces the source TCP port number with 5555, recomputes the checksums, and sends the resulting packets towards the web server. When the web server responds it sends a packet towards the router (IP address 85.255.31.209, port 5555). When the router receives this packet it looks in the translation table to find the corresponding IP address and port number of the host using the destination IP address and port number in the packet that the web server sent. In this case the router will find a matching entry, thus it will replace the destination IP address with 192.168.0.2 and it will replace the port number 5555 with port number 3333, recomputes the checksums, and forwards the resulting packet towards the host.

People within the IETF argued against the use of NAT for several reasons (see pages 387-388 of [6]). One of them being that IPv6 should be used instead of this short term, and patchy, solution to the shortage of IPv4 addresses. More importantly, NAT breaks the end to end property of IP communication, hence NAT causes problems for services such as peer-to-peer (P2P) file-sharing applications and voice over IP (VoIP) applications when both endpoints are behind different NATs.

2.16 IPv6 Security

There was only limited consideration of security when IPv4 was designed. IPv4 was meant for use by a closed community and it was not thought that IPv4 would be as widely deployed as it is today. However, security became a very important part of the specification of IPv6. This meant that security mechanisms, that were not part of the original IPv4 protocol, had to be applied in order to provide the desired security.

With this in mind when designing IPv6, built-in security was considered a requirement. This is achieved with IPsec. However, IPv4 is also able to use IPsec, but unlike the case for IPv4 – Ipv6 requires that every implementation of IPv6 include support for IPsec.

There are an extremely large number of attacks that can be performed on networks today. Some of the most common are[13]: denial of service; fabrication, modification, or deletion; and eavesdropping. Each of these types of attacks is described in the paragraphs below.

Denial of Service

A denial of service (DoS) attack is used to prevent the targeted service from being available. A DoS attack is easy to detect when the service becomes unavailable. Unfortunately, it is difficult to prevent a DoS attack and it is even difficult to detect the onset of such an attack. Common DoS attacks include overloading the target, i.e., to subject it to a load that is greater than it is capable of handling (thus slowing down valid service requests or perhaps even blocking them being handled at all), or disrupting vital network information (such as routing information) which can cause unexpected behavior of the network if nodes do not receive information that is current and operates based upon obsolete information.

Fabrication, modification, or deletion of information

These attacks be can used to forge information in order to fool someone/something to behave the way the attacker wants or just delete certain (or all) information. These attacks are hard to detect unless there is some form of sequence number and authentication.

Eavesdropping

Eavesdropping is often impossible to detect. An attacker can simply intercept packets and hence gain information without the knowledge of the victims, just as a person would eavesdrop on a conversation between two unsuspecting individuals. The man-in-the-middle attack is performed by a person identifying himself as “person B” in the conversation between person A and B in the eyes of “person A” and vice versa. A and B (who each think that they are talking directly to each other) sends their information to the man in the middle who relays the information flowing to and from them to the other party, the real person B. The two parties (A and B) will not discover that there is something wrong since they are getting all the information (as is the intruder). The intruder is now able to learn information that can be used against the victims, such as passwords.

While IPv6 provides new security features, it is still not flawless. Its new mechanisms also introduce new security issues. A host that has been able to gain access to a network could still cause a lot of damage by exploiting messages sent within a network. This includes forgery of neighbor advertisements (to conduct a man-in-the-middle-attack) and flooding of packets on the link and generating false router advertisements (two forms of DoS-attacks). The latter could even cause the target host to crash (i.e., fail to continue to operate correctly)[34].

The vulnerability of the ND protocol has caused the introduction of the Secure Neighbor Discovery (SEND) protocol[35]. Its purpose is to protect against threats when the link does not have physical security. To protect the ND protocol messages, SEND utilizes cryptographically generated addresses, RSA signatures, and nonces.

2.16.1 IPsec

IPsec is a framework that provides secure communication in networks at the network layer. IPsec is a mandatory component for all implementations of IPv6[36]. However, IPsec can be used with both IPv4 and IPv6; as it was designed for both protocols, but it needs to be retrofitted to IPv4 stacks

already in existence[13]. There are two types of IPsec headers: Authentication Header (AH) and Encapsulation Security Payload (ESP) header.

AH authenticates parts of the header and the payload[37]. AH can only protect the fields that are not intended to be changed, so called immutable fields. The AH header is in the same format as the other extension headers. It has a field indicating what type of the immediately following header is and the length of the AH header. However, the payload length field indicates the length in 4 octet units instead of 8 as with the other extension headers. The AH header is inserted between the payload and the IPv4 or IPv6 header[37]. The value of 51 in the next header field in case of IPv6 or the protocol field in case of IPv4, indicates that the next header is a AH header[38].

ESP does encryption and/or authentication of the payload of an IPv4 or IPv6 packet[37]. If you want only integrity protection you could use ESP for that by using the null encryption algorithm for your encryption. The ESP header and trailer, with the encrypted payload in between, are located after the IPv4 or IPv6 header. The value of 50 in the next header or protocol field indicates that the immediately following header is an ESP header[38].

AH and ESP can be used in two different modes: transport mode and tunnel mode[37]. In transport mode the IPsec information is added directly after the IP header in IPv4. In IPv6 the IPsec information is positioned after the IP header and the extension headers (except for the Destination Options header under certain circumstances mentioned in the extension header section), and before any upper layer protocols. Transport mode is mostly used when two end systems directly communicate – thus providing end-to-end security. In tunnel mode there is an IP header added outside the original header specifying the IPsec source and destination. In tunnel mode the IPsec information is added directly after the outer IP header and before the inner IP header. Tunnel mode is often used to create a secure tunnel between firewalls or between an end node and a firewall. The latter case can occur when a mobile user wishes to access the corporate network when they are away from their office. In this case the user will use IPsec to secure their communication to the corporate firewall thus creating an IPsec based virtual private network (VPN).

Before you can start securely sending packets a security association (SA) needs to be established[37]. The SA can be manually configured or established with IKE (Internet Key Exchange). The details of IKE are outside the scope of this report. For details about IKE see [37].

Implementation

IPsec can be implemented in three different ways:

- Integrated structure,
- Bump-in-the-stack (BITS), and
- Bump-in-the-wire (BITW).

Of these alternatives, the *integrated structure* is considered to be the best way, while *BITS* and *BITW* require software and hardware solutions[39].

Integrated structure

The preferred way of implementing IPsec is integrated into the IP stack, as the IPsec protocols are integrated with IP which will result in an easy implementation. As mentioned, IPsec a mandatory part of IPv6 thus making it an integrated part of any IPv6 implementation.

Bump-in-the-stack

Bump-in-the-stack (BITS) is a technique that is usually applied by IPv4 hosts. This approach implements IPsec as a separate layer between IP and the data link layer. IPsec perform its security transformation on the datagrams as they pass from the IP layer to the data link layer and the reverse at the destination.

The benefit of using BITS is that any IP device can adopt IPsec with the addition of suitable software. The downside is that using software to intercept the datagrams requires extra computing compared to the integrated structure. For example, in Windows one can implement an NDIS Device Driver that provide IPsec functionality (see the Windows OS file "ipsec.sys").

Bump-in-the-wire

Bump-in-the-wire (BITW) relies on hardware to implements IPsec functionality. Consider the scenario in Figure 2.2. In this scenario the routers do not implement IPsec (Network 1 and 2). Therefore we introduce an IPsec device (IPsec device 1 and 2) between the router and the Internet to provide IPsec functionalities. As datagrams passes out though the IPsec device, IPsec is applied; as datagrams passes

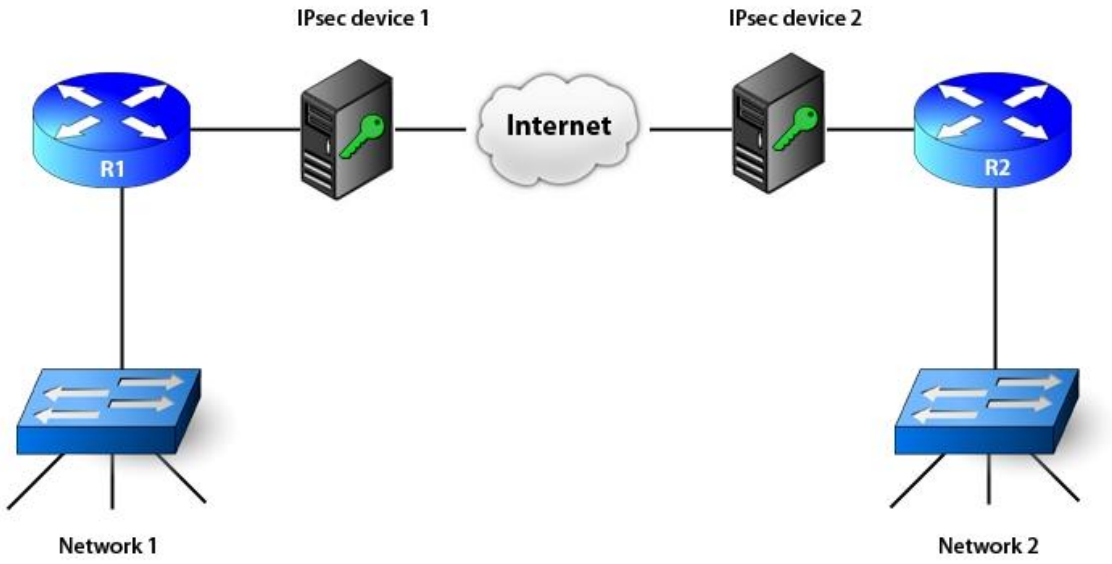


Figure 2.16.1.1 BITW

in through the IPsec device, IPsec is removed. The existence of an IPsec tunnel between the two IPsec devices is invisible to the routers.

The benefits of BITW are the same as for BITS. The downside is complexity and cost: new hardware needs to be bought, integrated into the existing network and configured. However, an advantage is that **no** other changes need to be made in the network. This assumes that R1 and R2 only want to communicate with each other. If they also way to send packets to and from the rest of the internet, then there needs to be a way to tell the IPsec devices which packets to **not** tunnel.

Both BITS and BITW provides the same functional outcome in the end, but one has to decide which alternative is best suited to a given application scenario. As mentioned earlier, the integrated structure (IPv6) is the preferred way of implementing IPsec. However, when IPsec has not been integrated – BITW and BITS provide a way of adding IPsec after the fact.

3. Routing protocols and IPv6

To be able to send IP packets to other subnets the router needs to know where to forward the packets so they get to the correct destination. Routing protocols solves this problem. In this section we are going to go through the routing protocols available to distribute connectivity information for IPv6.

3.1 RIPng

The Routing Information Protocol (RIP) is a commonly used intra domain routing protocol in small to moderate size networks (the maximum diameter of a network is 15 hops). RIP uses a Bellman Ford or other type of distance vector algorithm to calculate the best path in a network. RIP has its limitations, such as the low maximum number of hops for a path, the path cost is based *only* on the number of hops, and it has slow convergence[6]. Despite these limitations RIP is used because it is generally available and easy to configure.

RIPng is based on RIP, and thereby suffers from the same limitations, but is intended for IPv6 networks[40]. RIPng is not intended to be used in networks with both IP protocols. RIPng send its messages over UDP to port 521. Unsolicited response messages are sent every 30 seconds containing the whole routing table. Messages are also sent when triggered by route changes. There are two timers per route in the routing table, a timeout and a garbage-collector time. When the timeout expires the route is invalid, but it is kept in the routing table for a short amount of time so neighbors can be notified. When the garbage-collector time expires, the route is removed from the table. When a route is established the timeout timer is set and every time an update message received the timeout timer is reset. If the timeout is not reset after 180 seconds, then the route is expired and deleted.

3.2 OSPFv3

OSPF (Open Shortest Path First) is a widely used intra domain routing protocol based on Dijkstra's least-cost path algorithm for calculating the best paths to subnets[6]. Every router running OSPF makes its own complete map of the network before calculating the best path with itself as the root node. When routing information changes, or upon initialization, the router generates a link-state advertisement representing all link-states of the router. Link-states are exchanged by flooding. Every router that receives a link-state update saves it in its database and sends a copy to its neighboring routers. Then the best path is recalculated.

With OSPF an Autonomous System (AS) can be divided into areas[41]. Subsets of the routers are assigned to different areas. One, or more, of the border routers are set to be part of a backbone area that all communication between the areas goes through.

With OSPFv3, also known as OSPF for IPv6, much of the fundamental mechanism of OSPFv2 (OSPF for IPv4) remains unchanged[42]. In OSPFv3 protocol packets and in the main link-state advertisement types addresses are removed, making the core independent of the network-layer protocol. However, OSPFv3 is carried directly over IPv6, so IPv6 must be enabled on the interface. IP addresses are only present in the payload section. Authentication has been removed, instead the idea is to rely on the authentication provided by IPsec in IPv6.

With OSPFv3 on Cisco routers, one router process per address family (IPv4, IPv6, etc.) is allowed on the same interface[43]. This means that OSPFv3 can pass IPv4 and IPv6 routing information over the same network with dual stacks.

3.3 Integrated IS-IS

Integrated Intermediate System to Intermediate System (IS-IS) is another intra domain routing protocol[13]. The integrated part means that you can use the same routing protocol for several address families. This is possible by using a data field containing TLV (Type Length Value) entries. In the TLV entry the type of protocol is specified, the length, and the value. The number of the type of the network layer protocol is specified by ISO (International Organization for Standards). The value for IPv6 is 142.

Integrated IS-IS uses the same algorithm for all address families[44]. This routing protocol advertises link-state information to create a topology of the network, just as was the case for OSPF. As with OSPF, routers can be divided into areas[45]. Communication between the areas is made by level 2 routers that form a backbone. Routers that only know the topology within an area are level 1 routers.

3.4 BGP-4

Border Gateway Protocol 4 (BGP-4) is a inter domain routing protocol. It is used to transfer information about reachability to other networks between Autonomous Systems (ASs). BGP speaking routers peer with each other over TCP, thus BGP can be used over both IPv4 and IPv6[46]. The Network Layer Reachability Information (NLRI) field in the update message carries the prefixes and some attributes associated with them, such as the mandatory NEXT_HOP attribute, hence they are still IPv4 specific[47]. Fortunately, there exist multiprotocol extensions that define two new attributes: Multiprotocol Reachable NLRI and Multiprotocol Unreachable NLRI. These new attributes are able to carry information about what destination is reachable as well as not longer reachable. All BGP speakers still need to have an IPv4 address for certain functions[47].

3.5 MPLS

Multiprotocol Label Switching (MPLS) can forward packets from any network layer protocol, but MPLS is not really a routing protocol. Incoming packets are assigned to a Forwarding Equivalence Class (FEC). A FEC is a subset of all the packets that the router can forward. A FEC can be all packets destined to a specific address or all packets destined to this address that have a particular priority or distinguishing characteristic. All packets belonging to a FEC are assigned a specific label.

A MPLS header is inserted between the link layer header and the IP header. Subsequent forwarding is based on the label in the MPLS header. The route or Label Switched Path (LSP) is set up in advance with help from signaling protocols, such as RSVP, LDP, or BGP. RSVP can operate over both IPv4 and IPv6[48]. LDP and BGP use TCP or UDP as a transport protocol, so they can also operate over both network layer protocols[46], [49]. MPLS can thus be used to transport IPv6 packets over an IPv4 only network or vice versa.

MPLS is often used to create Virtual Private Networks (VPN). MPLS is mainly used in provider networks for traffic engineering[50].

4. Upper layer protocols

The effects of changing the network layer protocol to IPv6 from IPv4 on upper layer protocols are minimal.

Where this change does matter is when transport protocols use the IP header to calculate checksums[13]. Application layer protocols may compute checksums that include elements of the IP header. TCP, UDP, and DCCP uses a pseudo-header to calculate their checksum. In the specification for IPv6 there is also a pseudo-header specified for TCP and UDP (DCCP uses the same pseudo-header for IPv6[51]). The pseudo-header contains source address, destination address, upper layer packet length, zero, and next header fields[11]. The zero field is padding. Extension headers are not included in the pseudo-header. If the routing header extension header is used, then the destination address is the address for the ultimate destination. The checksum in a UDP packet is not optional when the UDP packet is originated by an IPv6 node.

FTP was designed to be used over IPv4[13]. Some of the commands use address information so they needed to be replaced for use over IPv6. RFC 2428 specifies an extension to enable FTP to work over IPv6. However, FTP works for both IPv4 and IPv6 with the extension.

Another popular protocol is Jabber or Extensible Messaging and Presence Protocol (XMPP). Jabber is used in a variety of applications, such as: instant messaging, presence, multi-party chat, voice and video calls, etc.[52]. No change to jabber is necessary to support IPv6[53].

5. Transition

Since there is such a large difference between IPv4 and IPv6, they cannot communicate directly with each other. A system that is capable of handling IPv6 traffic can be made backward compatible, but an already deployed system that handles only IPv4 is not able to handle IPv6 datagrams. This means that a major upgrade process would need to take place, involving hundreds of millions of machines, in order to make a complete transition to IPv6. This is way too expensive and time consuming and in any case will not happen overnight. The network world will most likely see a gradual transition to IPv6, where IPv6 will be integrated into the IPv4 world that exists today. As an owner of a network, you can run IPv6 while others (such as your ISP) still run IPv4 or vice versa. Slowly, IPv4 nodes will be phased out leading to an all IPv6 network.

In order to make the transition smoother and to facilitate the coexistence of the two protocols when possible, several transition techniques have been introduced.

5.1 Dual-stack

Dual-stack[54], or dual IP layer, requires that a node implement both IPv4 and IPv6. The node can therefore communicate with IPv4 nodes as well as IPv6 nodes. The node has full support for both protocols and has the ability to turn one of the stacks off, thus making it into an IPv4- or IPv6-only node. In order to be configured with addresses, the node uses static or DHCP configuration for IPv4 and static or autoconfiguration and/or DHCP for IPv6. A so called IPv6/IPv4 node will have at least one address for each version of IP.

The Domain Name System (DNS) is usable with both IPv6 and IPv4. An IPv6/IPv4 node that wants to resolve a domain name and IP address requires a DNS sever that supports both A and AAAA records.

5.2 Tunneling

Tunneling IPv6 traffic over an IPv4 network is another possibility. This approach allows the IPv6 traffic to be encapsulated in an IPv4 packet and forwarded, creating an IPv6 tunnel over the IPv4 infrastructure[54]. A scenario where that would be useful would be if you as an IPv6 network user want to reach another IPv6 network, but have to traverse an IPv4-only network. A tunnel can be created as a solution for transporting your IPv6 traffic, from your IPv6 node to the destination IPv6 node, over the IPv4-only network. A “virtual link” is created and, from the perspective of the two establishing IPv6 nodes, this appears as a point-to-point link[18].

The different types of tunneling techniques can be categorized into two types: manually configured- and automatic tunneling. A point-to-point link has to be manually configured, as the name suggests. For automatic tunneling, an IPv6 node can dynamically tunnel packets by using a 6to4 address (see the next section).

5.3 6to4

One problem is that ISPs do not deploy IPv6 unless there is a great demand for it from their customers; however, the customers do not demand it since their applications work well on the current infrastructure (IPv4 with NATs)[55]. The current infrastructure is what the developers of applications adapt to since ISPs have not deployed IPv6. Fortunately, 6to4 is a technique that meets (most of) the IPv6 user’s requirements, while meeting the ISP’s requirements in terms of costs and administration.

As mentioned above, 6to4 is an automatic type of tunneling that does not require configuration of explicit tunnels. Between the so called 6to4 gateways (6to4 routers) the communication treats the intermediate IPv4 network as a point-to-point link[56]. The gateway, not the host, encapsulates the

IPv6 packet as the payload of an IPv4 packet. Configuration will therefore have to take place at the gateway, but this will make the 6to4 tunneling mechanism an automated process from the host's point of view. One, or more, unique IPv4 unicast address must be available to make this configuration work[13].

The address prefix reserved for the 6to4 mechanism is defined in RFC 3056[56] as 2002::/16. Following after these 16 bits are 32 bits containing the global IPv4 address converted to its hexadecimal representation. The IPv4 address 85.255.31.209 would be written as 55ff:1fd1 which when combined with the prefix yields the final address 2002:55ff:1fd1::/48. This address specifies a valid IPv6 prefix that can be used to support IPv6 subnets and hosts attached to these subnets.

This type of tunneling is not meant to last, merely used during the coexistence period of IPv4 along with IPv6.

5.4 IPv6 rapid deployment

IPv6 rapid deployment, or 6rd, addresses a limitation of the 6to4 technique: the encapsulation in a 6to4 relay router has to be done in order for the packet to reach hosts of an ISP that has implemented 6to4. The native IPv6 sites that do not route towards a 6to4 relay will not be able to communicate with that ISP's customers.

What has been done in 6rd, compared to 6to4, is that (1) only packets, coming from the global Internet, destined to customer sites of an ISP traverse its 6rd gateway and (2) all IPv6 packets traverse its 6rd gateway if they are destined to 6rd customer sites of this ISP and come from anywhere else on the IPv6 Internet[55].

5.5 Bump in the Stack

In the initial phase of IPv6 we cannot expect applications to have the same availability for IPv6 as for IPv4. The more widely used applications will likely start (and already have started) to support IPv6, but it will take a long time before we will see an equal deployment of supported applications for each protocol. The "Bump-in-the-Stack" mechanism provides IPv6 hosts (dual-stack hosts really) the ability to use existing IPv4 applications. Its goal is, as with all the other transition techniques, to ease the transition between the two protocols. It may also be a potential way in the future to use old IPv4 applications that do not have an IPv6 successor[57].

RFC 2767[58] specifies three modules that are to be used by the hosts: a translator, an extension name resolver, and an address mapper. These are each described in detail in the following paragraphs.

The translator implements the Stateless IP/ICMP Translation Algorithm (SIIT)[59] to do the translation between IPv4 and IPv6 packet headers. The IPv4 packet that is to be sent gets its header translated into an IPv6 packet header, the IPv6 packet is fragmented (due to the header's larger size) and then sent out. The translator works the same way when receiving an IPv6 packet, but the packet does not need to be reassembled (as the final destination will do this).

The extension name resolver creates a query based on the query sent by the application to resolve "A" records. The new query resolves both "A" and "AAAA" records and the new query is sent to the DNS server. The resolver returns an "A" record to the application (since it is an IPv4 application) as the application acts differently based on whether the "A" or the "AAAA" record is resolved. If the "A" record is the one that gets resolved, then the translator does not need to get involved and the record is returned to the application. If, however, the "AAAA" record is resolved, then the mapper is requested

to assign an IPv4 address that corresponds to this IPv6 address before returning the “A” record to the application.

The address mapper returns an IPv4 address upon request by the resolver or translator. It has an IPv4 address pool and a table that contain pairs of IPv4 and IPv6 addresses. When requested, the address mapper will return an IPv4 address from its pool that corresponds to the IPv6 address. If the request is for an IPv6 address that has no mapping entry, then a new entry will be dynamically inserted into the table. Since the pool contains private addresses, a translation of IP addresses (just like NAT provides) needs to take place[58].

5.6 IPv6 Tunnel Broker

To set up and administer a tunnel can be difficult. An IPv6 tunnel broker provides a tunnel service to its customers with the intention of helping them to connect to an existing IPv6 network[60]. As described in the tunneling section, the tunnel routes IPv6 traffic over IPv4.

5.7 Teredo

The Teredo service is a service that makes IPv6/IPv4 nodes behind IPv4 NATs able to experience IPv6 connectivity. This is done by encapsulating the IPv6 packet within an IPv4 UDP packet and thereby enabling it to be routed through NATs and over IPv4[61].

The problem with Teredo addresses is that NATs usually do not translate protocols other than TCP or UDP; hence an IPv6 packet encapsulated in an IPv4 packet has its protocol field in the header set to 41 (i.e., Protocol 41)[62]. This requires manual configuration of the NAT. However, since NATs translate UDP traffic, Teredo encapsulates the IPv6 packet within an IPv4 UDP packet and thereby enabling it to pass through NATs (as long as a valid mapping exists in the NAT).

