VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Bijay Banstola

IPv6 IMPLEMENTATION, FIREWALL AND REDUDANCY

Faculty of Information Technology
2012

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Degree Programme in Information Technology

ABSTRACT

Author	Banstola Bijay
Title	IPv6 Implementation, Firewall and Redundancy
Year	2012
Language	English
Pages	100
Name of Supervisor	Gao Chao

The project was inspired by the fact that the last block of IPv4 address has been already assigned and the need of more address for further development in the field of Internet applications. Thus new Internet protocol was developed, IP next generation (IPng), now known as IPv6, Internet Protocol Version 6 which was proposed on early 1990s as a successor of IPv4. In the process of transition it is very important that the functions running over IPv4 must not be affected so these two protocols must run together and independently. So at the time being Dual Stack mode is preferred i.e. host running both protocols.

The goal of the project was to implement secure redundant IPv6 network in Vaasan Ammattikorkeakoulu (VAMK) which was achieved by configuring a pair of Dell PCs to act as gateway router together with firewall, DNS and DHCPv6 functions and are redundant to each other. FreeBSD 9.0 release was installed on those PCs. The firewall policy was to pass secure connections only which was achieved by using FreeBSD’s default firewall, PF firewall. Redundancy was achieved re-configuring a pair of routers with CARP and *pfsync* functions included. The system was tested on VAMK’s Information Technology department and the goal was achieved successfully.

This project covers the implementation of IPv6, Firewall and Redundancy. A complete new set of Firewall rules were implemented for IPv6 since firewall policy for IPv4 will not work for IPv6. After the completion of the project the campus have running IPv6 network and all the host and servers are dual stacked.

Keywords: Dual Stack, Autoconfiguration, Tunnel, Redundancy

CONTENTS

ABSTRACT
1. INTRODUCTION 10
  1.1 Project Background 10
  1.2 Purpose and Scope 11
  1.3 Project Flow 11
2. TECHNOLOGY BACKGROUND 12
  2.1. Addressing 12
    2.1.1 Link-Local addresses 13
    2.1.2 Interface Identifier (Modified EUI-64) 14
  2.2 Internet Control Message Protocol for IPv6 (ICMPv6) 17
  2.3 Neighbor Discovery (ND) 19
    2.3.1 Router Advertisement 19
    Essential Services from Neighbor Discovery Protocol 21
    2.3.2 Address Resolution 21
    2.3.3 Neighbor Unreachability Detection 22
    2.3.4 Duplicate Address Detection 22
    2.3.5 Path MTU Discovery and Fragmentation 23
  2.4 Autoconfiguration 23
    2.4.1 Stateful Address Autoconfiguration 24
    2.4.2 Stateless Address Autoconfiguration 24
    Address Lifecycle 26
    2.4.3 Tentative Address 26
    2.4.4 Preferred Address 26
    2.4.5 Deprecated Address 27
    Some More about Addresses 28
    2.4.6 Temporary Address 28
    2.4.7 Address Requirements 28
    2.4.8 Cases of Address Autoconfiguration 29
  2.5 Virtual Local Area Network (VLAN) 30
    2.5.1 Static VLANs 30

2.5.2 Dynamic VLANs 30
2.5.3 Tagged VLANs 30
2.6 Routing IPv6 31
2.7 Dynamic Host Configuration Protocol for IPv6 (DHCPv6) 32
2.7.1 Stateful DHCPv6 35
2.7.2 Stateless DHCPv6 36
2.7.3 Prefix Delegation 37
2.8 Domain Name Server for IPv6 (DNSv6) 38
2.8.1 DNS Message Format 39
2.8.2 Recursive method 41
2.8.3 Iterative method 41
2.8.4 Mixed method 42
2.8.5 Representing IPv6 information in DNS 43
2.9 Firewall 44
2.10 Redundancy 45
2.10.1 The CARP Protocol 45
2.10.2 The PFSYNC Protocol 46
3. IMPLEMENATATION OF PROJECT 49
3.1 Network Structure 49
3.2 Router Configuration 51
3.2.1 IPv6 Packets Forwarding 52
3.2.2 Router Advertisement Message Configuration 54
3.3 DHCPv6 Configuration 55
3.4 DNSv6 Configuration 56
3.5 PF Firewall Configuration 58
3.5.1 Macros 59
3.5.2 Options 59
3.5.3 Filter Rules 60
3.6 Linux Host Configuration 62
3.7 Redundancy 62
3.7.1 Rebuilding the kernel 63
3.7.2 CARP Configurations 64

3.7.3 PFSYNC Configuration....66
3.7.4 PF Firewall Configuration for Redundancy....67
4. TEST, RESULT AND ANALYSIS....68
4.1 Router Test....68
4.1.1 IPv6 Internet Connectivity and Router Advertisement Test....68
4.1.2 Address Autoconfiguration Test....70
4.1.3 Neighbor Unreachability Detection Test....71
4.1.4 Routing Tables....72
4.2 DHCPv6 Test....74
4.3 DNSv6 Test....75
4.4 PF Firewall Test....77
4.5 Linux Host Test....80
4.6 Redundancy Test....80
4.6.1 CARP Test....81
4.6.1.1 Effect of CARP on Clients....83
4.6.1.2 Linux Router Advertisement Daemon (radvd)....85
4.6.2 PFSYNC Test....86
5. CONCLUSIONS....88
REFERENCES....89
APPENDICES....91
Appendix A: rc.conf....91
Appendix B: RDNSS Configuration on rtadvd.conf....93
Appendix C: pf.conf....94
Appendix D: Redundancy and Firewall Synchronization Configuration....98
Appendix E: radvd.conf....100

ACKNOWLEDGMENT

Before being thankful to almighty god, I would like to remember my parents without whom my study wouldn’t have been successful. I do appreciate their enormous love, care and support.

I would like to express my deep gratitude to supervisor, Dr. Gao Chao for his feedback, comments, support and inspiration throughout my studies.

I am heartily grateful to Hannu Teulahti, VAMK Linux Administrator for his help, comments and willingness to share his vast knowledge.

I would like to thank entire VAMK unit for their wonderful support and learning environment.

Finally, I am thankful to all my friends for their infinite help and support throughout my studies.

Vaasa, 29 July 2012
Banstola Bijay

ABBREVIATIONS

ALTQ	Alternate Queuing
ARP	Address Resolution Protocol
BGP	Border Gateway Protocol
BIND	Berkeley Internet Name Daemon
BSD	Berkeley Software Distribution
CARP	Common Address Redundancy Protocol
CIDR	Classless Inter-Domain Routing
CRC	Cyclical Redundancy Check
DAD	Duplicate Address Detection
DHCPv6	Dynamic Host Configuration Protocol for IPv6
DNSv6	Domain Name Server for IPv6
DSTM	Dual Stack Transition Mechanism
DUID	DHCP Unique Identifier
EUI	Extended Unique Identifier
FUNET	Finnish University and Research Network
HMAC	Hash-Based Message Authentication Code
IA	Identity Association
IAID	Identity Association Identification
ICMPv6	Internet Control Message Protocol for IPv6

IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPng	Internet Protocol next generation
IPsec	Internet Protocol Security
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISATAP	Intra-site Automatic Tunnel Addressing Protocol
IS-IS	Intermediate System to Intermediate System
ISP	Internet Service Provider
LAN	Local Area Network
LIR	Local Internet Registry
MAC	Media Access Control
MTU	Maximum Transmission Unit
NAPT-PT	Network Address Port Translation and Protocol Translation
NAT	Network Address Translation
NAT-PT	Network Address Translation and Protocol Translation
ND	Neighbor Discovery
NDP	Neighbor Discovery Protocol
NUD	Neighbor Unreachability Detection

Opti-DAD	Optimistic Duplicate Address Detection
OSI	Open System Interconnection
OSPFv3	Open Short Path First version 3 for IPv6
PF	Packet Filter Firewall
PFSYNC	Packet Filter Synchronization
PMTUD	Path Maximum Transmission Unit Discovery
QoS	Quality of Service
RDNSS	Recursive DNS Server
RFC	Request for Comments
RIPng	Routing Information Protocol for next generation
RIR	Regional Internet Registry
SHA-1	Secure Hash Algorithm
TCP	Transmission Control Protocol
TTL	Time to Live
UDP	User Datagram Protocol
VAMK	Vaasan Ammattikorkeakoulu
VLAN	Virtual LAN
VRRP	Virtual Router Redundancy Protocol

1. INTRODUCTION

In the early 1990s, the IETF discovered that the available IPv4 address space was running out rapidly which prompted IETF to start work on ”IP next generation” (IPng), which result the creation of IPv6 standard. At the end of 1995 the first set of RFC on IPv6 was released as RFC 1883: Internet Protocol Version 6 (IPv6) Specification. [1] The new protocol was developed by Steven Deeing and Robert Hinden.

During the time of development it was predicted that IPv6 will replace IPv4 in near future but the wide deployment of NAT solutions weakened the main IPv6 driving force. IPv6 have many advantages over IPv4, security advantage is one of them. In IPv6 an Ethernet usually gets 64 bits to number hosts that results an attacker has much harder time scanning a single Ethernet subnet also than scanning the entire IPv4 internet. One major difference in IPv6 network is that routers are no longer required to fragment oversized packets, fragmentation is done by the host itself and the routers can simply signal the source to send the packets that can be transferred through that link depending upon MTU of the link. Secondly, multicast addresses replace broadcast addresses so the host listening for multicast is interrupted rather than the entire network in IPv4. Similarly, ARP mechanism is replaced by Neighbor Discovery (ND).

1.1 Project Background

The project was implemented in laboratory at VAMK also known as Vaasa University of Applied Sciences located at Vaasa, Finland. VAMK has stable running IPv4 network which is divided in three layers, core layer, distribution layer and access layer. Core layer provides optimal transport between two locations, the distribution layer of the network service to connect to the access layer, and the realization of security, communications loading and routing strategy. Access layer provides user access to a switch or hub. VAMK has two campuses, one in Wolffintie providing Technology and Communication studies and another in Raastuvankatu providing Business Economic and Tourism studies. The project was implemented in Wolffintie campus.

1.2 Purpose and Scope

The goal of the project was to add IPv6 network to existing IPv4 network without affecting the running services. VAMK has already stable running IPv4 network. This project only covers the IPv6 implementation in Local Area Network (LAN). Furthermore, this document explains the basic concept about IPv6 and briefly describes the transition technologies. Wireless IPv6 Network and Mobile IPv6 are beyond the scope of this project.

1.3 Project Flow

The project was carried out on FreeBSD 9.0 release. FreeBSD is an advanced operating system for modern server and KAME IPv6 stack was integrated to FreeBSD for IPv6 support since FreeBSD 6.1 releases. FreeBSD is free Unix-like operating system which is based on BSD Unix. This task can be divided into two parts:

- IPv6 Implementation and Firewall which consist the implementation of;
    - Router
    - DHCPv6 Server
    - Recursive DNS and
    - Firewall
- Building Redundancy using CARP and Firewall Synchronization using PFSYNC

This document follows the project flow, firstly IPv6 implementation and Firewall and adding Redundancy.

2. TECHNOLOGY BACKGROUND

In this chapter we will briefly describe about IPv6 and its major features. This chapter will only provide the quick overview about IPv6 and for details refer to the corresponding RFCs.

2.1. Addressing

IPv6 addresses are 128-bits long and are written in 32 hexadecimal digits. Every 4 digits forms a group, resulting 8 groups for readability and groups are separated by colons. For simplicity during writing leading zeros in a group can be omitted but not the trailing zeros and a sequence of all-zero groups can be replaced by pair of colons, condition applied that there can't be more than a pair of colons. IPv6 addresses are case insensitive. Prefix length replaces the subnet mask form IPv4 and are written in CIDR (Classless Inter-Domain Routing) notation similar to IPv4. The examples are shown below:

- *2001:708:230:50:1:ABCD:2020:100* (Global Unicast IPv6 address)
- *FE80::5072:0:0:FADE* (Link-local address)

In textural form an IPv6 address is represented as:

IPv6-address/prefix-length

Where, IPv6 address is regular address and Prefix-length is a decimal value specifying how many of the leftmost contiguous bits of the address the prefix comprises. In IPv6 usually a node gets /64 prefix using autoconfiguration but it can be altered according to the necessarity and structure of the network. A pair of colon or a full IPv6-address is followed by prefix-length, no other conditions applied. An example:

2001:0000:0000:230:0000:0000:7788:5687/64 in prefix notation can be written

2001:0:0:230::/64 indicating that the right part after 64-bits can be changed or also as *2001::230:0:0:7788:5687/64*.

Address are handled depending upon their types. There are three different types of address in IPv6 for which the delivery technique differs:

• Unicast Address	one-on-one delivery
• Multicast Address	one-to-many delivery
• Anycast Address	to the nearest one in the group

The details about addressing can be found on RFC 4291. [2] We will only list the types and briefly describe the Link-Local address and Interface Identifier.

2.1.1 Link-Local addresses

Link local addresses are mandatory address for IPv6 enabled devices. Every IPv6 enabled interface must create a link-local address by combining their MAC address with the prefix *FE80::/64.* A link-local address should never be routed outside the LAN since its use is limited for a single link. Link-local addresses are used for autoconfiguration mechanism, Neighbor Discovery, next-hop calculation in routing protocols and on a link without no routers to create temporary network. Using link-local address one can easily share files between two computer using a cross-over cable.

From Figure 2-1 we can see link-local address are directly connected to MAC address of the link but it is not a rare case that a single interface be connected to two or more different link interfaces. In such cases KAME-derived IPv6 stacks like FreeBSD provides *%* interface convention to represent the interface attached to that link. For example, *fe80::5072:57cb:bfd4:e2ef%bge0.368* represents the link-local address reachable via *bge0.368* interface i.e. the link are directly connected while *fe80::5072:57cb:bfd4:e2ef%bge0.25* also represents the same node now reachable via *bge0.25* interface. It can be also seen in some Linux systems.

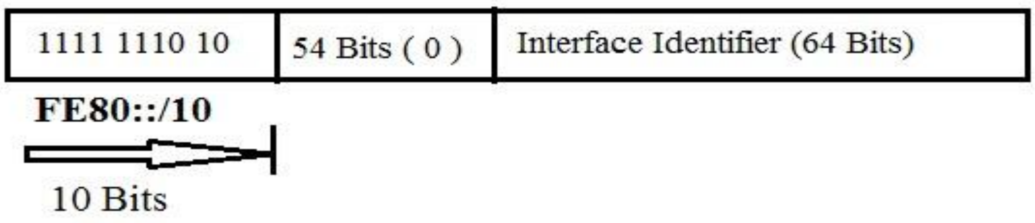


Figure 2-1. Link-Local address format

Link-local addresses are used over a single link. Site-local addresses (now depreciated) were intended to use inside a single site or organization. Site-local address should never be forwarded outside the network. Due to the insufficient definition of “site” in RFCs it is depreciated. RFC 3879 states that the site-local prefix *fec0::/10* is preserved for future use. [3]

2.1.2 Interface Identifier (Modified EUI-64)

Each network interface contains a 48 bits MAC address which is unique to that interface. Modified EUI-64 uses the 48 bits MAC address and change to the 64 bit interface identifier as shown in Figure 2-2.

Figure 2-2 shows the basic steps or rules to change 48 bits MAC address to EUI-64 bits Interface Identifier. The 7th bit of the 48 bits MAC address is inverted and *FFFE* is inserted between 3rd and 4th byte of the MAC address. For instance, an interface with MAC address of *00-23-54-90-01-7E* is changed to be *FE80::223:54FF:FE90:17E*. It can also be assigned manually by the network manager, if needed. In Windows later than Windows XP, link-local addresses are generated randomly but not being based on MAC address.

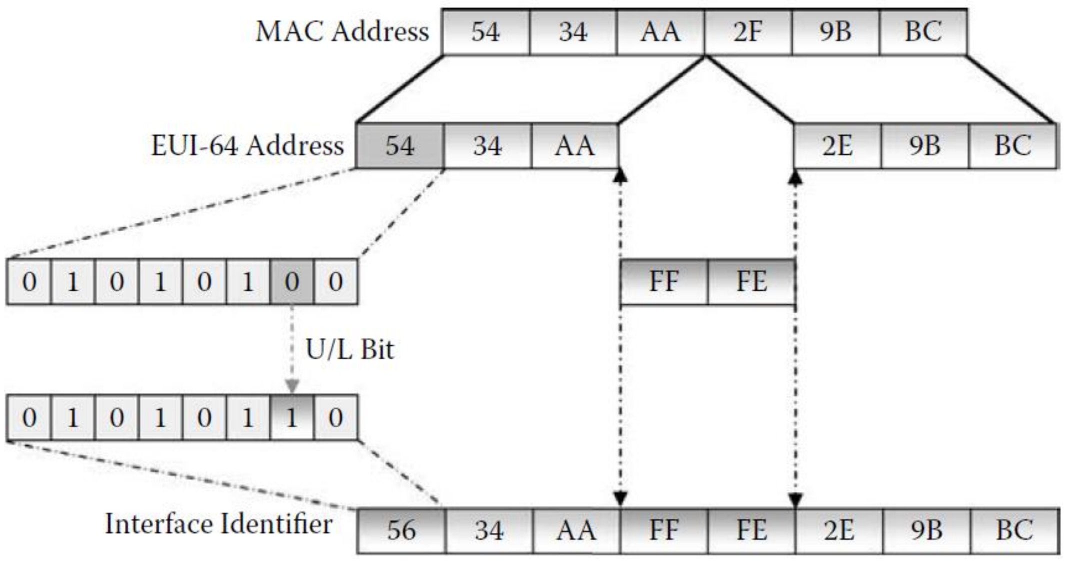


Figure 2-2. Generation of EUI-64 bits Interface Identifier [4]

Table 2-1 shows some well-known multicast addresses and their scope.

Multicast Address	*Type or Scope*	*Description*
FF01::1	Node or Interface Local	All Nodes
FF02::1	Link-Local	All Nodes
FF01::2	Node or Interface Local	All Routers
FF02::2	Link-Local	All Routers
FF05::2	Site-Local	All Routers
FF02::1:2	Link-Local	All DHCP agents

Table 2-1. Some Well-Known Multicast addresses

Table 2-2 shows the different types of unicast address and its structure.

IPv6 Unicast address	*Structure*
Unspecified address	::
Loopback address	::1
Link-Local address	FE80::/10
Global Unicast address	Normal IPv6 addresses (e.g. 2001::/3)
IPv4-Compatiable IPv6 address	::a.b.c.d ,where a.b.c.d is IPv4 address represented using dotted decimal (Deprecated in RFC 4291)
IPv4-Mapped IPv6 address	::F.a.b.c.d ,where Fis hexadeciaml number and a.b.c.d is IPv4 address.
6to4 address	2002:AABB:CCDD:Subnet ID:Interface ID, where :AABB:CCDD: is the hexadecimal representation of a.b.c.d public IPv4 address.
Teredo address	2001::/32
ISATAP address	64-bit IPv6 prefix:0:5EFE:a.b.c.d where a.b.c.d is private IPv4 address.

Table 2-2. Unicast addresses

Apart from all this address type 128-bit IPv6 address are divided into few areas according to their use as shown in Table 2-3.

IPv6 Address	*Description*
::/3	Special addresses types
FE80::/10	Link-local unicast addresses
FEC0::/10	Site-local unicast addresses (Deprecated)
FF00::/8	Multicast addresses
4000::/2 – FE00::/9	Reserved global unicast addresses
2000::/3	Allocated global unicast addresses

Table 2-3. IPv6 address space allocation

2.2 Internet Control Message Protocol for IPv6 (ICMPv6)

We are familiar with ICMP in IPv4 which can also be used as diagnostic tool. Similarly, IPv6 uses ICMPv6, a new protocol but several changes are made for IPv6. ICMPv6 is also used as diagnostic tool but it plays a vital role in Neighbor Discovery Protocol (NDP) and Path MTU Discovery Protocol. ICMPv6 messages can be divided into two types ICMPv6 Error Messages and ICMPv6 Informational message depending on their use elaborated in Table 2-4 and Table 2-5 respectively.

Message No.	Message Type	Code	Description
1	Destination Unreachable	0	No route to destination.
		1	Administratively prohibited destination
		2	Not assigned
		3	Unreachable Address
		4	Unreachable Port
2	Packet Too Big	N/A	No code field is defined, if set to 0 by the sender then ignored by the receiver
3	Time Exceeded	0	Hop Limit dropped to zero in transit
		1	Fragment reassembly time exceeded
4	Parameter Problem	0	Error in Header Field
		1	Unrecognized next header type encounters
		2	Unrecognized IPv6 option encounters

Table 2-4. ICMPv6 Error Messages

Message Type	*Type Field*	*Description*
Echo Request Echo Reply	128 129	Both used in Ping Command. Ping6 in FreeBSD and Unix-like Systems.
Multicast Listener Query Multicast Listener Report Multicast Listener Done	130 131 132	All three are used in Multicast Listener Discovery and will be employed in Multicast Group Management Protocol.
Router Solicitation	133	Used by node when it joins the network to search routers in the link. Routers replies with Router Advertisement message.
Router Advertisement	134	Advertisement by the Router either in response to Router Solicitation message or periodically. It contains the prefix information used for address configuration and on-link determination.
Neighbor Solicitation	135	Nodes advertise Neighbor Solicitation message when it joins the link and starts address duplication check. It is also used to determine the link-layer address of the neighbors and their reachability.
Neighbor Advertisement	136	Neighbor Advertisement message are sent in response to Neighbor Solicitation message announcing the link-layer address and its presence in the network.
Redirect	137	Routers send out the Redirect message to inform source hosts a better first hop to the destination.

Table 2-5. Some ICMPv6 Informational Messages

2.3 Neighbor Discovery (ND)

Neighbor Discovery is new protocol implemented in IPv6 which integrates ARP, RARP, router discovery and redirect services from IPv4 and adds news features like parameter discovery, address autoconfiguration, next-hop determination, Neighbor Unreachability Detection (NUD) and Duplicate Address Detection (DAD). It allows different nodes to inform its presence and to learn about the existence of neighbor on the same link. It is a mandatory function in IPv6 and must be implemented in all platforms. So ND is used to determine the routers in the link which can forward packets, to determine the link-layer address of the neighbors on the same link and to check whether the neighbors are reachable or not and to detect the change in link-layer address. All the messages used in NDP are contained in ICPMv6 Packets. Five new types of ICMP messages are available in ND described briefly in Table 2-5 and listed below:

- Router Solicitation (RS)
- Router Advertisement (RA)
- Neighbor Solicitation (NS)
- Neighbor Advertisement (NA)
- Redirect

Here we will have a close look at Router Advertisement message only. For details please refer RFC 4861. [5]

2.3.1 Router Advertisement

Router Advertisement (RA) messages are sent by the routers periodically or in response to Router Solicitation messages. It contains the prefix and the information from which the host configures the unicast address and the process how the address are configured, stateful using DHCPv6 or stateless autoconfiguration. The source address of RA contains the Link-Local address assigned to the sending interface. If the RA message is sent periodically then the destination address will be all-nodes multicast addresses i.e. *FF02::1* and if sent in response to RS messages the interface address of the source generating RS message. Hop Limit is set to 255.

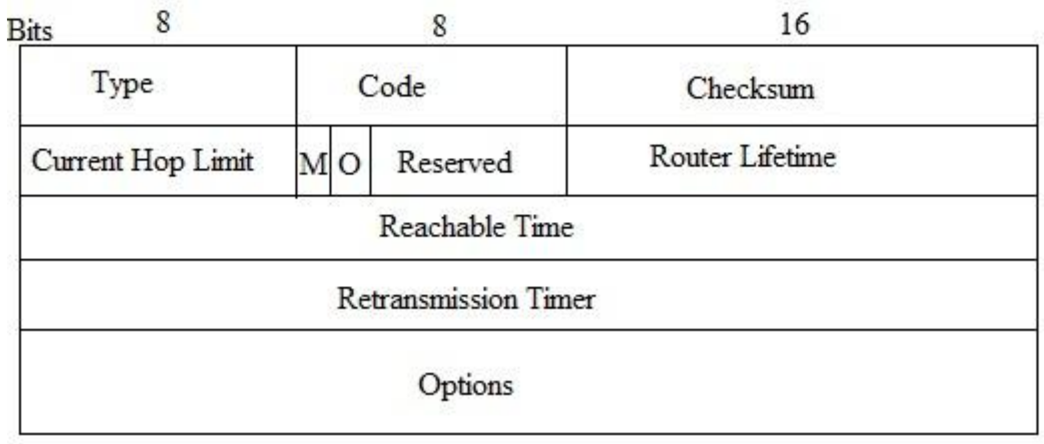


Figure 2-3. ICMPv6 Router Advertisement Message Format

In Figure 2-3, Type Field is 134, Code Filed 0 and Current Hop Limit should be set to the hop count field of the IP header as a default for outgoing packets and is 8-bit unsigned integer. If the Current Hop Limit field is set to 0, the forwarding router specifies it as ‘unspecified’.

The “*M”* Managed Address Configuration bit and “*O”* Other Stateful Configuration bit are 1-bits each and are special flags defined in RA. If *M* flag is set to 1, it specifies managed address configuration i.e. stateful using DHCPv6 and host starts communication with DHCPv6 server and if not set i.e. 0, specifies automatic address configuration. The *O* flag specifies the host to use other source for non-address information like DNS address or not. If *O* flag is set then the host uses stateful address autoconfiguration for non-address information else it specifies that RA message contains all those other information. So, the flag specifies wheatear to use DHCPv6 or not. The rest 6 bits are reserved and must be initialized to 0 by the sender and if not must be ignored by the receiver.

The Router Lifetime specifies if the RA sending router is a default router or not. The value set to 0 means the router can’t be used as default router and hence the receiving nodes mustn’t set the router as a default router in default router list. Any other value in this field specifies the Lifetime in seconds and informs host that the router can be used as default router. It is 16-bit unsigned integer with available maximum value of 18.2 hours.

The Reachable Time field is represented in milliseconds and is used by NUD algorithm to determine the neighbor status, either reachable or not. If the value set to 0 is determined by the receiving node it assumes that the sending node is unreachable and if set to any other value then 0, it is assumed that the sending node is reachable after that time. It contains 32-bit unsigned integer.

The Retransmission Timer field is used by address resolution and NUD algorithm and is 32-bit unsigned integer represented in milliseconds. It carries the information of the time between which NS messages are retransmitted.

Currently there are 3 possible values defined in Option filed: Source Link-Layer address, MTU (can be variable depending upon the link) and Prefix Information. In Source Link-Layer address option, sender’s interface address may be used. The router must include all the on-link prefixes that a node on the link must know in RA messages. Prefix Information is used for automatic address configuration.

RS and RA message are used for *router discovery* mechanism and NS and NA message used for *host-to-host discovery* mechanism.

Essential Services from Neighbor Discovery Protocol

2.3.2 Address Resolution

The process of determination of link-layer address of a neighbor by a node form the provided IP address is called Address Resolution. All sending nodes should have knowledge about the link-layer address of the neighbor on the same link according to the unicast address before sending any unicast packets, if not they must perform address resolution. Link-Layer addresses of the neighbor are obtained from the Neighbor Solicitation message, Router Advertisement message and Redirect message. Exception is Multicast address in which Address Resolution is never performed.

2.3.3 Neighbor Unreachability Detection

Hosts in the same network should know if the packets sent are correctly forwarded to the neighbor where a neighbor can be a host in same link or a router. This feature is provided by NUD. First a host sends out the Neighbor Solicitation Message, if it receives a Neighbor Advertisement Message in response which contains the link-layer address of the neighbor then the host updates it *Neighbor cache* entry adding the neighbor as reachable if not it either removes the neighbor link-layer address or mark it as unreachable in its *Neighbor cache*. Also whenever a link-layer address of a host changes it advertise a Neighbor Advertisement Message announcing the change. ND detects a failed connectivity so the packets or traffic are not sent to unreachable neighbor and hence solves the issue with expired ARP caches.

2.3.4 Duplicate Address Detection

DAD is the process of verifying the uniqueness of an address. It is mandatory before assigning an address to an interface. DAD is the process which allows hosts to find out if the address that it is trying to use is already used by other node in the link or not. DAD is performed by using a pair of Neighbor Solicitation and Neighbor Advertisement messages. DAD is performed for Link-Local addresses, Global Unicast Address, temporary generated Global Unicast Address and address created using stateless address autoconfiguration, discussed later, but not for Anycast address since Anycast address is used by multiple hosts. However DAD can be skipped for those addresses that have EUI-64 based Interface Identifier. [6]

In case, when the duplication rate is relatively low, it takes more time to complete DAD process and hosts are unable to start communication with peers which is solved by opti-DAD mechanism and the address during this process is marked as *optimistic* address but once the DAD process is complete the status of address must be changed to *preferred* or *deprecated* depending upon the address lifetime.

2.3.5 Path MTU Discovery and Fragmentation

Path MTU (PMTU) is the maximum packet size that can be transmitted across the delivery path without fragmentation of the packets. The minimum required link's MTU is 1280 bytes in IPv6 and if any link which may not support this MTU requirement, link-specific fragmentation must be provided by a layer below IPv6 stack. [7] PMTUD is mandatory in IPv6 where links with greater than 1280 bytes MTU are used since host fragment packets instead of routers and the fragmenting hosts should know the MTU of all the links from where the packets travels through which might be variable since each link employs different layer-2 technology. If the router receives a packet greater that the MTU of next forwarding link, it generates the ICMPv6 Packet Too Big Message and informs the source with MTU that can be used and the source fragment the message with the new MTU received. This process continues until the packets reach the destination. *"IPv6 fragmentation has the same problem as IPv4 fragmentation: the TCP or UDP port numbers are available only in the first fragment. This makes it hard for firewalls and the like to filter fragmented packets. Common solutions are to reassemble the packet prior to filtering or to discard all fragments."*[8]

2.4 Autoconfiguration

Autoconfiguration is new major feature provided by IPv6 which reduce the task of manual assignment of address to the network administrators. IPv6 nodes are able to configure their unique link-local and global address and getting network information themselves. In IPv4, DHCP server is used to assign address to the host which is no longer necessary in IPv6. IPv6 offers "plug and play" to nodes connected to IPv6 network. A node in the state of null information about the network can plug and start playing in the network; it's called Stateless address autoconfiguration. It becomes very hard to imagine IPv6 without autoconfiguration since it is one of the major features in IPv6. Manual configuration of address is called Stateful address autoconfiguration in IPv6.

2.4.1 Stateful Address Autoconfiguration

Stateful address autoconfiguration applies the same technique as in IPv4 i.e. assignment of the address using DHCP where a server stores and manages whole address information. DHCP for IPv6 is called DHCPv6 and configuration of address using DHCPv6 is categorized as stateful address autoconfiguration in IPv6. The pro of using stateful address autoconfiguration is management of address resources efficiently and the con is heavy burden in network administrators. A node invokes stateful address autoconfiguration if the *M* flag in the Router Advertisement is set to 1.

2.4.2 Stateless Address Autoconfiguration

Configuration on routers is enough for stateless address autoconfiguration and no more servers are required. In this process host learns all the information about the network through the router advertisement. All the address configured in any step should pass DAD mechanism for its uniqueness and can be used for the preferred or configured time period. The lifecycle of IPv6 address is discussed later. Stateless address autoconfiguration is performed if the *M* flag in the Router Advertisement is not set i.e. 0.

Both stateful and stateless address autoconfiguration can be used together or independently. If *M* flag is not set and *O* flag is set to 1, node configures its address using stateless autoconfiguration and asks for other information using stateful autoconfiguration. If both *M* and *O* flags are not set, meaning that all information are provided through the Router Advertisement message, node will use stateless address configuration mechanism only.

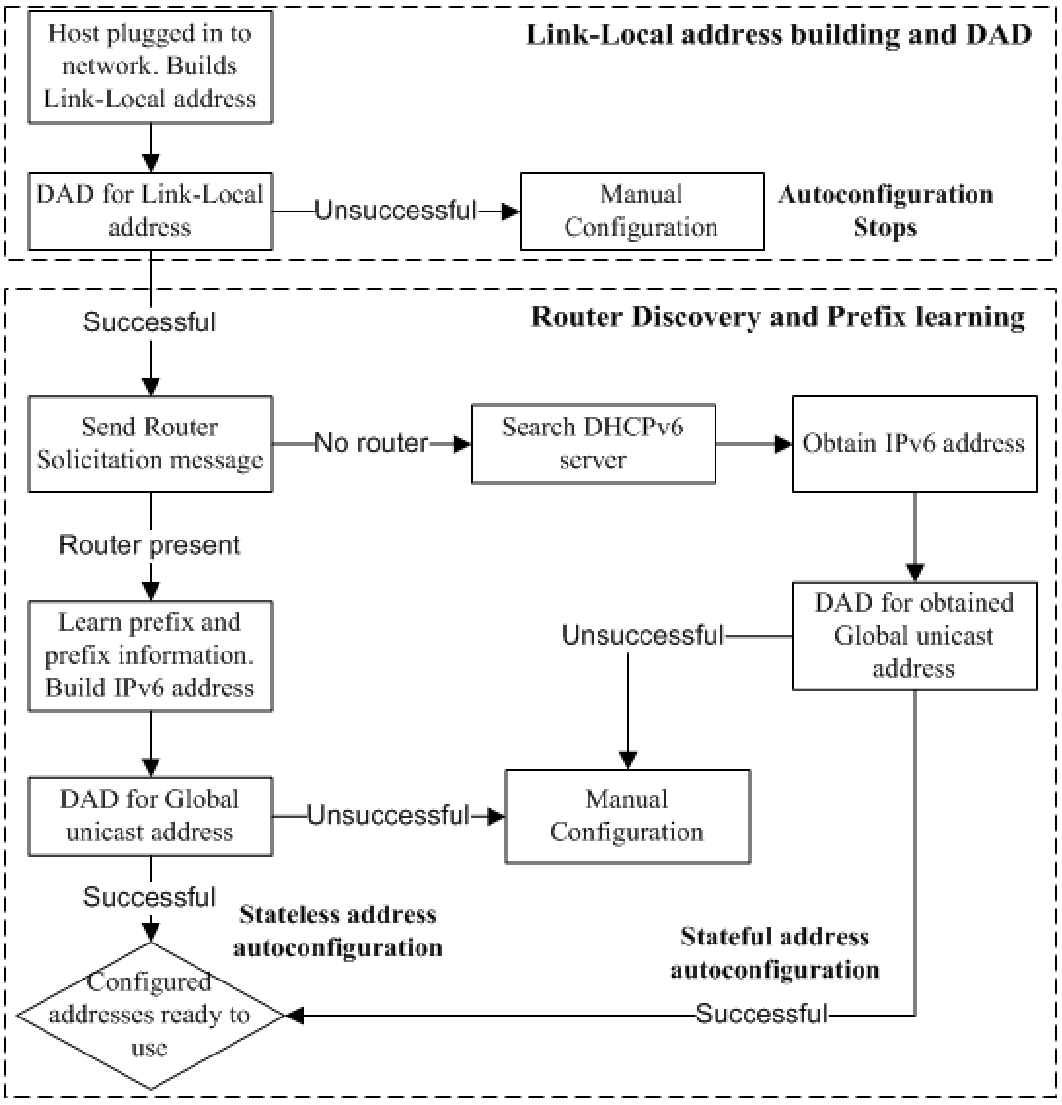

Link-Local address building and DAD
Host plugged in to network. Builds Link-Local address
DAD for Link-Local address
Unsuccessful
Manual Configuration
Autoconfiguration Stops
Successful
Router Discovery and Prefix learning
Send Router Solicitation message
No router
Search DHCPv6 server
Obtain IPv6 address
Router present
Learn prefix and prefix information. Build IPv6 address
DAD for obtained Global unicast address
Unsuccessful
DAD for Global unicast address
Unsuccessful
Manual Configuration
Successful
Stateless address autoconfiguration
Stateful address autoconfiguration
Configured addresses ready to use
Successful

Figure 2-4. Address configuration mechanism

From Figure 2-4, first the IPv6 enabled node generates 64-bit interface identifier based on its 48-bit MAC address and then generates link-local address from its 48-bit MAC address using *FE80::/10* prefix. Now the node joins the several multicast group in the link such as all-nodes multicast group identified by the address *FF01::1* and *FF02::2* and solicited-node multicast group identified by the *FF02::1:FFXX:XXXX* where *Xs* are lower 24 bits of 64 bits interface identifier. Generated link-local address is *tentative* and goes through DAD process for uniqueness test. If the address is found duplicated then autoconfiguration stops and manual configuration must start. After verifying the uniqueness of the link-local address, the node enters into router discovery stage in which the node sends

Router Solicitation message or waits for Router Advertisement message. If no routers are present in the link, the node gets no response to its Router Solicitation message and attempts stateful autoconfiguration in which the node ask DHCPv6 for address and non-address information. The configured address must go through the DAD process to verify its uniqueness and if fails manual configuration must start and if passed the address status is changed from *tentative* to *preferred*. If the router is present in the link, in response to Router Solicitation message it sends Router Advertisement message informing the prefix to be used and other information if configured to provide. Now the node is able to build global unicast address by appending interface identifier to the prefix information learned. The generated global unicast address now undergoes DAD process for its uniqueness verification.

Address Lifecycle

Despite DAD is performed to verify uniqueness of an address, it have certain lifetime. When the lifetime expires, the address can no longer be used i.e. invalid address. Every address generated has certain lifetime in IPv6. Depending upon the status, IPv6 address can be divided into 3 types as described below.

2.4.3 Tentative Address

This is the address under DAD stage which is waiting for the uniqueness verification. This address is not assigned to any interface but the address prior to the assignment. When a node uses Opti-DAD this address is marked as *Optimistic* address.

2.4.4 Preferred Address

This is the address which is verified unique and can be used without restriction. This address is assigned to an interface. The preferred lifetime information is obtained from the Router Advertisement message.

2.4.5 Deprecated Address

Deprecated address are those address whose lifetime is about to finish. This address can be used but are not recommended. A node can use this address to continue the on-going communication but not for new communication. The deprecated address is removed from the interface after the valid lifetime runs out. At this point a node either must generate new address or refresh the timer; otherwise the session using the deprecated address as source address will break. Figure 2-5 shows IPv6 address lifecycle. Deprecated address has advantages in Wireless network, if the link goes down and comes up quickly because of bad reception the sessions are still alive because the interface address status changes to deprecated one before it is counted as invalid.

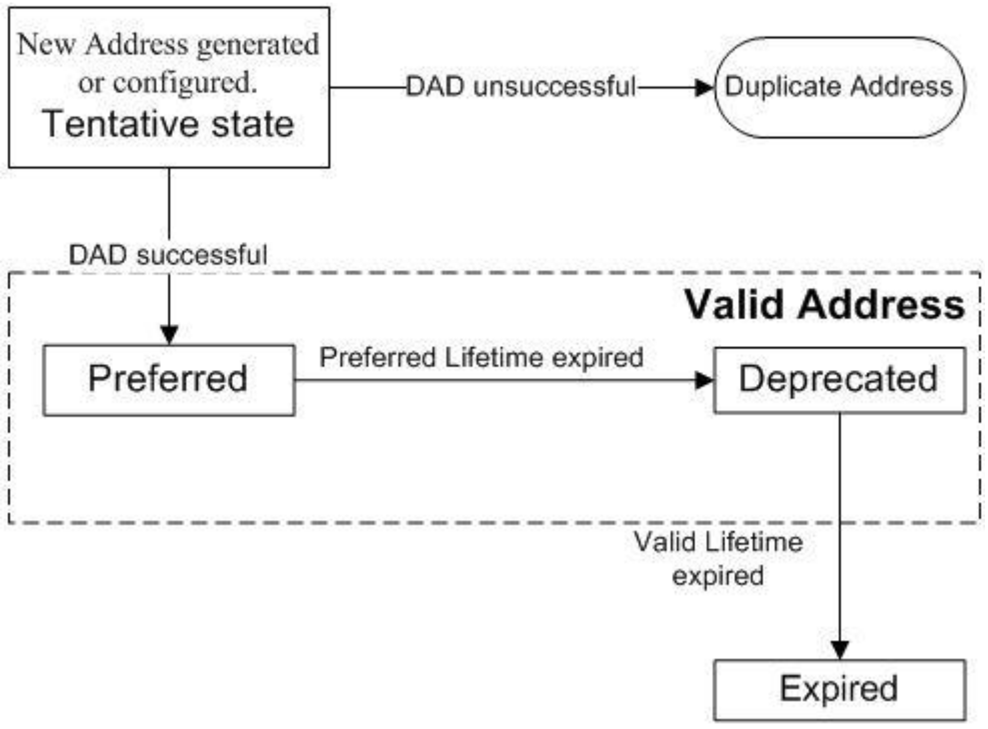


Figure 2-5. IPv6 address lifecycle

In most cases the address remains in preferred state because the lifetime is refreshed by the Router Advertisement message which is broadcast periodically by the router. If there is no further Router Advertisement message advertised by the router then the address enters the deprecated status and finally becomes invalid.

Some More about Addresses

Since IPv6 supports assigning the multiple addresses in an interface a node can sit in two or more different subnet. In this case the node must generate multiple addresses and use corresponding solicited multicast addresses.

2.4.6 Temporary Address

Using the MAC address of an interface to build the address is a perfect mechanism because there is no chance of duplicate address on the link but when it comes to security, it seems to be the worst method since it exposes the MAC address of a node to external environment and can be easily tracked by the use of MAC address. So, a hacker can follow the victim’s movement and can prepare for the attack. To solve this problem IPv6 introduces the privacy address since an interface can have more than one IPv6 address at the same time, temporary address can be generated and used for outbound traffic. Temporary address is generated by appending random number of 64-bits to the prefix learned. This address is similar to global unicast address and must go through DAD process and have valid lifetime. Microsoft has already implemented privacy address to its product later than Windows XP but Unix-like operating systems haven’t adopted it yet and must be manually configured to do so. If a node sits in more than one subnet the generated temporary address’s lower 64-bits can be same but are distinguished by the subnet prefix.

2.4.7 Address Requirements

In IPv4 a single interface has single IPv4 address which is not the case in IPv6. IPv6 requires multiple addresses in a single interface for its functionality and since it allows assigning multiple addresses in an interface. A node which doesn’t forward packets i.e. typically a host PC, printers, etc. require following address for IPv6 functionality.

- Each interface link-local address
- Manually or automatically configured unicast and Anycast address for that interface.
- Loop-back address
- All-nodes multicast addresses (*FF01:1, FF02::1*)
- Solicited-node multicast addresses corresponding to unicast and Anycast addresses configured
- Multicast addresses assigned which identify the groups the node joins

For a node that forwards packets typically a router needs even more addresses that that of host. It must support all the addresses required by the host and the addresses listed below:

- Subnet-router Anycast addresses for all interface where it acts as a router
- All assigned Anycast addresses
- All-routers multicast addresses (*FF01::2, FF02::2, FF05::2*)

2.4.8 Cases of Address Autoconfiguration

The address autoconfiguration depends upon the configured parameters in Router Advertisement message as below:

- If RA lifetime and prefix lifetime is valid, IPv6 work.
- If RA lifetime is valid but invalid prefix lifetime, the system has IPv6 default route but no global IPv6 address.
- If RA lifetime and prefix lifetime are invalid, IPv6 is disabled.
- If RA lifetime is invalid but valid prefix lifetime, the system has global IPv6 address but no IPv6 default route.

VAMK have native IPv6 connection available from its ISP, FUNET, Finish University and Research Network with the prefix of *2001:708:230::/*48. All the traffic generated from VAMK passes through FUNET network meaning that the default route for VAMK's router is FUNET router's address. Since our FreeBSD box acts as a Router, DHCPv6, DNS and Firewall and the internal network is divided using VLANs, let's have closer look at them.

2.5 Virtual Local Area Network (VLAN)

VLAN allows the logical segmentation of a LAN into different IP sub network. Using VLANs groups can be created logically instead of their physical location. Different users form different VLAN segments can be organized to share common resources. The big advantage of using VLAN is increase in security since groups that have sensitive data can be separated from network. VLANs can be broadly divided into Static VLANs, Dynamic VLANs and Tagged VLANs.

2.5.1 Static VLANs

Static VLANs are configured with a VLAN ID, VLAN name and port member where ports are manually assigned to VLANs. Data packets are bridged from source to destination port in same VLAN and cross domain broadcast packets are eliminated to save bandwidth in the switch.

2.5.2 Dynamic VLANs

Dynamic VLAN assignment is based in the source MAC address of the device connected to the port. It is configured with a special server that has database of all devices' MAC address to assign VLANs dynamically. The switch will dynamically assign the proper VLAN to the host even the host moves form one port to another.

2.5.3 Tagged VLANs

Tagged VLAN is point-to-point link between two network devices that carries more than one VLAN in network. VLAN tagging allows multiple bridge networks to transparently share the same physical network link without leakage of information between networks.

Tagged**:** Ports that compile with the 802.1Q standard including priority settings and allow the port to join multiple networks.
Untagged: Untagged port doesn’t use or forward 802.1Q VLAN tagging.

2.6 Routing IPv6

IPv6 defines routers as nodes forwarding packets and advertising Router Advertisement message. Router must support IPv6 to forward IPv6 packets. Usually, routers are configured with static IP address and don’t accept Router Advertisement message. Routing IPv6 is no different from routing IPv4. Routes can be defined statically or dynamically using routing protocols available. Currently there are 4 routing protocol working with IPv6, RIPng (Routing Information Protocol for next generation) based on RIP version 2, OSPFv3 (Open Shortest Path First for IPv6) which keeps the map of entire network topology and the traffic starts traveling through the available short path which is determined by using Shortest Path First algorithm, Integrated IS-IS (Integrated Intermediate System to Intermediate System) which is extended OSI routing protocol for IPv6 which for use within an organization and is popular within large ISPs because of its capability of handling large networks and BGP-4 (Border Gateway Protocol version 4) which is used between the networks of different organization such as organization connecting to two or more ISPs.

For a successful delivery of an IP datagram the network prefix of the an destined IP address must correspond to a unique data link layer network, routers and hosts that have a common network prefix must be able to exchange IP datagrams using a data link protocol e.g. Ethernet and every data link layer network must be connected to at least one other data link layer network via a router. The datagram will be sent to a different system if a forwarding node (routers/switch) receives a datagram which is not for the local system but if a receiving node (host PC) receives a datagram which it not for the local system, the datagram will be dropped.

Processing of IPv6 datagram at a router is similar to processing IPv4 datagram. The major differences are routers don’t fragment IPv6 packets nor calculates the checksum. Processing of an IPv6 datagram at a router after receiving a datagram occurs in following ways:

- IP header validation
- Process Next header
- Parsing the destination IP address
- Routing table lookup
- Decrement Hop Limit
- Transmit to next hop
- Send ICMPv6 packet (if required)

FUNET network uses OSPFv2 for IPv4 and IS-IS for IPv6 only. IPv4 infrastructure uses BGP with multiple autonomous systems but for IPv6 BGP has been set up at the border routers, and the rest use IS-IS. The idea to switch from OSPFv2 to IS-IS for both IPv4 and IPv6 has been a low priority task and probably the same kind of BGP infrastructure be built for IPv6 during the switching process. [9]

2.7 Dynamic Host Configuration Protocol for IPv6 (DHCPv6)

Well, DHCP is common in IPv4 network which is widely used to provide IPv4 address dynamically to the clients. DHCP for IPv6 is called DHCPv6 which is extended or modified for IPv6 support and is optional in IPv6. It depends upon the network administrator whether to use DHCPv6 or not depending upon the requirements. DHCPv6 can be omitted totally using stateless address autoconfiguration mechanism or stateless autoconfiguration can be omitted totally using DHCPv6 only i.e. using stateful address configuration only or can be used both. Depending upon the information in Router Advertisement message a client decides whether to use DHCPv6 or not and for what purpose i.e. address configuration or non-address information or even prefixes delegation and if no routers are present in the link, the client starts communication with DHCPv6 if available. Unlike DHCP only provides one IPv4 address to the client DHCPv6 can assign multiple IPv6 addresses according to the client request. If no DHCPv6 are present on the link, relay agent which is an intermediate node between DHCP clients and server can be used to communicate. The relay agent forwards the message for client and DHCPv6 sever. Some differences between DHCP and DHCPv6 are listed below:

- Clients can have multiple IPv6 address so clients can send multiple requests to DHCPv6 server.
- Client can configure IPv6 address from the DHCPv6 server even it is not directly connected to by using relay agents.
- 2-way handshake for simple configurations like request of non-address information i.e. stateless DHCPv6.
- Clients use link-local address as source address instead of unspecified IP like in IPv4.
- Communicating parties uses special multicast addresses for relay agents and servers.
- A server can send *RECONFIGURE-INIT* message to clients to reconfigure when there is update in information or the configuration changes. For instance the change in the prefix i.e. renumbering. This is optional.
- IPv6 doesn’t support BOOTP.
- Clients can send *DECLINE* message if the DAD fails.

The special multicast addresses used in DHCPv6 communication are *FF02::1:2*, All-DHCP-Relay-Agents-and-Servers address used when a DHCPv6 client wants to communicate with relay agents or servers on the link and *FF05::1:3*, All-DHCP-Servers address used by relay agent to communicate with server. And of course, all the communicating parties must be the members of the same multicast group. UDP port 546 called *Client port* is used by servers to reach relays or client and UDP port 547 called *Agent port* is used by clients to reach servers or relay agents. Relay agents use this port to reach servers. Table 2-6 shows messages types defined for DHCPv6.