Teredo is, like 6to4, an automatic type of tunneling technique. However, 6to4 does not solve all the issues introduced by NATs. RFC 4380[61] rejects the use of tunnel brokers as a way of solving this since the quality of service (QoS) may suffer. The traffic has to traverse the broker before reaching its destination point. This means that the broker must provide enough capacity to act as a relay and the traffic may have to make a substantial detour to traverse the broker’s link(s). It would therefore be better to locate relays within the ISP’s network to provide a direct path from source to destination without the need for a broker.

5.8 ISATAP

Intra-Site Automatic Tunnel Addressing Protocol (ISATAP) is a mechanism that connects nodes with a dual-stack over IPv4[63]. ISATAP uses the IPv4 network as a virtual link layer. It does not assume that IPv4 multicast is generally available, so it does not support IPv6 multicast. ISATAP is supposed to be used only within sites.

ISATAP creates an IPv6 address out of a specific prefix and the interface's IPv4 address [64]. The addresses is built up in the form ::0:5efe:W.X.Y.Z for private IPv4 addresses and in the form ::200:5efe:W.X.Y.Z if the IPv4 address is globally unique. The W.X.Y.Z part is the IPv4 address. ISATAP addresses can be merged with any valid 64 bit prefix. ISATAP supports both manual configuration and auto-configuration. The IPv4 address used to create the IPv6 address must only be unique on the network that the service is implemented in.

The Neighbor Discovery (ND) protocol that is used for IPv6 is specified in RFC 4861. This ND protocol is also used by ISATAP interfaces, but with some additions. For address resolution, router solicitations, and router advertisement multicast is usually used[65]. Neighbor discovery is trivial

since the IPv4 address is part of the IPv6 address. For a host to be able to send a router solicitation message it needs to know the IPv4 address of one or more ISATAP routers. This can be done through DHCP, DNS, manual configuration, etc. Once this address is known the router solicitation can be sent as unicast packet. The routers will only send router advertisements in response to solicitation messages and only as a unicast packet. Solicitation messages are sent regularly to known ISATAP routers by ISATAP hosts.

6. Software support

To be able to send IPv6 packets, operating systems need to have an IPv6 stack and applications needs to have support for IPv6. In this section we examine the state of IPv6 support in the most widely used software.

6.1 Operating systems

Linux kernel 2.2 (released in 1999) optionally implemented IPv6 although support for generating options and IPsec was initially missing[66]. Modern Linux kernels support IPv6[67]. Linux kernel 3.2 added support for transmitting IPv6 over IEEE 802.15.4 networks[68]. Windows XP implements IPv6, but IPv6 needs to be manually installed (with the command "netsh init ipv6 install")[69]. Windows Vista also includes a implementation for IPv6, but it is installed and enabled by default[70]. The same applies for Windows 7 and Windows Server 2008 R2[71]. This probably applies to Windows 8 as well. Windows 8 Consumer Preview and Windows 8 Server Beta are even said to have improved IPv6 functionality[72].

Apple's Mac OS X has had IPv6 support since Mac OS X v10.1 Puma[73], which was released September 25, 2001[74]. Mac OS X v10.3 Panther, released October 24, 2003[74] has IPv6 enabled by default[73]. FreeBSD has had "out-of-the-box" support for IPv6 since FreeBSD 4.0 was released in 2000[75]. In 2008 FreeBSD even released a kernel with only IPv6 support as a research project[76]. Solaris has supported IPv6 since March 2000 in Solaris 8[77]. In summary we can say that IPv6 dual stack support exists in all of the most widely used operating systems, and has existed in them for some time.

6.2 Applications

Having a network and an operating system that are compatible with IPv6 are not much good if your applications are not compatible with IPv6. We will examine some of the most common network applications and see if they support IPv6.

6.2.1 Web servers

According to Netcraft, a company that does regular web server surveys, in their survey of March 2012 420,337,139 sites utilize Apache web servers. This is 65.24% of all sites. While 13.81% of sites utilize Microsoft's IIS web servers, 10.15% utilize the nginx web server, and a further 3.28% of sites utilize Google's web server GWS[78]. Their survey results do not say what versions of these servers are in use. Google's web server is not open source[79] and they apparently do not discuss GWS[80], but it is probably compatible with IPv6 because Google's sub-domain ipv6.google.com is only accessible with IPv6. Apache has had IPv6 support since version 2.0[81] released on April 6 2002[82]. Microsoft's IIS server has had IPv6 support since IIS 6.0. The nginx web server does also support IPv6[83]. However, many sites such as www.kth.se are not accessible via IPv6.

6.2.2 Web browsers

If you want to fetch content from sites on a IPv6 only web server, or simply want to do it over IPv6 (on a web server with a dual-stack), your web browser needs to support IPv6. Today the most popular web browsers, such as Mozilla Firefox[84], Opera, Safari[85], Internet Explorer[86], and Chrome[87], all support IPv6.

6.2.3 Mail servers

Some of the most common mail servers are Microsoft's Exchange server, Sendmail, and Postfix. Exchange has had IPv6 support since Exchange 2007[88]. Sendmail has supported IPv6 since version

4.95[89]. Postfix has had IPv6 support since Postfix 2.0[90]. Other mail servers that support IPv6 natively are Exim, ZMailer, and the Courier mail server[84].

6.2.4 Mail clients

Popular mail client such as Microsoft's Outlook 2007[91] and Mozilla's Thunderbird[84] support IPv6. Other mail clients such as Apple Mail[92], Mutt, Sylpheed, KMail, Novel Evolution, Alpine, and Pine all support IPv6 as well[84].

6.2.5 DNS servers

Berkeley Internet Name Domain (BIND) is by far the most commonly DNS software used on the Internet[93]. BIND has had support for IPv6 since version 8.4.1[84]. Microsoft's DNS server is another common DNS server, with full IPv6 support from the version released with Windows server 2008[94]. The Windows 2000 DNS server also supports IPv6, but Windows 2000 did not include a native IPv6 stack[95]. Other DNS servers that support IPv6 natively are MaraDNS, NSD, and PowerDNS[84].

6.2.6 Firewalls

Firewalls that supports IPv6 first emerged in 2001 in the various operating systems, but more complete versions appeared in 2004[96].

There exist several open-source IPv6 firewalls[97]:

- MFfirewall based in Linux[98]
- M0n0wall based on FreeBSD[99]
- pfSense based on FreeBSD[100]
- FWBuilder provides a tool for filter setups[101]
- Checkpoint FW1 NGX R65 on SecurePlatform
- FortiGate/Fortinet (FortiOS 3.0 and up)[102]
- Juniper SSG (formerly Netscreen) (ScreenOS 5.4 and up)
- Cisco ASA (formerly PIX) (version 7.0 and up)

6.2.6 Other popular applications

One very popular Voice over IP (VoIP) application is Skype. Skype had in December 2010 around 663 million registered users of which an average of 145 million users connected every month the three months prior[103]. However, Skype does not currently have any inherent support for IPv6[104].

7. Background

In this chapter we examine how many autonomous systems that announces one or more IPv6 prefixes, compare IPv4 and IPv6 performance measurements done by others, and explain what you can do yourself at home to compare the performance of the two protocols.

7.1 Autonomous Systems announcing IPv6 prefixes

Réseaux IP Européens Network Coordination Center (RIPE NCC) has statistics on how many Autonomous Systems (ASs) announce IPv6 prefixes[105]. In the Asia-Pacific Network Information Centre (APNIC) region 17% of all ASs announce one or several IPv6 prefixes. The corresponding figures for the Latin America and Caribbean Network Information Centre (LACNIC) region is 15%, the RIPE NCC region: 15%, the African Network Information Centre (AfriNIC) region: 12%, and in the American Registry for Internet Numbers (ARIN) region 10% of the ASs announce one or several IPv6 prefixes. In Norway 48.9% of the ASs announce IPv6 prefixes. In Sweden 31.3% of all ASs announces a IPv6 prefix.

However, this information does not give any insight into how far the ASs have come in their deployment of IPv6, nor the size of the their network[106]. They could be announcing the prefix just for a test lab or for a full production network. My[1] ISP, for example, announces a IPv6 prefix[107], but they do not offer IPv6 to their customers[108]. Although some ISPs provides transition solutions, such as 6to4 and 6rd, pending their deployment of native IPv6.

7.2 What have others already done

In this section we look at some work done by others regarding IPv6 performance compared to IPv4 performance, in networks and in operating systems.

7.2.1 IPv4 and IPv6 performance differences

T. Bilski writes in [109] that router tests carried out by Cisco in 2007 came to the conclusion that IPv6 does not perform better than IPv4 in a dual-stack environment. With smaller datagram sizes (256 bytes and less) smaller software based routers showed a lower throughput with IPv6 than with IPv4. The highest rate without packets being dropped was for IPv4 75% of the link bandwidth and 50% for IPv6. With larger packet sizes the throughput was about the same. Larger hardware based routers showed no variance in throughput even with smaller packet sizes.

He also writes that similar tests by Spirent Communication in 2006 concludes that the increased header size together with IPsec in IPv6 uses up more bandwidth and that the packet sizes are linked to the relative amount of overload. In one of the tests to transport a 100 byte file over HTTP, IPv6 needed approximately 8% more bandwidth. With a file of 1 MB the excess bandwidth needed was below 1%.

He also writes that other studies of TCP connections between China Education and Research Network (CERNET) and approximately 1000 dual-stack web servers in 44 different countries confirms that performance is not improved with IPv6. They also noticed that connections with IPv6 had a packet loss rate of above 3% while the packet loss rate for IPv4 was less than 1%.

7.2.2 TCP/UDP performance in different operating systems with IPv4 and IPv6

Shaneel Narayan has done several performance analysis of TCP and UDP traffic over IPv4 and IPv6 with several different operating systems. In one of his papers [110] he compares several Windows

[1] Gilbert Lidholm

operating systems (Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008 and Windows 7). He concludes that the operating systems performance differ from 2 to 5% in throughput between the two protocols for TCP connections with IPv4 providing greater throughput. For UDP traffic the difference was almost 3% to the advantage of IPv4. Windows Server 2008 and Windows 7 throughput are compatible with Microsoft's earlier operating systems. The latency with TCP traffic varies between the two Internet protocols and operating system. However, Windows 7 on IPv6 did stand out with substantially higher latency values.

7.2.3 Performing measurements during World IPv6 Day

During World IPv6 Day on June 8, 2011, the RIPE NCC performed measurements of the performance of IPv4 and IPv6 [111]. Their measures were done between 48 participants, together with other dual-stack sites, and 40 vantage points. They listed a number of factors that can affect the performance between IPv4 and IPv6. These factors are:

Forwarding:

- If routing is done by hardware for one protocol and by software for the other. If routing is performed by software for IPv6 and by hardware for IPv4, then the routing for IPv6 is going to be a lot slower.
- More parts of the IPv4 header needs to be changed at each hop than in the IPv6 header, which should give IPv6 a relative performance advantage.
- The IPv6 header is larger than the IPv4 header, if no options are used, that could make IPv6 relatively slower than IPv4.

End nodes:

- The IPv4 and IPv6 addresses returned by DNS for a given hostname, could be in completely different networks, possibly leading to differences in performance.

Network:

- IPv6 packets may be forced to take another AS path in order to circumvent ASs without IPv6 compatibility.
- IPv4 and IPv6 packets may be treated differently inside an AS.

The relative performance between IPv4 and IPv6 was measured from one vantage point to one destination throughout a interval of ten minutes[111]. The data showed that IPv4 more often outperforms IPv6 than vice versa. However, there was still a significant chance that IPv6 could outperform IPv4.

This information could benefit applications that want real-time traffic, such as voice over IP (VoIP), games, and video. These applications could, for dual-stacked hosts, choose between IPv4 and IPv6 for the best performance to a dual-stack end host.

7.3 Test you can do from home

The site ipv6-test.com offers a free service where you can test you IPv6 and IPv4 connectivity, speed tests to servers around the globe, as well as latency tests[112]. The tests are done for both protocols at the same time so that you can compare the results between IPv4 and IPv6.

8. Hardware and topology

We used a computer with a server operating system to act as our gateway from our local network (shown in Figure 8.1) to our ISP. The gateway is directly connected to the IPv4 internet and connected to the IPv6 internet via a tunnel (see Figure 8.2). The gateway is connected to the LAN via a 10/100Mbps Ethernet switch. The clients are connected to the LAN via the same Ethernet switch. The full details of all of the hardware that was used for our testing are given in Appendix B.

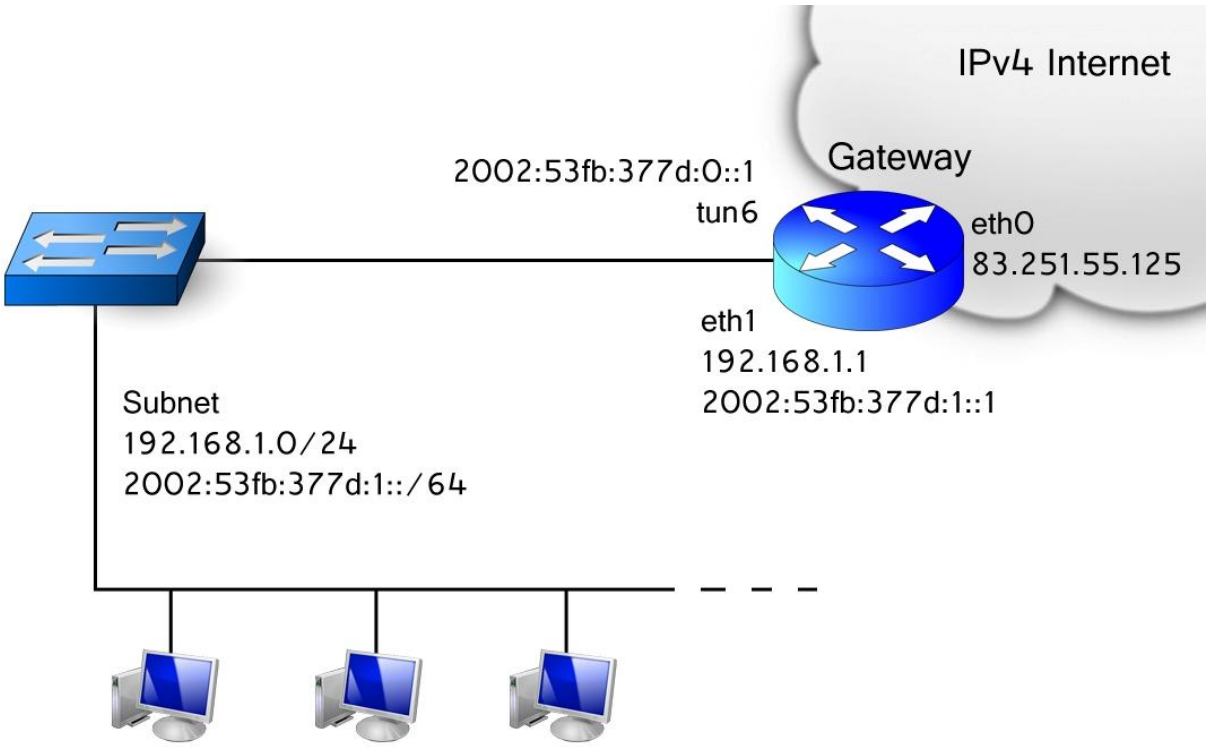


Figure 8.1 Topology of the LAN

Figure 8.2 illustrates how we, and other IPv6 “islands”, will be connected to the Internet. We will not initially connect to another IPv6 island, but will connect to IPv6 hosts connected to the IPv6 Internet.

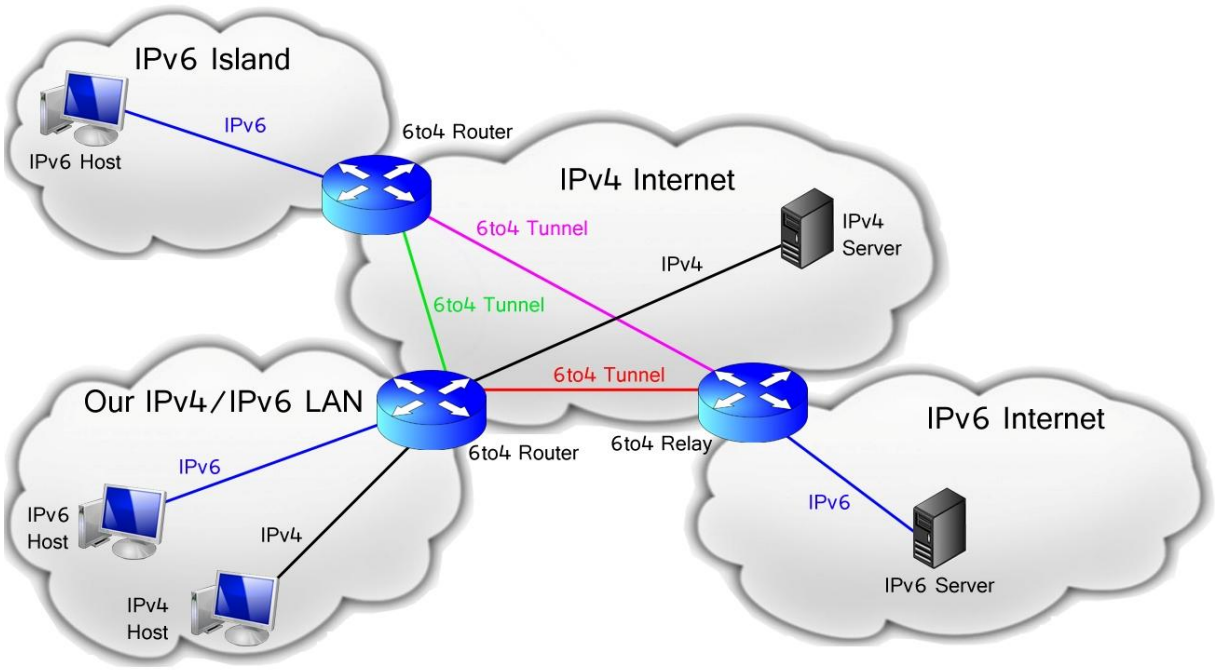


Figure 8.2 Connection to the Internet

9. Ubuntu Server

Ubuntu server is a Linux based open source free server operating system without any pre-installed GUI. We installed Ubuntu server 12.04 64-bit on the computer that we used as a gateway and will run all of the services on this machine.

9.1 Setting up the network

In this section we will explain the configuration of the server's IPv4 and IPv6 internet connection. In order to enable our clients to connect to the IPv4 internet we configured the server as a Network Address Translation (NAT). To enable our clients reach the IPv6 internet, we setup a 6to4 tunnel from this server to a 6to4 relay.

9.1.1 Enable IPv4 and IPv6 Routing

The process begins by modifying the file `/etc/sysctl.conf` to enable routing (as it is disabled by default). The required configuration lines are in the file; we simply needed uncomment one line for each protocol to enable IPv4 and IPv6 routing. The lines should look like this after the modification:

```
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.conf.default.forwarding=1

# Uncomment the next line to enable packet forwarding for IPv6
net.ipv6.conf.default.forwarding=1
```

If you want to immediately (and only temporarily) enable forwarding, you can simply type:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward

# echo 1 > /proc/sys/net/ipv6/conf/all/forward
```

which will replace the content of `ip_forward` and `forward` (which is initially 0) with a 1. This will only last until the next reboot (hence this change is only "temporary").

9.1.2 Set up a 6to4 Tunnel

A global IPv4 address is required to set up the tunnel. This IPv4 address should have been assigned to your interface dynamically based upon the DHCP response sent by your ISP. You can see this IPv4 address by using the `ifconfig` command (we assume that the external interface is eth0):

```
# ifconfig -a

eth0    Link encap:Ethernet  HWaddr 00:24:54:09:46:59
        inet addr:83.251.55.125  Bcast:83.251.63.255  Mask:255.255.240.0
```

A new network interface is created with a tunnel type called Simple Internet Transition (SIT) that utilizes IP in IP (IPIP) to interconnect isolated IPv6 networks[113]. IPIP is a simple type of tunnel with low overhead, but it has some limitations - such as not being able to support multicast-based protocols.

The interface can be named with a name of your own choice (in this case we call it tun6):

```
# ip tunnel add tun6 mode sit remote any local 83.251.55.125
```

Bring up the interface to activate it:

```
# ip link set dev tun6 up
```

The output from ifconfig for tun6 will look something like this:

```
tun6      Link encap:IPv6-in-IPv4
          inet6 addr: ::83.251.55.125/128 Scope:Compat
          UP RUNNING NOARP  MTU:1480  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

9.1.3 Addressing

Next we need to assign global IPv6 addresses to the interfaces. As mentioned in the *6to4* section, the IPv6 address is derived from the 6to4 prefix (2002::/16) and the 32 bit hexadecimal representation of the global IPv4 address (83.251.55.125 → 53fb:377d). Following this is the site-level aggregator (0 for the WAN interface and 1 for the LAN interface), yielding a /64 prefix, and lastly the host's interface specific address. The following lines configure the tunnel's IPv6 address and the gateway's LAN (eth1) address.

```
# ip -6 addr add dev tun6 2002:53fb:377d:0::1/64
```

```
# ip -6 addr add dev eth1 2002:53fb:377d:1::1/64
```

It is not necessary to use a global IPv6 address on the LAN interface since the link-local addresses (which are generated by the autoconfiguration process) are used when forwarding packets. However, it simplifies administration to be able to use the configured address. However, in order to make it work without the global address, you have to add a default route for the incoming packets:

```
# sudo ip -6 route add 2002:53fb:377d:1::/64 dev eth1
```

Otherwise the incoming packets will be dropped since they are destined for a network that is not present in the routing table.

9.1.4 IPv6 Routing

The default gateway must be set to a 6to4 relay in order to route the traffic to the IPv6 Internet. You can set the address of a specific relay if you know its IPv4 address. But a simple way is to send packets to the anycast address of 6to4 relays (192.88.99.1):

```
# ip -6 route add ::/0 via ::192.88.99.1 dev tun6 metric 1
```

`::/0` matches all IPv6 addresses and `::192.88.99.1` is a IPv4-compatible IPv6 address written in mixed notation.

9.1.5 NAT

We are using the software *iptables* to provide NAT-functionalities which was described in the section 2.14. The required commands to apply masquerading and forwarding are:

```
# iptables --table nat --append POSTROUTING --out-interface eth0 -j
MASQUERADE
```

```
# iptables --append FORWARD --in-interface eth1 -j ACCEPT
```

Explanations of the above commands are:

`--table`: specify the table to use

`nat`: alter IP address

`--append`: specify the chain which to append the rule(s) to:

`POSTROUTING`: apply after routing, just before sending out

`FORWARD`: filter packets that are to be forwarded

`--out-interface`: specifies the interface that the packet is going to be sent to

`--in-interface`: specifies the interface that the packet was received on

`-j`: if the packet matches the rule, then jump to specified target

`MASQUERADE`: Mapping to the specified IP address of the outgoing interface. The source address is modified (source NAT)

`ACCEPT`: packet is let through

9.1.6 Configure interface with static IPv4 address

Next we configure our eth1 (LAN) interface with a static IPv4 address. This is done with the following command:

```
# ifconfig eth1 192.168.1.1 netmask 255.255.255.0
```

9.2 Setting up services

We implemented and tested several services on the Ubuntu server. These include DHCP, DNS, DNSSEC, and FTP among others. The configurations for these services are described in the following subsections.

9.2.1 Implementing a DHCP server

To automatically assign IPv4 and IPv6 addresses, we implemented a DHCP server. We used the Internet System Consortium (ISC) DHCP server version 4.1.1-P1. To enable the ISC DHCP server to hand out IPv6 addresses a configuring file specific for IPv6 is needed. Unfortunately, ISC DHCP server cannot hand out both IPv6 addresses as well as IPv4 addresses at the same time. To hand out addresses for the two protocols, two instances of the ISC DHCP server need to be run.