Type Value	*Message Type*	*Description*
1	Solicit	Used to discover DHCPv6 by clients.
2	Advertise	Response to Solicit message.
3	Request	Information request by clients.
4	Confirm	Sent by clients to verify IP address used is valid or not.
5	Renew	Request to renew the information by client to the same server. For instance, address lifetime.
6	Rebind	Same as Renew message but can be any DHCPv6 server.
7	Reply	Response to Solicit, Request, Renew, Rebind, Confirm and Decline messages by server.
8	Release	Sent by clients to same server when it releases IP address assigned.
9	Decline	Sent by clients to server if assigned IP address fails DAD mechanism.
10	Reconfigure	Sent by server to notify the update or change. Client must initiate Renew/Reply or Information-Request/Reply transaction to update the changes.
11	Information-Request	Request for non-address information by clients.
12	Relay-Forward	Used by Relay agent to forward clients message to server. The client’s message is encapsulated in an option in Relay-Forward message.
13	Relay-Reply	Used by Relay agent to forward server message to client. The relay decapsulates the Relay-Reply message received from server which contains the encapsulated message to the client.

Table 2-6. DHCPv6 message types

So, the scope of DHCPv6 depends upon its use. It can be used for 3 different purposes described below:

2.7.1 Stateful DHCPv6

In this case, DHCPv6 is used for address configuration. The clients configure their address using DHCPv6. The client use stateful DHCPv6 when it receives the Router Advertisement message with *M* flag or bit set to 1 or if no routers are present in the link.

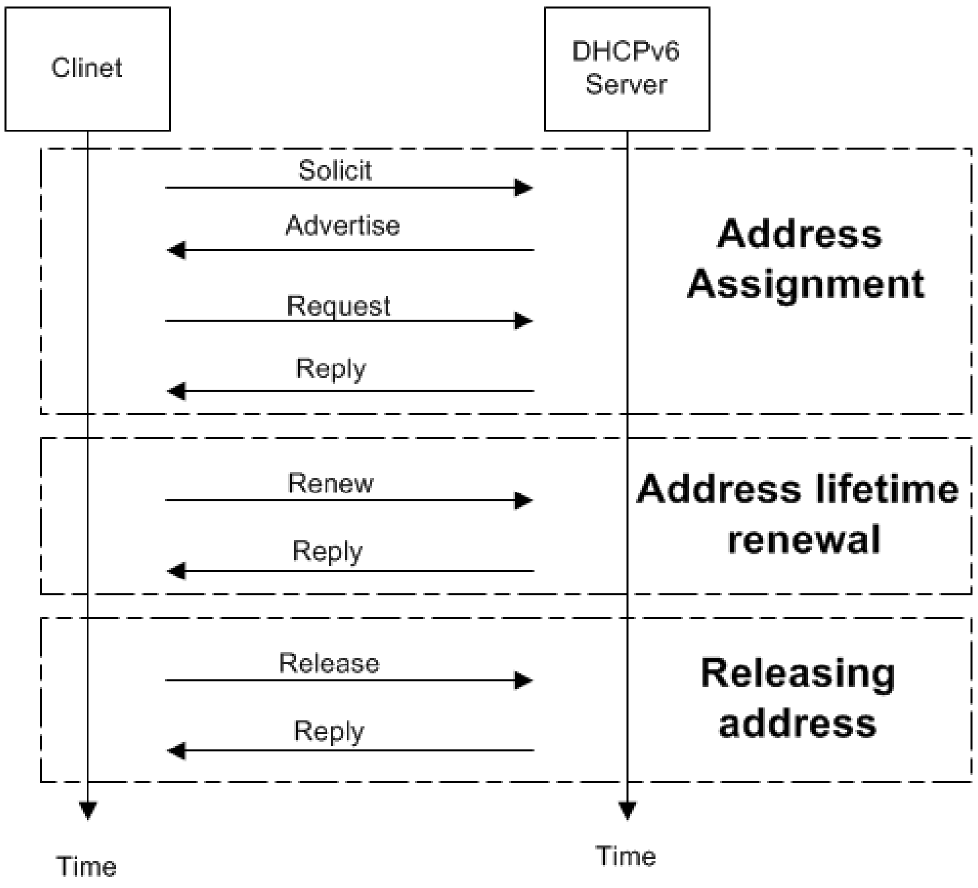


Figure 2-6. Stateful DHCPv6 mechanism

Figure 2-6 shows the process of assigning address, address lifetime renewal and releasing address. If relay is used then instead of DHCPv6 server there will be relay agent in between in the figure which then forwards the client message as *Relay-Forward* message and the server sends message to relay first as *Relay-Reply* message. The messages which are not shown in the figure are used in special cases as described in Table 2-6. A DHCPv6 client sends *Solicit* message to discover server or relay in the link and in response to this message the server replies with *Advertise* message. Before the DHCPv6 solicitation the client are required to build Identity Association (IA), a collection of address assigned to client interface which is used to identify and manage addresses. IA identification (IAID) identifies an IA which is assigned by a client uniquely over all IAs. A DHCPv6 node is identified by its DHCP Unique Identifier (DUID) which is assigned to each DHCP node and is globally unique. Then the client sends *Request* message and in response the server sends *Reply* message with offered IPv6 address.

The offered address has preferred and valid lifetime and a client can renew it before it expires by sending *Renew* message for which the server replies with *Reply* message and new valid lifetime. If the node is unplugged or the client want to release the assigned address or if the link gets down the client sends *Release* message and the server replies with *Reply* message. The released address can now be assigned to new interface.

2.7.2 Stateless DHCPv6

If the DHCPv6 is used to provide non-address information then it is said to be stateless DHCPv6. Clients use this method if it receives the Router Advertisement Message with *O* flag is set to 1. Information exchange process has been simplified and can be done in 2 steps.

Figure 2-7 shows the process of information request and reply. After receiving the Router Advertisement message with *O* flag set to 1, the client understands that there is a DHCPv6 server in the link to provide additional information other than address prefix, and then the client after configuring its address sends *Information-Request* message to the server for which the server replies with *Reply* message including the information requested.

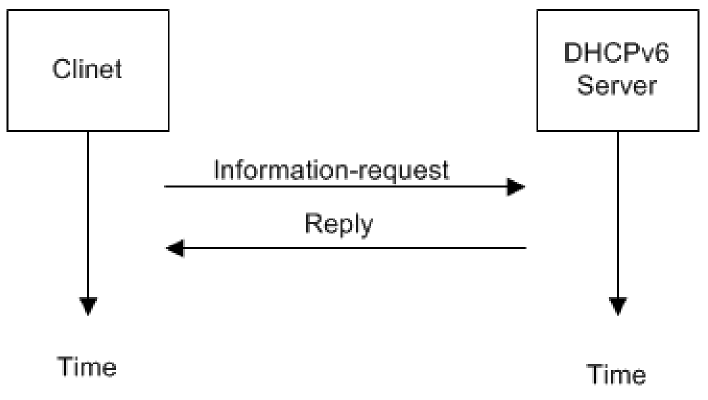


Figure 2-7. Stateless DHCPv6 mechanism

In case if the client receives the Router Advertisement message with *M* bit set to 1 but *O* bit unaltered i.e. 0, then the client configures its address using Stateful DHCPv6 and gets other non-address information form the Router Advertisement message. And in case both bits in Router Advertisement message are set to 1, then client uses both stateless and stateful DHCPv6 procedure to obtain address and non-address information.

2.7.3 Prefix Delegation

DHCPv6 can be used to delegate prefix(s) to the customer. The process may be used in ISPs. The customer's router or DHCPv6 communicates with ISPs edge DHCPv6 server which then selects the prefix(s) to be assigned. The assigned prefix can be used by customer to its LAN. Using this method it is not necessary to advertise prefix(s) information on Router Advertisement message but the default gateway must be informed to the client. For instance, a customer may be provided /48 prefix by ISP DHCPv6 server by using prefix delegation method which the customer can delegate to /64 prefix and use stateless autoconfiguration.

To prevent false DHCP messages, DHCPv6 provides authentication mechanism which allows the secure communication between communicating parties. The authentication mechanism is optional and must be configured on both server and clients resulting heavy load for network administrators.

2.8 Domain Name Server for IPv6 (DNSv6)

DNS is used to map domain name into IP address and vice-versa also. Operating system can't understand domain name and it is hard for users to remember IP addresses and even more IPv6 addresses. DNS in IPv6 follows the same hierarchy as it does in IPv4 and works in systematic delegation procedure. DNS in IPv6 world works in the same as it does in IPv4 but the new record type "*AAAA*" read as "*Quad A*" is defined to support 128-bit long IPv6 address and nibble method for reverse lookup under *ip6.arpa*. Although other record types are also defined like *DNAME, A6* and *Bitlabels, AAAA* is recommended by IETF and is defined as best common practice. *Ip6.int* is now deprecated.

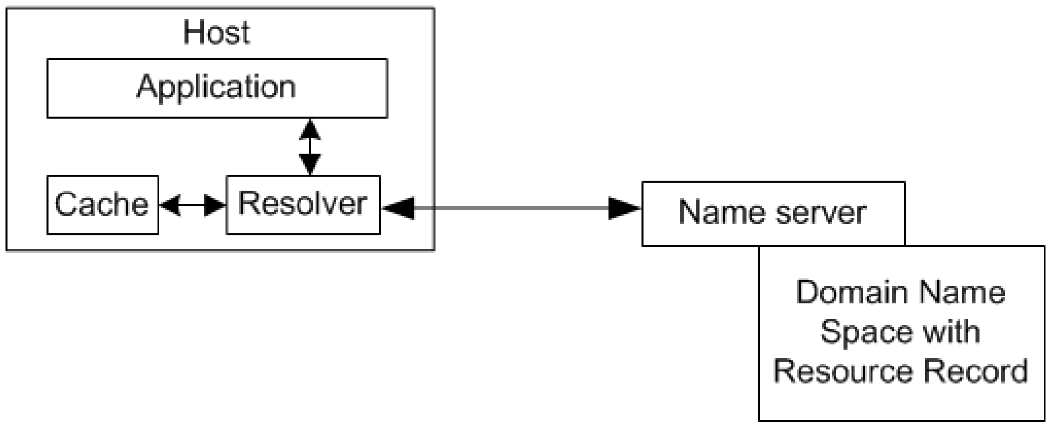


Figure 2-8. DNS architecture simplified

Figure 2-8 shows the simplified DNS architecture and shows the 3 major components of DNS, Domain Name Space with Resource Record, Name server and Resolver. Domain Name Space contains the record for domain names defined with specific type like *A, AAAA, A6, PTR* etc. The query message generated defines the type. Name server keeps the information about where those zones belong to. Resolver program is used to send queries from client to local name server and gets answer for generated queries. Name server and resolver may have capacity to keep some cache information so that if a client requests the same information frequently then instead of repeating the whole process information from cache can be used. Cache information keeps on out-dating after certain time limit and can no longer be used and new information must be provided.

2.8.1 DNS Message Format

DNS protocol is used to exchange Query and Response message between hosts and name servers. Both TCP and UDP protocols can be used to exchange the messages. As shown in Figure 2-9, the DNS message is divided into 5 parts, DNS header, Question, Answer, Authority and Additional.

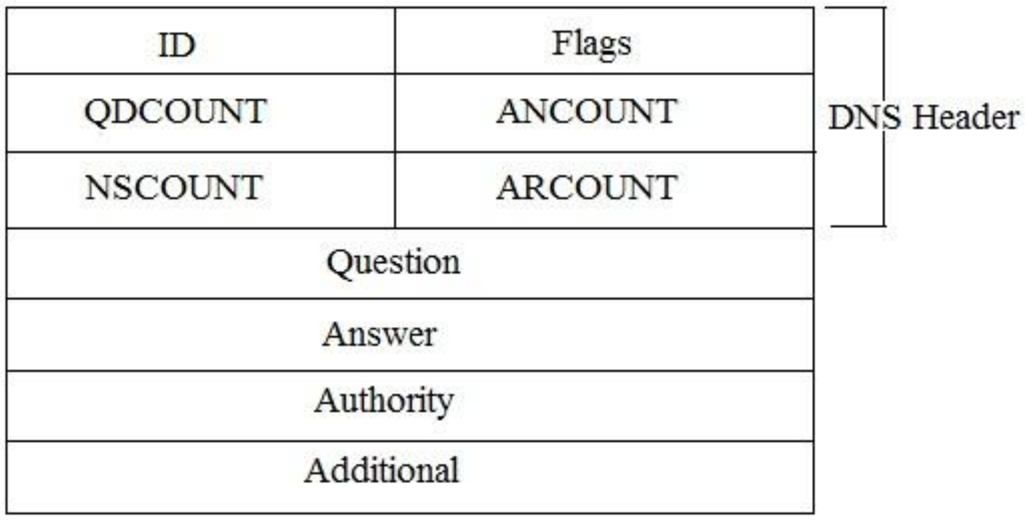


Figure 2-9. DNA message format

- The Header is always present in the message. The DNS Header is composed of 6 fields, such as ID, flags, QDCOUNT, ANCOUNT, NSCOUNT, and ARCOUNT fields.
  - ❖ The ID field is assigned by an application which triggers DNS query process and is 16-bit long. This field is copied into all response messages.
  - ❖ 7 different flags are defined for the DNS message; QR, Opcode, AA, TC, RD, RA and RCODE flags. The QR flag specifies whether the message is a query or a response and is 1-bit long. The Opcode flag notifies what kind of query is contained in the message; e.g. Standard query, Reverse query, multiple and single. The AA (Authoritative answer) flag notifies that the responding name server is an administrative name server for the queried domain name and is only valid in a response message. When the length of the message exceeds 512 characters the message will be

truncated and is notified using TC (truncation) flag. The RD (recursion desired) flag is set to request the recursive query processing. The RA (recursion available) flag is set to notify whether the recursive query processing is available in the name server. The RCODE (response code) flag is only used in the response message to notify the whether the response obtained contains error or not; e.g. No error condition, Format error, Server failure, Name error, Not completed and Refused.

    - ❖ The QDCOUNT specifies the number of records in the Question part of the query message, the ANCOUNT specifies the number of resource records in the Answer part of the response message, the NSCOUNT specifies the number of resource records in the Authority part of the response message and the ARCOUNT specifies the number of resource records in the Additional part of the response message. The QDCOUNT, ANCOUNT, NSCOUNT, and ARCOUNT fields contain unsigned 16-bit integer value.

- The detail contents for the query such as query type, query class and the query domain name are contained in the Question part.
- The Answer part contains the resource records for the query.
- The Authority part contains resource records which points to the authoritative name server managing the questioned domain.
- The Additional part contains the resource record related to the query.

DNS can be divided into two depending upon its use namely Forward DNS, which maps domain names into IP address and Reverse DNS which maps IP address to domain name. The query generated by client is forwarded to local name server using resolver. The local name server forwards the query to nearest name server until it gets the answer. The query can be handled in recursive, iterative or mixed way.

2.8.2 Recursive method

In recursive method the local name server looks up a query in behalf of client until it finds the answer. The query is forwarded to root name server from local name server which is forwarded to intermediate name server which also forwards to authoritative name server which replies with the answer to intermediate name server and is forwarded to root name server and then to local name server. Now the answer is replied to the client by local name server. Figure 2-10 shows the process.

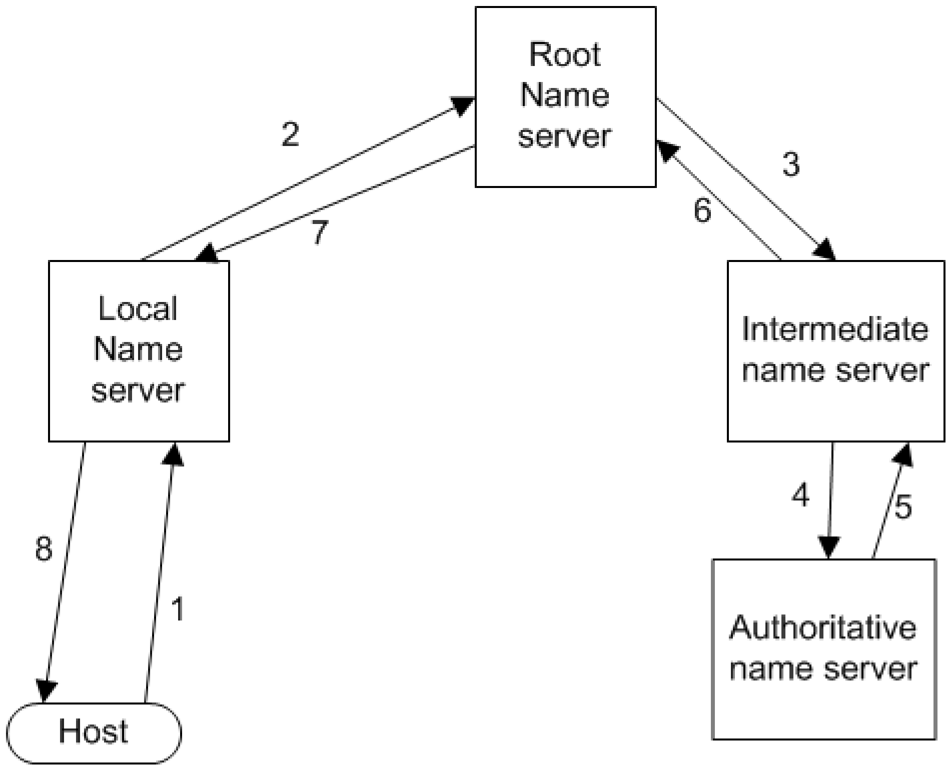


Figure 2-10. Recursive name lookup

2.8.3 Iterative method

In iterative method the local name server forwards the query which when finds the address of the referral name server sends to the client as an answer and the client keeps querying to another name server whose address is referred. Figure 2-11 shows the process.

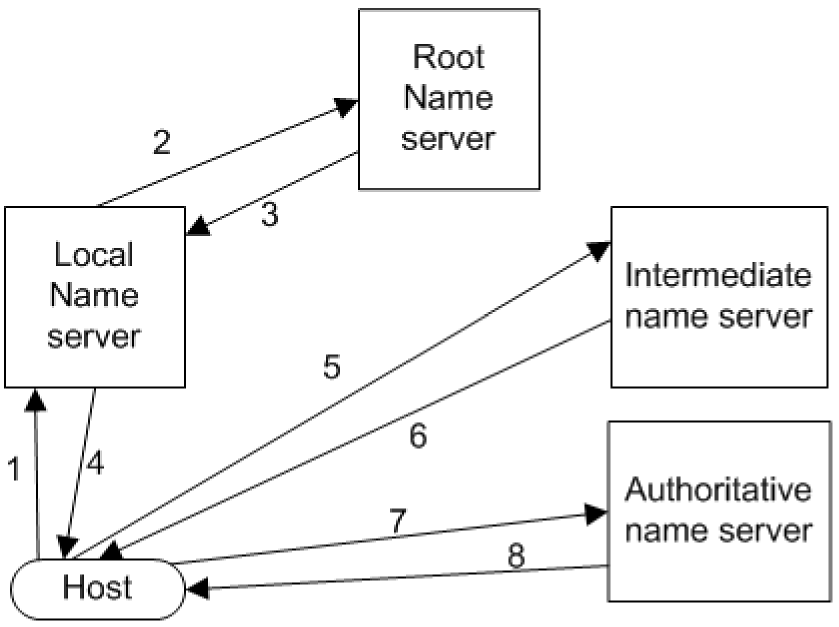


Figure 2-11. Iterative name lookup

2.8.4 Mixed method

In general, both recursive and iterative method is used. The query between local name server and client is handled recursively and the remaining lookup is handled iteratively. Figure 2-12 shows the process for *example.fi*. The client request for *example.fi* and the request are sent to local name server by resolver. The local name server forwards the query to root name server which points the referral to *.fi* top level name server. The local name server then sends query to *.fi* name server which still doesn’t have the answer but points to authoritative name server as an answer. Now, the local name server sends the query to *example.fi* authoritative name sever and hence get the correct answer when is then forwarded to client who requested the query.

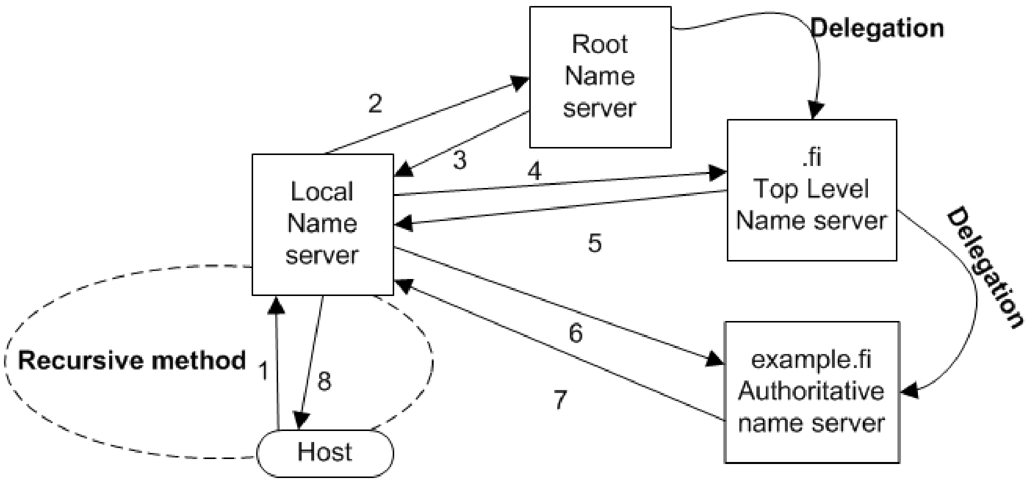


Figure 2-12 Mixed method name lookup

2.8.5 Representing IPv6 information in DNS

Like IPv4 information in DNS is represented using *A* record type, IPv6 is represented using *AAAA* record type. Although other possibilities were proposed to represent 128-bit long IPv6 information in DNS, they don't gain much popularity and hence are deprecated. Other record types proposed were *A6, DNAME* and *Bitlabels.*

When it comes to the reverse mapping, *in-addr.arpa* is used for IPv4 in which individual bytes in the address are reversed followed by *.in-addr.arpa*. Similarly IPv6 reverse mapping use *ip6.arpa* which is now the recommended method. Although *ip6.int* was used in the first place, it has now been depreciated. Using *ip6.arpa*, the reverse mapping is done by taking the hexadecimal digits of IPv6 address in reverse order and adding *ip6.arpa* at the end, condition applied that the sequences of zeros can't be omitted.

BIND (Berkeley Internet Name Daemon) is widely used DNS software which is also de facto standard on Unix-like operating systems. BIND 9 now fully supports IPv6 i.e. handling IPv6 queries, responding to those queries over IPv6 and querying other name servers over IPv6.

2.9 Firewall

Security has been the one of the important part in network. Firewall is mainly used to protect outside attacks and malicious connections. To keep inner network secure from the outside world it is necessary to filter the packets that are unwanted and allow access for authorized packets or connection for smooth running network which is achieved by using firewall. Firewall is used to block unwanted connection or packets and allow only the authorized packets. Many different types of firewall have been developed till now based on the filtering techniques.

It is obvious that the firewall rules for IPv4 will not work for IPv6, a complete new set of firewall rules must be implemented to secure IPv6 network. Before making the rule set for IPv6 it is very important to understand the features and mechanism on how IPv6 work. There are major features which should be allowed for IPv6 functionality like ICMPv6. Since the NDP functionality extensively uses ICMPv6 protocol it is like a day dream to wish for IPv6 functionality blocking the ICMPv6 packets.

FreeBSD comes with Packet Filter (PF) firewall by default in its installation which was imported from OpenBSD project in 2003 as third party software. [10] PF work on IP packet level and intercepts each IP packets passing through the kernel. Only those packets matching the rules defined are allowed and else are silently discarded or dropped. PF provides both stateless and stateful packet filtering. In stateless packet filtering, no information about connection is stored and filtering is based on the information contained in the packet itself. In stateful packet filtering, firewall keeps the track of connection which makes filtering more powerful. Rules in PF are evaluated from top to bottom and the last matching rule wins.

PF has many features else than firewalling like NAT, load balancing and traffic management. Often PF is combined with CARP (Common Address Redundancy Protocol) and *pfsysnc* to make secure and all time available i.e. redundant network.

2.10 Redundancy

Building a stable running network means there is no question about its availability and the service provided is all time available. Since a failure in the network may cause the entire group of machine to remain offline it is necessary to have backup. The problem comes more important when it comes to public servers since any damage may cause it unreachable throughout the world. To solve these issues CARP and *pfsysnc* were introduced in OpenBSD which was later imported to FreeBSD. The main purpose of CARP and *pfsync* is to make the failure transparent to the user and availability of the network resources all time. CARP and *pfsync* or commonly understood as *pfsense* is not available in default installation so, the kernel must be rebuilt with these features included.

2.10.1 The CARP Protocol

CARP is based on creating a single virtual network interface between multiple or group of machines to achieve redundancy. Using CARP the machines in redundancy group uses the same virtual IP address so that if one fails another from the group starts working which results the service continuity. Usually CARP is paired with *pfsync* protocol to synchronize the PF states when another machine takes the place of original one even though CARP isn’t any firewall specific protocol. CARP can also be used for load balancing purpose to a number of network services. On a redundancy group, one machine takes the role of *master* which will provide network services and other machine takes the role of *backup* which will provide the same services upon the *master* failure meaning all the *master* and *backup* shares the same virtual IP address which will be the default gateway for clients. So use of CARP makes work a lot easier during failure, patching, upgrading, and maintenance.

“The Common Address Redundancy Protocol (CARP) was developed as a non-patent-encumbered alternative to the Virtual Router Redundancy Protocol (VRRP), which was quite far along the track to becoming an IETF sanctioned standard, even though possible patent issues have not been resolved.” [11]

The *master* sends advertisement message at configured intervals from which the *backup* knows the master is still alive but when the *backup* doesn't hear any advertisement from *master* i.e. during failure or being offline, the *backup* takes over the duties and now acts as *master* remaining the service unaffected. When the main system comes back, it will hear advertisement form other system thus becomes *backup*. It is also possible to configure to try to become *master* again as shown in Figure 2-13. The multicast address used to advertise CARP message is 224.0.0.18 in IPv4 and *FF02::12* in IPv6.

CARP advertisement contains Virtual Host ID and may be configured with the password which identifies the CARP advertisement to which group it belongs to so it is possible to have multiple CARP interface in a network segment.

The CARP advertisement is protected by SHA-1 HMAC to prevent form malicious users. Since, multiple CARP interface are supported in a network segment special consideration must be made during the IP assignment. The IP address assigned to the CARP interface and physical interface must be in the same subnet to operate CARP in that physical interface. In OpenBSD *carpdev* may be used to specify the physical interface for the CARP to function over that interface but is not available in FreeBSD yet.

2.10.2 The PFSYNC Protocol

CARP is only used to create virtual interface between a groups of machines so a separate protocol *pfsync* must be implemented to synchronize the states between the machines. Unlike CARP *pfsync* is firewall specific and work with PF firewall. The change in state are sent in synchronized interface using IP multicast packets. The multicast IPv4 address used is 224.0.0.240 unless the IP address of the synchronized interface is configured. Usually *pfsync* is used with CARP to synchronize the states during graceful failover to achieve no apparent traffic disruption during failover.

Unlike CARP, *pfsync* protocol doesn't provide any cryptography or authentication mechanism so it is recommended to use a separate network i.e. separate VLAN or interface i.e. using crossover cable and also the loss of any updates result loss of connectivity during failover. It can be configured to use physical interface for synchronization and merge state table between firewalls. If the IP address of the firewall to be used for synchronization is stated then the IP is used as destination address of *pfsync* messages allowing the use of IPsec to protect the communication.

Each *pfsense* firewall in a CARP group has its own unique IP address assigned on the each interface and has the shared CARP Virtual IPs assigned as well. These CARP IPs are only active if the firewall is currently the master. If a failure of any network interface is detected, the next designated firewall switches to *master* on all interfaces. [12]

When the traffic starts flowing through the *master, pfsense* starts creating updates i.e. information of the packets passing through the interfaces and starts sending the updates to the *backup* which is shown as *state 1 creation* and *state 1 update* in Figure 2-13. In case the *master* fails the *backup* takes the role of new *master* making its CARP Virtual IP active and allows the ongoing connections informed via updates. During this process some packets may be lost. The new *master* now starts creating the updates shown as *state 2 creation* in Figure 2-13 and during the same moment CARP advertisement are sent at configured intervals to test whether the original *master* is available or not. When the original *master* returns, it request for the bulk updates. Bulk updates are synchronized at real time meaning that no packets are lost. Then the new *master* starts sending the bulk updates along with the updates for regular state synchronization. Until the bulk update is not completed, *pfsense* prevents CARP from preempting. After the synchronization is completes the original *master* takes the role of *master* and starts its service while the other changes it state from *master* to *backup.*

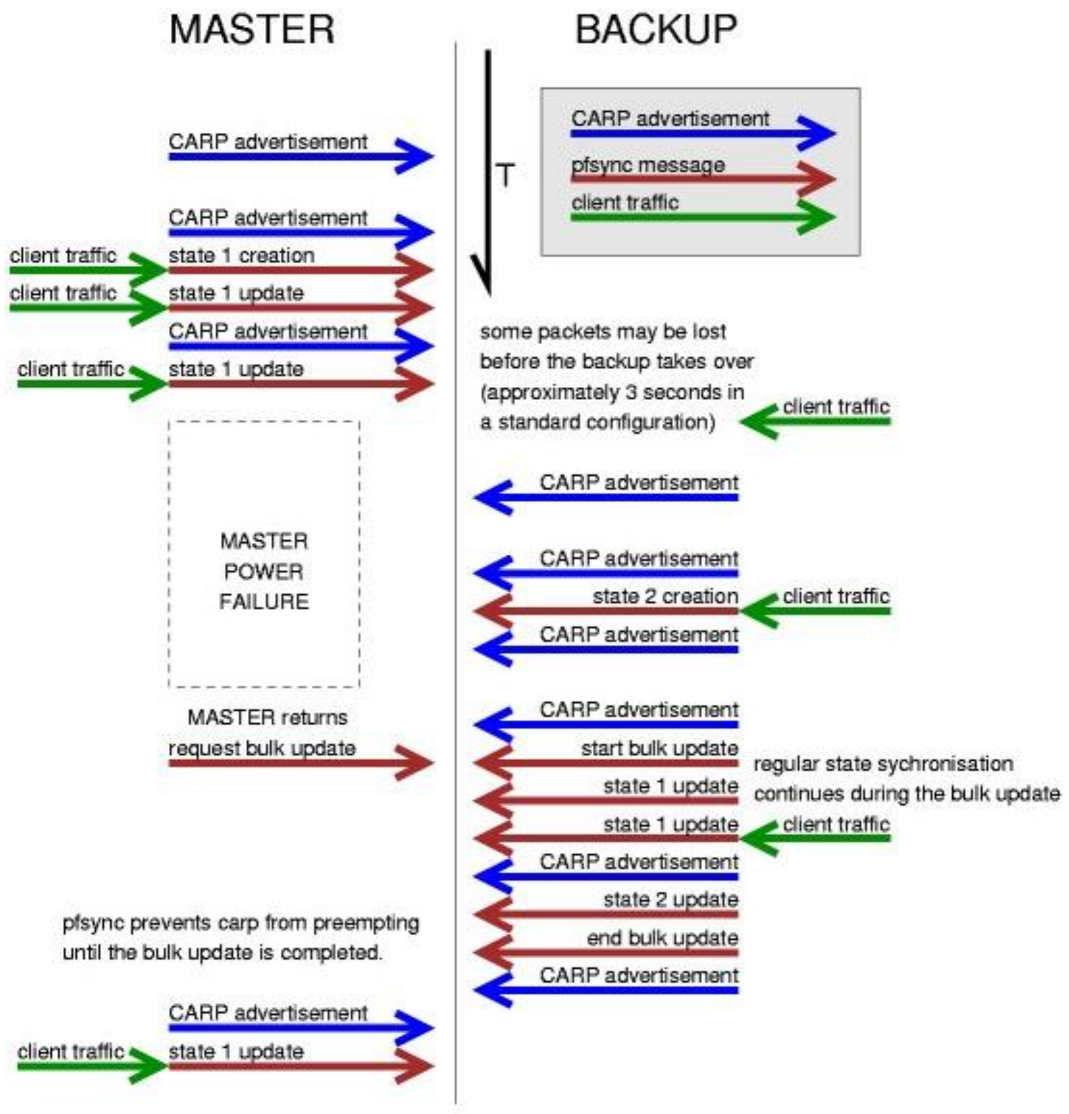


Figure 2-13. CARP and PFSYNC working mechanism [13]

From Figure 2-13, it must be noted that the incoming and outgoing traffic passes through the CARP interface. It is possible to use CARP without assigning address to the physical interface but the services requiring constant connection to the server like SSH will not be transparently transferred to the other system. It is also required to add the rules to pass CARP and *pfsync* message through the firewall.

3. IMPLEMENATATION OF PROJECT

As mentioned earlier, the project was carried out on VAMK. VAMK network is distributed into classrooms, staffs, peripheral devices and so on. The separation between the subnet uses VLANs.

3.1 Network Structure

VAMK has been assigned *2001:708:230::/48* IPv6 prefix. The remaining first 16 bits out of remaining 80 bits is used to divide network into different subnets and the last 64 bits for autoconfiguration. 16 bits are further divided into 8-8 bits to divide more subnets inside subnets. Table 3-1 shows the address division and VLANs used.

FUNET:	**2001:708:230::01/64**	**VLAN 302**
VAMK:	**2001:708:230::02/64**	**VLAN 302**
Routing	2001:708:230:0100::/56	-
Servers	2001:708:230:0200::/56	VLAN 24
Staffs	2001:708:230:0300::/56	VLAN 25
VPN	2001:708:230:0400::/56	-
Classrooms	2001:708:230:0500::/56	VLAN 368
Peripheral Devices	2001:708:230:0600::/56	-

Table 3-1. Address allocation

From Table 3-1, the default router in VAMK will have IPv6 address of *2001:708:230::*02 connected to FUNET edge router of IPv6 address *2001:708:230::*01. The entire classroom network in VAMK will have address in *2001:708:230:0500::/56* range and can be further divided into subnet to identify each classroom as *2001:708:230:0501::/64 – 2001:708:230:05FF::/64*.

Figure 3-1, shows the VAMK layer 3 IPv6 network. The FUNET edge router is connected to VAMK default router which also acts as DHCPv6 server, Recursive

DNS and Firewall. VAMK network is further sub-divided into classroom network, staff network and so on.

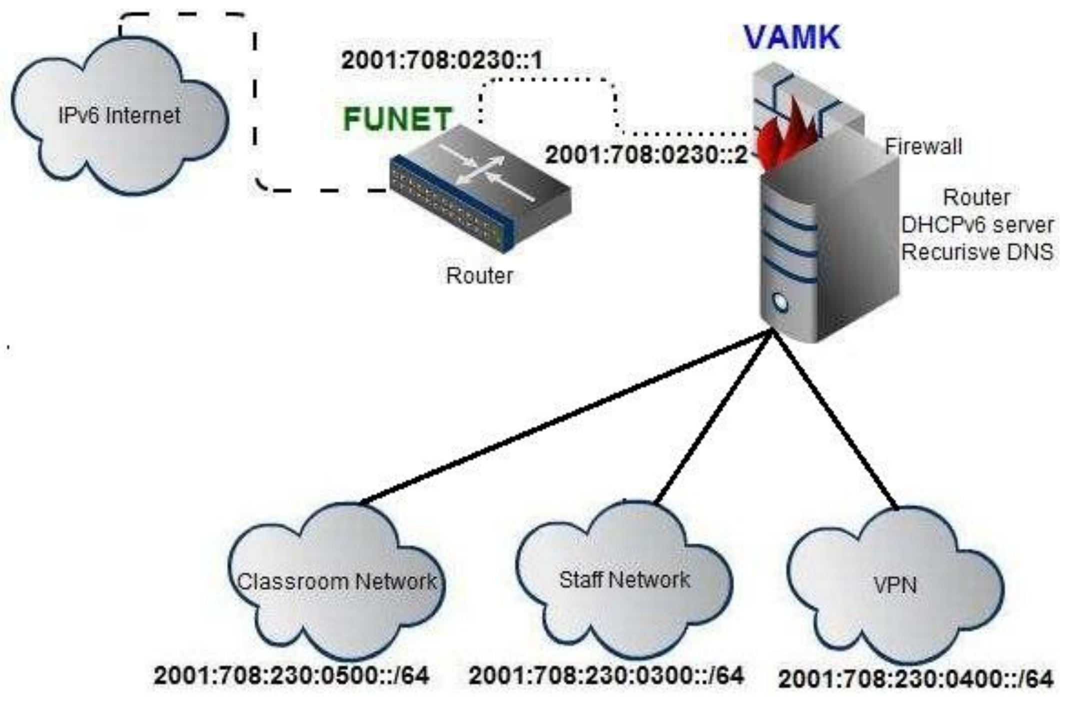


Figure 3-1. Layer 3 Network structure

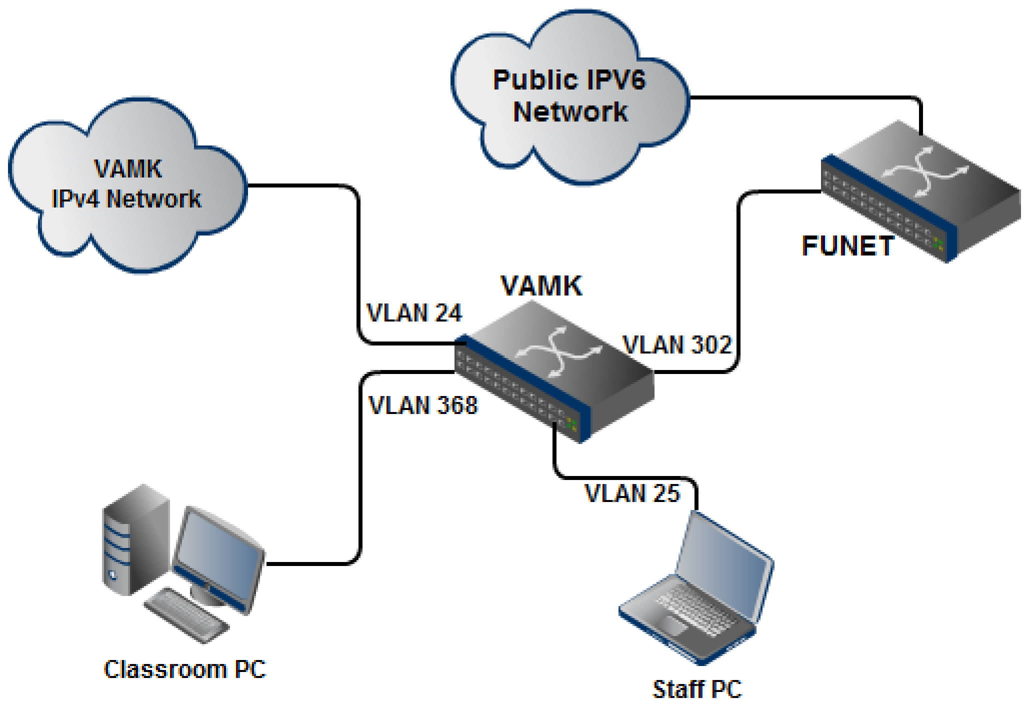


Figure 3-2. Layer 2 Network structure

Figure 3-2, shows layer 2 network structure. VLANs were used to separate network. VLANs shown in Figure 3-2 are tagged VLANs. IPv6 network is connected to VLAN 302, VAMK's IPv4 network to VLAN 24 and VLAN 368 and VLAN 25 are for classrooms and staff network respectively.

Before starting the configuration, firstly one must be root user or must provide root privilege of the FreeBSD system to configure the files and run commands and secondly, it should be noted that FreeBSD during boot time starts the services form */etc/defaults/rc.conf* which is the default setting for the system and can be overridden by the file */etc/rc.conf* which should be manually configured to meet the requirements. So, any changes to meet requirements must be performed on /*etc/rc.conf* not the */etc/defaults/rc.conf*. The system used in the project has a network interface card named "*bge0"*. VLANs were created with the network interface card named followed by the VLAN ID. For instance, VLAN for classroom network is VLAN 368 and the corresponding virtual interface created in the router is *bge0.368* which is connected to VLAN 368. This project includes the configuration of a subnet only, classroom subnet, since the remaining follows the same procedure. Table 3-2 shows the minimal system requirements to install FreeBSD/i368 and that of system being used.

	RAM	*Hard Drive*
FreeBSD/i368	64 MB	1.1 GB
System Used	2 GB	74.5 GB

Table 3-2. FreeBSD/i368 system requirement and actual system used

3.2 Router Configuration

FreeBSD 9 kernel supports IPv6 which can be achieved by selecting IPv6 support during the installation. Selecting IPv6 support is not enough, and needs manual configuration to activate in the interface which will create link-local address for the interface.

3.2.1 IPv6 Packets Forwarding

To enable IPv6 packet forwarding and link-local address generation the file */etc/sysctl.conf* was configured as below.

```
net.inet6.ip6.accept_rtadv=0
net.inet6.ip6.forwarding=1
net.inet6.ip6.auto_linklocal=1
```

The purposes of the configuration above are discussed below:

net.inet6.ip6.accept_rtadvd=0
By adding this line, the kernel will drop or reject all incoming Router Advertisement message. The purpose is to assign a static IPv6 address to the router.

net.inet6.ip6.forwarding=1
This is configuration for IPv6 packet forwarding. For a node which doesn’t forward packets, it is set to 0.

net.inet6.ip6.auto_linklocal=1
This will generate link-local address on the interfaces based on EUI-64 which can also be manually configured by setting this value to 0.

When the system reboots with the above settings, it will generate link-local address in the interface. For router the global unicast address is static and is manually configured in file */etc/rc.conf*. The full *rc.conf* configuration file is shown in the APPENDIX A and only the router configurations are shown and described below:

```
hostname="ip6.puv.fi"
keymap="finnish.iso.kbd"
sshd_enable="YES"
ipv6_network_interfaces="auto"
ipv6_activate_all_interfaces="YES"
ifconfig_bge0=up
```

```
vlans_bge0="302 24 368"
ifconfig_bge0_24="inet 193.166.140.9/24"
ifconfig_bge0_368_ipv6="inet6 2001:708:230:500::1"
ifconfig_bge0_302_ipv6="inet6 2001:708:230::2"
ipv6_gateway_enable="YES"
ipv6_defaultrouter="2001:708:230::1"
ipv6_route_default="default 2001:708:230::1"
```

The configuration *hostname* is used to name the local host so now on we will call our router with name *"IP6"*, *keymap* defines the keyboard layout to be used and *sshd_enable* is used to allow SSH connection to the machine.

ipv6_network_interfaces=auto
This is used to select network interfaces to be used for IPv6 and when set to "*auto*", will automatically detect the interfaces in the system. It will enable IPv6 on the interfaces.

ipv6_activate_all_interface="YES"
IPv6 will be activated to all the network interfaces. Now the interface is capable of receiving and forwarding IPv6 packets.

ifconfig_bge0=up & *vlans_bge0="302 24 368"*
The first line will activate the network interface *bge0* at boot time and the second line will create VLANs based on the given interface. If numbers are used the interface created will be based on *interface_name.number*. So the VLANs network interface created on the system are *bge0.302* for FUENT connection, *bge0.24* for VAMK IPv4 connection and *bge0.368* for classroom network. VLANs created follows IEEE 802.1Q standards.

ifconfig_bge0_24="inet 193.166.140.9/24"
Interface *bge0.24* is assigned static IPv4 address. The *inet* keyword specifies the IPv4 address family.

ifconfig_bge0_302_ipv6=”inet6 2001:708:230::2”

Interface *bge0.302* assigned a global unicast static IPv6 address. The *inet6* keyword specifies the IPv6 address family. If /64 prefix-length is to be used, then it is not necessary to specify prefix-length while assigning IPv6 address to an interface. The same applies for *bge0.368* interface.

ipv6_gateway_enable=”YES”

The local host will act as a router if set to “YES”. It will forward IPv6 packets between interfaces.

ipv6_defaultrouter=”2001:708:230::1” & *ipv6_route_default=”default IP”*

It defines the default router and default route. The second line is equivalent to defining the static route for default router case.

Furthermore, the FreeBSD kernel can tune address family policy. To prefer IPv6 over IPv4 the following configuration can be added to */etc/rc.conf* file.

```
ip6addrctl_enable="YES"
ip6addrctl_policy="ipv6_prefer"
```

3.2.2 Router Advertisement Message Configuration

The *rtadvd* is a daemon available in FreeBSD which is responsible for sending Router Advertisement message periodically or upon receiving Router Solicitation messages. The configuration file is located at */etc/rtadvd.conf.*

```
bge0.368:\
        :addrs#1:addr="2001:708:230:500::":prefixlen#64:tc=ether:raflags="o":
```

Router Advertisement messages are sent to interface *bge0.368* by *rtadvd* daemon. Items are separated by “:” and “\” can be used to concatenate the lines. Item *addr* defines the address prefix to be sent, *prefixlen* defines the prefix-length to be used and *raflags* defines the method of address autoconfiguration to be used by receiving nodes, if set to *“o”* means use other source for other non-address configuration.

The important variables not present in the configuration are *maxinterval* which defines the maximum interval in seconds to send unsolicited multicast Router Advertisement message and *mininterval* which defines the minimum interval in seconds to send unsolicited multicast Router Advertisement Message. If any items are omitted in the configuration, then the default values will be used. The default value for *maxinterval* is 1800 seconds and *mininterval* is 0.75**maxinterval*.

Also the preferred and valid lifetime for the address that the host will configure are not specified in the configuration meaning the default values will be used. The default preferred lifetime i.e. *pltime* is 604800 seconds (7 days) and valid lifetime i.e. *vltime* is 2592000 seconds (30 days).

To advertise the non-address information like DNS address using Router Advertisement message i.e. RDNSS with address autoconfiguration, *raflags* must be set to default value i.e. 0 and the configuration is shown in APPENDIX B.

To start *rtadvd* daemon during boot time, and start sending Router Advertisement message on certain interface (*bge0.368* in our case) the following lines must be added to */etc/rc.conf* file.

```
rtadvd_enable=”YES”
rtadvd_interfaces=”bge0.368”
```

To control *rtadvd* daemon manually following commands can be used:

```
service rtadvd start/stop/restart
/etc/rc.d/rtadvd start/stop/restart
```

3.3 DHCPv6 Configuration

Despite of fact that that KAME IPv6 stack is integrated in FreeBSD, KAME DHCPv6 is not available by default and can be downloaded from KAME FTP site *ftp.kame.net* in the directory */pub/kame/misc* under the name *kame-dhcp6* followed by the version number. The developers have stopped working in

DHCPv6 so it doesn’t support address assignments. It comes with 3 daemons namely DHCP6 client, server and relay. Since we are using as a server, the configuration is performed to server daemon and the file is located at */usr/local/etc/dhcp6s.conf.*

From *rtadvd* configuration, we set *raflag=”o”*, so DHCPv6 server will be used by clients for non-address information only. The configuration is shown below:

```
option domain-name-servers 2001:708:230::2;
option domain-name "ip6.puv.fi";
```

It is necessary to create *dhcp6sctlkey* file under same directory to control the behavior of DHCPv6 server. By the configuration above, DHCPv6 server will send DNS IPv6 address and domain-name upon receiving the client request. To start DHCPv6 server at boot time and start service on certain interface, following lines must be added to */etc/rc.conf* file.

```
dhcp6s_enable="YES"
dhcp6s_interface="bge0.368"
```

To control DHCPv6 server manually following commands are used:

```
service dhcp6s start/stop/restart
/usr/local/etc/rc.d dhcp6s start/stop/restart
```

3.4 DNSv6 Configuration

FreeBSD by default comes with BIND support and the latest version of BIND i.e. BIND 9 is available. Various files have to be configured for DNS to work properly and the main configuration file is */var/named/etc/namedb/named.conf*. The following changes must be made under *option* section for the BIND to listen on IPv6 which is disabled by default.

```
listen-on-v6    { any; };
```

It is also possible to specific network also as shown below:

```
listen-on-v6     { 2001:708:230::/48; };
```

Now, the BIND process is attached to the specified network. Since name server configured in this project will not act as authoritative name server, it is enough to define local host to be used inside the network only. So a *zone* file is created at the *named.conf* file as shown below:

```
zone "0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.arpa"{
        type master;
        file "/var/named/etc/namedb/v6";
};
```

As shown above the zone file points to another file */var/named/etc/namedb/v6* which is manually created and edited as shown:

```
$TTL 2d
$ORIGIN 0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.arpa.
@       IN      SOA     ip6.puv.fi.     Email-ID.edu.vamk.fi.(
3       ;serial number defines the version no. of file
3h      ;refresh time in hours
15m     ;update retry in minutes
3w      ;expiry in weeks.
3h      ;minimum in hours
)
        NS      ip6.puv.fi.
1.0.0.0.0.0.0.0 IN      PTR     localhost.
```

TTL is Time to Live, *SOA* Start of Authority, *NS* name server and *PTR* pointer. Local-host is defined for reverse mapping in this file and forward mapping can be defined in file */etc/hosts* file as follows:

```
::1                 localhost       ip6.puv.fi
```

To make name server to search local DNS files for local-host resolution we have to edit */etc/nsswitch.conf* file and add the following line:

```
hosts:          file dns
```

It is necessary to create or update the file */var/named/etc/namedb/named.root* which holds the information on root name servers needed to initialize cache of Internet domain name servers and is available from INTERNIC FTP site *ftp.interic.net* in the directory */domain* under the name *named.root*.