To install the ISC DHCP server on a Linux machine using the Advanced Packaging Tool (APT) you simply type:

```
# apt-get install isc-dhcp-server
```

To manually start the ISC DHCP server from the command line, you use the `dhcpd` command. To specify that DHCPv6 should be used you add the `-6` option. With the `-cf` option you can specify which configuration file should be used (see appendix A.2 for the DHCPv4 configuration file and A.3 for the DHCPv6 configuration file). You need to specify on what interface the server should listen, if you do not it will attempt to listen on all interfaces. Note that the interface(s) specified must have an IP address belonging to a subnet declared in the configuration file, otherwise the DHCP server will not

listen on that interface. When we start the instance for handing out IPv4 addresses we only need to specify the configuring file and the interface to listen on. Starting the two servers from the command line is shown below.

For IPv6:

```
# dhcpd -6 -cf /etc/dhcp/dhcpd6.conf eth1
```

For IPv4:

```
# dhcpd -cf /etc/dhcp/dhcpd.conf eth1
```

Upon starting dhcpd6, the following message might appear:

```
Can't open lease database /var/lib/dhcp/dhcpd6.leases: no such file
or directory --
```

This can be solved by creating that file in `/var/lib/dhcp/` and changing its owner to dhcpd:

```
# touch dhcpd6.leases
```

```
# chown dhcpd:dhcpd dhcpd6.leases
```

The file `dhcpd6.leases` contains the database for leases that the server has assigned. New entries are appended at the end of the file when a new lease is added or when an old one is renewed or released. This means that there will be duplicates of leases after a while. When this occurs, the last entry in the file is the one that is in effect.

In order to find out the cost in time for `dhcpd` to search through the lease file, we compared the time it took for a client to obtain a lease from the server when the file was empty and when it contained a lot of leases (462 entries). The testing showed that the size of the lease file might have a slight effect on the time (in this case it was under a millisecond greater). However, the difference that we saw was less than the standard deviation so we cannot draw that conclusion (for all the data and calculations see appendices K and L respectively). We have been unable to find out the behavior of the search algorithm so we do not know if the order of the leases has a theoretical effect on the performance. If the entire lease file is not searched through every time, e.g. it might start from the bottom and stop as soon as it finds a valid lease, it will not matter if there are many leases in the rest of the file. But even with and without a valid lease in the file, the time did not change noticeable. A new lease file is created periodically by `dhcpd` so that the old one does not get too large[114]. The potentially increased time will probably not be an issue because of this. But it is not stated in the manual how large the file can get before it is replaced.

Unlike DHCPv4, DHCPv6 cannot assign the gateway a host should use[115]. Other than this, a DHCPv6 server and a DHCPv4 server do not differ much in what they do. In order for an IPv6 client to get a default gateway, router advertisements have to be sent by a router. We installed a router advertisement daemon (radvd) on our Ubuntu Linux gateway. This program is an open source program that sends out router advertisements. A client can obtain an IPv6 address by autoconfigurating an interface with the help of these router advertisements. This is an alternative to having a DHCP server assigning the client a gateway and address. You can specify with DHCP from what range the addresses should be allocated from and reserve addresses for specific hosts based upon the interface's MAC address.

9.2.2 Install and configure radvd

The stateless autoconfiguration process requires information that Router Advertisements (RA) can provide. In order to periodically send out these RAs, we use thw *radvd* (RA deamon), which is an open-source program. In order to install this software you simply give the following command:

```
# apt-get install radvd
```

The radvd configuration file (`/etc/radvd.conf`) is empty so it needs to be edited after the installation. In this case we have placed the following into the file:

```
interface eth1 {
        AdvSendAdvert on;
        MinRtrAdvInterval 3;
        MaxRtrAdvInterval 10;
     AdvManagedFlag on;
     AdvDefaultLifetime 600;
        prefix 2002:53fb:377d:1::/64 {
                AdvOnLink on;
                AdvAutonomous on;
                AdvRouterAddr on;
        };
};
```

The daemon will not start after the installation (since the configuration file is empty), therefore you have to manually start the daemon with the following command:

```
# /etc/initd.d/radvd start
```

If any modifications have to be made in the file, the daemon has to be restarted before the changes takes effect. This can be done with the following command:

```
# /etc/initd.d/radvd restart
```

DHCPv6, as well as DHCPv4, can tell the client what DNS servers to use. This can, for IPv6, also be accomplished with router advertisements. In the radvd configuring file, located in `/etc/radvd.conf`, a list of DNS servers can be specified to be advertised. However, the client needs a program that listens to this information and updates it own list of name servers to use.

In the configuration file for radvd we set the M flag in the router advertisement to indicate that IPv6 addresses are available via DHCPv6[22]. There is also an O flag that indicates that additional configuration information is available via DHCPv6 such as DNS information. However, with the M flag set, DHCPv6 will return DNS information as well. With the variable AdvAutonomous set to “off” in the radvd configuration file, hosts are not allowed to autoconfigure their own addresses with the announced prefix, but rather they must instead rely on DHCPv6 or manually configured addresses. The default value for this option is “on”.

One problem that we encountered was that the IPv6 default gateway kept disappearing in Windows 7. The gateway disappeared if an IPv6 connection was idle for a while. However, in Ubuntu 11.10 this problem did not appear. To counteract this we made the default router advertised by radvd valid for 10 minutes instead of 3 times the maximum time between advertisements (30 seconds in our case) as our default value. However, this is not a general fix to the problem, but rather provided an ease of the symptoms exhibited by a Windows 7 host.

9.2.3 BIND9

The Domain Name System (DNS) is used to map names to IP addresses. As a DNS server we installed BIND9 version 9.8.1-P1. To install Bind9 on a Linux machine using APT you simply type:

```
# apt-get install bind9
```

BIND9 is configured as a caching server by default (for complete configuration see appendix A.5). All you need to do to use it as a caching server is to add forwarders to forward the queries to. You do this by modifying the file `/etc/bind/named.conf.options`. Simply uncomment and edit the following lines:

```
forwarders {
        <IPAddressToDNS1>
        <IPAddressToDNS2>
        ...
};
```

Because we do not have a domain name at the moment we simply added ComHem's DNS servers IPv4 address and Google's IPv6 address for their DNS server (2001:4860:4860::8888 and 2001:4860:4860::8844) to forward the queries to. Nothing needed to be altered in the default configuration to enable IPv6.

Later we added a fictitious domain in `/etc/bind/named.conf.local` called ex.jobb. To add a domain or zone you add:

```
zone "ex.jobb" {
      type master;
      notify no;
      file "/etc/bind/ex.jobb.db";
};
```

`type master`: Defines that the DNS server is the primary server for the zone.

`notify no`: NOTIFY should not be sent to anyone upon alteration.

`file`: Specifies where the zone file (see below) is located.

ex.jobb.db

`/etc/bind/ex.jobb.db` defines the names belonging to specific addresses.

```
$TTL 86400
@    IN   SOA  ns1.ex.jobb. daboss.ex.jobb. (
               20120531  ; Serial
                604800       ; Refresh
                 86400       ; Retry
               2419200       ; Expire
                 86400 ) ; Negative Cache TTL
;
ex.jobb.  IN   NS   ns1.ex.jobb.
gateway   IN   A    192.168.1.1
gateway   IN   AAAA 2002:53fb:377d:1::1
```

```
ns1  IN   A    192.168.1.1
www  IN   CNAME     gateway
```

`NS`: Name server record. The IP address or CNAME of the name server of the domain.

`A`: IPv4 Address record. Maps the name to an IPv4 address.

`AAAA`: IPv6 address record. Maps the name to an IPv6 address.

`CNAME`: Canonical name record. An alias' canonical name.

Reverse lookups

Reverse lookups are used when you want to know the name associated with a certain address. Add the following to `/etc/bind/named.conf.local` to make reverse lookups possible for the subnet 192.168.1.0/24:

```
zone "1.168.192.in-addr.arpa" {
          type master;
          notify no;
          file "/etc/bind/1.168.192.in-addr.arpa";
};
```

The zone has to be the subnet typed backwards.

1.168.192.in-addr-arpa

Add the following to `/etc/bind/1.168.192.in-addr.arpa` to define what address belongs to a specific name:

```
$TTL      86400
@         IN   SOA  ns1.ex.jobb. daboss.ex.jobb. (
                    20120531  ; Serial
                     604800       ; Refresh
                      86400       ; Retry
                    2419200       ; Expire
                      86400 ) ; Negative Cache TTL
;
          IN   NS   ns1.ex.jobb.
1         IN   PTR  gateway.ex.jobb.
```

`PTR`: A domain name pointer. Defines the (last octet of the) address on the subnet for which a certain name belongs to.

Equivalent for IPv6

Add the following to `/etc/bind/named.conf.local` in order to make reverse lookups for IPv6 possible for the subnet 2002:53fb:3271:1::/64:

```
zone "1.0.0.0.1.7.2.3.b.f.3.5.2.0.0.2.ip6.arpa" {
          type master;
          notify no;
          file "/etc/bind/ipv6.ip6.arpa";
};
```

Add the following to `/etc/bind/ipv6.ip6.arpa` to define the address on the subnet for which a certain name belongs to:

```
$TTL       86400
@          IN   SOA  ns1.ex.jobb. daboss.ex.jobb. (
                           1         ; Serial
                      604800         ; Refresh
                       86400         ; Retry
                     2419200         ; Expire
                       86400 )  ; Negative Cache TTL
;
           IN   NS   ns1.ex.jobb.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0 IN   PTR  gateway.ex.jobb.
```

9.2.4 DNSSEC

Domain Name System Security Extensions (DNSSEC)[116] adds security to some of the data sent by DNS. DNSSEC utilizes public-key cryptography in order to sign the records used in DNS lookups. Four new record types are introduced: Resource Record Signature (RRSIG), DNS Public Key (DNSKEY), Delegation Signer (DS), and Next Secure (NSEC). Two header bits, Checking Disabled (CD) and Authenticated Data (AD), are also added. An RRSIG record contains a digital signature that can be associated with a DNS resource record (RR) set. The signature is created by using a DNSKEY. The DS record is used in authenticating DNSKEYs. The NSEC record provides security against spoofing. These signatures can be used to prove that certain names do exist in the domain. We implemented DNSSEC using BIND9.

To enable DNSSEC we add the following lines to `/etc/bind/named.conf.options`:

```
dnssec-enable yes;
dnssec-validation yes;
dnssec-lookaside auto;

include "/etc/bind/bind.keys";
```

`dnssec-enable` enables DNSSEC, while `dnssec-validation` enables DNSSEC validation.

Next we need to generate a key using the following command:

```
zonesigner -genkeys -usensec3 -zone ex.jobb ex.jobb.db
```

`-genkeys` means that a new key signing key (KSK), a new active zone signing key (ZSK), and a passive ZSK are to be generated. The default algorithm is RSA SHA1. NSEC3 is used (`-usensec3`) to avoid zone walking. Specify the name of the domain that is to be signed after –zone, and then the

name of the zone file. New files will be created. These include a signed zone file (ex.jobb.db.signed).

Replace "ex.jobb.db" with "ex.jobb.db.signed" in /etc/bind/named.conf.local:

```
zone "ex.jobb" IN {
     type master;
     notify no;
     file "/etc/bind/ex.jobb.db.signed";
};
```

In order to add or modify RRs you alter the original zone file and then sign the file with:

```
zonesigner -zone ex.jobb ex.jobb.db
```

To see if it works you can use the dig command. First you need to copy the keys to a file. You can get the keys by simply typing:

```
# dig @127.0.0.1 ex.jobb –t dnskey
```

We make sure that we query our own DNS server by adding @ and the IP address (in this case the IP address is the loopback address). Then create a file and copy the keys to the file so it looks like this:

```
ex.jobb.       86400     IN   DNSKEY    257 3 8
AwEAAcM2rjIOhWl5bQISXvKKeA4OG4ZUdGQbzuyL4Va9ULWXhitA3c4y
DYeRu1TeDoUnDyyhKohDxrR9+u0CoR0eQYZ4yWdRb9yiPhsu4m42iSL1
1lChSWOr1SKOYqvF7zBFtG6ZovU2Ea7BBh455Hhss45aVuauXXfgZtX6
riFvbOGQUoVEMaNghXbhqx4/2MTju33tmlEu0FwhBei+ieQvXOE9rfbm
PIJGmb74kcstGhuTWlMcb6xAxHIXUONHR4OH/dP1CTBwnWfklEHC/cXH
aE6V20OAKsYwaU+Phzqsp2qDqDamV3aA6vf55xdukPIIiT3tR8Y6LI7F
RNSaXu1ALdc=
ex.jobb.       86400     IN   DNSKEY    256 3 8
AwEAAbYahFxwvcNsdHwrrL45L2YyHYcpM8I1CDiQ+ErhLTPELkcUtz+U
aMTQzAQ/69xMH+UBrzVnn11vYxTYU+/ZHKS05n6iIV/CTKoqhVOp6GgZ
OWfHn13fhB4rXKgV/jDrZ04rnwdJWzZMkdJzd2WRZuMIW6DMXKeDnulM lt8IUu/V
ex.jobb.       86400     IN   DNSKEY    256 3 8
AwEAAcRj8vKnyZy9Sfv5oDIKO3f3Sx+yRZmV7wY1fu6CPQC3ixJTmjFF
rggY6rXKGmwhn0ph6VAvzx40RbmPm/iJ5R9h1sh0fIL+8jCGsPYxbJmg
L2Z5WJSn4NfG8RjQUQpzmjdC0dfSDm+8HwPjFPP0DU8dOj8kb89WxaYX 34HwVZdR
```

Run the following command to verify that it works correctly:

```
# dig @127.0.0.1 +sigchase +trusted-key=./key ex.jobb. any
```

The option +sigchase means that dig should verify the records and try to climb the trust chain until it finds a record signed by a trusted key. The +trusted-key=./key option specifies the key(s) that we trust and that it is in the file "key". In this example ex.jobb is the domain we want to fetch records from and the any option means that we want to fetch all types of records. The only record we got is the SOA record for ex.jobb. To verify that record, dig will fetch the RRSIG record belonging to the SOA record which is a digital signature for the SOA record[117]. The signature was made by our Zone Signing Key (ZSK) so dig fetches the DNSKEY set of resource records. Given the response, dig verifies the ZSK by fetching the ZSKs RRSIG record. The RRSIG for the ZSK is

signed by itself or one of the other keys in the DNSKEY set. Normally `dig` then fetches a DS record for the domain from the parent zone that includes a digest of the Key Signing Key (KSK), signed by the parent zone's key. However, there is no job domain that `dig` can fetch a DS record from as our domain is phony. Now `dig` has enough material to start verifying the records. As the trusted keys are the DNSKEY set that `dig` fetched from the server, the validation will ultimately be successful.

9.2.5 Web server

As a web server we installed Apache version 2.2.20. To install it with APT you type:

```
# apt-get install apache2
```

In the `/etc/apache2/ports.conf` you can specify on what address(es) and on what port(s) the server should listen. You can specify both IPv4 and IPv6 addresses. IPv6 addresses are specified in brackets []. If only an IPv4 address is specified, then the instance of the apache server will only listen on this specific IPv4 address. In the default configuration only port 80 is specified which means that it listens on all of this host's addresses, i.e., both IPv4 and IPv6 for all interfaces.

9.2.6 Network File System

The Network File System (NFS) is a protocol that allows a user to access a storage device over the network similar to the way they would access a local device. To add a device or directory to be remotely accessed in Linux you edit the `/etc/exports` file of the NFS server. To make a local directory remotely accessible you add the line:

```
/my/directory <IPAddress> (options)
```

Both IPv4 and IPv6 addresses work in a standard configuration. To mount a remote directory on a Unix type operating system you type:

```
# mount <IP address>:/path/to/directory/ /path/to/local/directory/
```

To mount a device over IPv6 you add brackets around the address.

To unmount a device you type:

```
# umount /path/to/local/directory
```

However, when we tried to unmount a device that was mounted over IPv6 it returned an error saying that the mount was not found in `/proc/mounts` even though it was there. We never encountered this problem when mounting over IPv4, but it occurred every time we mounted the device over IPv6. This is a known bug, see [118]. For some reason this problem stopped occurring by itself without any action on our part.

9.2.7 File Transport Protocol

The File Transfer Protocol (FTP) allows users to send and retrieve files over the network. We installed a couple of the commonly used FTP servers to test if they supported IPv6, specifically we installed the Very Secure FTP Daemon (VSFTPD) version 2.3.5, Pure-FTPd, and ProFTPD version 1.3.4a. All of these supported IPv6 in their default configuration.

To access the content shared by the FTP server in Linux you type:

```
# ftp <UserName>@<IPAddressOrServerName>
```

Alternatively to use SSH FTP (SFTP):

```
# sftp <UserName>@<IPAddressOrServerName>
```

To use an IPv6 address you simply put brackets around the address.

We tested the FileZilla FTP client in Windows 7. The FileZilla client supports IPv6 by default, but we needed to be specific about which port to use and what protocol to use (FTP or SFTP).

9.2.8 Streaming

To test streaming we installed vlc-noX, which is a non-graphical version of VideoLAN, on our gateway with Ubuntu Server 12.04. To stream media you need to create a `vlm.config` file. The `vlm.conf` file is a text file with a list of commands that say how to stream, what to stream, and how to access it from the outside (see appendix A.6 for the configuration file that was used for our testing). We chose to stream over HTTP. To launch the server you type:

```
# cvlc -I http --vlm-conf /path/to/vlm.conf
```

`cvlc` stand for Console VLC which is VLC without a graphical user interface. The option `-I http` means that it should use the HTTP interface. This starts a small HTTP server that is used both for streaming and remote control. Further details can be found in [119]. With the `--vlm-conf` option you can specify a specific vlm configuration file.

To access the stream via VLC running on a client computer you press on the Media tab and choose Open Network Stream or you can simply press Ctrl + N. In the window that pops up you type `http://<ServerIP>:<PortNumber>/<`*`PathSpecifiedInConfigurationFile`*`>`. To access the stream over IPv6 you simply put brackets around the IPv6 address.

If you want to access the stream using the Microsoft Windows Media Player you right click in the upper part of the window and select File and choose Open URL or press Ctrl + U. In the window that pops up type `http://<ServerIP>:<PortNumber>/<PathSpecifiedInConfigurationFile>`. To access the stream over IPv6 you put brackets around the IPv6 address just as in VLC.

9.2.9 SSH

Secure Shell (SSH) allows for secure communication between computers over an insecure network. Our test setup consisted of three machines: an Ubuntu machine that acted as the server and two clients, running Windows and Ubuntu. We used SSH to remotely executing commands on the server from the clients. A connection is established via either IPv4 or IPv6 depending on which type of address you specify in the command.

On the Ubuntu client you simply enter the command:

```
# ssh <UserNameOnServer>@<IPAddress>
```

It will prompt you to enter the password for the user and then the connection is established.

As a Windows client we used *putty* and specified the IP address in the *Host Name (or IP address)* field and port number (22) in the *Port* field. Putty will prompt you to enter the username and then the password. A secure connection is established upon receiving the correct credentials.

For our testing we monitored the traffic sent between the client and the server to see that IPv4 or IPv6 addresses were being used as you expect. Note that it is not possible to see inside the encrypted

payloads, unless you have the server's private key. If you do, then it is possible to tell Wireshark to use this key, thus enabling you to see all of the actual communications. See for example [120].

9.2.10 VoIP

As stated earlier in the report, Skype has no support for IPv6. It works well on IPv4, but if only an IPv6 address is present on the computer, then you are unable to connect.

Mumble is an open source VoIP program mainly focused on the gaming community. It has IPv6 support by default.

A server can be set up that clients can connect to. We are using an Ubuntu machine that is running the Mumble server. The server is installed with the command:

```
# apt-get install mumble-server
```

Next add a folder is created to store the user database:

```
# mkdir /etc/murmur
```

Next we change to this directory and add a password for the super user to the database with the following commands:

```
# cd /etc/murmur
```

```
# murmurd -supw <password>
```

To install the mumble client you type:

```
# apt-get install munmble
```

Now you can start the client. Go to Server and press Connect or press Ctrl + O. At the server you add the name for the "connection", IP address, port number, and user name. To connect to an IPv6 server you simply type the IPv6 address without brackets. Select the connection you just created and click `connect` to connect to the server.

Mumble was able to use both IPv4 and IPv6, as was confirmed by monitoring the traffic and checking which version of addresses were being used.

9.2.11 Subversion

Apache Subversion (SVN) is a popular open source program that allow version controlled storage and retrieval of files. Little concern was given to security in SVN. Here we merely test for IPv6 compatibility. Passwords are sent in plaintext unless you use encryption.

Subversion is installed by typing:

```
# apt-get install subversion
```

Create a directory for the files that you want to keep under version control:

```
# mkdir /home/userName/svn/projectName
```

Create a group with the name *subversion*:

```
# sudo groupadd subversion
```

Add your user and the Apache user (in this case www-data) to the subversion group:

```
# sudo usermod –a –G subversion www-data

# sudo usermod –a –G subversion userName
```

Change ownership of the file and set the permissions by typing:

```
# sudo chown –R www-data:subversion projectName

# sudo chmod –R g+rws projectName
```

Create a repository:

```
# sudo svnadmin create /home/userName/svn/projectName
```

In order to get access to these files via HTTP, `/etc/apache2/mods-available/dav_svn.conf` needs to be edited to contain the following lines:

```
<Location /svn/projectName>
     DAV svn
     SVNPath /home/userName/svn/projectName
     AuthType Basic
     AuthName "myproject subversion repository"
     AuthUserFile /etc/subversion/passwd
     <LimitExcept GET PROPFIND OPTIONS REPORT>
        Require valid-user
     </LimitExcept>
  </Location>
```

Restart the web server in order for the changes to take effect:

```
# sudo /etc/initd.d/apache2 restart
```

Users should be added with passwords to provide authentication. When a new copy of SVN is installed, the `/etc/subversion/passwd` file does not exist. Use the `-c` switch when adding the **first** user in order to create it:

```
# sudo htpasswd –c /etc/subversion/passwd userName
```

Remove the `-c` switch when adding additional users.

9.2.12 Mail

We will be using Postfix which is an open source mail transfer agent (MTA). A MX (Mail exchange record) record has to be added in the zone file (our: `/etc/bind/ex.jobb.db`) in order for this to work. The MX maps a domain name to MTAs. An example of such a record is shown in DNS server entry below:

```
$TTL 86400
@    IN   SOA  ns1.ex.jobb. daboss.ex.jobb. (
               20120533  ; Serial
                604800        ; Refresh
                 86400        ; Retry
               2419200        ; Expire
                 86400 ) ; Negative Cache TTL
;
     IN   NS   ns1.ex.jobb.
gateway   IN   A    192.168.1.1
gateway   IN   AAAA 2002:53fb:377d:1::1
ns1  IN   A    192.168.1.1
mail IN   A    192.168.1.1
mail IN   AAAA 2002:53fb:377d:1::1
www  IN   CNAME     gateway
ex.jobb.  IN   MX   10   mail.ex.jobb.
```

Proceed with the installation of Postfix:

```
# sudo apt-get install postfix
```

During the installation we chose “Internet Site” as the general configuration, followed by the domain name “ex.jobb”.

The mailx package contains the `mail` command that will be used later on:

```
# sudo apt-get install mailutils
```

During the installation we chose “no” when asked if we should force synchronous updates on the mail queue.

Test setup

To perform our testing we added an initial user that can be used to receive the mails sent:

```
# sudo useradd –m –s /bin/bash testuser
```

```
# sudo passwd testuser
```

This will prompt you to enter a password for `testuser`.

Make sure that postfix is running, by testing to see if it is working correctly:

```
# telnet localhost 25
```

Type the following SMTP commands in the terminal (each line is followed by enter) to create the mail and end with “.”:

```
ehlo localhost
mail from: root@localhost
rcpt to: testuser@localhost
data
Subject: A test mail

A test mail
```

```
.
quit
```

Change to your created user (`testuser`) and check if the mail has arrived:

```
# su – testuser

# mail
```

You should be able to see the new mail and can choose to read it by entering the number of the mail message.