The resolver configuration file is located at */etc/resolv.conf* which is configured with the name server IP address and domain where the query must be forwarded or resolved as shown below:

```
search          ip6.puv.fi      puv.fi
nameserver      2001:708:230::2
```

The line *options rotate timeout:1 attempts:5* can be added at the end of *resolv.conf* file if multiple name server are present, to try different name server if one fails to reply in defined time interval and attempts. To start *named* daemon during boot time following line must be added to */etc/rc.conf* file:

```
named_enable=”YES”
```

BIND can be controlled manually through *named* daemon using commands:

```
service named start/stop/restart
/etc/rc.d/named start/stop/restart
```

3.5 PF Firewall Configuration

As mentioned earlier, rules in PF are evaluated from top to bottom and last matching rule wins, it is very important to write the rules in order. The full configuration file can be found in APPENDIX C. The configuration file is

/etc/pf.conf and the structure used in this project follows the order as described in this section. The common basic syntax rule is shown below:

pass|block in|out on <interface> [address_family] [proto <protocol_type>] from <source_IP> [port <source_port>] to <target_IP> [port <target_port>]

Where,

Interface is a network interface name; address family can be *inet* or *inet6*; protocol type defines the protocol like *icmp6, tcp, udp* etc; source IP can be IP address of source or even the range and same applies for target IP.

3.5.1 Macros

Macros are user defined variable for the sake of simplicity. It is usually used for interface name and frequently used IP address. Any references to macros in the configuration must start from *"$"* character and lines starting with *"#"* are comments and are not evaluated. It can be also used to list the services which make the configuration compact. Any item enclosed in *"{ }"* bracket is expanded into separate item or rules when the PF interprets the configuration file. Example used in this project:

```
funet_if="bge0.302"
staff_if="bge0.40"
classroom_if="bge0.368"
ipv4_if="bge0.24"
```

3.5.2 Options

In this part we define the default behavior of PF. The default policy is to block all packets, skipping loopback interface and log the packets passing through the *funet_if* i.e *bge0.302* interface as shown:

```
block all
set loginterface $funet_if
set skip on lo0
```

3.5.3 Filter Rules

Rules are configured in this part in order. Rules follow the basic syntax shown above. If *quick* word is present in the rules then no more rules are evaluated and the rule is applied immediately. In our case all the IPv4 traffic must be passed to *ipv4_if* connected to VAMK IPv4 network without any filtering as shown:

```
pass quick in $ipv4_if inet
block quick on $ipv4_if inet6
```

The rest of the configuration file follows the basic filtering rules like allowing SSH, HTTP requests and so on but there are some important aspects to be dealt with ICMPv6 since basic communication before or in some case after address configuration use ICMPv6 protocol. For instance Router Solicitation message must be passed to router and Router Advertisement message from router to clients and so on. The Table 3-3 shows some of the important ICMPv6 types supported by PF. The following shows an example to pass Router Solicitation message from LAN and Router Advertisement message out on LAN.

```
pass out on { $classroom_if, $staff_if } inet6 proto icmp6 all icmp6-type routersol
pass in on { $classroom_if, $staff_if } inet6 proto icmp6 all icmp6-type routeradv
```

Multicast messages from *ff02::/16* and *fe80::/16* must be allowed and also the packets from unspecified IP i.e. *::*. To enable or start PF at boot time and start loading rules from certain file the following line must be added to */etc/rc.conf* file.

```
pf_enable=”YES”
pf_rules=”/etc/pf.conf”
```

PF supports logging of the packets passing through which can be used later for security proofs. To enable packet logging the following lines must be added to */etc/rc.conf* file where */var/log/pflog* is also the default location to store the logged information.

pflog_enable="YES"

pflog_logfile="/var/log/pflog"

Various commands can be used for debugging purpose as shown:

pfctl –nf /etc/pf.conf	Check syntax errors of file */etc/pf.conf*
pfctl –s info	Status check
pfctl –vf /etc/pf.conf	Check the rules in expanded form without loading
pfctl –e	Enable PF
pfctl –d	Disable PF

To flush all the rules currently loaded and load entire new set of rules from the defined file (*/etc/pf.conf* in our case), the following command is used:

pfctl –F all –f /etc/pf.conf

Description	***Type***	***Code***
Echo Request	128	echoreq
Echo Reply	129	echoreq
Membership query	130	groupqry
Membership report	131	grouprep
Membership termination	132	groupterm
Router Solicitation	133	routersol
Router Advertisement	134	routeradv
Neighbor Solicitation	135	neighbrsol
Neighbor Advertisement	136	neighbradv
Redirection	137	redir

Table 3-3. Some ICMPv6 types supported by PF

3.6 Linux Host Configuration

Since UNIX-like operating system doesn’t create temporary address, manual configuration is necessary to generate and use temporary address as shown below in file */etc/sysctl.conf*.

```
net.ipv6.conf.all.use_tempaddr=2
net.ipv6.conf.default.use_tempaddr=2
```

3.7 Redundancy

A new identical machine is added to the project to build a redundant network. The new machine hostname is set to *IPng*. The new network interface cards were added for *pfsync* operation and connected using a cross over cable.

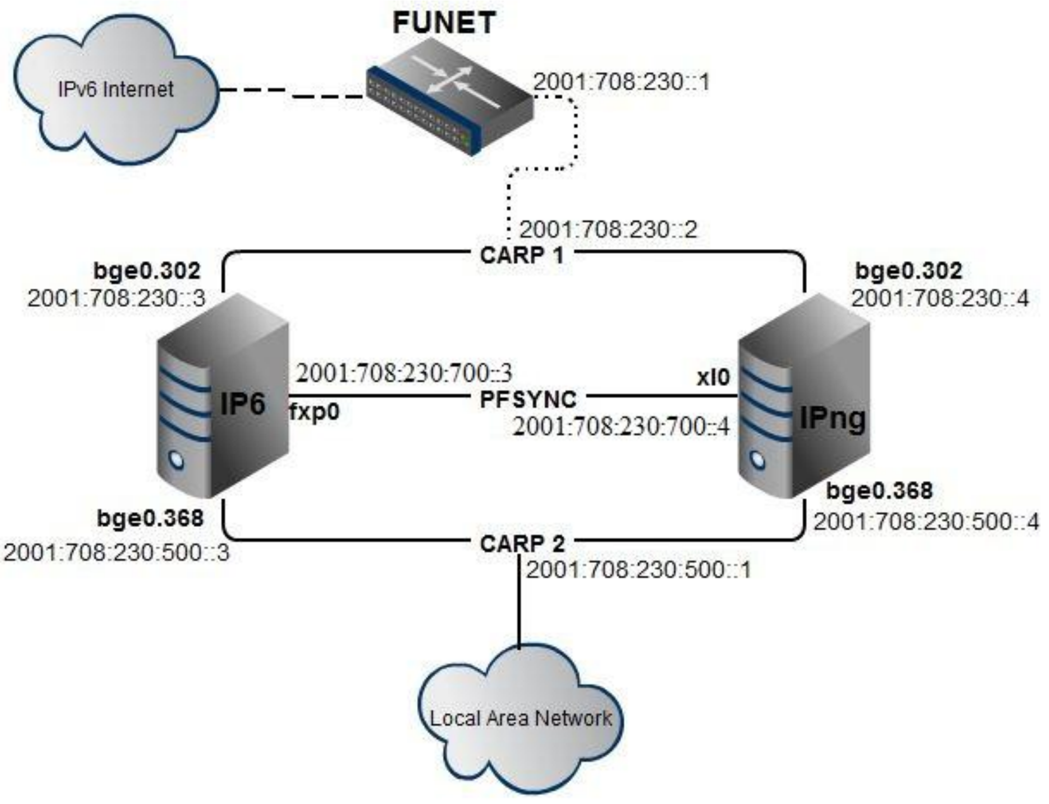


Figure 3-3. Modified network structure

From Figure 3-3, the IPv6 address assigned to *bge0.302* interface in Figure 3-1 now has been assigned to CARP virtual interface, CARP 1 and that of interface *bge0.368* to CARP 2. The physical interfaces are assigned with new IPv6 address in same subnet. Figure 3-3 shows LAN for classroom network only known from its prefix assigned to CARP 2. Similarly, more number of CARP interfaces can be created for other networks. New network adapter were added on each machine, *fxp0* on *IP6* and *xl0* on *IPng* and configured with the IPv6 address belonging to same subnet. These two interfaces were connected using cross-over cable and are only used for firewall synchronization purpose i.e. used by *pfsync* protocol.

3.7.1 Rebuilding the kernel

The default kernel *"GENERIC"* is located at */usr/src/sys/amd64/conf/* directory. The following lines must be added to the file to support CARP and *pfsense*. It is recommended to copy the original file and edit on the new one.

```
device      carp
device      pf
device      pflog
device      pfsync
```

To create a new kernel of name NEW_KERNEL and with the features included and load it during the next boot time following steps were applied.

```
cp GENERIC NEW_KERNEL
cd /usr/src
make buildkernel KERNCONF=NEW_KERNEL
make installkernel KERNCONF=NEW_KERNEL
```

After the above steps the new kernel will be copied to */boot/kernel/* directory and the old kernel is moved to */boot/kernel.old/*directory.

3.7.2 CARP Configurations

The new IPv6 address assignments to the network interfaces are not included in this configuration part but the modification required and redundancy configuration is shown in APPENDIX D for both machines. Rebuilding the kernel with CARP included is not enough; the following lines must be added to *sysctl.conf* file which allows CARP to work in the machine.

```
net.inet.carp.allow=1
net.inet.carp.preempt=1
```

The variable *net.inet.carp.allow* allows the CARP to work on the system while the variable *inet.inet.carp.preempt* forces another CARP interface to go down if one fails in case of multiple CARP interfaces so that the traffic passes through another machine in the group. The CARP configuration is almost identical on both machines. The configuration file is */etc/rc.conf*.

Configuration on both *IP6* and *IPng*:

```
cloned_interfaces="carp1 carp2"
ifconfig_carp1=up
ifconfig_carp2=up
```

The variable *cloned_interface=”carp1 carp2”* creates the clonable network interfaces on the host meaning the created interface will be cloned with other interface. The created CARP interfaces are automatically appended to network interfaces. The variable *ifconfig_carp*=up* brings the created cloned *carp** interface up. Now we can see *carp1* and *carp2* interfaces in output of *ifconfig* command. Since the IP address used in CARP interfaces are same, the configuration differs in the role i.e. how the machine will act either *master* or *backup*.

Configuration on IP6:

ifconfig_carp1_ipv6="inet6 2001:708:230::2 vhid 1 advbase 5 advskew 0 pass c1"
ifconfig_carp2_ipv6="inet6 2001:708:230:500::1 vhid 2 advbase 5 advskew 0 pass c2"

Configuration on IPng:

ifconfig_carp1_ipv6="inet6 2001:708:230::2 vhid 1 advbase 5 advskew 100 pass c1"
ifconfig_carp2_ipv6="inet6 2001:708:230:500::1 vhid 2 advbase 5 advskew 100 pass c2"

The configuration parameters are discussed below:

vhid 1
It is Virtual Host ID used to identify the redundancy group to which the CARP interface belongs to and must match the same CARP interface on both machines. As we can see the *vhid* for *carp1* interface is 1 on both machines. The *vhid* is set equal to CARP number for simplicity. The value ranges from 1 to 255. The virtual MAC address generated by CARP interface is based on *vhid*. The configured *vhid* is converted into hexadecimal format and MAC address for that CARP interface is generated as 00-00-5E-00-01-*vhid_in_hexadecimal*.

advbase 5 and *advskew 0*
The variable *abvbase* and *advskew* are used to calculate the time duration in seconds that the CARP advertisements are sent. The *advbase* value can be as low as 1 but doing so it will increase the traffic sent on the network. These variables determine the priority of being a *master*. The lower the *advskew* value the higher the priority on choosing the *master* on the redundancy group when it comes back. The default value for *advskew* is 0 and range from 0 to 254.

The *advskew* value of 0 indicates that the machine will always become *master* in the group and also takes the role of *master* upon its availability after failure.

In the configuration the *IP6* is forced to be *master* whenever it is available since the *advskew* value is 0 while *IPng* will become *master* if *IP6* is unavailable since the value is set to 100. The time interval between the two consecutive CARP advertisement is calculated by the formula *(advbase + (advskew/255))*. [14] In our configuration the results is of 5 for *IP6* and 5.39 for *IPng* meaning *IP6* advertise more frequently than *IPng* making it *master* whenever available. If the *advbase* value is same then the one that is already *master* will keep its *master* status.

pass c1
It is an authentication password used when talking with other CARP interface and must be same on all members of the group. In our configuration *c1* is the password for *carp1* interface and *c2* for *carp2* interface. The *vhid* and *pass* variable determines the *carp** interface on other machine with which it should work.

Depending upon the prefix the *carp1* interface is automatically attached to *bge0.302* interface and *carp2* to *bge0.368* interface.

3.7.3 PFSYNC Configuration

The *pfsync* configuration is same on both firewalls and assignment of IP address to the interface is not required for *pfsync* to work on that interface. The configuration file is */et/rc.conf*. We have two different name *fxp0* and *xl0* which can be named one by using following variable in *rc.conf* file.

```
ifconfig_xl0_name=”fxp0”
```

The *pfsync* configuration for both firewalls is shown below.

```
ifconfig_fxp0=”up”
pfsync_enable="YES"
ifconfig_pfsync0=”syncdev fxp0"
```

The first variable brings the *fxp0* interface up and the second variable activates the *pfsync* protocol on the kernel when the system starts and creates the new *pfsync0* interface on the machine.

ifconfig_pfsync0=”syncdev fxp0”
This variable specifies the name of the network interface through which *pfsync* must operate. All the *pfsync* updates are sent on this link.

ifconfig_pfsync0=”IP-address-of-other-machine”
This is an optional parameter and specifies the IP address to which the updates are sent. By default the updates are sent using multicast IP address but if this variable is specified then the traffic is sent using the destination IP address specified in this variable. The interfaces must be assigned with IP address to use this parameter.

3.7.4 PF Firewall Configuration for Redundancy

New firewall rule must be added to pass CARP advertisements and *pfsync* updates. Since we are using entirely new link for *pfsync* communication only, the interface can be skipped from the filtering rules which is faster and cheaper than filtering and passing.

```
pass on { $funet_if, $classroom_if } proto carp
set skip on $fxp0
```

4. TEST, RESULT AND ANALYSIS

Since Wireshark was not available in router, *tcpdump* command was used to capture the packets passing through an interface. It is also possible to specify the protocol name in the command. Due to limited page margin, the sequence number, time and length field from Wireshark are not shown in any Wireshark capture appearing in this document.

4.1 Router Test

Since router plays an important part in IPv6 world several tests were done in step-by-step mode as shown.

4.1.1 IPv6 Internet Connectivity and Router Advertisement Test

After the router configuration, the connection between *IP6* and FUNET router was tested using echo-request-reply as shown in Figure 4-1.

```
ip6# ping6 2001:708:230::1
PING6(56=40+8+8 bytes) 2001:708:230::2 --> 2001:708:230::1
16 bytes from 2001:708:230::1,  icmp_seq=0  hlim=64   time=4.836 ms
16 bytes from 2001:708:230::1,  icmp_seq=1  hlim=64   time=4.757 ms
16 bytes from 2001:708:230::1,  icmp_seq=2  hlim=64   time-4.599 ms
16 bytes from 2001:708:230::1,  icmp_seq=3  hlim=64   time-4.759 ms
```

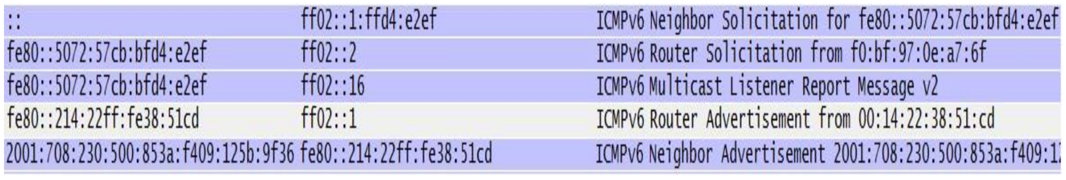
Figure 4-1. Echo Request from FUNET router

There are few IPv6 only sites available at the moment. KAME and FreeBSD official sites can run with IPv6 only internet. We can view the dancing turtle in KAME official site if used IPv6 HTTP connection. Figure 4-2 shows the echo reply from FreeBSD official site verifying the connectivity to IPv6 internet.

```
ip6# ping6 www.freebsd.org
PING6(56=40+8+8 bytes) 2001:708:230: :2 --> 2001:4f8:fff6: :22
16 bytes from 2001:4f8:fff6::22, icmp_seq=0 hlim=53 time=200.927 ms
16 bytes from 2001:4f3:fff6::22, icmp_seq=1 hlim=53 time=201.526 ms
16 bytes from 2001:4f3:fff6::22, icmp_seq=2 hlim=53 time=206.165 ms
16 bytes from 2001:4f3:fff6::22, icmp_seq=3 hlim=53 time=201.319 ms
```

Figure 4-2. Echo Reply from FreeBSD official site

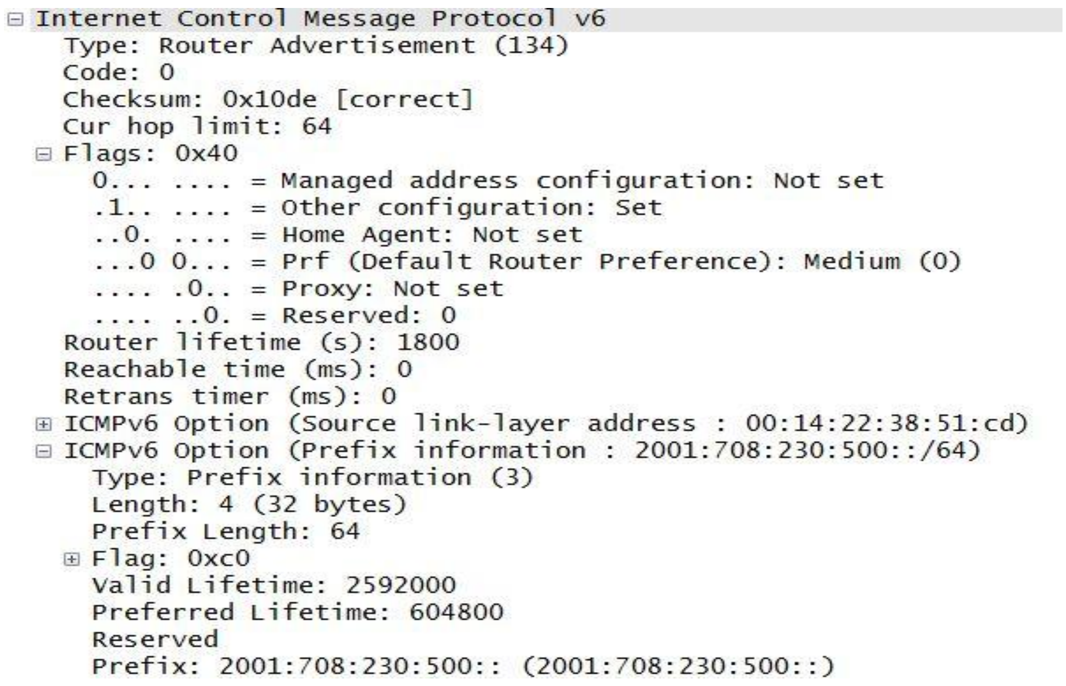
After the successful connection to IPv6 internet, the client was connected. The Figure 4-3 shows the communication.



```
::                                     ff02::1:ffd4:e2ef           ICMPv6 Neighbor Solicitation for fe80::5072:57cb:bfd4:e2ef
fe80::5072:57cb:bfd4:e2ef              ff02::2                     ICMPv6 Router Solicitation from f0:bf:97:0e:a7:6f
fe80::5072:57cb:bfd4:e2ef              ff02::16                    ICMPv6 Multicast Listener Report Message v2
fe80::214:22ff:fe38:51cd               ff02::1                     ICMPv6 Router Advertisement from 00:14:22:38:51:cd
2001:708:230:500:853a:f409:125b:9f36 fe80::214:22ff:fe38:51cd      ICMPv6 Neighbor Advertisement 2001:708:230:500:853a:f409:1
```

Figure 4-3. Link-local address generation and prefix learning

From Figure 4-3, we can see the client first sends Neighbor Solicitation message including the link-local address and configures the address if no reply for duplicate address is found. Next, the client joins the multicast group and sends Router Solicitation message. The router in response to Router Solicitation message sends Router Advertisement message containing flags set, prefix information, valid lifetime and preferred lifetime as shown in Figure 4-4. After receiving the Router Advertisement message the client configures the global unicast IPv6 address using the prefix learned and advertises its presence in the link via Neighbor Advertisement message.



```
⊟ Internet Control Message Protocol v6
    Type: Router Advertisement (134)
    Code: 0
    Checksum: 0x10de [correct]
    Cur hop limit: 64
  ⊟ Flags: 0x40
      0... .... = Managed address configuration: Not set
      .1.. .... = Other configuration: Set
      ..0. .... = Home Agent: Not set
      ...0 0... = Prf (Default Router Preference): Medium (0)
      .... .0.. = Proxy: Not set
      .... ..0. = Reserved: 0
    Router lifetime (s): 1800
    Reachable time (ms): 0
    Retrans timer (ms): 0
  ⊞ ICMPv6 Option (Source link-layer address : 00:14:22:38:51:cd)
  ⊟ ICMPv6 Option (Prefix information : 2001:708:230:500::/64)
      Type: Prefix information (3)
      Length: 4 (32 bytes)
      Prefix Length: 64
    ⊞ Flag: 0xc0
      Valid Lifetime: 2592000
      Preferred Lifetime: 604800
      Reserved
      Prefix: 2001:708:230:500:: (2001:708:230:500::)
```

Figure 4-4. Router Advertisement detailed

From Figure 4-4, we can see the prefix advertised is *2001:708:230:500::/64* with the default valid and preferred lifetime. The Other Configuration flag is set to 1 meaning the client has to ask another source for non-address related information.