Adding own domain

Your domain has to be added to `mydestinations` (this will overwrite the previous content):

```
# sudo postconf –e "mydestination = mail.ex.jobb, localhost.ex.jobb, localhost, ex.jobb"
```

Adding local network

Adding the LAN to Postfix allows remote clients connecting from these addresses to have greater privileges than clients connecting from other addresses [121]. This addition is done for several LANs with a command of the form:

```
# sudo postconf –e "mynetworks = 127.0.0.0/8, 192.168.1.0/24, [::1]/128,[2002:53fb:377d:1::]/64"
```

Receive on all interfaces:

```
# sudo postconf –e "inet_interfaces = all"
```

The "`inet_protocols = all`" is set by default, this enables the use of both IPv4 and IPv6.

Restart Postfix to apply the changes:

```
# sudo /etc/init.d/postfix restart
```

It should now be possible to send mail from one computer to another computer in the LAN using both IPv4 and IPv6. We confirmed this by sending mail from a computer using the following lines:

```
netcat mail.ex.jobb 25
ehlo ex.jobb
mail from: aUser@ex.jobb
rcpt to: testuser@ex.jobb
data
Subject: A mail from me to you

I like carrots!
.
quit
```

Check the mail of `testuser`:

```
#su – testuser

#mail
```

10. Windows Server 2008 R2

Windows Server 2008 R2 is a Microsoft 64-bit server operating system. We installed Windows Server 2008 R2 Enterprise with service pack 1 on a computer as our gateway to the network and to run all the test services on.

10.1 Setting up the network

As on the Ubuntu server we setup a NAT so that the clients connected to the internal LAN could have IPv4 internet connectivity and setup a tunnel to provide IPv6 internet connectivity.

10.1.2 NAT

We use the "Add Roles" wizard to add the role "Routing and remote access service" (includes subcategories: "Routing" and "Remote access service"). No configuration options were entered during the initial setup with the wizard.

In the Server Manager, go to "Roles" and select "Network Policy and Access Services". Right click on "Routing and Remote Access" and click on "Configure and Enable Routing and Remote Access". This will bring up the "Routing and Remote Access Server Setup" wizard. After the welcome screen, select "NAT" and click "Next". Mark the interface towards the Internet and then select "Finish".

10.1.3 Enable IPv6

In the Server Manager, go to "Roles" and select "Network Policy and Access Services". Right click on "Routing and Remote Access" and click on "Properties". Check "IPv6 routing LAN only".

Add an IPv6 address on the LAN interface.

10.1.4 Tunnel

By default, several versions of Windows (including Server 2008 R2), automatically add a 6to4 tunnel. But for some reason ours disappeared, so we added our own. A lot of the configuration options are similar to that in Ubuntu. Refer to that set up section for further explanations.

In PowerShell (using our current public IPv4 address 83.251.48.135 as our example), type:

```
# netsh

# interface ipv6

# add v6v4tunnel tun6 83.251.48.135 192.88.99.1
```

to set up a 6to4 tunnel with the name "tun6", with the entry point 83.251.48.135 and exit point 192.88.99.1. Add an IPv6 address:

```
# add address tun6 2002:53fb:3087:0::1
```

Add a default route:

```
# add route ::/0 tun6 ::192.88.99.1 metric=1
```

Advertise the default gateway on the LAN interface in order for clients to be able to automatically configure it:

```
# set interface INT_NR adv=enabled managed=enabled other=disabled

# set interface INT_NR advertisedefaultroute=enabled
```

In order for you to be able to ping the Windows Server 2008 R2 over IPv4 and IPv6, its firewall must accept inbound ICMPv4 and ICMPv6 packets. Details for this are given in many places, such as [122].

10.2 Setting up services

On Windows server 2008 R2 we setup and tested a web server and a DHCP server.

10.2.1 Internet Information Server (IIS)

In the Windows service manager add the role “Webserver (IIS)” and click next a few times until finished. It works without needing further configuration.

10.2.2 DHCP

To be able to automatically assign IP addresses to hosts attached to the LAN we implemented a DHCP server. To setup a DHCP server use the Server Manager and click on add a role. After the Add Roles wizard comes up, check the option “DHCP Server” and press next. First we configured the server for IPv4. Select which interface the server should use to serve clients and press next. Add the parent domain, preferred DNS IP, alternate DNS IP address, and press next. Check "WINS is not required for applications on this network", as we do not have a Windows Internet Name Service (WINS) server or any applications that require WINS, and press next. Press “add” to add a scope, add a name to the scope, specify the range of addresses that should be handed out, the subnet mask, and the default gateway for the clients; then press OK and next.

Next we configured the server for IPv6. We chose to configure the server for IPv6 without using the Add Roles wizard as the wizard did not give any options for specifying the range of addresses to allocate to the clients. For this reasons we simply check "Disable DHCPv6 stateless mode for this server" and press next. Then click “install” to install the server.

To configure the server for IPv6 you need to open the Server Manager, click on “roles”, click on “DHCP Server”, click on the computer name, right click on IPv6 and select “New Scope” then press next. Enter the name of the scope and description and press next (the name of the scope was LANET and we left the description field blank). Enter the prefix of the subnet (in this case we entered “2002:53fb:3087:1”) and the preference (we used the default value of 0). The preference tells the client what server to use if there are multiple DHCPv6 servers on the network[123]. The client chooses the server with the highest preference value first, but the client may choose a server with a lower value if it has better advertised parameters. Click next. You can add a range of addresses to *exclude* from being distributed. Click next. Leave the lease time values as they are set by default and press next. Check yes to activate the scope and press finish.

To select which interface the server should serve IPv6 addresses on you open the Server Manager, click on “Roles”, “DHCP Server”, click on the computer name, right click on IPv6 and select Properties. Click on the advanced tab and click on bindings. Select the interface in the list and press OK and OK again.

You optionally distribute a list of DNS servers for the clients to use. We chose not to as the server already distributes a list of DNS servers via DHCPv4. Although the clients cannot communicate with these servers via IPv6, AAAA records can be fetched via IPv4 instead.

There is no option in DHCPv6 to tell the client what default gateway to use. This is instead done automatically based upon the router advertisements. To configure Windows Server to send routing advertisements you simply open a PowerShell and type:

```
PS C:\Users\Administrator> netsh

netsh>interface ipv6

netsh interface ipv6>set interface INT_NR adv=enabled managed=enabled
other=disabled
```

To see the numbers of the interfaces you type:

```
netsh interface ipv6>show interface
```

11. Performance tests

In this chapter we will describe a series of tests that we conducted to see how IPv4 and IPv6 perform, both locally and from locally to remote computers.

11.1 Local test

To compare TCP and UDP performance between IPv4 and IPv6 we started by using a program called *pcattcp*. This is the Windows version of *ttcp*, which is a UNIX benchmarking tool. The command interface to the program and its options are shown below.

```
# pcattcp -t   transmitter
          -r   receiver
          -6   IPv6 (default: IPv4)
          -u   UDP (default: TCP)
```

The setup consisted of two computers running Windows 7 (acting as transmitter and receiver) interconnected by an Ethernet switch (this is the same switch as described in Chapter 8). We divided the test into four groups: IPv4 with TCP, IPv4 with UDP, IPv6 with TCP, and IPv6 with UDP, and ran each test-group 50 times (for a total of 200 tests). We calculated the estimated expected value and the standard deviation for each group. The results are shown in Table 11.1.1 (expressed in KB/sec).

Table 11.1.1 Bandwidth (in KB/sec) results with PCATTCP

	Expected value (mean μ)	Standard deviation (σ)
IPv4 TCP	10333	56
IPv4 UDP	11597	37
IPv6 TCP	9779	744
IPv6 UDP	11400	19

During the test we noticed that TCP for IPv6 was highly unstable in that the throughput jumped between about 10400 KB/sec and roughly 9000 KB/sec between the tests (see appendix C). This caused a huge standard deviation and an unreliable result (see results in table 11.1.1 and calculations in appendix D). Given this behavior we chose not to draw any conclusions other than not to recommend *pcattcp* as a testing tool since its measurements seem to be unreliable, and decided to change to testing with *iperf*, another network testing tool. The command interface to this program and its options are shown below.

```
# iperf -c   client (transmitter)
        -s   server (receiver)
        -V   IPv6 (default: IPv4)
        -u   UDP (default: TCP)
        -i   interval between bandwidth reports (in seconds)
        -t   time to transmit for (in seconds)
```

The test consisted of the same setup, only this time the OS on each of the computers was Ubuntu. Here we ran the test groups for 250 seconds and *iperf* reported back every 5 seconds (yielding 50 reports per group) with the average throughput of the past 5 seconds. The results (expressed in KB/sec) are shown in Table 11.1.2.

Table 11.1.2 Bandwidth (in KB/sec) results with iperf

	Expected value (mean μ)	Standard deviation (σ)
IPv4 TCP	11493	0.5
IPv4 UDP	11679	6.4
IPv6 TCP	11334	0.7
IPv6 UDP	10874	5.1

As we can see in Table 11.1.2, these results here are much more stable (see details of the calculations in appendix F). UDP was not faster than TCP (as might have been expected since UDP is more lightweight than TCP). TCP for IPv4 was ever so slightly faster than TCP for IPv6, with almost an identical standard deviation. UDP for IPv4 was slightly faster than UDP for IPv6, with a very similar standard deviation.

During the test the speed of the interface was negotiated to 100Mbit full duplex mode and the NICs was configured to do checksums. The Ethernet frames were 1514 byte with TCP and 1512 byte with UDP. However, we noticed that the packets got fragmented during the tests for UDP over IPv6 so the frame size alternated between 1510 and 92 bytes. This might affect the speed. We did not succeed in making *iperf* work without fragmenting the packets. The computer acting as server was Client 2 and the computer sending was Client 3 (see appendix B for the hardware specifications). The UDP datagram size was set to 1470 byte for both IPv4 and IPv6. The TCP Window size was set to 85.3 KB for both IPv4 and IPv6. Timestamps were used for TCP with both IPv4 and IPv6.

We believe that the difference is marginal and the fluctuation so similar between the protocols, that we cannot draw any conclusions as to whether IPv4 or IPv6 would be faster or better, performance wise, than the other. The two protocols seem to be on a par with each other.

We also concluded that the switch was a major bottleneck in this setup by running a few tests without it. The throughput was much higher (see appendix E). As a result in our subsequent testing we focused on the relative difference and stability, rather than the absolute maximum throughput, by keeping the switch.

The theoretical upper bound in our test topology for IPv4 would be[124]:

- 11,767,896 bytes/second for TCP with timestamps
- 11,865,420 bytes/second for TCP without timestamps
- 11,963,589 bytes/second for UDP

The theoretical upper bound in our test topology for IPv6 as we calculated (see appendix M) would be:

- 11,605,356 bytes/second for TCP with timestamps
- 11,702,880 bytes/second for TCP without timestamps
- 11,800,404 bytes/second for UDP

11.2 Ping test

To compare our IPv6 connection to our IPv4 connection we created a script that pinged and traced the route to different dual stacked websites. An instance of the script ran every hour from 11 am to 3 pm during working days. The data showed that IPv6 was almost always some milliseconds slower (from about 1 to about 30 milliseconds), except to www.xbox.com where the latency varied much more (from about 1 to about 95 milliseconds). Only on some rare occasions was IPv6 faster than IPv4. This

performance difference is probably because the IPv6 traffic is tunneled to a 6to4 relay before it is routed to its destination, thus incurring added delay at the gateway and possibly the traffic has to take a suboptimal route. The delay and route to the relay itself was relatively stable. See the results in appendix G.

11.3 Traceroute test

Due to the results (described above) of the ping test (the varying latency) we made a traceroute test. This test was performed using two computers, to collect a traceroute to two different dual-stacked sites at the same time. By doing so we wanted to see whether it was changes in the tunnel or if it was competing traffic outside of the tunnel that caused the latency spikes. If the spikes occur at the same time on both traceroutes, it is more likely that a common denominator is the cause, which could be due to something that is happening in the tunnel. The trace towards the relay was always consistent in terms of the paths chosen, so we assumed that the path to and from the relay does not change during the tests.

We were not able to replicate the latency spikes to a degree that we feel would give a definitive answer, but the data points towards the tunnel **not** being the cause of the difference in the traceroute results. Consider the following excerpt from the data at hop 9 (this is a sample from one of all the traces we made):

```
9  2001:428::205:171:203:158 (2001:428::205:171:203:158)  118.687 ms
215.261 ms  119.297 ms  119.73 ms  120.996 ms
```

Here we see a spike of 100ms in the latency of the trace towards Xbox, but nothing of the sort occurs in the entire trace towards FreeBSD (see appendix J for the whole example). This indicates that the delay is caused somewhere outside of the tunnel on the path from the gateway to the xbox.com server.

After observing the highly fluctuating latency, we wanted to see if there was any pattern to it. We compiled all the data into a graph illustrating the latency towards xbox.com. This data is shown in figure 11.3.1 – where the vertical bars represent one sample of the ten samples collected at every hour, starting at 11:00 am and ending at 3:00 pm.

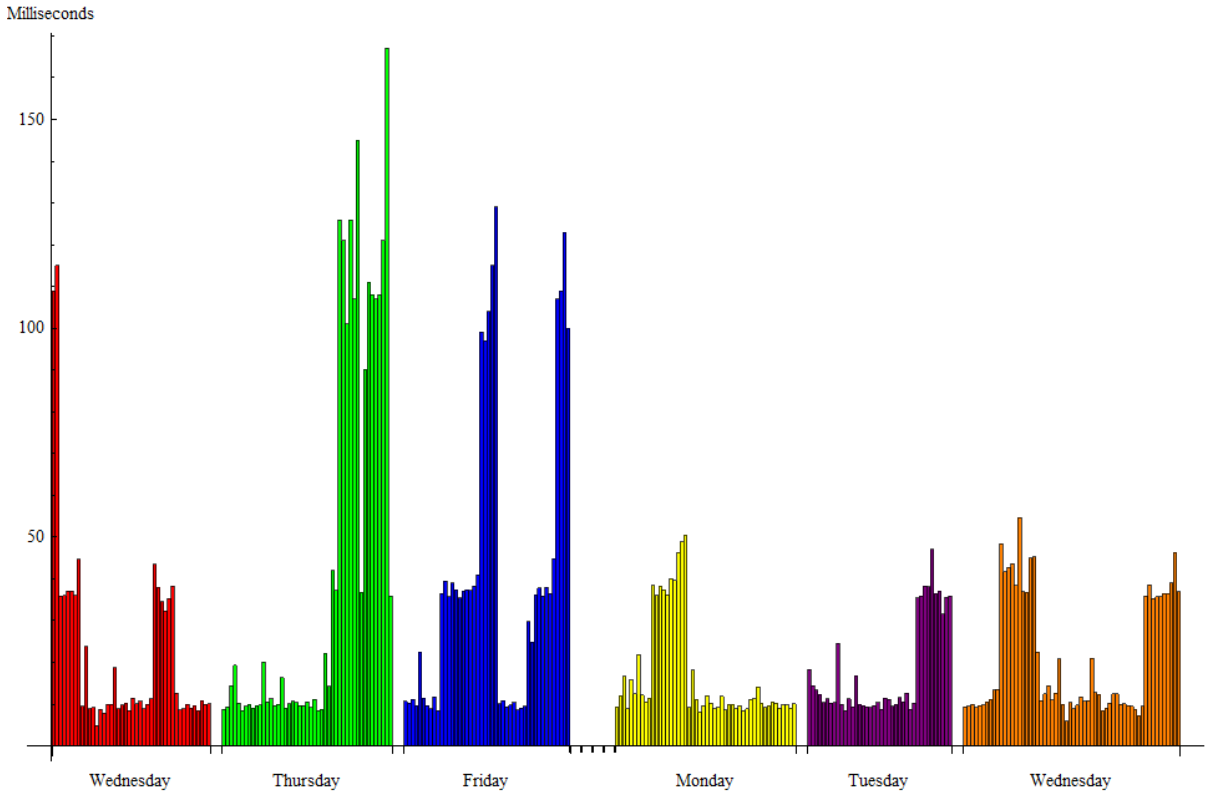


Figure 11.3.1 Latency towards xbox.com

As shown in the graph, there is no definite pattern to the latency. Sometimes it is elevated and spiking, but not consistently during a particular time of the day or part of the day (for example towards later in the afternoon). The high latency is probably caused by heavy load during some random hours of the day. This could also explain why the trace we did was very inconsistent, with different destination servers, which we believe is due to load balancing as a result of the DNS lookup, i.e., different IP addresses are being returned for the same host name at different times.

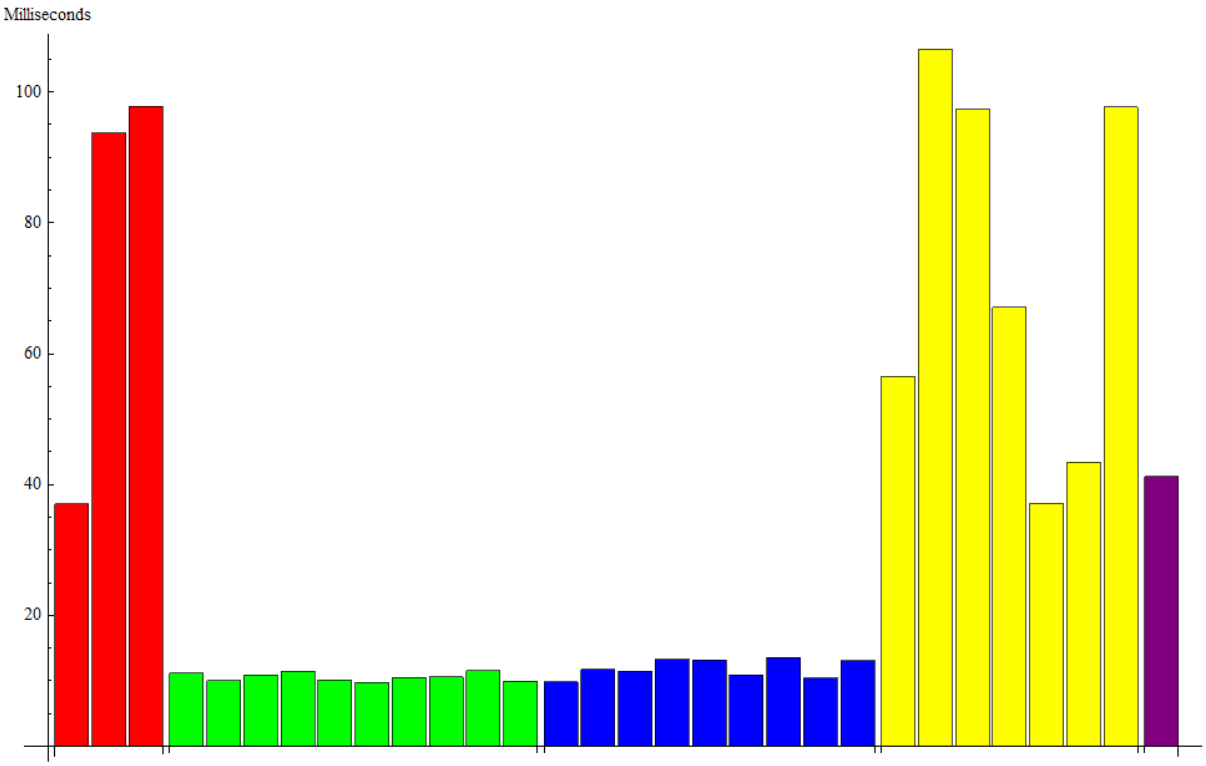


Figure 11.3.2 Latency towards different destination servers

This graph illustrates the latency towards each individual destination server. Each vertical bar is the mean of the ten pings sent at each time and the colors represents IP addresses. It is clear that the latency is dependent on the server that is being used. E.g. the servers in green and blue responds a lot faster than the server in yellow.

11.4 Web server performance test

To measure if there is any performance difference in serving a webpage over IPv4 or IPv6, we made a test using a very simple webpage with one large picture and fetched it over both IPv4 and IPv6 from a laptop running Windows Vista. We did the tests 20 times for each protocol and when running both the IIS and Apache web servers. Afterwards we calculate the estimated expected value and the standard deviation. The results are shown in Table 11.4.1 (with the time measured in seconds).

Table 11.4.1 Results (in seconds) from the web server tests

	Expected value (mean μ)	Standard deviation (σ)
Apache IPv4	0.2333	0.0181
Apache IPv6	0.2373	0.0168
IIS IPv4	0.2324	0.0278
IIS IPv6	0.2373	0.0167

The results, presented in table 11.4.1, show that the estimated expected value for IPv4 is a bit smaller than IPv6 when fetching the page from both servers. For the Apache server the estimated expected value for IPv4 is 4 milliseconds less than for IPv6. For the IIS server the corresponding difference is 4.93 milliseconds. The estimated expected values for IPv4 on IIS and Apache differed by 0.85 milliseconds. For IPv6 the estimated expected value differed only with 0.08 milliseconds. The estimated standard deviation was greater for IPv4 than it was for IPv6 on both servers. On the IIS server it was 66.3% greater, i.e., 2.78 milliseconds. However, on the Apache server it was only 7.4% greater, i.e., 1.68 milliseconds. A small number of measurements were much greater than the others during the test, causing the distribution for IPv4 on the IIS server to become wider. One should note that even if the values are small, the relative difference is noticeable. Since the observed difference between IPv4 and IPv6 is less than the standard deviation, we concluded that there is no significant difference in performance between IPv4 and IPv6 (for all the data and calculations see appendices H and I respectively). A very interesting result of this test is the nearly identical performance for both IIS and Apache for IPv6.

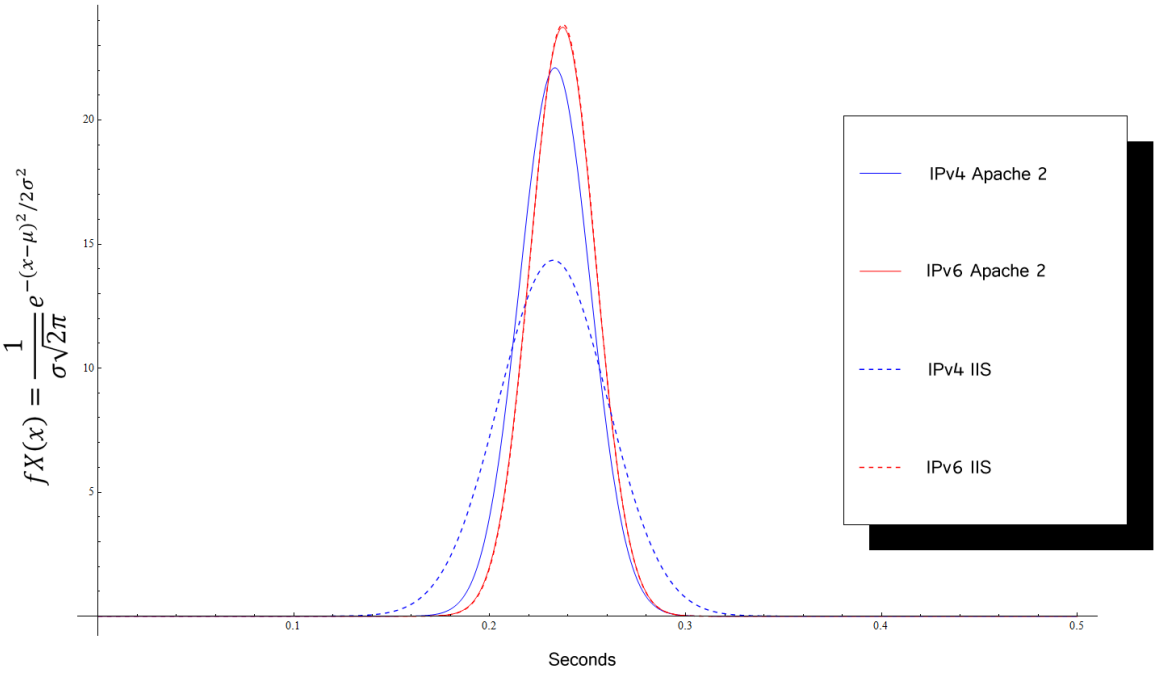


Figure 11.4.1 Graph of the delay time distribution

12. Routing protocols

Vyatta is free, software-based (Linux) program that acts as a virtual router. We will use it in order to test the IPv6 compatibility of the routing protocols RIP, RIPng, OSPF, OSPFv3, and BGP.