4.1.2 Address Autoconfiguration Test

Figure 4-5 shows the automatic configured global unicast IPv6 address. The Figure also shows the address generated using link-local address and the randomly generated temporary address. *FE80::5072:57CB:BFD4:E2EF* is link-local IPv6 address and the corresponding global unicast IPv6 address generated is *2001:708:230:500:5072:57CB:BFD4:E2EF* and randomly generated address is *2001:708:230:500:853A:F409:125B:9F36* shown in Figure 4-5. The default gateway is the link-local address of router.

```
C:\Users\Bijay>ipconfig
Windows IP Configuration
Ethernet adapter Local Area Connection:
Connection-specific DNS Suffix . :  ad.puv.fi
IPv6 address . . . . . . . . . . . . . . . : 2001:708:230:500:5072:57cb:bfd4:e2ef
Temporary IPv6 address. . . . . .  :  2001:708:230:500:853a:f409:125b:9f36
Link-local IPv6 address . . . . . . . :  fe80::5072:57cb:bfd4:e2ef%11
Default Gateway . . . . . . . . . . . . :   fe80::214:22ff:fe38:51cd%11
```

Figure 4-5. Address Autoconfiguration

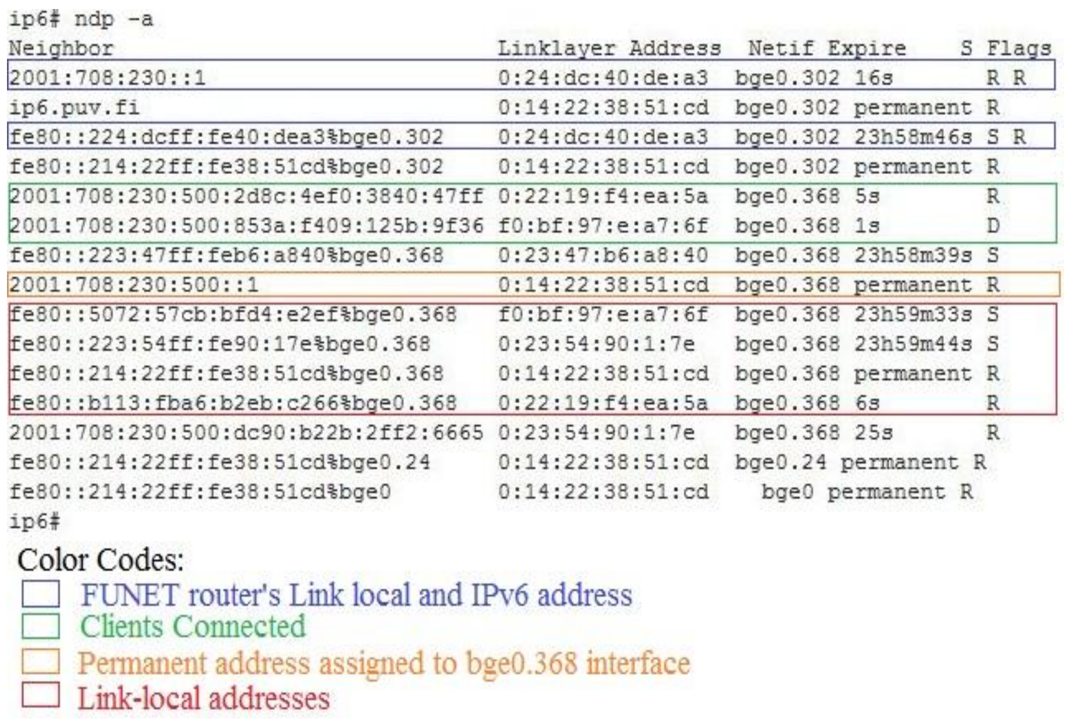
Figure 4-6 shows echo reply received from router verifying the connection between client and router.

```
C:\Users\Bijay>ping 2001:708:230::2
Pinging 2001:708:230::2 with 32 bytes of data:
Reply from 2001:708:230::2: time<1ms
Reply from 2001:708:230::2: time<1ms
Reply from 2001:708:230::2: time<1ms
Reply from 2001:708:230::2: time<1ms
Ping statistics for 2001:708:230::2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Figure 4-6. Echo Reply from router

4.1.3 Neighbor Unreachability Detection Test

The command *ndp –a* in FreeBSD shows the neighboring nodes connected. All the IPv6 running nodes have their *Neighbor cache* where neighbor status is updated.



```
ip6# ndp -a
Neighbor                             Linklayer Address  Netif Expire     S Flags
2001:708:230::1                      0:24:dc:40:de:a3  bge0.302 16s       R R
ip6.puv.fi                           0:14:22:38:51:cd  bge0.302 permanent R
fe80::224:dcff:fe40:dea3%bge0.302    0:24:dc:40:de:a3  bge0.302 23h58m46s S R
fe80::214:22ff:fe38:51cd%bge0.302    0:14:22:38:51:cd  bge0.302 permanent R
2001:708:230:500:2d8c:4ef0:3840:47ff 0:22:19:f4:ea:5a  bge0.368 5s        R
2001:708:230:500:853a:f409:125b:9f36 f0:bf:97:e:a7:6f  bge0.368 1s        D
fe80::223:47ff:feb6:a840%bge0.368    0:23:47:b6:a8:40  bge0.368 23h58m39s S
2001:708:230:500::1                  0:14:22:38:51:cd  bge0.368 permanent R
fe80::5072:57cb:bfd4:e2ef%bge0.368   f0:bf:97:e:a7:6f  bge0.368 23h59m33s S
fe80::223:54ff:fe90:17e%bge0.368     0:23:54:90:1:7e   bge0.368 23h59m44s S
fe80::214:22ff:fe38:51cd%bge0.368    0:14:22:38:51:cd  bge0.368 permanent R
fe80::b113:fba6:b2eb:c266%bge0.368   0:22:19:f4:ea:5a  bge0.368 6s        R
2001:708:230:500:dc90:b22b:2ff2:6665 0:23:54:90:1:7e   bge0.368 25s       R
fe80::214:22ff:fe38:51cd%bge0.24     0:14:22:38:51:cd  bge0.24 permanent R
fe80::214:22ff:fe38:51cd%bge0        0:14:22:38:51:cd    bge0 permanent R
ip6#
```

Color Codes:

- □ FUNET router's Link local and IPv6 address
- □ Clients Connected
- □ Permanent address assigned to bge0.368 interface
- □ Link-local addresses

Figure 4-7. Neighbors Connected to router

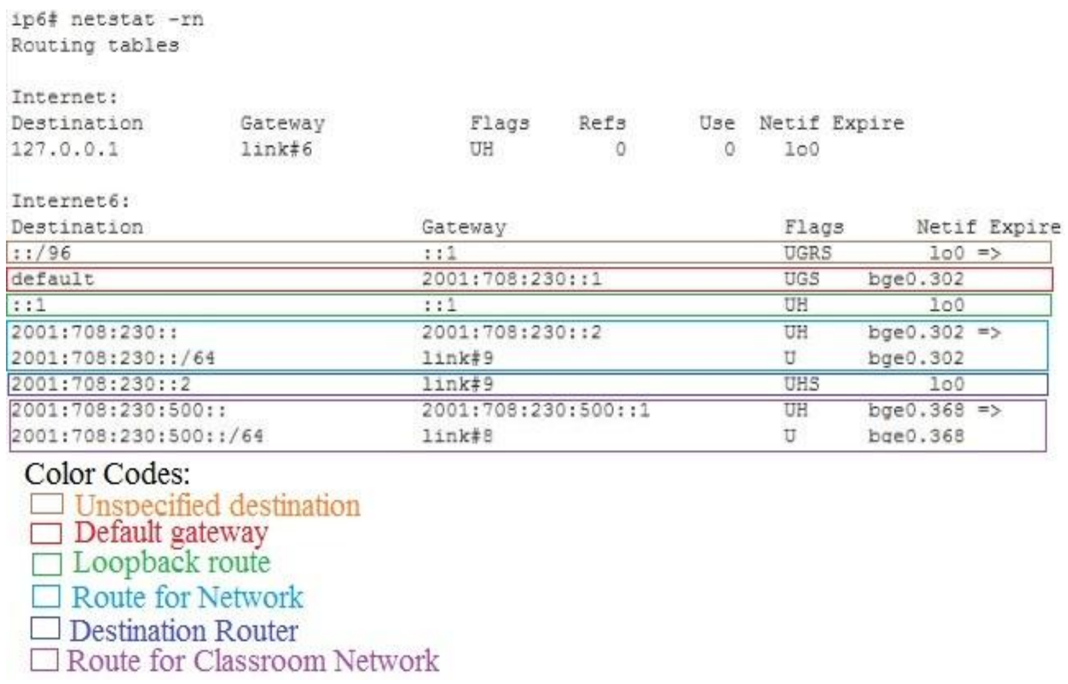
Figure 4-7 is self-explanatory where we can see the nodes connected to the router. It also shows the nature of the connection either permanent or the time to expire. Further it shows through which interface the nodes are reachable. Based on Figure 4-7, the clients connected must be reachable via *bge0.368* interface i.e. from *2001:708:230:500::1* which is shown in Figure 4-8.

```
ip6# ping6 2001:708:230:500:853a:f409:125b:9f36
PING6(56=40+8+3 bytes) 2001:708:230:500::1:--> 2001:708:230:500:853a:f409:125b:9f36
16 bytes from 2001:708:230:500:353a:f409:125b:9f36, icmp_seq=0 hlim=64 time=0.795 ms
16 bytes from 2001:708:230:500:353a:f409:125b:9f36, icmp_seq=1 hlim=64 time=0.541 ms
16 bytes from 2001:708:230:500:853a:f409:125b:9f36, icmp_seq=2 hlim=64 time=0.537 ms
^C
-–— 2001:703:230:500:353a:f409:125b:9f36 ping6 statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round—trip min/avg/max/std-dev = 0.537/0.624/0.795/0.121 ms
```

Figure 4-8. Neighbor Reachability test

4.1.4 Routing Tables

Routing tables plays the important role when delivering the packets. The basic idea behind the routing table is to maintain the better and possible shortest path for the packet flow. Figure 4-9 shows the routing table of router which can be viewed using command *netstat –rn*.

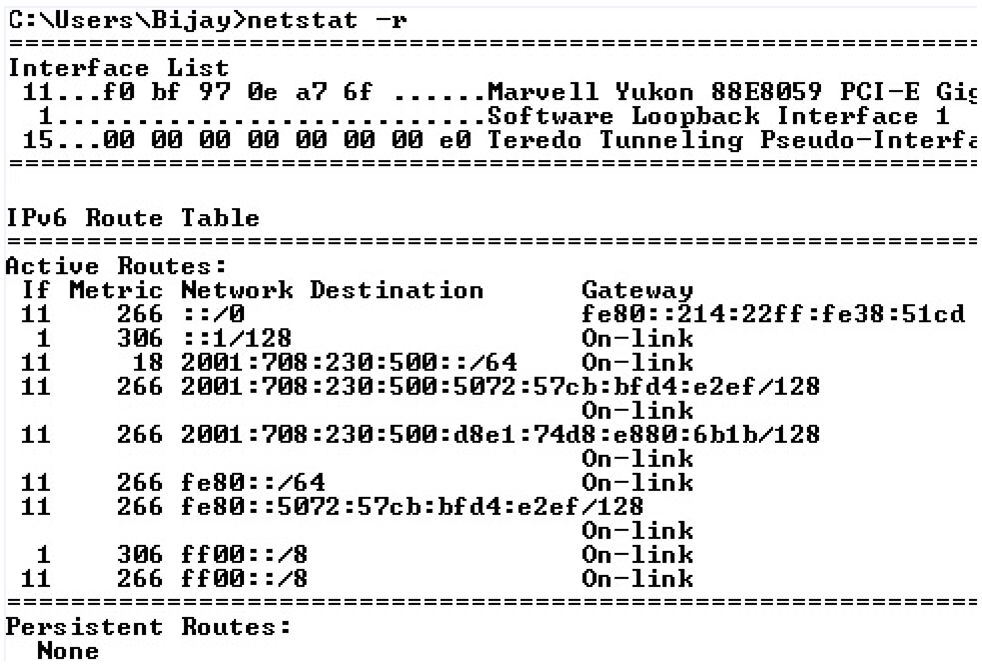


```
ip6# netstat -rn
Routing tables

Internet:
Destination        Gateway            Flags    Refs      Use  Netif Expire
127.0.0.1          link#6             UH          0        0    lo0

Internet6:
Destination                        Gateway                        Flags      Netif Expire
::/96                              ::1                            UGRS         lo0 =>
default                            2001:708:230::1                UGS      bge0.302
::1                                ::1                            UH           lo0
2001:708:230::                     2001:708:230::2                UH       bge0.302 =>
2001:708:230::/64                  link#9                         U        bge0.302
2001:708:230::2                    link#9                         UHS          lo0
2001:708:230:500::                 2001:708:230:500::1            UH       bge0.368 =>
2001:708:230:500::/64              link#8                         U        bge0.368
```

Figure 4-9. Router’s routing Table

From Figure 4-9, we can see the packets with unspecified destination are not forwarded beyond the router; default gateway for outgoing packets is FUNET router via *bge0.302* interface and the loopback route for packets originated by router to itself. We can see the gateway for entire network is *2001:708:230::2* via *bge0.302* interface and for the classroom network is *2001:708:230:500::1* via *bge0.368* interface.

Figure 4-10 shows the routing table of a client connected where we can see the packets with the unspecified destination goes to the router which is used in IP address learning stage. The second line shows the loopback route while the third line shows that the packets originated passes through the gateway of classroom network i.e. *2001:708:230:500::/64* and the status is on-link.



```
C:\Users\Bijay>netstat -r
===========================================================================
Interface List
 11...f0 bf 97 0e a7 6f ......Marvell Yukon 88E8059 PCI-E Gi
  1...........................Software Loopback Interface 1
 15...00 00 00 00 00 00 00 e0 Teredo Tunneling Pseudo-Interf
===========================================================================

IPv6 Route Table
===========================================================================
Active Routes:
 If Metric Network Destination      Gateway
 11    266 ::/0                     fe80::214:22ff:fe38:51cd
  1    306 ::1/128                  On-link
 11     18 2001:708:230:500::/64    On-link
 11    266 2001:708:230:500:5072:57cb:bfd4:e2ef/128
                                    On-link
 11    266 2001:708:230:500:d8e1:74d8:e880:6b1b/128
                                    On-link
 11    266 fe80::/64                On-link
 11    266 fe80::5072:57cb:bfd4:e2ef/128
                                    On-link
  1    306 ff00::/8                 On-link
 11    266 ff00::/8                 On-link
===========================================================================
Persistent Routes:
  None
```

Figure 4-10. Client's routing table

Till now it must be noted that the default gateway for the client is router's link-local address but not the global IPv6 address. But for the outgoing packets to the internet, global unicast IPv6 address must be used. It must also be noted that the generated link-local address for different network interface in router are the same since we are using virtual interfaces over a real physical network card. As mentioned earlier, they can be distinguished by the *"%"* notation followed by the interface name in some operating system.

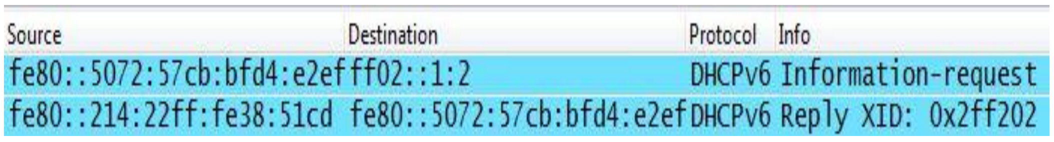
```
ip6# ping6 fe80::5072:57cb:bfd4:e2ef%bge0.368
PING6(56=40+8+8 bytes) fe80::214:22ff:fe38:51cd%bge0.368 -->
fe80::5072:57cb:bfd4:e2ef%bge0.368
16 bytes from fe80::5072:57cb:bfd4:e2ef%bge0.368, icmp_seq=0 hlim=64 time=0.633 ms
16 bytes from fe80::5072:57cb:bfd4:e2ef%bge0.368, icmp_seq=1 hlim=64 time=0.559 ms
16 bytes from fe80::5072:57cb:bfd4:e2ef%bge0.368, icmp_seq=2 hlim=64 time=0.538 ms
```

Figure 4-11. Echo request for client's link-local address

Figure 4-11 shows the echo reply in response to request done on client's link-local address followed by the interface it is attached to.

4.2 DHCPv6 Test

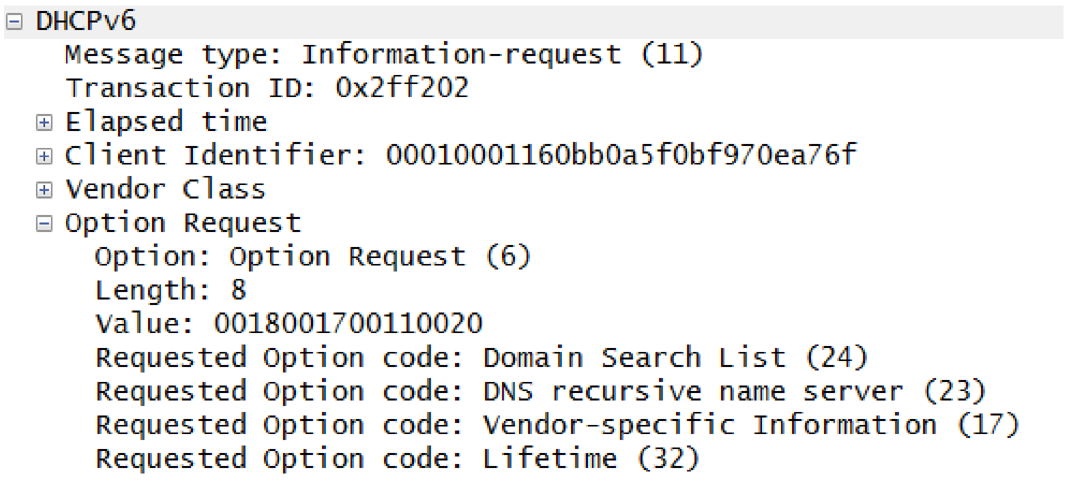
Upon receiving Router Advertisement message, the client configures IPv6 address and sends *Information-Request* message to DHCPv6 server since the Other Configuration flag in Router Advertisement message received is set to 1. Figure 4-12 shows the *Information-Request* message sent by client and server responding to that message.



```
Source                     Destination                Protocol Info
fe80::5072:57cb:bfd4:e2ef  ff02::1:2                  DHCPv6   Information-request
fe80::214:22ff:fe38:51cd   fe80::5072:57cb:bfd4:e2ef  DHCPv6   Reply XID: 0x2ff202
```

Figure 4-12. Information Request and Reply

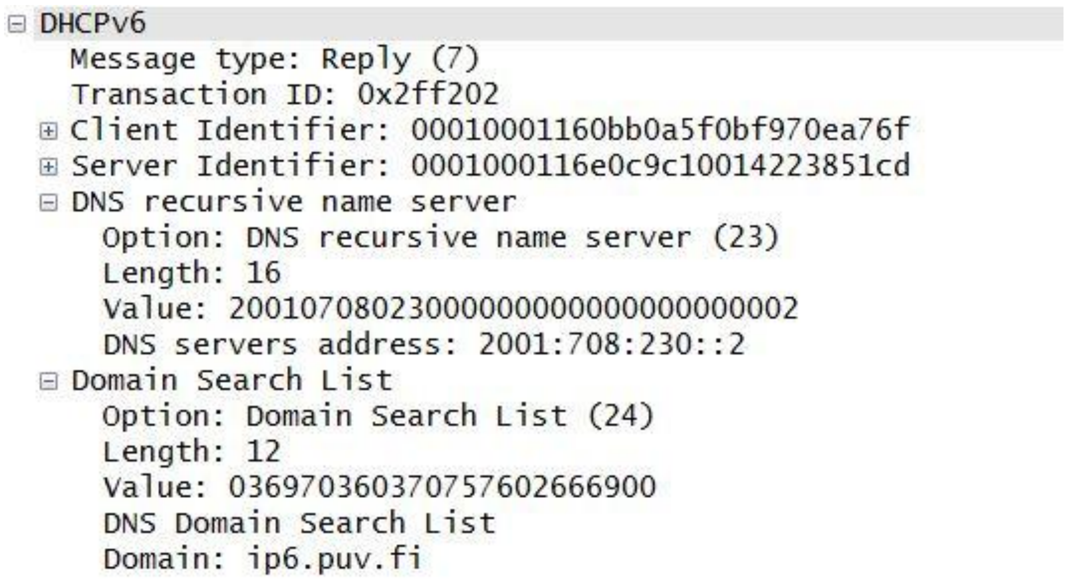
Figure 4-13 shows the information requested by the client. The information requested are Domain Search List, DNS recursive name server, Vendor-specific Information and Lifetime.



```
⊟ DHCPv6
    Message type: Information-request (11)
    Transaction ID: 0x2ff202
  ⊞ Elapsed time
  ⊞ Client Identifier: 00010001160bb0a5f0bf970ea76f
  ⊞ Vendor Class
  ⊟ Option Request
      Option: Option Request (6)
      Length: 8
      Value: 0018001700110020
      Requested Option code: Domain Search List (24)
      Requested Option code: DNS recursive name server (23)
      Requested Option code: Vendor-specific Information (17)
      Requested Option code: Lifetime (32)
```

Figure 4-13. Information-Request message

In response to the *Information-Request* message the DHCPv6 server sends *Reply* message which contains the information requested. The client identifier shown in Figure 4-13 is IA assigned by the client over the communicating interface. Figure 4-14 shows the *Reply* message from DHCPv6 server to client in response of *Information-Request* message.



```
⊟ DHCPv6
    Message type: Reply (7)
    Transaction ID: 0x2ff202
  ⊞ Client Identifier: 00010001160bb0a5f0bf970ea76f
  ⊞ Server Identifier: 0001000116e0c9c10014223851cd
  ⊟ DNS recursive name server
      Option: DNS recursive name server (23)
      Length: 16
      Value: 20010708023000000000000000000002
      DNS servers address: 2001:708:230::2
  ⊟ Domain Search List
      Option: Domain Search List (24)
      Length: 12
      Value: 036970360370757602666900
      DNS Domain Search List
      Domain: ip6.puv.fi
```

Figure 4-14. DHCPv6 Reply message

From Figure 4-14 we can see the information sent by the server which contains the DNS recursive name server IP address and Domain search list. We can also see the server identifier i.e. DUID on the message.

4.3 DNSv6 Test

DNS must be capable to translate IP address to domain name and vice-versa. The Figure 4-15 shows the forward and reverse mapping of domain *ip6.puv.fi* using *host* command.

```
ip6# host ip6.puv.fi
ip6.puv.fi has IPv6 address 2001:703:230::2

ip6# host 2001:703:230::2
2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.3.2.0.8.0.7.0.1.0.0.2.ip6.arpa domain
name pointer: ip6.puv.fi.
```

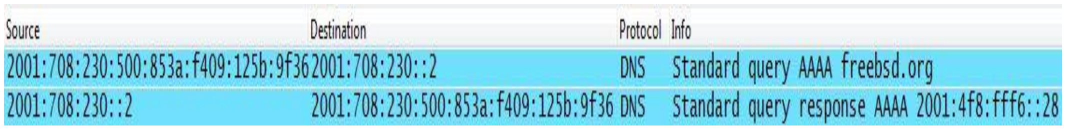
Figure 4-15. Forward and Reverse mapping of *ip6.puv.fi*

While to check name resolution by DNS, *dig* command was used. Figure 4-16 shows searching *ip6.puv.fi* for KAME name server. The answer returns the IP address of different types of name server used by KAME.