The network consists of three computers (from now on: Routers A, B, and C) acting as routers each running Vyatta. Router A is connected to Router B which is also connected to Router C (as illustrated in Figure 12.1).

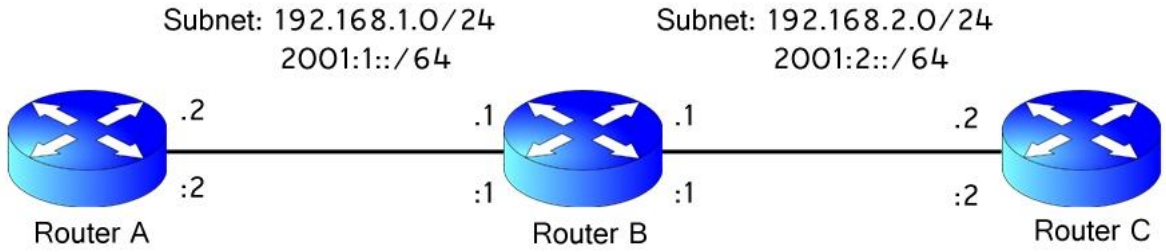


Figure 12.1 Topology for testing routing protocols

Enter `configure` at the Vyatta prompt to enter configuration mode. Use `edit` to enter (in order to edit) existing elements and `set` to add new ones. `Show` will show all the elements from the level that you are currently in. Using Tab will provide the possible commands that can follow what you have already typed. To apply the changes, type `commit`.

Below is an example of how to set the IP addresses of Router B (see appendix A.1 and A.1.1 for the complete configuration of Router B, the addresses are changed when configuring routers A and C):

```
$ configure

# edit interfaces

# edit Ethernet eth1

# set address 192.168.1.1/24

# set address 2001:1::1/64

# up

# edit Ethernet eth2

# set address 192.168.2.1/24

# set address 2001:2::1/64
```

RIPng and OSPFv3 worked as desired with IPv6, although we had trouble making BGP work properly. BGP peering was successfully set up and working, but we were unable to figure out how to make the peers distribute IPv6 networks properly to each other. The manual pages referred to elements that did not seem to exist. However, it is possible to distribute IPv6 networks using BGP as well as other network protocols, such as IPX, using multiprotocol extensions.

13. Conclusion, future work and reflections

This chapter will bring this thesis to a close with conclusions, suggestions for future work, and required reflections.

13.1 Conclusion

IPv6 has many advantages over IPv4, such as a larger address space, streamlined header, extension headers, and no need for NATs. Almost all of the most used software for implementing services in a network support IPv6 with their default configuration. Although it is a bit more complicated to implement automatic configuration of IP addresses, DNS server addresses, and default gateways for clients -- as you need software to transmit router advertisements as well as a DHCP server.

One big issue is the transition from IPv4 to IPv6. The support from hardware and software vendors is there, but the demand from customers generally controls the rate at which something changes. While everything works well over IPv4 there will be little demand for IPv6. At least in the beginning phase when (and where) there still are addresses left. Given the "IPv6 world launch" occurred on 6 June 2012, we will hopefully see a larger and larger scale of implementation of IPv6 in order to accommodate the need for addresses. IPv6 will soon be the only viable option for growing networks. The transition should be as transparent for the end users as possible. We noticed for instance that an IPv6 compatible router (not part of the lab setup) had suddenly set up a 6to4 tunnel automatically. This might have occurred in association with the IPv6 launch, but we are not certain of this.

As we saw in the local performance test of IPv4 and IPv6 (UDP and TCP traffic) there is little that indicates that one protocol would be preferable over the other in terms of throughput. Both protocols seem to be on par with each other. Regarding performance towards distant sites there is also not a lot of difference. Any difference seems to be due to the route that is chosen and the amount of competing traffic.

Tests can easily be made from locally to remote sites by users visiting test sites. One should consider that a route can be highly unstable (as shown in the trace test), fluctuate, and perform differently on a daily or even hourly basis.

With the knowledge we gained during this project we are not sure if there is anything that we would have done differently if we had to do it again. We chose to work as independent as possible and search for information ourselves rather than asking for help. This might have caused the project to progress slower, as we encountered several problems along the way, but we feel that it has been very educational to solve the majority of them by ourselves.

Our goal was to examine if there were any differences in implementing services in an IPv6 and IPv4 network, if there was any performance differences as well as how you implement the network itself. We think that we achieved that goal. There are still several more services that you can implement and test and other ways of setting up the network.

13.2 Future work

As for future work, there are some things that have been left undone that could be investigated and tested further. First an expansion of the test network could be made to add several routers, subnets, additional services, and to utilize components that do not cause bottlenecks as did our switch as we described in section 11.1). Services such as active directory, OpenLDAP, content delivery network (CDN), additional web servers, Kerberos, databases, DNS running on a Microsoft Windows server,

and firewalls could be implemented and tested. For details about configuring an IPv6 firewall with ip6tables see [125]. Transition techniques other than 6to4 could also be evaluated.

13.3 Required reflections

Regarding the economical aspect we relied solely on our own and borrowed equipment. We did not need to invest in any new equipment. This limited the size of our test network and we used an ordinary computer(s) to act as a router(s). Using our existing equipment caused problems by limiting performance in some of our test cases, but all of these could be resolved to a satisfactorily level with the exception of our switch. However, our tests showed us that an IPv4 and IPv6 network with Internet connectivity can be achieved using existing equipment and does not require investing in newer equipment. For this reason we believe that our results should have wide applicability by others and should facilitate others deciding to utilize IPv6. However, better results in terms of performance could be achieved with more modern equipment.

No issues were encountered throughout the course of this thesis regarding social and ethical aspects. Considerations were taken to avoid any real harm occurring outside of the test network while we were implementing and testing the components.

References

[1] "ARPAnet - The First Internet." [Online]. Available: http://inventors.about.com/library/weekly/aa091598.htm. [Accessed: 02-Apr-2012].

[2] "IANA — Number Resources." [Online]. Available: http://www.iana.org/numbers/. [Accessed: 05-Aug-2012].

[3] "Free Pool of IPv4 Address Space Depleted | The Number Resource Organization." [Online]. Available: http://www.nro.net/news/ipv4-free-pool-depleted. [Accessed: 02-Apr-2012].

[4] "APNIC - Two /8s allocated to APNIC from IANA." [Online]. Available: http://www.apnic.net/publications/news/2011/delegation. [Accessed: 02-Apr-2012].

[5] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," *Internet Request for Comments*, vol. RFC 1883 (Proposed Standard), Dec. 1995.

[6] J. F. Kurose and K. W. Ross, *Computer networking : a top-down approach*. Boston, Mass.: Pearson, 2010.

[7] "Unicast IPv6 addresses: IPv6." [Online]. Available: http://technet.microsoft.com/en-us/library/cc759208(v=ws.10).aspx. [Accessed: 03-Apr-2012].

[8] "FAQ: ISPs — RIPE Network Coordination Centre." [Online]. Available: http://www.ripe.net/lir-services/member-support/info/faqs/isp-related-questions. [Accessed: 09-Aug-2012].

[9] "IPv6 Address Allocation and Assignment Policy — RIPE Network Coordination Centre." [Online]. Available: http://www.ripe.net/ripe/docs/ripe-481#_8._IPv6_Provider. [Accessed: 09-Aug-2012].

[10] "IBM i information center masthead." [Online]. Available: http://publib.boulder.ibm.com/infocenter/aix/v6r1/topic/com.ibm.aix.doc/doc/base/banner.htm. [Accessed: 02-Apr-2012].

[11] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," *Internet Request for Comments*, vol. RFC 2460 (Draft Standard), Dec. 1998.

[12] J. Postel, "Internet Protocol," *Internet Request for Comments*, vol. RFC 791 (Standard), Sep. 1981.

[13] S. Hagen, *IPv6 essentials [integrating IPv6 into your IPv4 network*. Beijing; Cambridge [Mass.]; Farnham [England]: O'Reilly, 2002.

[14] S. Amante, B. Carpenter, S. Jiang, and J. Rajahalme, "IPv6 Flow Label Specification," *Internet Request for Comments*, vol. RFC 6437 (Proposed Standard), Nov. 2011.

[15] D. Borman, S. Deering, and R. Hinden, "IPv6 Jumbograms," *Internet Request for Comments*, vol. RFC 2675 (Proposed Standard), Aug. 1999.

[16] J. Reynolds, "Assigned Numbers: RFC 1700 is Replaced by an On-line Database." [Online]. Available: http://tools.ietf.org/html/rfc3232. [Accessed: 02-Apr-2012].

[17] “Protocol Numbers.” [Online]. Available: http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml. [Accessed: 09-Apr-2012].

[18] A. Conta and S. Deering, “Generic Packet Tunneling in IPv6 Specification,” *Internet Request for Comments*, vol. RFC 2473 (Proposed Standard), Dec. 1998.

[19] R. Hinden and S. Deering, “IP Version 6 Addressing Architecture,” *Internet Request for Comments*, vol. RFC 2373 (Proposed Standard), Jul. 1998.

[20] “The TCP/IP Guide - IPv6 Multicast and Anycast Addressing.” [Online]. Available: http://www.tcpipguide.com/free/t_IPv6MulticastandAnycastAddressing.htm. [Accessed: 02-Apr-2012].

[21] G. Malkin, “Traceroute Using an IP Option,” *Internet Request for Comments*, vol. RFC 1393 (Experimental), Jan. 1993.

[22] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, “Neighbor Discovery for IP version 6 (IPv6),” *Internet Request for Comments*, vol. RFC 4861 (Draft Standard), Sep. 2007.

[23] S. Deering, “ICMP Router Discovery Messages,” *Internet Request for Comments*, vol. RFC 1256 (Proposed Standard), Sep. 1991.

[24] S. Thomson and T. Narten, “IPv6 Stateless Address Autoconfiguration,” *Internet Request for Comments*, vol. RFC 2462 (Draft Standard), Dec. 1998.

[25] “Cisco Security Appliance Command Line Configuration Guide, Version 8.0 - Configuring IPv6 [Cisco ASA 5500 Series Adaptive Security Appliances] - Cisco Systems.” [Online]. Available: http://www.cisco.com/en/US/docs/security/asa/asa80/configuration/guide/ipv6.html. [Accessed: 02-Apr-2012].

[26] M. Blanchet, *Migrating to IPv6 : a practical guide to implementing IPv6 in mobile and fixed networks*. Chichester, England; Hoboken, NJ: J. Wiley & Sons, 2006.

[27] T. Hardie, “Distributing Authoritative Name Servers via Shared Unicast Addresses,” *Internet Request for Comments*, vol. RFC 3258 (Informational), Apr. 2002.

[28] “DNS Records Explained with Syntax and examples, DNS Records Tutorials.” [Online]. Available: http://www.debianhelp.co.uk/dnsrecords.htm. [Accessed: 02-Apr-2012].

[29] S. Thomson, C. Huitema, V. Ksinant, and M. Souissi, “DNS Extensions to Support IP Version 6,” *Internet Request for Comments*, vol. RFC 3596 (Draft Standard), Oct. 2003.

[30] “The TCP/IP Guide - DNS Message Processing and General Message Format.” [Online]. Available: http://www.tcpipguide.com/free/t_DNSMessageProcessingandGeneralMessageFormat-3.htm. [Accessed: 02-Apr-2012].

[31] P. V. Mockapetris, “Domain names - implementation and specification,” *Internet Request for Comments*, vol. RFC 1035 (Standard), Nov. 1987.

[32] B. Carpenter, S. Jiang, and D. Conrad, “Moving A6 to Historic Status.” [Online]. Available: http://tools.ietf.org/html/rfc6563. [Accessed: 02-Apr-2012].

[33] P. Srisuresh and M. Holdrege, “IP Network Address Translator (NAT) Terminology and Considerations,” *Internet Request for Comments*, vol. RFC 2663 (Informational), Aug. 1999.

[34] Fredrik Folke and Netterberg Marcus, “Neighbor Discovery,” 17-May-2011. [Online]. Available: http://dl.dropbox.com/u/15473045/ND%20IPv6%20Security.pdf.

[35] J. Arkko, J. Kempf, B. Zill, and P. Nikander, “SEcure Neighbor Discovery (SEND),” *Internet Request for Comments*, vol. RFC 3971 (Proposed Standard), Mar. 2005.

[36] “IPv6.com - IPv6 and IPSec - Securing the Next Generation Internet.” [Online]. Available: http://ipv6.com/articles/security/IPsec.htm. [Accessed: 02-Apr-2012].

[37] C. Kaufman, R. Perlman, and M. Speciner, *Network security : private communication in a public world*. Upper Saddle River, NJ: Prentice Hall PTR, 2002.

[38] S. Kent and K. Seo, “Security Architecture for the Internet Protocol,” *Internet Request for Comments*, vol. RFC 4301 (Proposed Standard), Dec. 2005.

[39] “IPsec architectures and implementation methods.” [Online]. Available: http://searchenterpriselinux.techtarget.com/feature/IPsec-architectures-and-implementation-methods. [Accessed: 02-Apr-2012].

[40] G. Malkin and R. Minnear, “RIPng for IPv6,” *Internet Request for Comments*, vol. RFC 2080 (Proposed Standard), Jan. 1997.

[41] “OSPF Design Guide - Cisco Systems.” [Online]. Available: http://www.cisco.com/en/US/tech/tk365/technologies_white_paper09186a0080094e9e.shtml. [Accessed: 02-Apr-2012].

[42] R. Coltun, D. Ferguson, J. Moy, and A. Lindem, “OSPF for IPv6,” *Internet Request for Comments*, vol. RFC 5340 (Proposed Standard), Jul. 2008.

[43] “Cisco IOS IPv6 Configuration Guide, Release 12.4 - Implementing OSPF for IPv6   [Cisco IOS Software Releases 12.4 Mainline],” *Cisco*. [Online]. Available: http://www.cisco.com/en/US/docs/ios/ipv6/configuration/guide/ip6-ospf.html. [Accessed: 03-Apr-2012].

[44] “Cisco IOS IPv6 Configuration Guide, Release 12.4 - Implementing IS-IS for IPv6   [Cisco IOS Software Releases 12.4 Mainline],” *Cisco*. [Online]. Available: http://www.cisco.com/en/US/docs/ios/ipv6/configuration/guide/ip6-is-is.html. [Accessed: 02-Apr-2012].

[45] “Introduction to Intermediate System-to-Intermediate System Protocol.” [Online]. Available: http://www.cisco.com/warp/public/cc/pd/iosw/prodlit/insys_wp.pdf. [Accessed: 03-Apr-2012].

[46] Y. Rekhter, T. Li, and S. Hares, “A Border Gateway Protocol 4 (BGP-4),” *Internet Request for Comments*, vol. RFC 4271 (Draft Standard), Jan. 2006.

[47] T. Bates, R. Chandra, D. Katz, and Y. Rekhter, “Multiprotocol Extensions for BGP-4,” *Internet Request for Comments*, vol. RFC 4760 (Draft Standard), Jan. 2007.

[48] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, “Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification,” *Internet Request for Comments*, vol. RFC 2205 (Proposed Standard), Sep. 1997.

[49] L. Andersson, I. Minei, and B. Thomas, “LDP Specification,” *Internet Request for Comments*, vol. RFC 5036 (Draft Standard), Oct. 2007.

[50] “What is MPLS? Multi-Protocol Label Switching stack software, a mechanism for packet forwarding in network routers.” [Online]. Available: http://network-technologies.metaswitch.com/mpls/what-is-mpls-and-gmpls.aspx. [Accessed: 02-Apr-2012].

[51] E. Kohler, M. Handley, and S. Floyd, “Datagram Congestion Control Protocol (DCCP),” *Internet Request for Comments*, vol. RFC 4340 (Proposed Standard), Mar. 2006.

[52] “About – The XMPP Standards Foundation.” [Online]. Available: http://xmpp.org/about-xmpp/. [Accessed: 12-Apr-2012].

[53] “amessage: Protocol for Jabber over IPv6.” [Online]. Available: https://web.amessage.eu/IPv6/protokoll. [Accessed: 12-Apr-2012].

[54] E. Nordmark and R. Gilligan, “Basic Transition Mechanisms for IPv6 Hosts and Routers,” *Internet Request for Comments*, vol. RFC 4213 (Proposed Standard), Oct. 2005.

[55] R. Despres, “IPv6 Rapid Deployment on IPv4 Infrastructures (6rd),” *Internet Request for Comments*, vol. RFC 5569 (Informational), Jan. 2010.

[56] B. Carpenter and K. Moore, “Connection of IPv6 Domains via IPv4 Clouds,” *Internet Request for Comments*, vol. RFC 3056 (Proposed Standard), Feb. 2001.

[57] Jivika Govil, Jivesh Govil, N. Kaur, and H. Kaur, “An examination of IPv4 and IPv6 networks : Constraints and various transition mechanisms,” in *IEEE Southeastcon, 2008*, 2008, pp. 178–185.

[58] K. Tsuchiya, H. Higuchi, and Y. Atarashi, “Dual Stack Hosts using the ‘Bump-In-the-Stack’ Technique (BIS),” *Internet Request for Comments*, vol. RFC 2767 (Informational), Feb. 2000.

[59] E. Nordmark, “Stateless IP/ICMP Translation Algorithm (SIIT),” *Internet Request for Comments*, vol. RFC 2765 (Proposed Standard), Feb. 2000.

[60] A. Durand, P. Fasano, I. Guardini, and D. Lento, “IPv6 Tunnel Broker,” *Internet Request for Comments*, vol. RFC 3053 (Informational), Jan. 2001.

[61] C. Huitema, “Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs),” *Internet Request for Comments*, vol. RFC 4380 (Proposed Standard), Feb. 2006.

[62] “Teredo Overview.” [Online]. Available: http://technet.microsoft.com/en-us/library/bb457011.aspx. [Accessed: 02-Apr-2012].

[63] F. Templin, T. Gleeson, and D. Thaler, “Intra-Site Automatic Tunnel Addressing Protocol (ISATAP),” *Internet Request for Comments*, vol. RFC 5214 (Informational), Mar. 2008.

[64] “Windows Server 2008 IPv6 Transition Technologies,” Feb-2008. [Online]. Available: http://download.microsoft.com/download/1/2/4/124331bf-7970-4315-ad18-0c3948bdd2c4/IPv6Trans.doc. [Accessed: 03-Apr-2012].

[65] “An IPv6 Deployment Guide.” The 6NET Consortium, Sep-2005.

[66] “ipv6(7) - Linux manual page.” [Online]. Available: http://www.kernel.org/doc/man-pages/online/pages/man7/ipv6.7.html. [Accessed: 02-Apr-2012].

[67] “IPv6-ready kernel.” [Online]. Available: http://mirrors.bieringer.de/Linux+IPv6-HOWTO/systemcheck-kernel.html. [Accessed: 11-Apr-2012].

[68] “Linux 3.2 - Linux Kernel Newbies.” [Online]. Available: http://kernelnewbies.org/Linux_3.2. [Accessed: 11-Apr-2012].

[69] “How to install and uninstall IPv6 in Windows XP.” [Online]. Available: http://support.microsoft.com/kb/2478747. [Accessed: 02-Apr-2012].

[70] “The Cable Guy - May 2006.” [Online]. Available: http://technet.microsoft.com/en-us/library/bb878057.aspx. [Accessed: 02-Apr-2012].

[71] “Support for IPv6 in Windows Server 2008 R2 and Windows 7.” [Online]. Available: http://technet.microsoft.com/sv-se/magazine/2009.07.cableguy(en-us).aspx. [Accessed: 02-Apr-2012].

[72] “IPv6 - Technology overview.” [Online]. Available: http://technet.microsoft.com/library/hh831730.aspx. [Accessed: 02-Apr-2012].

[73] “Configuring IPv6 in Mac OS X v10.6.7 or later.” [Online]. Available: http://support.apple.com/kb/HT4667. [Accessed: 02-Apr-2012].

[74] “Robservatory » Blog Archive » A useless analysis of OS X release dates.” [Online]. Available: http://www.robservatory.com/?p=46. [Accessed: 02-Apr-2012].

[75] “IPv6 in FreeBSD.” [Online]. Available: http://www.freebsd.org/ipv6/. [Accessed: 02-Apr-2012].

[76] “FreeBSD IPv6-only Support.” [Online]. Available: http://www.freebsd.org/ipv6/ipv6only.html. [Accessed: 02-Apr-2012].

[77] “IPv6 Support in Solaris.” [Online]. Available: http://www.softpanorama.org/Net/Internet_layer/ipv6.shtml. [Accessed: 02-Apr-2012].

[78] “2012 | Netcraft.” [Online]. Available: http://news.netcraft.com/archives/2012/. [Accessed: 02-Apr-2012].

[79] “Can you use Google Web Server (GWS) in your projects? - O’Reilly Answers.” [Online]. Available: http://answers.oreilly.com/topic/2473-can-you-use-google-web-server-gws-in-your-projects/. [Accessed: 02-Apr-2012].

[80] “Google mystery server runs 13% of active websites • The Register.” [Online]. Available: http://www.theregister.co.uk/2010/01/29/google_web_server/. [Accessed: 02-Apr-2012].

[81] “Overview of new features in Apache HTTP Server 2.0 - Apache HTTP Server.” [Online]. Available: http://httpd.apache.org/docs/trunk/new_features_2_0.html. [Accessed: 02-Apr-2012].

[82] “Apache Week. Apache 2 Release.” [Online]. Available: http://www.apacheweek.com/features/ap2. [Accessed: 02-Apr-2012].

[83] “Enabling IPv6 Support in nginx,” *Oleksiy Kovyrin*. [Online]. Available: http://kovyrin.net/2010/01/16/enabling-ipv6-support-in-nginx/. [Accessed: 02-Apr-2012].

[84] “Current Status of IPv6 Support for Networking Applications.” [Online]. Available: http://www.deepspace6.net/docs/ipv6_status_page_apps.html. [Accessed: 02-Apr-2012].

[85] I. van Beijnum, *Running IPV6 : [a practical guide to configuring IPV6 for Windows XP, MacOS X, FreeBSD, Red Hat Linux, Cisco routers, DNS and BIND, Zebra and Apache 2]*. Berkeley, Calif: Apress, 2006.

[86] “Using Internet Explorer to Access IPv6 Websites.” [Online]. Available: http://msdn.microsoft.com/en-us/library/windows/desktop/ms740593(v=vs.85).aspx. [Accessed: 02-Apr-2012].

[87] “How To Enable IPV6 Using Google Chrome.” [Online]. Available: http://chromestory.com/2011/02/how-to-enable-ipv6-using-google-chrome/. [Accessed: 02-Apr-2012].

[88] “MSDN Blogs.” [Online]. Available: http://blogs.msdn.com/b/exchangefaqs/archive/2008/02/01/will-there-be-any-support-for-ipv6-in-exchange-2003.aspx. [Accessed: 02-Apr-2012].

[89] “How-to: IPv6 workstation.” [Online]. Available: http://www.viagenie.qc.ca/en/ipv6/fullipv6ws/how-to.shtml. [Accessed: 02-Apr-2012].

[90] “Postfix IPv6 Support.” [Online]. Available: http://www.postfix.org/IPV6_README.html. [Accessed: 02-Apr-2012].

[91] “Outlook 2007 supports Internet Protocol version 6 (IPv6).” [Online]. Available: http://support.microsoft.com/kb/924469. [Accessed: 02-Apr-2012].

[92] “National Advanced IPv6 Centre of Excellence.” [Online]. Available: http://www.nav6.org/Research/ipv6%20application%20support.php. [Accessed: 02-Apr-2012].

[93] “BIND | Internet Systems Consortium.” [Online]. Available: http://www.isc.org/software/bind. [Accessed: 02-Apr-2012].

[94] “What’s New in DNS in Windows Server 2008.” [Online]. Available: http://technet.microsoft.com/en-us/library/cc753143(WS.10).aspx. [Accessed: 02-Apr-2012].

[95] “Domain Name Service (DNS).” [Online]. Available: http://technet.microsoft.com/library/bb726935. [Accessed: 02-Apr-2012].

[96] “IPv6firewallsandSecurity_eng - Campus IPv6 Wiki.” [Online]. Available: http://ipv6.niif.hu/m/IPv6firewallsandSecurity_eng. [Accessed: 02-Apr-2012].

[97] “IPv6 Firewalls - ARIN IPv6 Wiki.” [Online]. Available: http://www.getipv6.info/index.php/IPv6_Firewalls. [Accessed: 02-Apr-2012].

[98] “mf-firewall - Linux Full Firewall for IPv6 and IPv4, Routing, Interface setup, and QOS - Google Project Hosting.” [Online]. Available: http://code.google.com/p/mf-firewall/. [Accessed: 02-Apr-2012].

[99] “m0n0wall - Screenshots.” [Online]. Available: http://m0n0.ch/wall/screenshots.php. [Accessed: 02-Apr-2012].

[100] “pfSense Open Source Firewall Distribution - Home.” [Online]. Available: http://pfsense.com/index.php?id=26. [Accessed: 02-Apr-2012].

[101] “Screenshots | Firewall Builder.” [Online]. Available: http://www.fwbuilder.org/4.0/screenshots.html. [Accessed: 02-Apr-2012].

[102] “FortiGate IPv6 support Technical Note.” 03-Oct-2008.

[103] “Amendment No. 2 To The Registration Statement On Form S-1/A about Skype.” [Online]. Available: http://www.sec.gov/Archives/edgar/data/1498209/000119312511056174/ds1a.htm. [Accessed: 12-Apr-2012].

[104] J. Arkko and A. Keranen, “Experiences from an IPv6-Only Network.” [Online]. Available: http://tools.ietf.org/html/draft-arkko-ipv6-only-experience-05. [Accessed: 12-Apr-2012].

[105] “Networks with IPv6 - One Year Later — RIPE Labs.” [Online]. Available: https://labs.ripe.net/Members/mirjam/networks-with-ipv6-one-year-later. [Accessed: 11-Apr-2012].

[106] “Interesting Graph - Networks with IPv6 over Time — RIPE Labs.” [Online]. Available: https://labs.ripe.net/Members/emileaben/interesting-graph-networks-with-ipv6-over-time. [Accessed: 11-Apr-2012].

[107] “RIPEstat — Internet Measurements and Analysis.” [Online]. Available: https://stat.ripe.net/AS39651. [Accessed: 11-Apr-2012].

[108] “För dig som undrar vad som gäller med övergången till IPv6 - Com Hem.” [Online]. Available: http://www.comhem.se/comhem/bredband/bredband-fran-comhem/overg-ng-till-ipv6/-/6260/611416/-/index.html. [Accessed: 11-Apr-2012].

[109] T. Bilski, “Network performance issues in IP transition phase,” in *Networked Computing and Advanced Information Management (NCM), 2010 Sixth International Conference on*, 2010, pp. 39 –44.

[110] S. Narayan and Y. Shi, “TCP/UDP network performance analysis of windows operating systems with IPv4 and IPv6,” in *Signal Processing Systems (ICSPS), 2010 2nd International Conference on*, 2010, vol. 2, pp. V2–219 –V2–222.

[111] “Measuring World IPv6 Day - Comparing IPv4 and IPv6 Performance — RIPE Labs.” [Online]. Available: https://labs.ripe.net/Members/emileaben/measuring-world-ipv6-day-comparing-ipv4-and-ipv6-performance. [Accessed: 10-Apr-2012].

[112] “IPv6 test - IPv6/4 connectivity and speed test.” [Online]. Available: http://ipv6-test.com/. [Accessed: 11-Apr-2012].

[113] “tunneling | The Linux Foundation.” [Online]. Available: http://www.linuxfoundation.org/collaborate/workgroups/networking/tunneling. [Accessed: 16-Jul-2012].

[114] “dhcpd(8): Dynamic Host config Protocol Server - Linux man page.” [Online]. Available: http://linux.die.net/man/8/dhcpd. [Accessed: 07-Sep-2012].

[115] “Various DHCPv6 Server and Client Configuration Examples.” [Online]. Available: http://linux.ardynet.com/ipv6setup.php. [Accessed: 16-Jul-2012].

[116] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, “DNS Security Introduction and Requirements,” *Internet Request for Comments*, vol. RFC 4033 (Proposed Standard), Mar. 2005.

[117] “DNSSEC verification with dig « \1.” [Online]. Available: http://backreference.org/2010/11/17/dnssec-verification-with-dig/. [Accessed: 03-Aug-2012].

[118] “Bug #1000202 ‘Unable to unmount NFSv3 or NFSv4 mounts when mount...’ : Bugs : ‘util-linux’ package : Ubuntu.”[Online]. Available: https://bugs.launchpad.net/ubuntu/+source/util-linux/+bug/1000202. [Accessed: 09-Aug-2012].

[119] “Chapter 4. Advanced use of VLC.” [Online]. Available: http://www.videolan.org/doc/play-howto/en/ch04.html#id590873. [Accessed: 16-Jul-2012].

[120] “Decrypt HTTPS Traffic Using Wireshark And Key File,” *segfault.in*. [Online]. Available: http://segfault.in/2010/11/decrypt-https-traffic-using-wireshark-and-key-file/. [Accessed: 21-Aug-2012].

[121] “Postfix Configuration Parameters.” [Online]. Available: http://www.postfix.org/postconf.5.html#mynetworks. [Accessed: 16-Jul-2012].

[122] “Configure Packet Filters to Allow ICMP Traffic.” [Online]. Available: http://technet.microsoft.com/en-us/library/ee649189%28v=ws.10%29. [Accessed: 08-Aug-2012].

[123] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney, “Dynamic Host Configuration Protocol for IPv6 (DHCPv6),” *Internet Request for Comments*, vol. RFC 3315 (Proposed Standard), Jul. 2003.

[124] “LAN Ethernet Maximum Rates, Generation, Capturing & Monitoring - NST Wiki.” [Online]. Available: http://wiki.networksecuritytoolkit.org/nstwiki/index.php/LAN_Ethernet_Maximum_Rates,_Generation,_Capturing_%26_Monitoring. [Accessed: 07-Sep-2012].

[125] F. Folke, “Security for home, small and medium sized enterprises IPv6 networks.” [Online]. Available: http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/120621-Fredrik_Folke-with-cover.pdf. [Accessed: 17-Aug-2012].

Appendix A - Configuration files

A.1 Routing Protocols

```
interfaces {
     Ethernet eth1{
     address 192.168.1.1/24
     address 2001:1::1/64
          }
     Ethernet eth2{
     address 192.168.2.1/24
          address 2001:2::1/64
                    }
}
protocols {
     ospf {
      area0 {
       network{
                    192.168.1.0/24
                    192.168.2.0/24
               }
             }
          }
     ospfv3 {
      area 0.0.0.0 {
                    interface eth1
                    interface eth2
                        }
          parameters {
                    router-id 1.1.1.1
            }
     redistribute {
                    connected {}
          }
                  }
     bgp 1 {
          neighbor 2001:2::2 {
                    remote-as 2
                    }
          network 192.168.5.0/24{}
          parameters {
                    router-id 1.1.1.1
                    }
          redistribute {
                    connected {}
                    }
          }
}
```

A.1.1 With RIP and RIPng

```
Protocols {
     rip {
      networks 192.168.1.0/24
      networks 192.168.2.0/24
     }
     ripng {
      network 2001:1::/64
      network 2001:2::/64
     }
}
```

A.2 DHCPD

Location: `/etc/dhcp/dhcpd.conf`

Content:

```
option domain-name-servers 130.237.72.246, 130.237.72.250, 8.8.8.8,
8.8.4.4;
option ip-forwarding off;

default-lease-time 600;
max-lease-time 7200;

subnet 192.168.1.0 netmask 255.255.255.0 {
     range 192.168.1.10 192.168.1.50;
     option routers 192.168.1.1;
}
```

A.3 DHCPD6

Location: `/etc/dhcp/dhcpd6.conf`

Content:

```
default-lease-time 600;
max-lease-time 7200;
log-facility local7;
option dhcp6.name-servers 2620:0:ccc::2;
subnet6 2002:53fb:377d:1::/64 {
        # Range for clients
        range6 2002:53fb:377d:1::20 2002:53fb:377d:1::50;
        # Additional options
        #option domain-name-servers 2620:0:ccc::2;
        #option dhcp6.domain-search "domain.example";
        # Prefix range for delegation to sub-routers
        #prefix6 2001:db8:0:100:: 2001:db8:0:f00:: /56;
        # Example for a fixed host address
        #host specialclient {
```

```
        #       host-identifier option dhcp6.client-id
00:01:00:01:4a:1f:ba:e3:60:b9:1f:01:23:45;
        #       fixed-address6 2001:db8:0:1::127;
        #}
}
```

A.4 Radvd

Location: `/etc/radvd.conf`

Content:

```
interface eth1 {
        AdvSendAdvert on;
        MinRtrAdvInterval 3;
        MaxRtrAdvInterval 10;
     AdvManagedFlag on;
     AdvDefaultLifetime 600;
        prefix 2002:53fb:377d:1::/64 {
                AdvOnLink on;
                AdvAutonomous on;
                AdvRouterAddr on;
        };
};
```

A.5 DNS

A.5.1 1.168.192.in-addr.arpa

Location: `/etc/bind/`

Content:

```
$TTL 86400
@    IN   SOA  ns1.ex.jobb. daboss.ex.jobb. (
               20120531  ; Serial
                604800       ; Refresh
                 86400       ; Retry
               2419200       ; Expire
                 86400 ) ; Negative Cache TTL
;
     IN   NS   ns1.ex.jobb.
1    IN   PTR  gateway.ex.jobb.
```

A.5.2 ex.jobb.db

Location: /

Content:

```
$TTL 86400
@    IN   SOA  ns1.ex.jobb. daboss.ex.jobb. (
               20120533  ; Serial
                604800       ; Refresh
                 86400       ; Retry
```

```
              2419200       ; Expire
                86400 ) ; Negative Cache TTL
;
     IN   NS   ns1.ex.jobb.
gateway   IN   A    192.168.1.1
gateway   IN   AAAA 2002:53fb:377d:1::1
ns1  IN   A    192.168.1.1
www  IN   CNAME     gateway
```

A.5.3 ipv6.ip6.arpa

Location: /etc/bind/

Content:

```
$TTL 86400
@    IN   SOA  ns1.ex.jobb. daboss.ex.jobb. (
                     1       ; Serial
                604800       ; Refresh
                 86400       ; Retry
                2419200       ; Expire
                  86400 ) ; Negative Cache TTL
;
     IN   NS   ns1.ex.jobb.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0   IN   PTR  gateway.ex.jobb.
```

A.5.4 named.conf

Location: /etc/bind/

Content:

```
include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";
```

A.5.5 named.conf.local

Location: /etc/bind/

Content:

```
zone "ex.jobb" {
     type master;
//   allow-transfer {192.168.1.21;};
//   also-notify {192.168.1.21;};
     notify no;
     file "/etc/bind/ex.jobb.db.signed";
};

zone "1.168.192.in-addr.arpa" {
     type master;
     notify no;
     file "/etc/bind/1.168.192.in-addr.arpa";
};
```

```
zone "1.0.0.0.1.7.2.3.b.f.3.5.2.0.0.2.ip6.arpa" {
        type master;
        notify no;
        file "/etc/bind/ipv6.ip6.arpa";
};
```

A.5.6 named.conf.options

Location: `/etc/bind/`

Content:

```
options {
     directory "/var/cache/bind";

     // If there is a firewall between you and nameservers you want
     // to talk to, you may need to fix the firewall to allow multiple
     // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

     // If your ISP provided one or more IP addresses for stable
     // nameservers, you probably want to use them as forwarders.
     // Uncomment the following block, and insert the addresses replacing
     // the all-0's placeholder.

      forwarders {
          83.255.245.11;
          2001:4860:4860::8888;
      };

     //========================================================================
====
     // If BIND logs error messages about the root key being expired,
     // you will need to update your keys.  See https://www.isc.org/bind-
keys
     //========================================================================
====


     auth-nxdomain no;    # conform to RFC1035
     listen-on-v6 { any; };
     dnssec-enable yes;
        dnssec-validation auto;
        dnssec-lookaside auto;
        //bindkeys-file "/etc/bind/bind.keys";
};

//include "/etc/bind/bind.keys";

managed-keys {
     "ex.jobb." initial-key 257 3 8
"AwEAAbq8RkyJ+8ie87JjHBDfxAqlKxAzfk9cJT/s0Tk4ljlgC9rSRnu1
prwwcNBfX1CcwCU3pQoXEIApiO/PiEA4Alq04KjUpEmUYvi7O3t3h8iV
Zy+9HaFQdqbpTGuAjhu5ojzU1K9MIJKNk4kS95iIQDOc6Sqh4ohAgSat
Pj/RKt0wCqDkYkzjeDSjSGml2lqcDPaNDaNjJZtEfetFcN1Gk1EZ+Amx
k5VF1S1bQQFMdMnsP/e0eLPeXAmzfSzdmpUCVRMriPXdkpXQX3FWUgjr
K1LxaBMSvxwuXo3YMbukpqDe9M0ltawnRCzc0qQz4EKZzuiSm06tBe5T TAaypMBGBx8=";
};
```

A.6 vlm.conf

Location: wherever suitable

Content:

```
new channel1 broadcast enabled

load playlist.conf

setup channel1 output #http{mux=asf,dst=:8090/live1}


setup channel1 option http-reconnect

setup channel1 option no-sout-rtp-sap option no-sout-standard-sap
option ttl=1 option sout-keep


control channel1 play
```

Appendix B - Hardware

Server and gateway:

Processor: Intel Core i5-2500 3.30GHz
Motherboard: ASUS P8 Z68-V LX
Chipset: Intel Z68
RAM: Kingston 2 x 4GB 1600MHz DDR3
Graphics card: AMD Radeon HD 6870 1GB GDDR5
Hard drive: Western Digital Caviar Green 1 TB
NIC: Realtek 8111E Gigabit LAN Controller
Operating System: Ubuntu Server 12.04 64-bit and Windows Server 2008 R2 Enterprise

Client 1:

Brand: HP Compaq 6710b
Processor: Intel Core 2 Duo T9300 2.5GHz
Chipset: Mobile Intel® GM965
RAM: 2GB DDR2 SDRAM, 667-MHz
Graphics card: Mobile Intel 965 express chipset family
Hard drive: Fujitsu MHZ2250BH G2 SATA 250GB 5400rpm
NIC: Broadcom NetLink Gigabit Ethernet 10/100/1000Mbps
Operating System: Windows Vista Business Service Pack 2 and Xubuntu Linux

Client 2:

Brand: Samsung NC10
Processor: Intel Atom N270 1.60GHz
RAM: 1GB SO-DIMM DDR2 667 MHz
Graphics card: GMA 950 128MB
Hard drive: HM160HI 160GB SATA 5400rpm
NIC: Marvell Yukon 88e8040 PCI-E Fast Ethernet controller 10/100Mbps
Operating System: Windows XP Home Edition Service Pack 3 and Ubuntu Desktop 12.04

Client 3:

Brand: Acer Aspire 5739G
Processor: Intel Core 2 Duo P8700 2.53GHz
RAM: 4GB DDR3 SDRAM
Graphics card: GeForce GT 240M
Hard drive: Seagate 500GB 5400rpm SATA-II
NIC: Atheros AR8131 PCI-E Gigabit Ethernet Controller
OS: Windows 7 Professional, 64-bit (SP1)

Switch:

Brand: Zonet ZFS3008
Ports: 8 x 10/100Mbps

Appendix C - Data collected with PCATTCP

Expressed in KB/s.

IPv4		IPv6	
TCP	UDP	TCP	UDP
10556.10	11578.13	10468.41	11432.69
10252.23	11578.13	10508.71	11416.77
10329.81	11602.73	9011.59	11400.88
10408.55	11556.40	8996.74	11385.04
10233.02	11610.96	8981.94	11424.73
10310.30	11578.13	8986.87	11385.04
10336.31	11586.32	8996.74	11424.72
10323.29	11521.14	10495.23	11408.82
10323.29	11569.97	10495.24	11392.96
10303.81	11553.64	8942.72	11400.88
10297.34	11586.32	10448.38	11432.71
10297.34	11602.74	10481.82	11432.69
10252.23	11627.43	10428.43	11385.03
10329.80	11635.70	9001.68	11377.12
10310.30	11594.53	8972.10	11385.04
10329.81	11586.32	8986.87	11400.87
10355.91	11496.89	8972.11	11377.13
10355.92	11627.44	10475.10	11432.71
10233.01	11594.53	10488.51	11385.03
10271.51	11594.53	9011.59	11369.24
10401.95	11594.52	10349.38	11392.96
10349.37	11569.95	8942.72	11424.73
10336.31	11610.96	10435.07	11377.13
10375.59	11594.52	8899.00	11377.12
10336.32	11569.97	8972.11	11392.96
10316.79	11594.52	10395.35	11400.88
10369.03	11488.82	10336.32	11392.96
10290.87	11537.37	10349.38	11392.96
10428.43	11594.52	10481.82	11392.94
10355.92	11578.13	9016.55	11392.96
10388.75	11586.33	9011.59	11392.96
10316.80	11643.97	8991.80	11424.73
10362.47	11602.74	10481.80	11408.82
10277.96	11610.96	10488.51	11385.04
10355.91	11643.97	10501.97	11424.72
10271.51	11643.97	10475.10	11424.72
10362.46	11652.25	10415.17	11424.73
10355.92	11610.95	10481.80	11392.96
10233.01	11619.19	8932.97	11385.04
10323.30	11610.96	10475.10	11377.12
10329.81	11619.20	10461.73	11392.96
10355.91	11594.53	8962.29	11377.12
10388.75	11610.95	10475.10	11432.71
10382.17	11635.70	9026.48	11392.94
10329.80	11660.53	8967.19	11385.04
10369.03	11610.96	9036.44	11392.96
10303.82	11602.74	10488.51	11392.69
10277.96	11586.33	10508.70	11400.88
10375.60	11635.70	10461.73	11385.03
10323.29	11660.53	8972.11	11424.73

Appendix D – Mathematica applied to PCATTCP measurements

Performance test IPv4 TCP

x1={10556.10,10252.23,10329.81,10408.55,10233.02,10310.30,10336.31,10323.29,10323.29,10303.81,10297.34,10297.34,10252.23,10329.80,10310.30,10329.81,10355.91,10355.92,10233.01,10271.51,10401.95,10349.37,10336.31,10375.59,10336.32,10316.79,10369.03,10290.87,10428.43,10355.92,10388.75,10316.80,10362.47,10277.96,10355.91,10271.51,10362.46,10355.92,10233.01,10323.30,10329.81,10355.91,10388.75,10382.17,10329.80,10369.03,10303.82,10277.96,10375.60,10323.29};

n=50;

Calculating the estimated expected value for the IPv4 TCP test.

$$\mu 1 = \frac{1}{n} * \sum_{i=1}^{n} \mathrm{x1}[[i]]$$

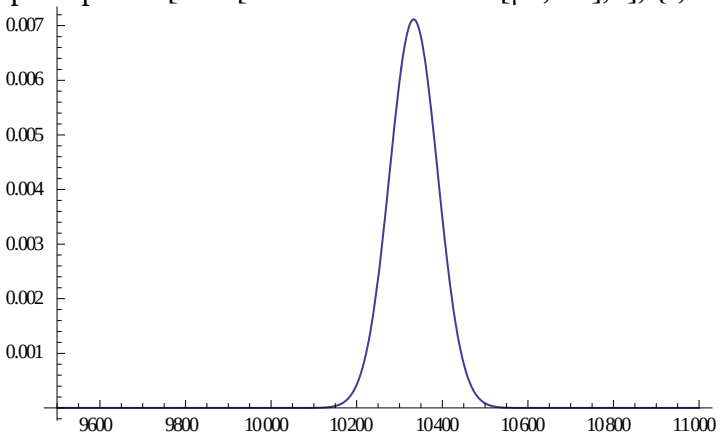
10333.1

Calculating the estimated standard deviation for the IPv4 TCP test.

$$\sigma 1 = \sqrt{\frac{1}{n-1} * (\sum_{i=1}^{n} (\mathrm{x1}[[i]] - \mu 1)^2)}$$

56.0711

ipv4tcp=Plot[PDF[NormalDistribution[μ1, σ1], t], {t, 9500, 11000}, PlotRange -> All]



Performance test IPv4 UDP

x2={11578.13,11578.13,11602.73,11556.40,11610.96,11578.13,11586.32,11521.14,11569.97,11553.64,11586.32,11602.74,11627.43,11635.70,11594.53,11586.32,11496.89,11627.44,11594.53,11594.53,11594.52,11569.95,11610.96,11594.52,11569.97,11594.52,11488.82,11537.37,11594.52,11578.13,11586.33,11643.97,11602.74,11610.96,11643.97,11643.97,11652.25,11610.95,11619.19,11610.96,11619.20,11594.53,11610.95,11635.70,11660.53,11610.96,11602.74,11586.33,11635.70,11660.53};

n=50;

Calculating the estimated expected value for the IPv4 UDP test.

$$\mu 2 = \frac{1}{n} * \sum_{i=1}^{n} \mathrm{x2}\left[[i]\right]$$

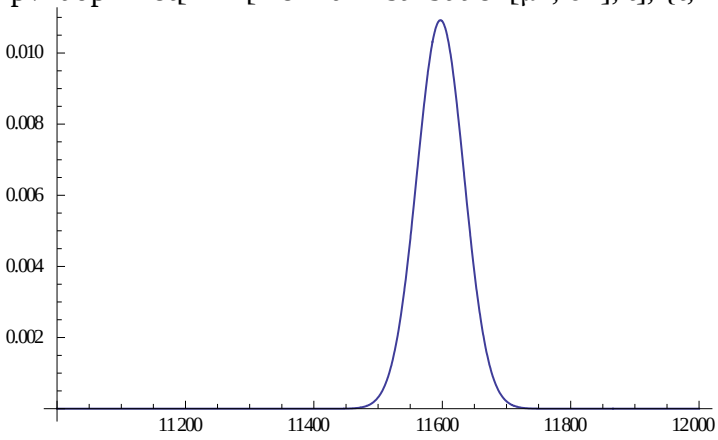
11597.2

Calculating the estimated standard deviation for the IPv4 UDP test.

$$\sigma 2 = \sqrt{\frac{1}{n-1} * (\sum_{i=1}^{n} (\mathrm{x2}[[i]] - \mu 2)^2)}$$

36.5246

ipv4udp=Plot[PDF[NormalDistribution[μ2, σ2], t], {t, 11000,12000}, PlotRange -> All]



Performance test IPv6 TCP

x3={10468.41,10508.71,9011.59,8996.74,8981.94,8986.87,8996.74,10495.23,10495.24,8942.72,10448.38,10481.82,10428.43,9001.68,8972.10,8986.87,8972.11,10475.10,10488.51,9011.59,10349.38,8942.72,10435.07,8899.00,8972.11,10395.35,10336.32,10349.38,10481.82,9016.55,9011.59,8991.80,10481.80,10488.51,10501.97,10475.10,10415.17,10481.80,8932.97,10475.10,10461.73,8962.29,10475.10,9026.48,8967.19,9036.44,10488.51,10508.70,10461.73,8972.11};

n=50;

Calculating the estimated expected value for the IPv6 TCP test.

$$\mu 3 = \frac{1}{n} * \sum_{i=1}^{n} \text{x3}[[i]]$$

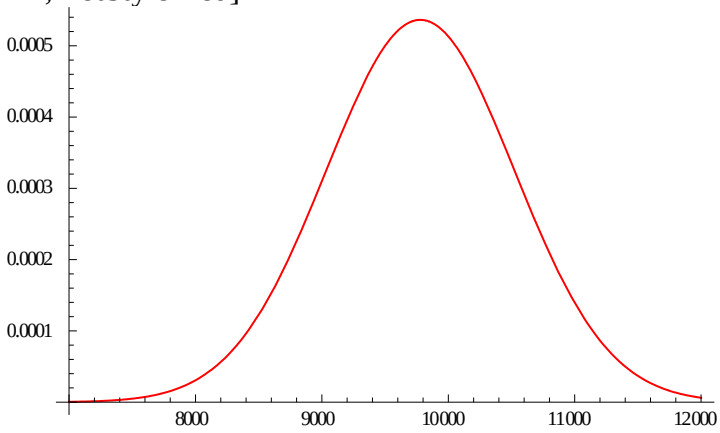
9778.89

Calculating the estimated standard deviation for the IPv6 TCP test.