```
ip6# dig @::1 www.kame.net
; <<>> DiG 9.6.-ESV-R3 <<>> @::1 www.kame.net
; [1 server found]
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44444
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 3
;; QUESTION SECTION:
;www.kame.net.                    IN        A
;; ANSWER SECTION:
www.kame.net.    86262    IN        CNAME           orange.kame.net.
orange.kame.net.          86262    IN        A                    203.173.141.194
;; AUTHORITY SECTION:
kame.net.                 86262    IN        NS        mango.itojun.org.
kame.net.                 86262    IN        NS        orange.kame.net.
;; ADDITIONAL SECTION:
mango.itojun.org. 3462             IN        A         210.155.141.200
mango.itojun.org.         3462               IN        AAAA     2001:2f0:0:3300::1:1
mango.itojun.org.         3462               IN        AAAA     2001:2f0:0:3300:206:5bff:fe3d:940
;; Query time: 0 msec
;; SERVER: ::1#53(::1)
;; WHEN: Wed Mar 7 10:46:49 2012
;; MSG SIZE rcvd: 183
```

Figure 4-16. KAME name resolution

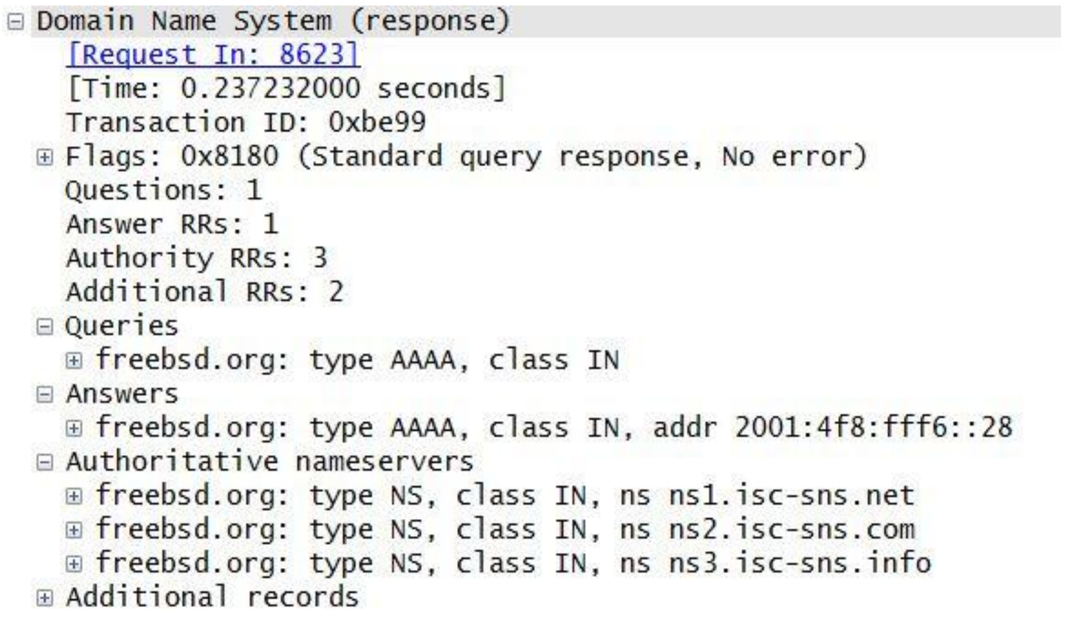
Figure 4-17 shows the query generated by client and the reply from the DNS server. We can see that the client is requesting IPv6 address i.e. AAAA type address for the given domain name. Figure 4-17 shows that the client is sending the query to the *IP6* server and the response by the server i.e. the server is searching the answer in the behalf of the clients meaning that the procedure used to query is recursive in nature.



Source	Destination	Protocol	Info
2001:708:230:500:853a:f409:125b:9f36	2001:708:230::2	DNS	Standard query AAAA freebsd.org
2001:708:230::2	2001:708:230:500:853a:f409:125b:9f36	DNS	Standard query response AAAA 2001:4f8:fff6::28

Figure 4-17. AAAA query generated by client

Figure 4-18 shows the response message in details. It shows the query was for AAAA record of FreeBSD official site followed by the answer received. Form the answer we can say that *2001:4F8:FFF6::28* holds the information for FreeBSD official site. Further we can see the response from the authoritative name servers meaning that the client is able to load the page requested. At this point the dancing turtle in KAME official site was achieved meaning the connection was made using IPv6 protocol.

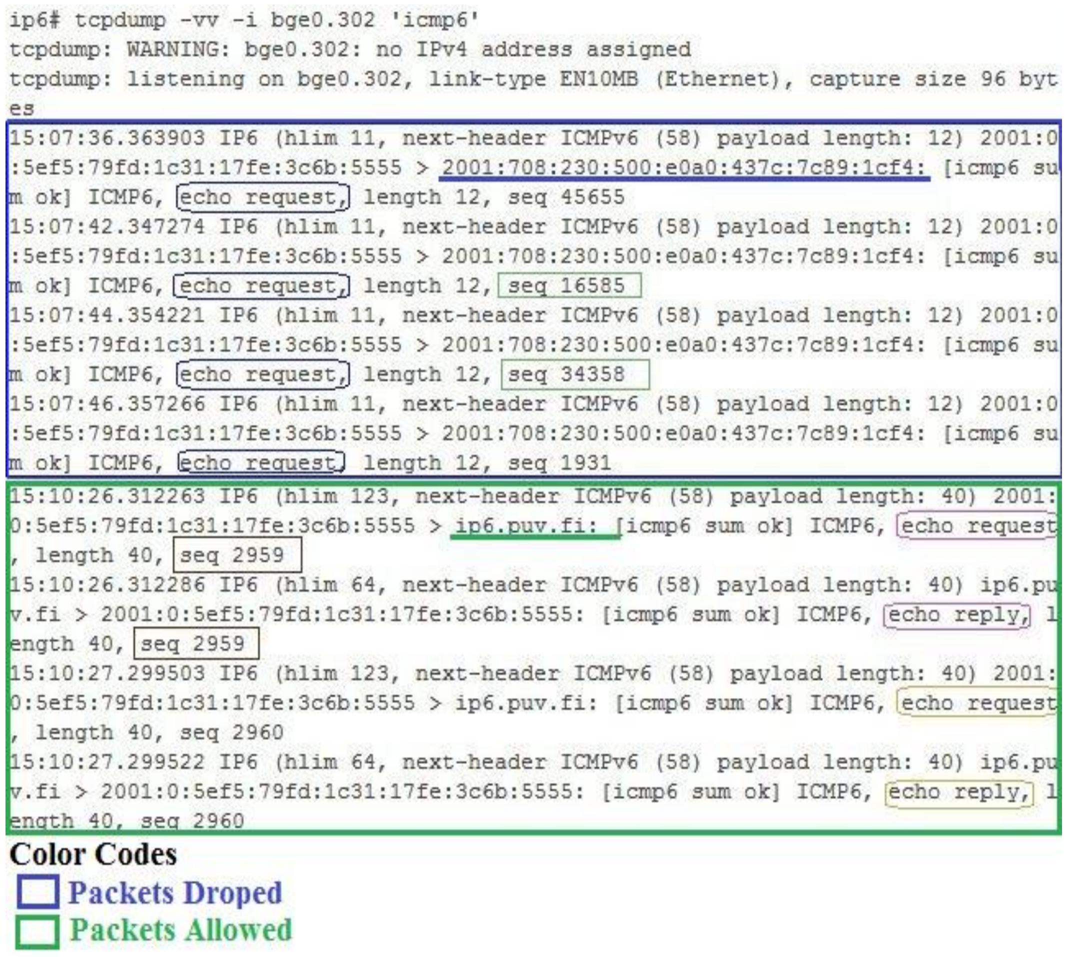


```
⊟ Domain Name System (response)
    [Request In: 8623]
    [Time: 0.237232000 seconds]
    Transaction ID: 0xbe99
  ⊞ Flags: 0x8180 (Standard query response, No error)
    Questions: 1
    Answer RRs: 1
    Authority RRs: 3
    Additional RRs: 2
  ⊟ Queries
    ⊞ freebsd.org: type AAAA, class IN
  ⊟ Answers
    ⊞ freebsd.org: type AAAA, class IN, addr 2001:4f8:fff6::28
  ⊟ Authoritative nameservers
    ⊞ freebsd.org: type NS, class IN, ns ns1.isc-sns.net
    ⊞ freebsd.org: type NS, class IN, ns ns2.isc-sns.com
    ⊞ freebsd.org: type NS, class IN, ns ns3.isc-sns.info
  ⊞ Additional records
```

Figure 4-18. DNS response

4.4 PF Firewall Test

In IPv4, NAT provides security to certain limits since all hosts have private IP address. But when it comes to IPv6 all hosts have globally unique IPv6 address and can be tracked based on the MAC address of the client which is not desirable. The Figure 4-19 shows the packets dropped and passed at interface *bge0.302*.



```
ip6# tcpdump -vv -i bge0.302 'icmp6'
tcpdump: WARNING: bge0.302: no IPv4 address assigned
tcpdump: listening on bge0.302, link-type EN10MB (Ethernet), capture size 96 byt
es
15:07:36.363903 IP6 (hlim 11, next-header ICMPv6 (58) payload length: 12) 2001:0
:5ef5:79fd:1c31:17fe:3c6b:5555 > 2001:708:230:500:e0a0:437c:7c89:1cf4: [icmp6 su
m ok] ICMP6, echo request, length 12, seq 45655
15:07:42.347274 IP6 (hlim 11, next-header ICMPv6 (58) payload length: 12) 2001:0
:5ef5:79fd:1c31:17fe:3c6b:5555 > 2001:708:230:500:e0a0:437c:7c89:1cf4: [icmp6 su
m ok] ICMP6, echo request, length 12, seq 16585
15:07:44.354221 IP6 (hlim 11, next-header ICMPv6 (58) payload length: 12) 2001:0
:5ef5:79fd:1c31:17fe:3c6b:5555 > 2001:708:230:500:e0a0:437c:7c89:1cf4: [icmp6 su
m ok] ICMP6, echo request, length 12, seq 34358
15:07:46.357266 IP6 (hlim 11, next-header ICMPv6 (58) payload length: 12) 2001:0
:5ef5:79fd:1c31:17fe:3c6b:5555 > 2001:708:230:500:e0a0:437c:7c89:1cf4: [icmp6 su
m ok] ICMP6, echo request, length 12, seq 1931
15:10:26.312263 IP6 (hlim 123, next-header ICMPv6 (58) payload length: 40) 2001:
0:5ef5:79fd:1c31:17fe:3c6b:5555 > ip6.puv.fi: [icmp6 sum ok] ICMP6, echo request
, length 40, seq 2959
15:10:26.312286 IP6 (hlim 64, next-header ICMPv6 (58) payload length: 40) ip6.pu
v.fi > 2001:0:5ef5:79fd:1c31:17fe:3c6b:5555: [icmp6 sum ok] ICMP6, echo reply, l
ength 40, seq 2959
15:10:27.299503 IP6 (hlim 123, next-header ICMPv6 (58) payload length: 40) 2001:
0:5ef5:79fd:1c31:17fe:3c6b:5555 > ip6.puv.fi: [icmp6 sum ok] ICMP6, echo request
, length 40, seq 2960
15:10:27.299522 IP6 (hlim 64, next-header ICMPv6 (58) payload length: 40) ip6.pu
v.fi > 2001:0:5ef5:79fd:1c31:17fe:3c6b:5555: [icmp6 sum ok] ICMP6, echo reply, l
ength 40, seq 2960
```

Figure 4-19. Packets dropped and allowed by PF

From Figure 4-19, we can see that the echo requests to client inside the network are dropped while echo request to router are allowed. Since PF keeps the state of a connection, we can see the sequence number *seq* in Figure 4-19 that the echo reply to the request is of same sequence number. It must be noted that only ICMPv6 packets are captured form the interface *bge0.302* in the Figure 4-19. We can also see that the decision is made on each packet. The packets matching the rule are allowed and if not dropped.

Figure 4-20 shows the state table of PF. In Figure we can see *“No ALTQ support”*. ALTQ is a feature provided by PF for traffic shaping and bandwidth allocation purpose and must be manually enabled by rebuilding the kernel. From

Figure 4-20, we can see the number of packets passed and dropped by PF both incoming and outgoing. One example of outgoing packets being dropped is Router Advertisement message since the purpose is to sever the clients inside the network only.

```
ip6# pfctl -s info
No ALTQ support in kernel
ALTQ related functions disabled
Status: Enabled for 0 days 00:39:37                  Debug: Urgent
Interface Stats for bge0.302              IPv4                IPv6
Bytes In                                     0             1239399
Bytes  Out                                   0              617723
Packets In
Passed                                       0                6143
Blocked                                      0                 122
Packets Out
Passed                                       0                 6335
Blocked                                      0                  171
```

Figure 4-20. PF state table

Figure 4-21 shows the clip from */var/log/pflog* file where we defined to store the logged information. Form Figure 4-21 we can see that Neighbor Advertisement message is passed out on *bge0.368* and the echo request to client inside the network are dropped at *bge0.302* interface. We can also see whether the packet matched the rule or not.

```
ip6# tcpdump -n -e -ttt -r /var/log/pflog l more
reading from file /var/log/pflog, link-type PFLOG (OpenBSD pflog file)
00:00:12.843566 rule 1/0(match): pass out on bge0.368: 2001:708:230:
500::1 > 2001:708:230:500:cee:8331:eacf:aba4: ICMP6, neighbor
advertisement [ |icmp6]
00:00:17.215529 rule 0/0(match): pass in on bge0.302: 2001:0:5ef5:79fd:
1c31:17fe:3c6b:5555 > 2001:708:230:500:1c72:758:12ad:2e8f: ICMP6,
echo request, seq 56686, length 12
00:00:04.934882 rule 0/0(match): pass in on bge0.302: 2001:0:5ef5:79fd:
1c31:17fe:3c6b:5555 > 2001:708:230:500:1c72:758:12ad:2e8f: ICMP6,
echo request, seq 2, length 40
```

Figure 4-21. Output from *pflog* file

4.5 Linux Host Test

Since multiple addresses are allowed in IPv6, a node can have more than one IPv6 address at a time. Figure 4-22 shows the output of *ifconfig* command from Linux host after configured to generate temporary address. We can see two global IPv6 address in the interface *eth0*. The first one from the top is temporary generated IPv6 address with the advertised prefix while the second one is generated using *modified EUI-64* format. We can compare the hardware address and link-local address with the global address to verify that the address generated is from the hardware address of the network interface. The use of temporary address depends upon the preference in the configuration. For our configuration above temporary address is preferred over the address generated using network interface hardware address. Now it is possible to reach the host globally using any of the global address configured.

```
user@teu-901:~$ ifconfig
eth0      Link encap:Ethernet Hwaddr 00:23:54:90:01:7e
          inet6 addr: 2001:708:230:500:1c72:758:12ad:2e8f/64 Scope:Global
          inet6 addr: 2001:708:230:500:223:54ff:fe90:17e/64 Scope:Global
          inet6 addr: fe80::223:54ff:fe90:17e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:1565 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1583 errors:0 dropped:0 overruns:0 carrier:2
          collisions:0 txqueuelen:1000
          RX bytes:418577 (418.5 KB) TX bytes:184013 (184.0 KB)
          Interrupt:44
```

Figure 4-22. Temporary address generation on Linux host

4.6 Redundancy Test

While testing CARP, it is not necessary to shut down or reboot the system. It can simply be tested by bringing the CARP interfaces or physical interfaces down and also by making the system offline i.e. unplugging the network connection cable.

4.6.1 CARP Test

The Figure 4-23 shows the *IP6* in *master* state and assigned IPv6 address and Figure 4-24 shows the *IPng* in *backup* state with the same IPv6 address assigned.

```
ip6# ifconfig carp1
carp1:     flags=49<UP,LOOPBACK,RUNNING> metric 0 mtu 1500
           inet6 2001:70B:230::2 prefixlen 64
           nd6 options=21<PERFORMNUD,AUTO_LINKLOCAL>
           carp: MASTER vhid 1 advbase 5 advskew 0

ip6# ifconfig carp2
carp2:     flags=49<UP,LOOPBACK,RUNNING> metric 0 mtu 1500
           inet6 2001:703:230:500::1 prefixlen 64
           nd6 options=21<PERFORMNUD,AUTO_LINKLOCAL>
           carp: MASTER vhid 2 advbase 5 advskew 0
```

Figure 4-23. *Master* state of *IP6*

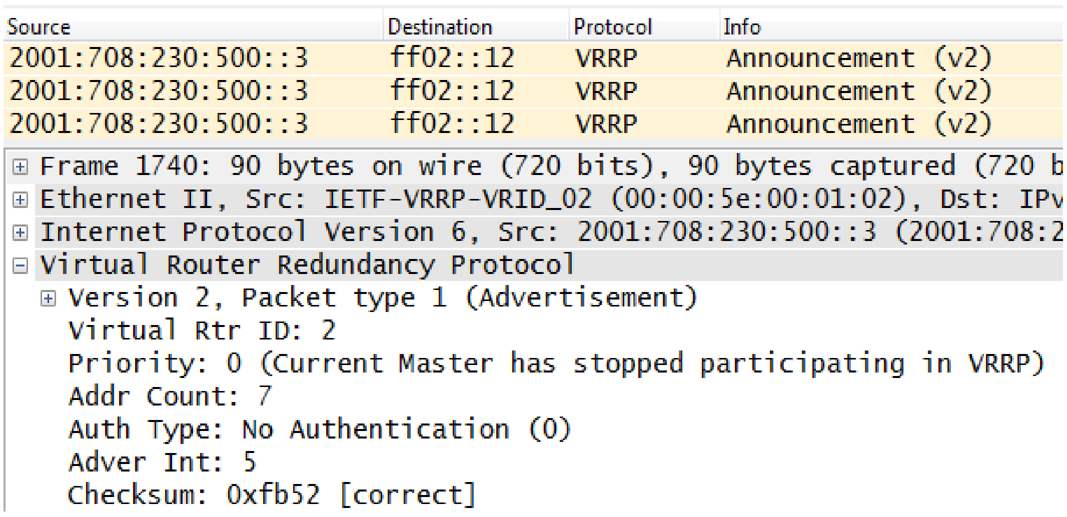
```
IPng# ifconfig carp1
carp1:      flags=49<UP,LOOPBACK,RUNNING> metric 0 mtu 1500
            inet6 2001:703:230::2 prefixlen 64
            nd6 options=21<PERFORMNUD,AUTO_LINKLOCAL>
            carp: BACKUP vhid 1 advbase 5 advskew 100

IPng# ifconfig carp2
carp2:      flags=49<UP,LOOPBACK,RUNNING> metric 0 mtu 1500
            inet6 2001:703:230:500::1 prefixlen 64
            nd6 options=21<PERFORMNUD,AUTO_LINKLOCAL>
            carp: BACKUP vhid 2 advbase 5 advskew 100
```

Figure 4-24. *Backup* state of *IPng*

```
C:\Users\Bijay>ping 2001:708:230:500::1 -t
Pinging 2001:708:230:500::1 with 32 bytes of data:
Reply from 2001:708:230:500::1: time<1ms
Reply from 2001:708:230:500::1: time<1ms
Reply from 2001:708:230:500::1: time<1ms
Request timed out.
Request timed out.
Reply from 2001:708:230:500::1: time<1ms
Reply from 2001:708:230:500::1: time<1ms
Reply from 2001:708:230:500::1: time<1ms
Reply from 2001:708:230:500::1: time<1ms
Reply from 2001:708:230:500::1: time<1ms
Reply from 2001:708:230:500::1: time<1ms
Reply from 2001:708:230:500::1: time<1ms
```

Figure 4-25. Echo-reply from virtual IPv6 address

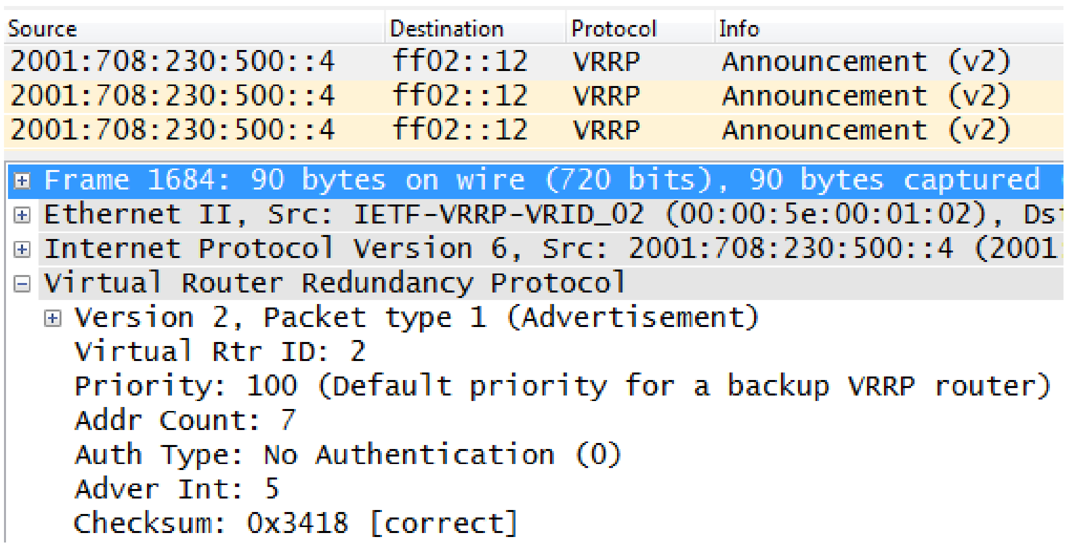
Then the gateway router addresses to FUNET is PINGed form the client and at the same time the *IP6* machine as shown in Figure 3-3 is made offline representing the failure in the network. Figure 4-25 shows the output. From Figure 4-25, we can see that the virtual IPv6 address is working fine. The service is discontinuous for a while when another machine takes over the *master* role but when the original machine, *IP6* in our case which is configured to be *master* whenever available comes back no discontinuity in the services is noticed.



```
Source                  Destination   Protocol   Info
2001:708:230:500::3     ff02::12      VRRP       Announcement (v2)
2001:708:230:500::3     ff02::12      VRRP       Announcement (v2)
2001:708:230:500::3     ff02::12      VRRP       Announcement (v2)

⊞ Frame 1740: 90 bytes on wire (720 bits), 90 bytes captured (720 b
⊞ Ethernet II, Src: IETF-VRRP-VRID_02 (00:00:5e:00:01:02), Dst: IPv
⊞ Internet Protocol Version 6, Src: 2001:708:230:500::3 (2001:708:2
⊟ Virtual Router Redundancy Protocol
  ⊞ Version 2, Packet type 1 (Advertisement)
    Virtual Rtr ID: 2
    Priority: 0 (Current Master has stopped participating in VRRP)
    Addr Count: 7
    Auth Type: No Authentication (0)
    Adver Int: 5
    Checksum: 0xfb52 [correct]
```

Figure 4-26. CARP advertisements from *master* router

Figure 4-26 shows the CARP multicast advertisements from *carp2* interface since the *Virtual Rtr ID* is 2 from which the *carp2* interface in another machine knows that *master* is still alive. The *Priority* field is 0 meaning the advertising host is taking the role of *master*. The *Priority* for a backup router is 100, as we configured which is shown in Figure 4-27.



```
Source                 Destination   Protocol   Info
2001:708:230:500::4    ff02::12      VRRP       Announcement (v2)
2001:708:230:500::4    ff02::12      VRRP       Announcement (v2)
2001:708:230:500::4    ff02::12      VRRP       Announcement (v2)

⊞ Frame 1684: 90 bytes on wire (720 bits), 90 bytes captured
⊞ Ethernet II, Src: IETF-VRRP-VRID_02 (00:00:5e:00:01:02), Ds
⊞ Internet Protocol Version 6, Src: 2001:708:230:500::4 (2001
⊟ Virtual Router Redundancy Protocol
  ⊞ Version 2, Packet type 1 (Advertisement)
    Virtual Rtr ID: 2
    Priority: 100 (Default priority for a backup VRRP router)
    Addr Count: 7
    Auth Type: No Authentication (0)
    Adver Int: 5
    Checksum: 0x3418 [correct]
```

Figure 4-27. CARP advertisements from *backup* router

4.6.1.1 Effect of CARP on Clients

Figure 4-28 shows the client configured with two default gateways after receiving Router Advertisement message from two different machine of different link-local address.

```
C:\Users\Bijay>ipconfig
Windows IP Configuration
Ethernet adapter Local Area Connection:
Connection-specific DNS Suffix  . : elisa-laajakaista.fi
IPv6 Address . . . . . . . . . . . . . . . :  2001:708:230:500:5072:57cb:bfd4:e2ef
Temporary IPv6 Address . . . . . .  : 2001:708:230:500:414f:6e64:6792:a3fc
Link-local IPv6 Address . . . . . . .  : fe80::5072:57cb:bfd4:e2ef%11
Default Gateway . . . . . . . . . . . . . : fe80::219:b9ff:fe0a:e976%11
                                              fe80::218:8bff:fe13:7000%11
```

Figure 4-28. Client configured with two default gateways

The purpose of CARP is to make a single redundant network which doesn’t match with the result from Figure 4-28 since the client configures two default gateways meaning the client is connected to two different networks. This is expected phenomena form IPv6 view since the client must configure the IPv6 address according to the parameters in the Router Advertisement message but form CARP point of view the client must be configured with the default gateway of virtual IP address to pass the client traffic from the virtual interface. It is not possible to configure the Router Advertisement message to be sent when the router is in *master* state only since the clients will only add a new default gateway to the routing table and keeps using the first default gateway until it is marked unreachable which is achieved by NUD process resulting temporary loss in connection and loss of packets. Figure 4-29 shows the client traffic passing through the physical interface *bge0.368* instead of *carp2*.

```
C:\Users\Bijay>tracert 2001:708:230::1
Tracing route to 2001:708:230::1 over a maximum of 30 hops
1       <1 ms   <1 ms     <1 ms     2001:708:230:500::3
2         *       *         *       Request timed out.
3         *       *         *       Request timed out.
4         *       *         *       Request timed out.

C:\Users\Bijay>tracert 2001:708:230::1
Tracing route to 2001:708:230::1 over a maximum of 30 hops
1       <1 ms   <1 ms     <1 ms     2001:708:230:500::4
2         *       *         *       Request timed out.
3         *       *         *       Request timed out.
4         *       *         *       Request timed out.
```

Figure 4-29. Traffic Flow

Figure 4-29 was captured during the failure of *master*. The client configures the first default gateway after receiving the first Router Advertisement message and updates the second gateway after receiving Router Advertisement message from another router. The client keeps on using the first default gateway until it is reachable and in case of failure after the delay of 1 to 1.5 minutes the client traffic passes through the next default gateway since the first one is marked unreachable after NUD process. From this result we can analyze that CARP configuration doesn’t make any sense since the expected behavior is to pass the traffic through the CARP interface.

So the only way to solve this problem is either to advertise Router Advertisement message on the CARP interface where the client are connected to or to advertise the CARP virtual IPv6 global address, not link-local as a default gateway through the Router Advertisement message.