$$\sigma 3 = \sqrt{\frac{1}{n-1} * (\sum_{i=1}^{n} (\text{x3}[[i]] - \mu 3)^2)}$$

743.88

ipv6tcp=Plot[PDF[NormalDistribution[μ3, σ3], t], {t, 7000,12000}, PlotRange -> All,PlotStyle Red]



Performance test IPv6 UDP

x4={11432.69,11416.77,11400.88,11385.04,11424.73,11385.04,11424.72,11408.82,11392.96
,11400.88,11432.71,11432.69,11385.03,11377.12,11385.04,11400.87,11377.13,11432.71,113
85.03,11369.24,11392.96,11424.73,11377.13,11377.12,11392.96,11400.88,11392.96,11392.9
6,11392.94,11392.96,11392.96,11424.73,11408.82,11385.04,11424.72,11424.72,11424.73,11
392.96,11385.04,11377.12,11392.96,11377.12,11432.71,11392.94,11385.04,11392.96,11392.
69,11400.88,11385.03,11424.73};

Calculating the estimated expected value for the IPv6 UDP test.

$$\mu 4 = \frac{1}{n} * \sum_{i=1}^{n} \text{x4}[[i]]$$

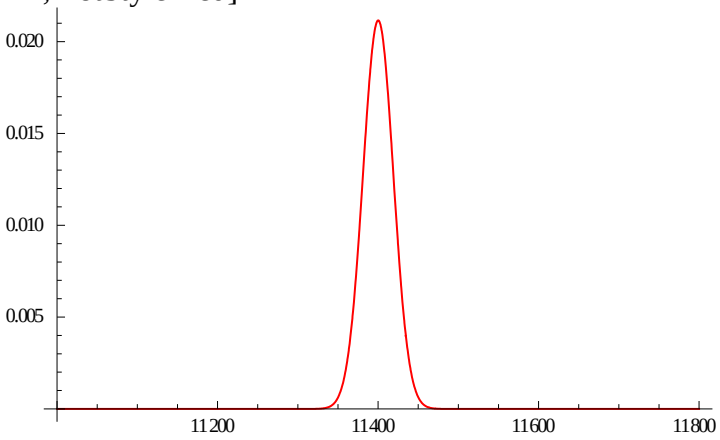
11400.1

Calculating the estimated standard deviation for the IPv6 UDP test.

$$\sigma 4 = \sqrt{\frac{1}{n-1} * (\sum_{i=1}^{n} (\text{x4}[[i]] - \mu 4)^2)}$$

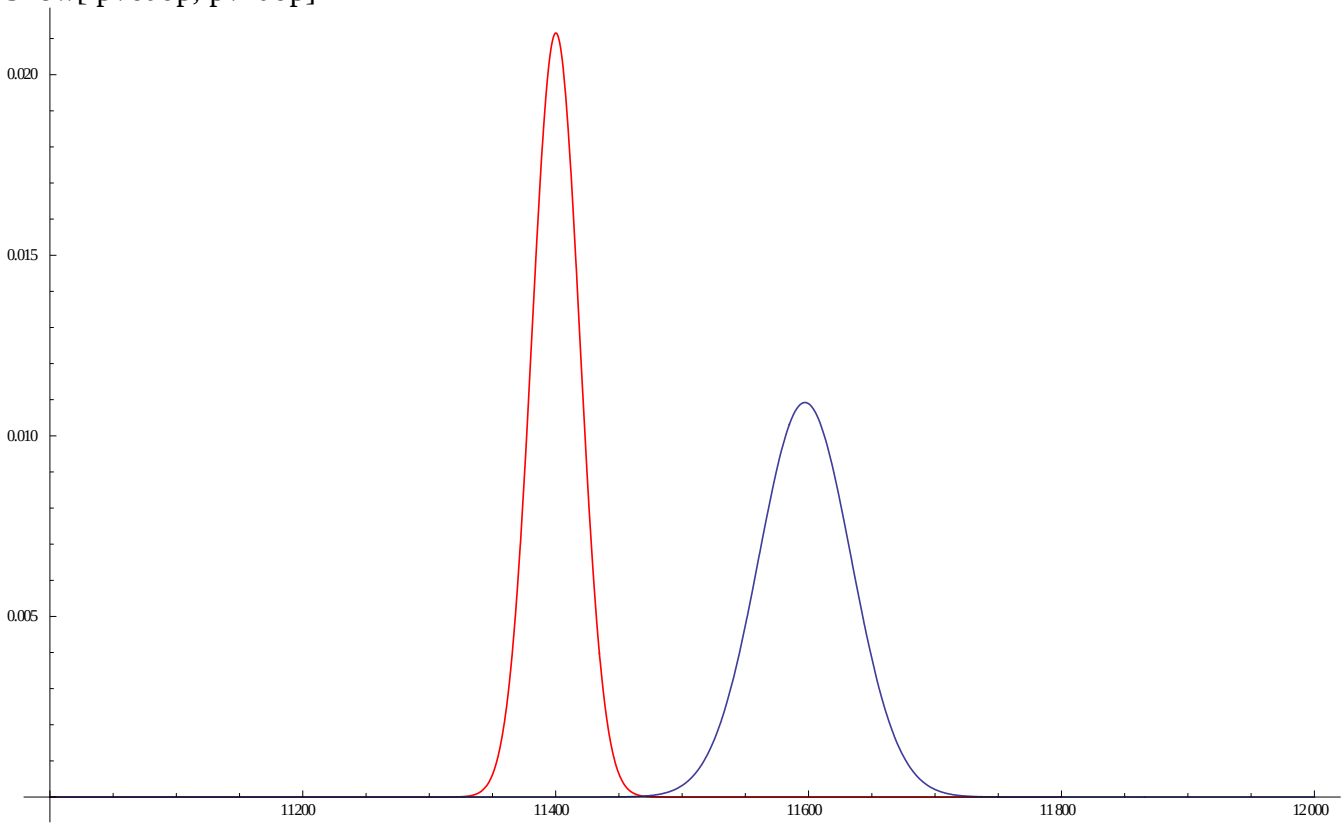
18.8595

ipv6udp=Plot[PDF[NormalDistribution[μ4, σ4], t], {t, 11000,11800}, PlotRange ->
All,PlotStyle Red]



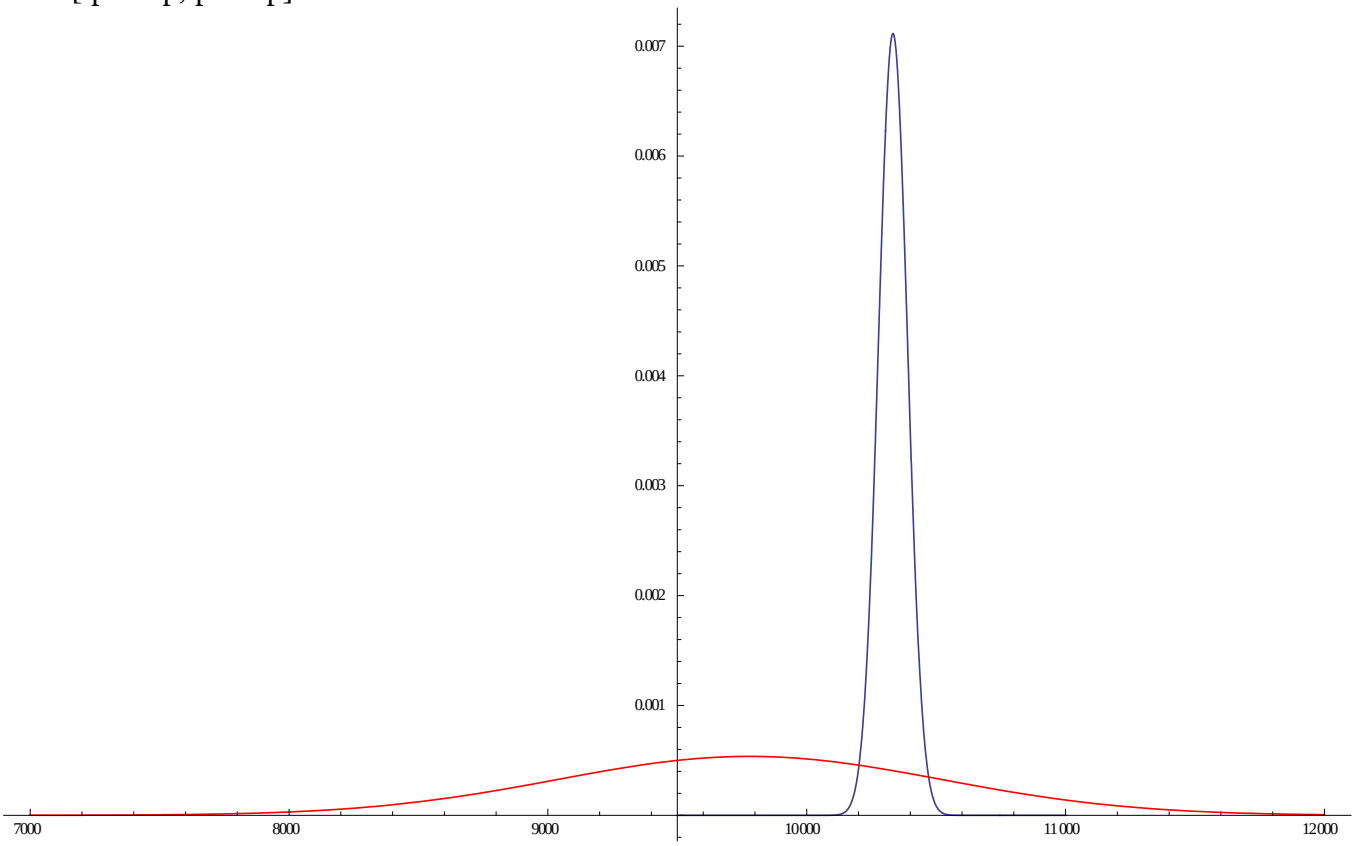
The IPv6 UDP and the IPv4 UDP distribution plotted.

Show[ipv6udp,ipv4udp]



The IPv6 TCP and the IPv4 TCP distribution plotted.

Show[ipv4tcp,ipv6tcp]



Appendix E - Data collected with iperf

Expressed in KB/s.

IPv4		IPv6	
TCP	UDP	TCP	UDP
11494	11679	11329	10870
11493	11680	11334	10875
11492	11681	11334	10875
11493	11682	11334	10875
11493	11682	11334	10864
11492	11682	11334	10875
11493	11681	11334	10875
11493	11681	11334	10875
11493	11671	11334	10875
11492	11666	11334	10875
11493	11681	11334	10875
11492	11652	11334	10875
11493	11682	11334	10875
11492	11682	11334	10875
11493	11682	11334	10872
11493	11682	11334	10875
11492	11683	11334	10875
11493	11682	11334	10875
11493	11682	11334	10875
11493	11682	11334	10875
11492	11680	11334	10875
11493	11682	11334	10875
11492	11682	11334	10875
11492	11672	11334	10864
11493	11663	11334	10875
11492	11678	11334	10875
11493	11668	11334	10875
11493	11683	11334	10875
11492	11682	11334	10875
11493	11682	11334	10875
11493	11682	11334	10875
11493	11682	11334	10871
11492	11682	11334	10875
11493	11682	11334	10843
11493	11682	11334	10875
11492	11682	11334	10875
11493	11666	11334	10875
11493	11682	11334	10875
11493	11682	11334	10875
11492	11682	11334	10875
11493	11682	11334	10875
11493	11683	11334	10875
11492	11682	11334	10875
11493	11682	11334	10868
11493	11682	11334	10875
11493	11682	11334	10873
11493	11670	11334	10875
11492	11682	11334	10875
11492	11682	11334	10875
11493	11682	11334	10875

Appendix F – Mathematica analysis of iperf measurements

Speed test with iperf with TCP over IPv4.

x1={11494,11493,11492,11493,11493,11492,11493,11493,11493,11492,11493,11492,11493,
11492,11493,11493,11492,11493,11493,11493,11492,11493,11492,11492,11493,11492,1149
3,11493,11492,11493,11493,11493,11492,11493,11493,11492,11493,11493,11493,11492,11
493,11493,11492,11493,11493,11493,11493,11492,11492,11493};

n=50;

Calculating the estimated expected value for TCP over IPv4.

$$\mu 1 = \frac{1}{n} * \sum_{i=1}^{n} \text{x1}[[i]] \,//N$$

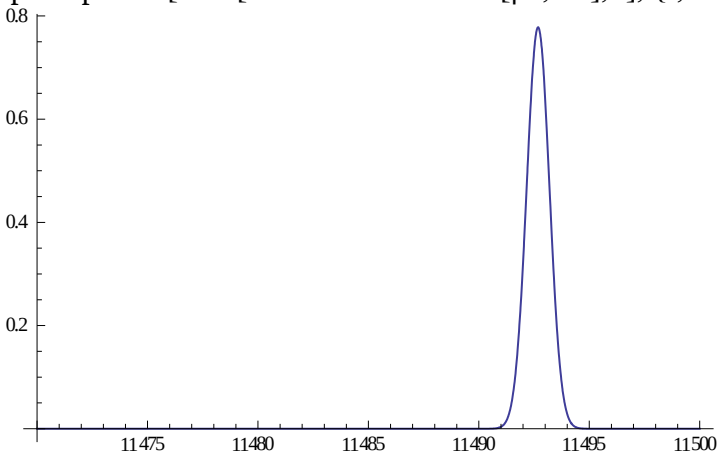
11492.7

Calculating the estimated standard deviation for TCP over IPv4.

$$\sigma 1 = \sqrt{\frac{1}{n-1} * (\sum_{i=1}^{n} (\text{x1}[[i]] - \mu 1)^2)}$$

0.512696

ipv4tcp=Plot[PDF[NormalDistribution[μ1, σ1], t], {t, 11470, 11500}, PlotRange -> All]



Speed test with iperf with UDP over IPv4.

x2={11679,11680,11681,11682,11682,11682,11681,11681,11671,11666,11681,11652,11682,11682,11682,11682,11683,11682,11682,11682,11680,11682,11682,11672,11663,11678,11668,11683,11682,11682,11682,11682,11682,11682,11682,11682,11666,11682,11682,11682,11682,11683,11682,11682,11682,11682,11670,11682,11682,11682};

n=50;

Calculating the estimated expected value for UDP over IPv4.

$$\mu 2 = \frac{1}{n} * \sum_{i=1}^{n} \text{x2}[[i]] \,//N$$

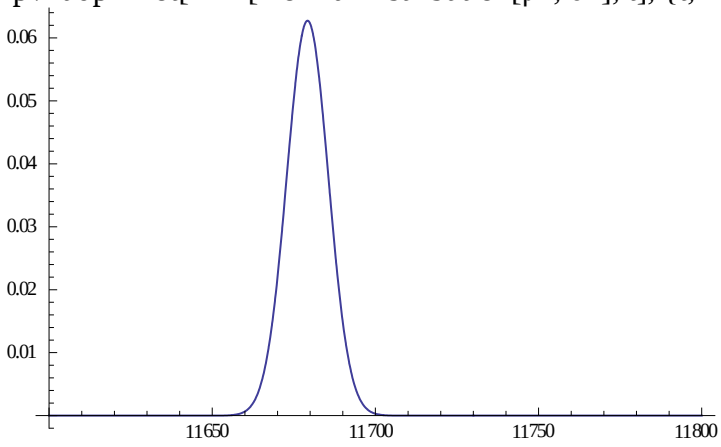
11679.2

Calculating the estimated standard deviation for UDP over IPv4.

$$\sigma 2 = \sqrt{\frac{1}{n-1} * (\sum_{i=1}^{n} (\text{x2}[[i]] - \mu 2)^2)}$$

6.35995

ipv4udp=Plot[PDF[NormalDistribution[μ2, σ2], t], {t, 11600,11800}, PlotRange -> All]



Speed test with iperf with TCP over IPv6.

x3={11329,11334,11334,11334,11334,11334,11334,11334,11334,11334,11334,11334,11334,
11334,11334,11334,11334,11334,11334,11334,11334,11334,11334,11334,11334,1133
4,11334,11334,11334,11334,11334,11334,11334,11334,11334,11334,11334,11334,11
334,11334,11334,11334,11334,11334,11334,11334,11334,11334};

n=50;

Calculating the estimated expected value for TCP over IPv6.

$$\mu 3 = \frac{1}{n} * \sum_{i=1}^{n} \text{x3}[[i]] \,//N$$

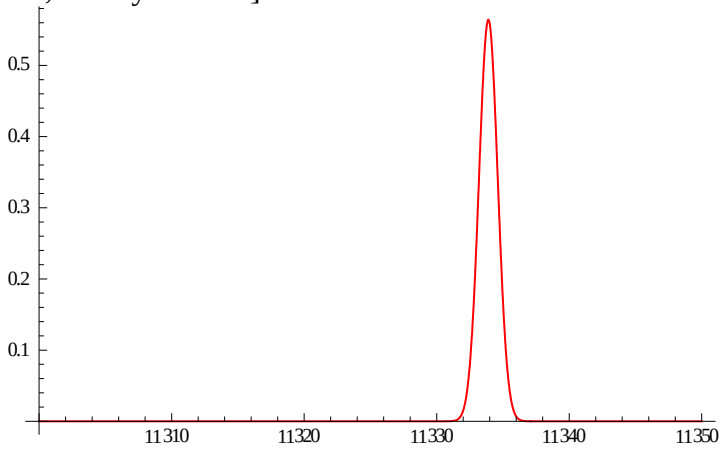
11333.9

Calculating the estimated standard deviation for TCP over IPv6.

$$\sigma 3 = \sqrt{\frac{1}{n-1} * (\sum_{i=1}^{n} (\text{x3}[[i]] - \mu 3)^2)}$$

0.707107

ipv6tcp=Plot[PDF[NormalDistribution[μ3, σ3], t], {t, 11300,11350}, PlotRange ->
All,PlotStyle->Red]



Speed test with iperf with UDP over IPv6.

x4={10870,10875,10875,10875,10864,10875,10875,10875,10875,10875,10875,10875,10875,
10875,10872,10875,10875,10875,10875,10875,10875,10875,10875,10864,10875,10875,1087
5,10875,10875,10875,10875,10871,10875,10843,10875,10875,10875,10875,10875,10875,10
875,10875,10875,10875,10868,10875,10873,10875,10875,10875,10875};

n=50;

Calculating the estimated expected value for UDP over IPv6.

$$\mu 4 = \frac{1}{n} * \sum_{i=1}^{n} \text{x4}[[i]] \,//N$$

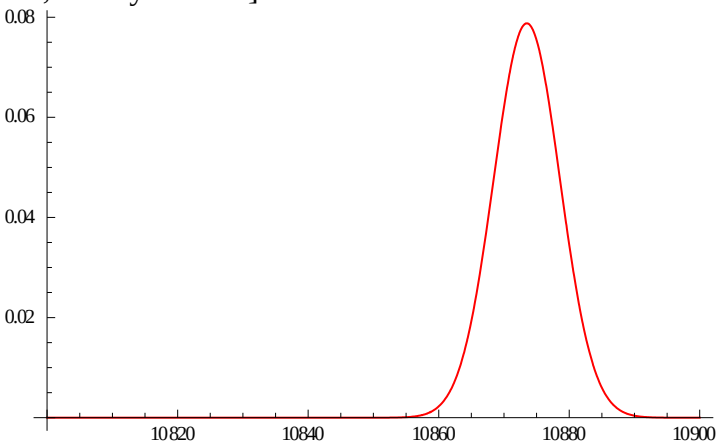
10873.5

Calculating the estimated standard deviation for UDP over IPv6.

$$\sigma 4 = \sqrt{\frac{1}{n-1} * (\sum_{i=1}^{n} (\text{x4}[[i]] - \mu 4)^2)}$$

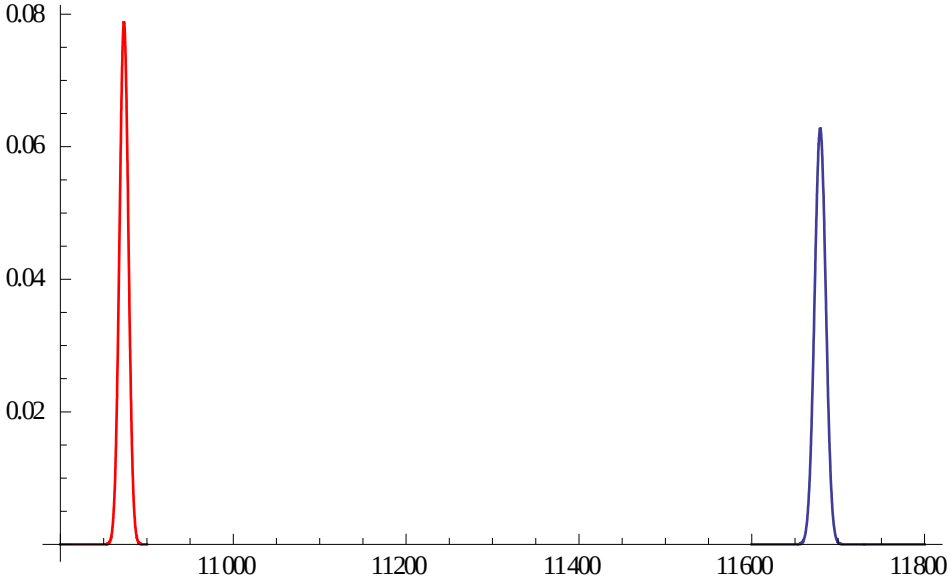
5.06388

ipv6udp=Plot[PDF[NormalDistribution[μ4, σ4], t], {t, 10800,10900}, PlotRange ->
All,PlotStyle->Red]



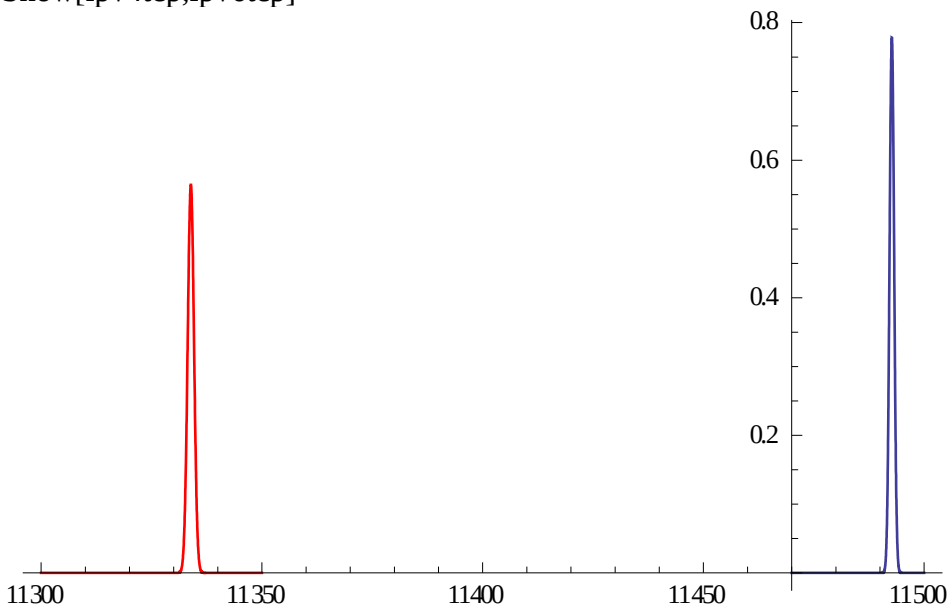
The IPv6 UDP and the IPv4 UDP distribution plotted.

Show[ipv6udp,ipv4udp]



The IPv6 TCP and the IPv4 TCP distribution plotted.

Show[ipv4tcp,ipv6tcp]



Appendix G – Ping test

	www.freebsd.org				www.glesys.se				www.isc.org				www.teddy.ch				www.xbox.com				6to4 relay (192.88.99.1)	
	PING (ms)	PING6 (ms)	Traceroute	Traceroute6	PING (ms)	PING6 (ms)	Traceroute	Traceroute6	PING (ms)	PING6 (ms)	Traceroute	Traceroute6	PING (ms)	PING6 (ms)	Traceroute	Traceroute6	PING (ms)	PING6 (ms)	Traceroute	Traceroute6	PING (ms)	Traceroute
"2012-05-09 12:02:00"	178.970	187.288	13	14	24.670	32.528	11	12	184.728	188.906	13	12	37.839	57.016	17	13	8.592	56.536	8	5	19.753	5
"2012-05-09 12:21:00"	175.242	185.807	13	14	25.254	34.907	11	12	183.568	189.683	13	12	37.828	56.822	17	13	9.731	11.177	7	5	9.406	5
"2012-05-09 13:00:00"	175.150	188.251	13	14	21.125	33.429	11	12	184.763	189.679	13	12	39.708	68.798	17	13	12.366	10.007	8	5	22.788	5
"2012-05-09 14:00:00"	176.942	185.095	14	14	20.280	33.112	11	12	183.028	182.683	13	12	37.141	68.800	17	13	9.256	37.008	7	5	9.835	5
"2012-05-09 15:00:00"	178.645	187.931	14	14	25.636	34.343	11	12	184.372	186.896	13	12	39.016	70.825	17	13	9.530	9.816	7	5	10.320	5
"2012-05-10 11:12:00"	179.375	185.334	13	14	23.538	35.144	11	12	184.599	185.276	13	12	38.612	69.124	17	13	11.392	10.885	8	5	9.212	5
"2012-05-10 12:00:00"	180.592	187.307	13	14	23.398	35.533	11	12	184.010	185.313	13	12	38.768	70.219	17	13	9.856	11.772	7	5	10.559	5
"2012-05-10 13:00:00"	178.494	185.953	13	14	22.640	34.938	11	12	190.308	187.518	13	12	38.668	72.363	17	13	10.941	11.427	7	5	12.455	5
"2012-05-10 14:00:00"	180.467	189.027	13	14	24.263	36.646	11	12	189.451	186.946	13	12	37.710	69.743	17	13	12.015	93.735	7	5	12.269	9
"2012-05-10 15:00:00"	184.444	189.174	13	14	25.116	37.970	11	12	189.275	190.401	13	12	41.121	70608	17	13	12.623	106.492	8	5	12.980	5
"2012-05-11 11:02:00"	175.326	184.360	13	14	22.671	34.068	11	12	182.665	184.420	13	12	39.590	70.860	17	13	8.839	11.486	8	5	16.171	5
"2012-05-11 12:00:00"	174.922	186.898	13	14	22.614	34.982	11	12	184.293	185.479	13	12	39.034	69.758	17	13	10.701	37.351	7	5	8.931	5
"2012-05-11 13:00:00"	175.699	192.794	13	14	22.958	34.426	11	12	186.119	185.769	13	12	40.012	69.019	17	13	9.130	97.801	7	5	30.887	5
"2012-05-11 14:00:00"	177.688	188.278	13	14	25.626	35.452	11	12	185.531	192.654	13	12	41.421	75.447	17	13	12.943	13.250	8	5	10.719	5
"2012-05-11 15:00:00"	175.628	186.462	13	14	24.418	34.736	11	12	187.168	186.386	13	12	38.189	57.984	17	13	12.399	67.070	8	5	29.368	5
"2012-05-14 11:20:00"	180.332	193.056	13	14	22.375	37.033	11	12	184.143	189.488	13	12	42.961	71.843	17	13	10.450	13.152	8	5	9.027	5
"2012-05-14 12:46:00"	181.854	192.121	13	14	22.851	38.303	11	12	185.249	190.406	13	12	39.458	68.912	17	13	9.908	41.198	8	5	12.328	5
"2012-05-14 13:00:00"	176.183	190.955	11	14	22.261	37.623	11	12	184.684	190.079	13	12	37.854	69.485	17	13	9.646	10.891	7	5	29.132	5
"2012-05-14 14:00:00"	175.899	193.404	12	14	25.796	37.283	11	12	184.363	191.000	13	12	39.288	70.507	17	13	9.678	10.121	8	5	9.041	5
"2012-05-14 15:00:00"	176.311	192.980	13	14	24.276	38.781	11	12	186.350	191.157	13	12	37.644	70.969	17	13	12.695	9.774	8	5	11.069	5
"2012-05-15 12:13:00"	175.751	191.037	13	14	22.018	35.609	11	12	183.614	189.644	13	12	37.288	69.188	17	13	10.707	13.546	7	5	9.127	5
"2012-05-15 13:00:00"	176.088	191.106	13	14	23.322	38.217	11	12	189.194	189.216	13	12	37.680	70.010	17	13	9.785	10.417	7	5	9.768	5
"2012-05-15 14:00:00"	175.815	192.791	13	14	23.817	38.461	11	12	190.959	189.272	13	12	39.148	71.530	17	13	9.093	10.483	8	5	8.693	5
"2012-05-15 15:00:00"	183.132	193.443	13	14	29.175	39.297	11	12	189.660	190.724	13	12	39.934	69.416	17	13	8.805	37.165	8	5	10.247	5
"2012-05-16 10:46:00"	177.144	191.148	13	14	22.129	36.025	11	12	187.963	190.066	13	12	38.151	70.637	17	13	9.205	10.626	7	5	11.782	5
"2012-05-16 11:00:00"	177.392	190.351	13	14	22.027	41.045	11	12	189.464	190.155	13	12	37.925	69.955	17	13	9.574	43.419	8	5	9.145	5
"2012-05-16 12:00:00"	178.446	191.440	13	14	24.186	36.452	11	12	189.353	190.046	13	12	40.244	70.848	17	13	12.776	13.127	7	5	10.212	5
"2012-05-16 13:00:00"	177.150	192.887	13	14	22.839	38.857	11	12	190.131	189.661	13	12	37.485	68.746	17	13	9.337	11.574	7	5	10.577	5
"2012-05-16 14:00:00"	177.704	190.464	13	14	23.232	36.037	11	12	188.743	190.893	13	12	38.746	70.118	17	13	8.911	9.970	8	5	10.935	5
"2012-05-16 15:00:00"	181.149	190.215	13	14	26.401	40.098	11	12	189.670	189.913	13	12	39.858	69.851	17	13	10.032	37.691	7	5	9.454	5

Appendix H – Web server performance test

Expressed in seconds

IPv4	Apache 2.2.20	
GET	200 OK	Difference
1.799791	2.054842	0.255051
0.015326	0.248507	0.233181
1.086806	1.308566	0.221760
0.015998	0.241060	0.225062
0.018645	0.264008	0.245363
0.577569	0.797521	0.219952
0.990334	1.239562	0.249228
0.015261	0.247970	0.232709
0.017561	0.247601	0.230040
0.014542	0.234200	0.219658
0.674533	0.896927	0.222394
0.014076	0.233567	0.219491
0.016230	0.238724	0.222494
0.015439	0.239381	0.223942
0.385996	0.630266	0.244270
0.014049	0.308813	0.294764
0.015067	0.237829	0.222762
0.017423	0.244907	0.227484
0.017929	0.237925	0.219996
0.016159	0.251865	0.235706
	Mean:	0.2332654

IPv6		
GET	200 OK	Difference
4.177618	4.405910	0.228292
0.247153	0.470831	0.223678
0.228884	0.461123	0.232239
0.597767	0.820580	0.222813
2.557280	2.777084	0.219804
0.015666	0.239809	0.224143
0.015339	0.249713	0.234374
2.700591	2.930594	0.230003
1.544999	1.812002	0.267003
0.881586	1.113496	0.231910
0.015585	0.239259	0.223674
0.372334	0.625076	0.252742
0.014523	0.282975	0.268452
0.014040	0.239201	0.225161
0.016240	0.260927	0.244687
0.189437	0.443555	0.254118
0.013871	0.285535	0.271664
0.014177	0.239248	0.225071
0.036453	0.278160	0.241707
2.385795	2.609579	0.223784
	Mean:	0.237266

IPv4	IIS 7.5	
GET	200 OK	Difference
0.024256	0.316205	0.291949
0.033858	0.352257	0.318399
0.017170	0.243508	0.226338
0.019225	0.242280	0.223055
0.028025	0.239899	0.211874
0.016376	0.222884	0.206508
0.791343	1.004756	0.213413
0.027185	0.251618	0.224433
0.018059	0.231906	0.213847
0.016970	0.235927	0.218957
0.016076	0.259675	0.243599
0.017395	0.241786	0.224391
0.016670	0.234598	0.217928
0.016407	0.235114	0.218707
0.016269	0.235709	0.219440
0.017487	0.244829	0.227342
0.017345	0.271296	0.253951
0.558498	0.786745	0.228247
0.873272	1.118925	0.245653
0.654045	0.874232	0.220187
	Mean:	0.232411

IPv6		
GET	200 OK	Difference
4.177618	4.405910	0.228292
0.247153	0.470831	0.223678
0.228884	0.461123	0.232239
0.597767	0.820580	0.222813
2.555728	2.777084	0.221356
0.015666	0.239809	0.224143
0.015339	0.249713	0.234374
2.700591	2.930594	0.230003
1.544999	1.812002	0.267003
0.881586	1.113496	0.231910
0.015585	0.239259	0.223674
0.372334	0.625076	0.252742
0.014523	0.282975	0.268452
0.014040	0.239201	0.225161
0.016240	0.260927	0.244687
0.189437	0.443555	0.254118
0.013871	0.285535	0.271664
0.014177	0.239248	0.225071
0.036453	0.278160	0.241707
2.385795	2.609579	0.223784
	Mean:	0.237344

Appendix I – Mathematica calculations of web server test

GET over IPv4 time differences. Apache 2.2.20

x1={0.255051,0.233181,0.22176,0.225062,0.245363,0.219952,0.249228,0.232709,0.23004,0.219658,0.222394,0.219491,0.222494,0.223942,0.24427,0.294764,0.222762,0.227484,0.219996,0.235706};

n=20;

Calculating estimated expected value for IPv4 on Apache.

$$\mu 1 = \frac{1}{n} * \sum_{i=1}^{n} \text{x1}[[i]]$$

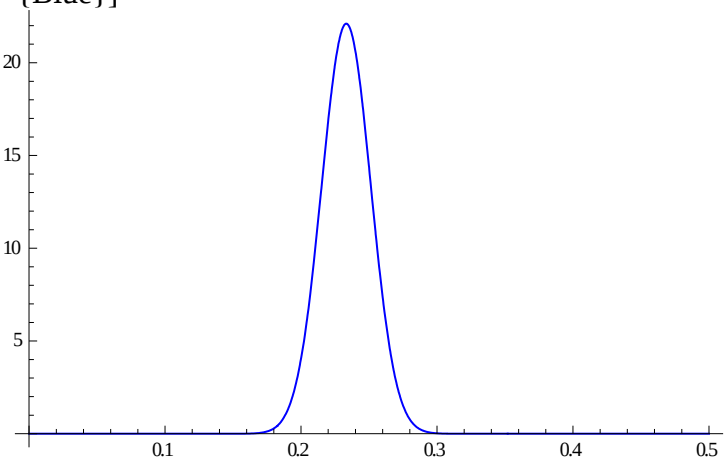
0.233265

Calculating estimated standard deviation for IPv4 on Apache.

$$\sigma 1 = \sqrt{\frac{1}{n-1} * (\sum_{i=1}^{n} (\text{x1}[[i]] - \mu 1)^2)}$$

0.0180473

ipv4apache=Plot[PDF[NormalDistribution[μ1, σ1], t],{t,0,.5},PlotRange -> All,PlotStyle->{Blue}]



GET over IPv6 time differences. Apache 2.2.20

x2={0.228292,0.223678,0.232239,0.222813,0.219804,0.224143,0.234374,0.230003,0.267003
,0.23191,0.223674,0.252742,0.268452,0.225161,0.244687,0.254118,0.271664,0.225071,0.24
1707,0.223784};

n=20;

Calculating estimated expected value for IPv6 on Apache.

$$\mu 2 = \frac{1}{n} * \sum_{i=1}^{n} \text{x2}[[i]]$$

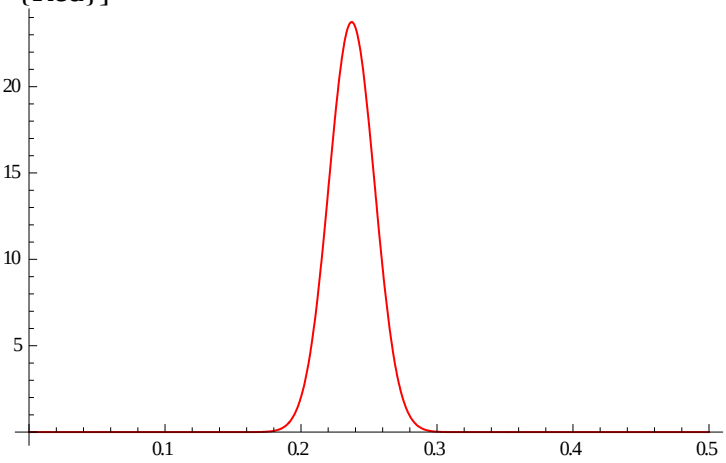
0.237266

Calculating estimated standard deviation for IPv6 on Apache.

$$\sigma 2 = \sqrt{\frac{1}{n-1} * (\sum_{i=1}^{n} (\text{x2}[[i]] - \mu 2)^2)}$$

0.016806

ipv6apache=Plot[PDF[NormalDistribution[μ2, σ2], t],{t,0,.5},PlotRange -> All,PlotStyle-
>{Red}]



Get a picture over IPv4 time differences. IIS

x3={0.291949,0.318399,0.226338,0.223055,0.211874,0.206508,0.213413,0.224433,0.213847
,0.218957,0.243599,0.224391,0.217928,0.218707,0.21944,0.227342,0.253951,0.228247,0.24
5653,0.220187};

n=20;

Calculating estimated expected value for IPv4 on IIS.

$$\mu 3 = \frac{1}{n} * \sum_{i=1}^{n} \text{x3}[[i]]$$

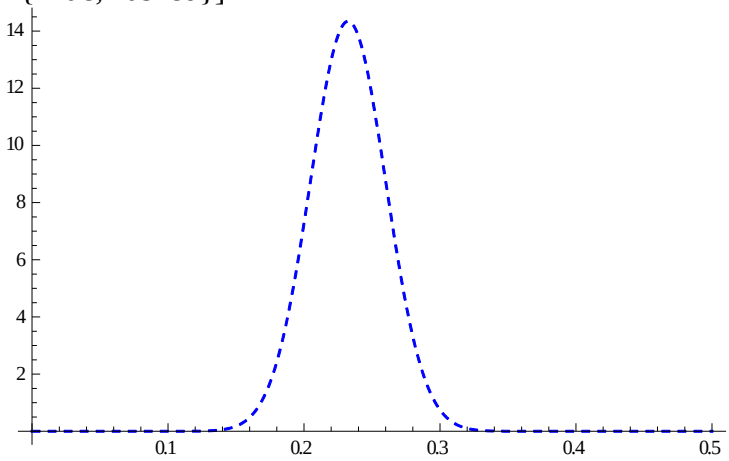
0.232411

Calculating estimated standard deviation for IPv4 on IIS.

$$\sigma 3 = \sqrt{\frac{1}{n-1} * (\sum_{i=1}^{n} (\text{x3}[[i]] - \mu 3)^2)}$$

0.0278087

ipv4iis=Plot[PDF[NormalDistribution[μ3, σ3], t],{t,0,.5},PlotRange -> All,PlotStyle-
>{Blue,Dashed}]



Get a picture over IPv6 time differences. IIS

x4={0.228292,0.223678,0.232239,0.222813,0.221356,0.224143,0.234374,0.230003,0.267003
,0.23191,0.223674,0.252742,0.268452,0.225161,0.244687,0.254118,0.271664,0.225071,0.24
1707,0.223784};

n=20;

Calculating estimated expected value for IPv4 on IIS.

$$\mu 4 = \frac{1}{n} * \sum_{i=1}^{n} \text{x4}[[i]]$$

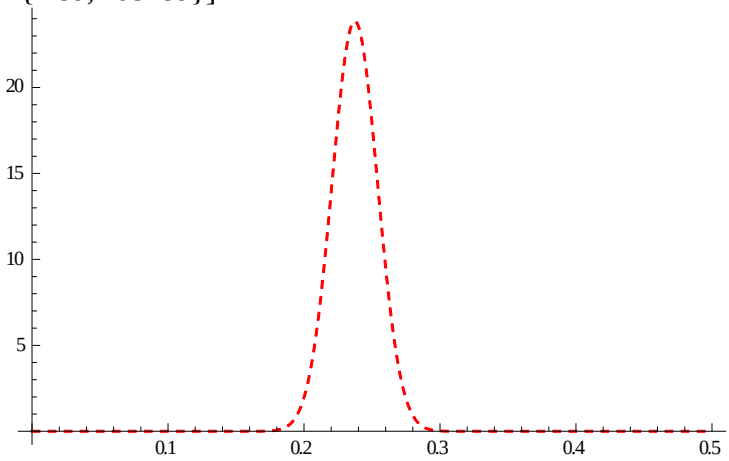
0.237344

Calculating estimated standard deviation for IPv4 on IIS.

$$\sigma 4 = \sqrt{\frac{1}{n-1} * (\sum_{i=1}^{n} (\text{x4}[[i]] - \mu 4)^2)}$$

0.0167245

ipv6iis=Plot[PDF[NormalDistribution[μ4, σ4], t],{t,0,.5},PlotRange -> All,PlotStyle-
>{Red,Dashed}]



Trace to xbox.com:

```
1  gateway.ex.jobb (2002:53fb:3271:1::1)  0.587 ms  0.335 ms  0.388 ms  0.304 ms  0.284 ms
 2  * * * * *
 3  * * * * *
 4  * * * * *
 5  10gigabitethernet1-4.core1.lon1.he.net (2001:470:0:3f::1)  43.903 ms  41.225 ms  48.689 ms  50.121 ms *
 6  10gigabitethernet7-4.core1.nyc4.he.net (2001:470:0:128::1)  109.118 ms  113.199 ms  108.019 ms  116.575 ms  109.185 ms
 7  10gigabitethernet2-3.core1.ash1.he.net (2001:470:0:36::1)  121.7 ms  127.171 ms  121.363 ms  124.936 ms  129.945 ms
 8  dcp-brdr-04-xe-0-2-0.qwest.com (2001:504:0:2::209:1)  119.125 ms  120.511 ms  130.43 ms  120.142 ms  129.022 ms
 9  2001:428::205:171:203:158 (2001:428::205:171:203:158)  118.687 ms  215.261 ms  119.297 ms  119.73 ms  120.996 ms
10  2001:428:2402:107::4179:d123 (2001:428:2402:107::4179:d123)  118.615 ms  119.72 ms  120.149 ms  121.556 ms  120.059 ms
```

Trace to freebsd.org:

```
1  gateway.ex.jobb (2002:53fb:3271:1::1)  0.269 ms  0.289 ms  0.396 ms  0.399 ms  0.4 ms
 2  * * * * *
 3  * * * * *
 4  * * * * *
 5  * 10gigabitethernet1-4.core1.lon1.he.net (2001:470:0:3f::1)  54.117 ms *  48.145 ms *
 6  10gigabitethernet7-4.core1.nyc4.he.net (2001:470:0:128::1)  121.7 ms  109.54 ms  114.358 ms  109.668 ms  115.316 ms
 7  10gigabitethernet2-3.core1.ash1.he.net (2001:470:0:36::1)  140.548 ms  124.455 ms  124.679 ms  126.356 ms  124.431 ms
 8  10gigabitethernet1-2.core1.atl1.he.net (2001:470:0:1b5::2)  141.24 ms  132.713 ms  141.935 ms  133.146 ms  140.602 ms
 9  isc.gige-g2-1.core1.atl1.he.net (2001:470:0:ce::2)  187.227 ms  187.899 ms  187.566 ms  188.804 ms  188.99 ms
10  iana.r1.atl1.isc.org (2001:500:61:6::1)  188.596 ms  198.803 ms  187.909 ms  192.637 ms  188.393 ms
11  int-0-5-0-1.r1.pao1.isc.org (2001:4f8:0:1::49:1)  193.359 ms  198.506 ms  192.712 ms  191.555 ms  198.553 ms
12  2001:4f8:1:13::84 (2001:4f8:1:13::84)  189.147 ms  189.384 ms  197.948 ms  190.261 ms  189.795 ms
13  ipv6gw-isc.freebsd.org (2001:4f8:0:1::3e:2)  190.707 ms  192.166 ms  190.753 ms  191.872 ms  189.978 ms
14  red.freebsd.org (2001:4f8:fff6::22)  190.207 ms  190.592 ms  191.035 ms  191.407 ms  191.406 m
```

Appendix K – Data collected from DHCP leases tests

Many entries

Discover	Offer	Difference
14.760184	15.762160	1.001976
0.000000	1.002070	1.002070
0.000000	1.002524	1.002524
0.000000	1.002069	1.002069
0.000000	1.002080	1.002080
0.000000	1.002041	1.002041
0.000000	1.002037	1.002037
0.000000	1.002098	1.002098
0.000000	1.002063	1.002063
0.000000	1.002018	1.002018

Few entries

Discover	Offer	Difference
2.08947	3.09163	1.00216
5.96602	6.96797	1.00195
2.18639	3.18837	1.00199
0.00000	1.00174	1.00174
3.93013	4.93191	1.00178
2.12814	3.13013	1.00199
13.78850	14.79050	1.00201
8.57196	9.57313	1.00117
9.02820	10.03020	1.00200
9.76295	10.76440	1.00141

Appendix L – Mathematica calculations of DHCP leases tests

DHCP leases test.

few={1.002161,1.001947,1.001987,1.001738,1.001781,1.001986,1.002012,1.001171,1.001998,1.001405};
n =10;

Calculating the estimated expected value for few entries.

$$\mu\text{few} = \frac{1}{n} * \sum_{i=1}^{n} \text{few}[[i]]$$

1.00182

Calculating the estimated standard deviation for few entries.

$$\sigma\text{few} = \sqrt{\frac{1}{n-1} * (\sum_{i=1}^{n} (\text{few}[[i]] - \mu\text{few})^2)}$$

0.000308771

many={1.001976,1.00207,1.002524,1.002069,1.00208,1.002041,1.002037,1.002098,1.002063,1.002018};

Calculating the estimated expected value for many entries.

$$\mu\text{many} = \frac{1}{n} * \sum_{i=1}^{n} \text{many}[[i]]$$

1.0021

Calculating the estimated standard deviation for many entries.

$$\sigma\text{many} = \sqrt{\frac{1}{n-1} * (\sum_{i=1}^{n} (\text{many}[[i]] - \mu\text{many})^2)}$$

0.000153804

Appendix M – Calculations of the theoretical speed limits for IPv6

MTU: 1500

Ethernet frame plus MAC preamble, plus Start Frame Delimiter, plus CRC, plus inter-frame gap are 1538 bytes.

Ethernet frame = 1514 bytes

MAC preamble = 7 bytes

Start Frame Delimiter = 1 byte

CRC = 4 bytes

inter-frame gap = 12 bytes

1514 + 7 + 1 + 4 + 12 = 1538

(100Mb/second) / ((8 bits) * (1538 bytes/frame)) = 8,127 frames/second

Max data per frame over IPv6 with TCP and time stamps:
1538 - 40 - 32 - 38 = 1428

∴

8127 * 1428 = 11,605,356 bytes/second

Max data per frame over IPv6 with TCP without time stamps:
1538 - 40 - 20 - 38 = 1440

∴

8127 * 1440 = 11,702,880 bytes/second

Max data per frame over IPv6 with UDP:
1538 - 40 - 8 - 38 = 1452

∴

8127 * 1452 = 11,800,404 bytes/second

TRITA-ICT-EX-2012:191

www.kth.se