```
IPng# tail -f /var/log/messages
Jul 12 14:34:47 IPng kernel: carp2: MASTER --> BACKUP (more frequent
advertisement received)
Jul 12 14:35:52 IPng kernel: fxp0: link state changed to UP
Jul 12 14:38:46 IPng rtadvd[2751]: <sock_mc_join> IPV6_JOIN_GROUP(link) on
carp2: Can't assign requested address
Jul 12 14:39:06 IPng rtadvd[2768]: <sock_mc_join> IPV6_JOIN_GROUP(link) on
carp2: Can't assign requested address
```

Figure 4-30. Router Advertisement message in CARP interface

Figure 4-30 shows that the Router Advertisement message can’t be advertise on the CARP interface. The *rtadvd* daemon doesn’t understand the CARP virtual MAC address yet.

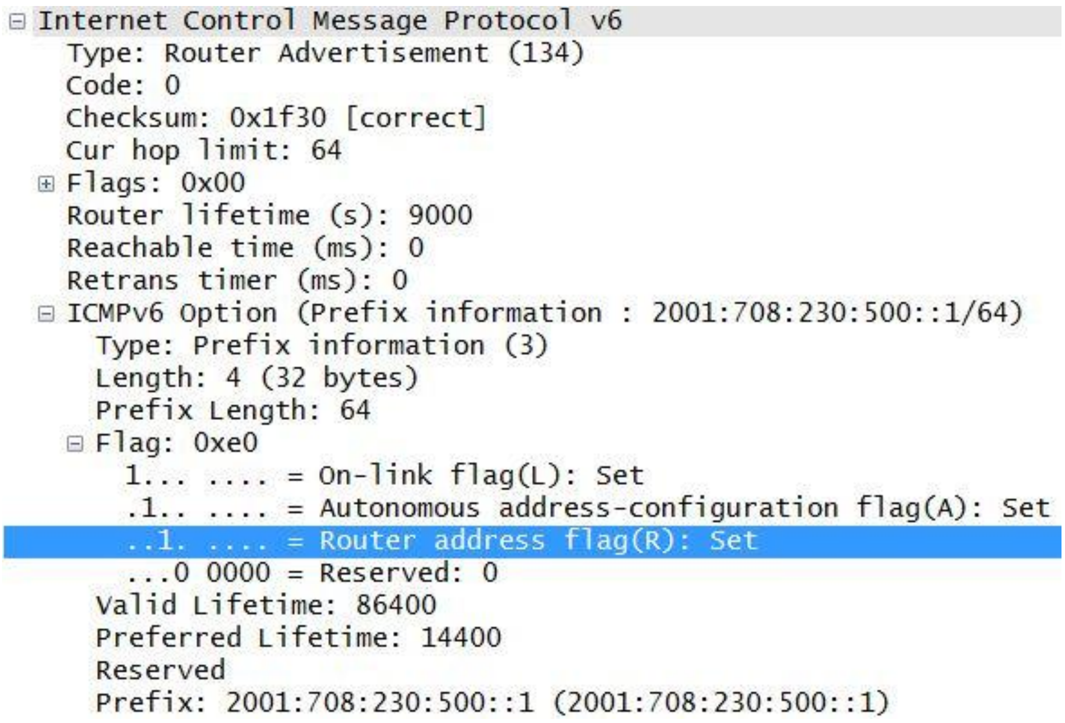
During the writing of this document the *rtadvd* daemon doesn’t support router’s global IPv6 address advertisement. There is also another daemon *radvd;* Linux Router Advertisement Daemon for Linux which can be downloaded.

4.6.1.2 Linux Router Advertisement Daemon (radvd)

The configuration file is locates at */usr/local/etc/* under file name *radvd.conf*. The location may differ according to the installation. The configuration is shown in APPENDIX E. From the configuration, *AdvRouterAddr* is set to *on* which is the parameter to advertise on-link router’s address.

From Figure 4-31, we can see that router address flag is set to 1. So *radvd* daemon is capable of advertising the router’s global IPv6 address. In the configuration, the address must be configured with the full router’s global IPv6 address not only the prefix to use this feature.

Even though the Wireshark capture shows that the router’s address is advertised, the clients connected still configure the default gateway to be the link-local address and still the traffic didn’t flow using the advertised address. The same results were captured as shown in Figure 4-28 and Figure 4-29 from which we can analyze that the Windows client doesn’t understand this parameter yet.



```
⊟ Internet Control Message Protocol v6
    Type: Router Advertisement (134)
    Code: 0
    Checksum: 0x1f30 [correct]
    Cur hop limit: 64
  ⊞ Flags: 0x00
    Router lifetime (s): 9000
    Reachable time (ms): 0
    Retrans timer (ms): 0
  ⊟ ICMPv6 Option (Prefix information : 2001:708:230:500::1/64)
      Type: Prefix information (3)
      Length: 4 (32 bytes)
      Prefix Length: 64
    ⊟ Flag: 0xe0
        1... .... = On-link flag(L): Set
        .1.. .... = Autonomous address-configuration flag(A): Set
        ..1. .... = Router address flag(R): Set
        ...0 0000 = Reserved: 0
      Valid Lifetime: 86400
      Preferred Lifetime: 14400
      Reserved
      Prefix: 2001:708:230:500::1 (2001:708:230:500::1)
```

Figure 4-31. Router Address Advertisement

From the results above, we can analyze that CARP doesn’t work properly or at all with IPv6. Despite of this fact the actual problem lies on the *rtadvd* software implemented in FreeBSD which was originally imported for KAME IPv6 stack. The *rtadvd* must be modified so that it understands the CARP virtual MAC address making it possible to advertise Router Advertisement message through the CARP interface.

4.6.2 PFSYNC Test

The Figure 4-32 shows the state of *pfsync0* interface before assigning any IP address to the physical interface *fxp0*.

```
ip6# ifconfig pfsync0
pfsync0:   flags=41<UP,RUNNING> metric 0 mtu 1500
           nd6 options=21 <PERFORMNUD, AUTO_LINKLOCAL>
           pfsync: syncdev: fxp0 syncpeer: 224.0.0.240 maxupd: 128
```

Figure 4-32. Interface *pfsync0* status

From Figure 4-32, we can see the synchronizing physical interface is *fxp0* and the synchronizing peer is the multicast address where the *pfsync* updates are sent. The *pfsync* protocol automatically assigns the IPv4 multicasts address to *syncpeer* option even though no IPv4 address is assigned to the synchronizing interface *fxp0*.

```
ip6# ifconfig pfsync0 syncpeer 2001:708:230:700::4
ifconfig: error in parsing address string: hostname nor servname provided,
or not known
```

Figure 4-33. Synchronizing using IPv6 address

Figure 4-33 shows the result while assigning IPv6 address to the *syncpeer* options. The Figure 4-33 was captured after assigning the IPv6 address to the interface *fxp0* on both machines as shown in Figure 3-3. Form the result we can analyze that *pfsync* protocol doesn't support IPv6 yet and no IPv6 multicast addresses has been reserved to advertise the updates. Since we have configure the kernel to prefer IPv6 over IPv4 explicitly the output from Figure 4-32 must have been IPv6 multicast address instead of IPv4 multicast address.

In reality, it doesn't matter whether *pfsync* protocol uses the IPv4 or IPv6 since the purpose is to synchronize the firewall state and usually separate link is used for this purpose. But using IPv6 the process can be speeded up since the minimum MTU for IPv6 is 1280 bytes more updates can be sent on the same time than that of using IPv4 thus reducing the connectionless point during the failure. Figure 4-34 shows the updates being sent and the IP protocol used is IPv4.

```
ip6# tcpdump -i fxp0
tcpdump: WARNING: fxp0: no IPv4 address assigned
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on fxp0, link-type EN10ME (Ethernet), capture size 65535 bytes
09:27:20.601634    IP   0.0.0.0 > 224.0.0.240:      pfsync 676
09:27:20.865897    IP   0.0.0.0 > 224.0.0.240:      pfsync 1453
09:27:21.602635    IP   0.0.0.0 > 224.0.0.240:      pfsync 946
09:27:22.603629    IP   0.0.0.0 > 224.0.0.240:      pfsync 714
```

Figure 4-34. *pfsync* updates

5. CONCLUSIONS

IPv6 not only solves address space shortage problem but also provides new features and possibilities. The advantages are plenty of global IP address, simplified header format, route optimization, integrated security, interoperability and autoconfiguration. Shifting to IPv6 is inevitable since many new devices are emerging which needs to be connected to network through Internet which can only be solved by extended address space in IPv6. Since we are targeting "Internet of Things", IPv6 provides the possibilities to make it happen. The vital autoconfiguration feature not only makes implementation simple but also provides "Plug and Play" feature and make renumbering of the network easy.

IPv6 supports multiple addresses assignment in an interface which makes it possible for a node to sit in more than one network and utilize the resources available at the same time. Since the two protocols IPv6 and IPv4 can work together but independently, they might co-exist for longer time than expected so it is sensible to be dual stack than IPv6-only at current situation.

Well, the challenging part of any network is its high time availability, so as we do face this challenge in this project. CARP and *pfsync* were widely used in BSD systems to build a redundant network. CARP does support IPv6 but the Router Advertisement message advertising daemons doesn't understand the virtual MAC address of CARP yet. To make CARP working, it is necessary to teach *rtadvd* or *radvd* daemons the virtual address of CARP. The *pfsync* protocol doesn't support IPv6 yet. At end, IPv6 is already mature enough and we need to upgrade the advantageous features from IPv4 to cope with IPv6 to be benefited.

REFERENCES

/1/ S. Deering and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification”, RFC 1883, December 1995. Available in www-from:
<URL: http://www.ietf.org/rfc/rfc1883>

/2/ /1/ S. Deering and R. Hinden, “IP version 6 Addressing Architecture“, RFC 4291, February 2006. Available in www-from:
<URL: http://www.ietf.org/rfc/rfc4291>

/3/ C. Huitema and B. Carpenter, “Deprecating Site Local Addresses”, RFC 3879, September 2004. Available in www-from:
<URL: http://www.ietf.org/rfc/rfc3879>

/4/ Johnson I. Agbinya, “IP Communications and Services for NGN”, CRC Press, 2010. (pg.105)

/5/ T. Narten, E. Nordmark, W. Simpson and H. Soliman, “Neighbor Discovery for IP version 6 (IPv6)”, RFC 4861, September 2007. Available in www-form:
<URL:http://tools.ietf.org/html/rfc4861>

/6/ Silvia Hagen, “IPv6 Essentials- Integrating IPv6 into Your IPv4 Network”, O’Really Media, July 2002. (pg. 56)

/7/ Youngsong Mun and Hyewon K. Lee, “Understanding IPv6”, Springer Inc., 2010. (pg. 21)

/8/ Lljitsch V. Beijnum, “Running IPv6- The Expert’s Voice in Networking”, Apress, 2006. (pg. 166)

/9/ Martin Dunmore, “6net- An IPv6 Deployment Guide”, September 2005, (pg. 325). Available in www-form:
<URL: http://www.6net.org/book/deployment-guide.pdf>

/10/ Babak Farrokhi, “Network Administration with FreeBSD 7”, Packt Publishing, April 2008. (pg. 183)

/11/ Peter N.M. Hansteen, “The Book of PF”, 2nd edition, No Starch Press, 2007. (pg. 119)

/12/ Christopher M. Buechler and Jim Pingle, “Pfsense: The Definitive Guide”, Reed Media Services, 2009. (pg. 378)

/13/ Ryan McBride, “Firewall Failover with pfsync and CARP”, Countersiege Systems Corporation, Available in www-form:
<URL: http://www.countersiege.com/doc/pfsync-carp/>

/14/ Daniele Mazzoocchio, “Redudant firewalls with OpenBSD, CARP and pfsync”, November 2010. (pg. 8)

APPENDICES

Appendix A: rc.conf

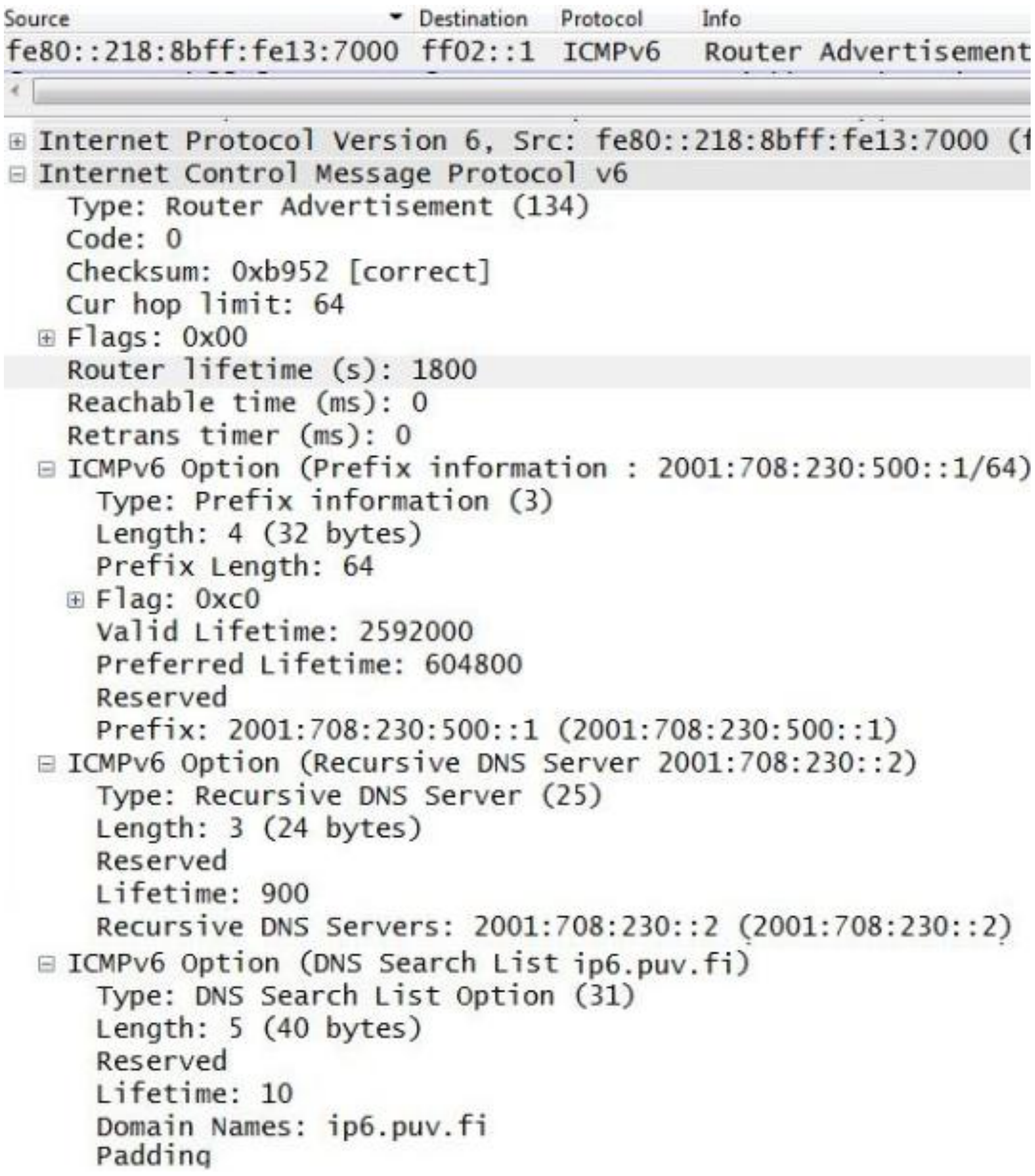
```
############ General Configurations ############
hostname="ip6.puv.fi"
keymap="finnish.iso.kbd"
zfs_enable="YES"
sshd_enable="YES"

########## IPv6 General Configurations ##########
ipv6_network_interfaces="auto"
ipv6_activate_all_interfaces="YES"
ifconfig_bge0=up
vlans_bge0="302 24 368"

############ IPv6 Address Assignment ############
ifconfig_bge0_24="inet 193.166.140.9/24"
ifconfig_bge0_368_ipv6="inet6 2001:708:230:500::1"
ifconfig_bge0_302_ipv6="inet6 2001:708:230::2"

############# Router Configuration ##############
ipv6_gateway_enable="YES"
ipv6_defaultrouter="2001:708:230::1"
ipv6_route_default="default 2001:708:230::1"

########## Router Advertisement Daemon ###########
rtadvd_enable="YES"
rtadvd_interfaces="bge0.368"

################# BIND daemon ###################
named_enable="YES"
```

```
################# PF Firewall ####################
pf_enable="YES"
pf_rules="/etc/pf.conf"
pflog_enable="YES"
pflog_logfile="/var/log/pflog"

################ DHCPv6 daemon ###################
dhcp6s_enable="YES"
dhcp6s_interface="bge0.368"

################ Protocol Preferences ###########
ip6addrctl_enable="YES"
ip6addrctl_policy="ipv6_prefer"
```

Appendix B: RDNSS Configuration on rtadvd.conf

```
bge0.368:\
        :addrs#1:addr="2001:708:230:500::":prefixlen#64:tc=ether:raflags#0:\
        :rdnss="2001:708:230::2":\
        :dnns1="ip6.puv.fi":
```

DNS options included in Router Advertisement message is shown in Figure a.



```
Source                         Destination  Protocol  Info
fe80::218:8bff:fe13:7000       ff02::1      ICMPv6    Router Advertisement

⊞ Internet Protocol Version 6, Src: fe80::218:8bff:fe13:7000 (f
⊟ Internet Control Message Protocol v6
    Type: Router Advertisement (134)
    Code: 0
    Checksum: 0xb952 [correct]
    Cur hop limit: 64
  ⊞ Flags: 0x00
    Router lifetime (s): 1800
    Reachable time (ms): 0
    Retrans timer (ms): 0
  ⊟ ICMPv6 Option (Prefix information : 2001:708:230:500::1/64)
      Type: Prefix information (3)
      Length: 4 (32 bytes)
      Prefix Length: 64
    ⊞ Flag: 0xc0
      Valid Lifetime: 2592000
      Preferred Lifetime: 604800
      Reserved
      Prefix: 2001:708:230:500::1 (2001:708:230:500::1)
  ⊟ ICMPv6 Option (Recursive DNS Server 2001:708:230::2)
      Type: Recursive DNS Server (25)
      Length: 3 (24 bytes)
      Reserved
      Lifetime: 900
      Recursive DNS Servers: 2001:708:230::2 (2001:708:230::2)
  ⊟ ICMPv6 Option (DNS Search List ip6.puv.fi)
      Type: DNS Search List Option (31)
      Length: 5 (40 bytes)
      Reserved
      Lifetime: 10
      Domain Names: ip6.puv.fi
      Padding
```

Figure a. Router Advertisement message with RDNSS

Appendix C: pf.conf

```
###Define interfaces
funet_if="bge0.302"
staff_if="bge0.25"
classroom_if="bge0.368"
ipv4_if="bge0.24"

###IP assignments or subnet declaration
vamk_net="2001:708:0230:0::/48"
staff_net="2001:708:0230:0300::/56"
classroom_net="2001:708:0230:0500::/56"
server_net="2001:708:230:0020::/64"

###Static IP assigned
ip6_server="2001:708:230::2"
classroom_ip = "2001:708:230:500::1"
staff_ip = "2001:708:230:300::1"

####Macros
icmp6_types = "{ neighbradv, neighbrsol, routersol, listqry }"
tcp_services = "{ ssh, http, www, https, ftp, smtp, pop3s, domain, ntp }"
udp_services = "{ domain, ntp }"

#####Expire state connection early
set block-policy return
set loginterface $funet_if

####Drop all packets or default deny policy
block all

####skip loopback
set skip on lo0
```

```
####pass inet to IPv4 interface and block inet6
pass quick on $ipv4_if inet
block quick on $ipv4_if inet6

####For Default Router Connected to Internet
pass in on $funet_if inet6 proto icmp6 all icmp6-type $icmp6_types keep state
pass out on $funet_if inet6 proto icmp6 all icmp6-type $icmp6_types keep state
pass out on $funet_if inet6 proto icmp6 all icmp6-type echoreq keep state

####Only Server responding to ICMP request to external interface
pass in on $funet_if inet6 proto icmp6 from any to $ip6_server icmp6-type
echoreq keep state

#####TCP and UDP to external interface
pass out on $funet_if inet6 proto tcp from any to port $tcp_services keep state
pass out on $funet_if inet6 proto udp to port $udp_services keep state
pass in on $funet_if inet6 proto tcp from any to port $tcp_services keep state
pass in on $funet_if inet6 proto udp from any to port $udp_services keep state

####Allow traceroute
pass out on $funet_if inet6 proto udp to port 33433 >< 33626

####FTP out
pass out on $funet_if inet6 proto tcp from any to any port ftp
pass out on $funet_if inet6 proto tcp from any to any port > 1023

####Allow everything in and out on local network
pass in on { $staff_if, $classroom_if } inet6 proto { tcp, udp } from { $staff_net,
$classroom_net }
pass out on { $staff_if, $classroom_if } inet6 proto { tcp, udp } from { $staff_net,
$classroom_net }
```

```
##############For clients, to obtain IPv6 address #####################
######Multicast address and link-local address in and out from LAN
pass out on { $classroom_if, $staff_if } inet6 proto icmp6 from { ::, fe80::/16,
ff02::/16 }
pass in on { $classroom_if, $staff_if } inet6 proto icmp6 from { fe80::/16,
ff02::/16 }

#######Router Solicitation from LAN
pass out on { $classroom_if, $staff_if } inet6 proto icmp6 all icmp6-type routersol

#######Router Advertisement to LAN
pass in on { $classroom_if, $staff_if } inet6 proto icmp6 all icmp6-type routeradv

######Neighbor solicitation from LAN
pass out on { $classroom_if, $staff_if } inet6 proto icmp6 all icmp6-type
neighbrsol

######Neighbor Advertisement to LAN
pass in on { $classroom_if, $staff_if } inet6 proto icmp6 all icmp6-type
neighbradv

######Information from client that have joined group
pass out on { $classroom_if, $staff_if } inet6 proto icmp6 all icmp6-type grouprep

######ND and NS after DAD
pass out on { $classroom_if, $staff_if } inet6 proto icmp6 from { $classroom_net,
$staff_net } icmp6-type neighbrsol
pass in on { $classroom_if, $staff_if } inet6 proto icmp6 from any to {
$classroom_net, $staff_net } icmp6-type neighbradv
```

```
######DHCP request and reply
pass out quick on { $staff_if, $classroom_if } inet6 proto udp from fe80::/16
pass in quick on { $staff_if, $classroom_if } inet6 proto udp from fe80::/16

######Allow ICMP reply
pass in on { $staff_if, $classroom_if } inet6 proto icmp6 from { $staff_net,
$classroom_net } icmp6-type echoreq keep state

#####Block student to access staff network
block from $classroom_net to $staff_net
pass in on $staff_if inet6 proto icmp6 from $classroom_net to $staff_net icmp6-
type echoreq keep state

#####Allow ICMP request originated by router
pass out on $classroom_if inet6 proto icmp6 from $classroom_ip to
$classroom_net icmp6-type echoreq keep state
pass out on $staff_if inet6 proto icmp6 from $staff_ip to $staff_net icmp6-type
echoreq keep state
```

Appendix D: Redundancy and Firewall Synchronization Configuration

Required Modification on IP6 (Master) rc.conf

```
############ IPv6 address assignment to Interfaces ############

ifconfig_bge0_368_ipv6="inet6 2001:708:230:500::3"
ifconfig_bge0_302_ipv6="inet6 2001:708:230::3"

##########CARP Configurations ##############
cloned_interfaces="carp1 carp2"

ifconfig_carp1=up
ifconfig_carp1_ipv6="inet6 2001:708:230::2 vhid 1 advbase 5 advskew 0 pass c1"

ifconfig_carp2=up
ifconfig_carp2_ipv6="inet6 2001:708:230:500::1 vhid 2 advbase 5 advskew 0
pass c2"

########### PFSYNC Configurations ###################
ifconfig_fxp0="up"
ifconfig_fxp0_ipv6="inet6 2001:708:230:700::3"
pfsync_enable="YES"
ifconfig_pfsync0="syncdev fxp0"
```

Required Modification on IPng (Backup) rc.conf

```
############ IPv6 address assignment to Interfaces ###############
hostname="IPng.puv.fi"
ifconfig_bge0_302_ipv6="inet6 2001:708:230::4"
ifconfig_bge0_368_ipv6="inet6 2001:708:230:500::4"

##########CARP Configurations ##############
cloned_interfaces="carp1 carp2"

ifconfig_carp1=up
ifconfig_carp1_ipv6="inet6 2001:708:230::2 vhid 1 advbase 5 advskew 100 pass
c1"

ifconfig_carp2=up
ifconfig_carp2_ipv6="inet6 2001:708:230:500::1 vhid 2 advbase 5 advskew 100
pass c2"

########### PFSYNC Configurations ###################
ifconfig_xl0_name="fxp0"
ifconfig_fxp0="up"
ifconfig_fxp0_ipv6="inet6 2001:708:230:700::4"
pfsync_enable="YES"
ifconfig_pfsync0="syncdev fxp0"
```

Appendix E: radvd.conf

```
interface bge0.368
{
        AdvSendAdvert on;
        AdvManagedFlag off;
        AdvOtherConfigFlag on;
        AdvDefaultLifetime 9000;
        MaxRtrAdvInterval 60;
        prefix 2001:708:230:500::1/64
        {
                AdvOnLink on;
                AdvAutonomous on;
                AdvRouterAddr on;
        };
}
```