

IPv6 and Multicast Routing

SOI ASIA Operators Workshop

Institut Teknologi Bandung, 16-17 August 2004
Asian Institute of Technology, 23-24 August 2004

Contents

1	Introduction to IPv6	2
2	Name Resolution	13
3	IPv6 Routing	18
4	Multicast Routing	30

Chapter 1

Introduction to IPv6

Problems of IPv4

The Internet has been using its protocol, IPv4, for more than a quarter of a century. In this development, the Internet saw its deployment found the tipping point in early 1990s with the popularity of World Wide Web. This fast pace development, however, creates problems for IPv4:

- Exhaustion of IPv4 addresses.
- Routing table explosion.
- Proliferation of NAT.

Exhaustion of IPv4 addresses. IPv4 address is 32 bits long, hence it can handle $2^{32} = 4.3$ billion hosts, which is less than the human population. With the current deployment pace, IPv4 address will be exhausted in 2008. It is difficult to get IPv4 allocation from Internet Registries these days.

Routing table explosion. IPv4 address allocation scheme does not allow effective routing information aggregation at the core of the Internet. As of July 2004, the number of prefixes in the Internet routing table has more than 130 thousand prefixes before aggregation and more than 95 thousand entries after aggregation. Routing table explosion burdens core routers, and may create instability problems and routing accidents.

Proliferation of NAT. New networks resort to use private IP addresses and Network Address Translation (NAT) mechanism because they cannot get enough IP address space. NAT breaks the end-to-end connectivity between hosts behind a NAT router and hosts on the Internet, and limits the use of some applications.

IPv6 Features

IPv6 fixes the IPv4 address exhaustion problem and several other problems related to IPv4. It also adds some improvements and features to the current IPv4 protocol, such as zero configuration and better security. Briefly, the features of IPv6 are:

- Larger address space

- New header format
- Efficient and hierarchical addressing and routing infrastructure
- Built-in security
- Better support for quality of service
- Extensibility

Larger address space. IPv6 has 128 bit address, supporting up to 3.4×10^{38} possible combinations. IPv6 will not have this many possible addresses, since it is designed for hierarchical subnetting and address allocation; however the total possible addresses in IPv6 are very large.

New header format. IPv6 header is only twice that of IPv4, even though it has four times the address size. This is achieved by streamlining the header, removing nonessential and optional fields in IPv4 header. Furthermore, IPv6 headers have boundaries in the multiples of 32 bits for faster processing.

Efficient and hierarchical addressing and routing infrastructure. The IPv6 address has multiple subnetting hierarchy, that allows aggregation at the core of the Internet. Address aggregation will result in an efficient routing at the Internet core, where routing tables will consist of only several thousand entries.

Built-in security. IPsec is included in the IPv6 protocol requirements. Therefore, every hosts have a standard mechanism to ensure secure communications.

Better support for quality of service. IPv6 has a Traffic Class and a Flow Label field to support QoS. Intermediate routers give traffic priority based on the content of Traffic Class field, while Flow Label allows router to identify and give a special handling to the packet.

Extensibility. Each IPv6 header has a Next Header field. This allows an IPv6 packet to have many headers.

IPv6 Addressing

IPv6 addresses are 128-bit identifiers of interfaces and sets of interfaces. There are three types of IPv6 addresses:

- **Unicast** An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address.
- **Anycast** An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to one of the interfaces identified by that address.
- **Multicast** An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address.

Notation

An IPv6 address is written using 8 groups of 16-bit block separated by a colon. For example, 2001:1D80:0000:3FC6:0000:0000:4AB7:5E91. 16-bit blocks whose value are 0 can be compressed using a double colon (::) to simplify the address notation with a limitation that there can be no more than one double colon in an address. Table 1.1 shows the correct and incorrect IPv6 address notation of the previous address example. Notation number 4 in the table is incorrect because it includes the zero in 1D80 to the double colon, therefore changes the address into 2001:01D8:0000:3FC6:0000:0000:4AB7:5E91.

Table 1.1: Simplifying the notation of 2001:1D80:0000:3FC6: 0000:0000:4AB7:5E91

No.	Notation	Correct
1	2001:1D80:0:3FC6::4AB7:5E91	Yes
2	2001:1D80::3FC6:0:0:4AB7:5E91	Yes
3	2001:1D80::3FC6::4AB7:5E91	No
4	2001:1D8::3FC6:0:0:4AB7:5E91	No

IPv6 also uses prefixes to identify subnets and routes, as in IPv4 CIDR (Classless Interdomain Routing). An IPv6 prefix address is written as *ipv6-address/prefix-length*. If the address in the previous example has a route prefix with length of 48 bits, the prefix is 2001:1D80::/48.

Address type identification

The type of an IPv6 address is identified by the high-order bits of the address, as in Table 1.2.

Table 1.2: Address type identification

Address type	Binary prefix	IPv6 notation
Unspecified	000..0 (128bits)	::/128
Loopback	000..1 (128bits)	::1/128
Multicast	11111111	FF00::/8
Link-local unicast	1111111010	FE80::/10
Site-local unicast	1111111011	FEC0::/10
Global unicast	(everything else)	

Unicast address

There are several types of unicast addresses in IPv6; for example global unicast, site-local unicast, and link-local unicast addresses. New address types may also be allocated in the

future.

Interface identifier. For all unicast addresses, except those that start with binary value 000, the IPv6 address structure consists of a 64-bit subnet prefix and a 64-bit interface identifier constructed by a Modified EUI-64 format. Interface identifiers must be unique for each interface on a subnet, and may be unique globally. The interface identifiers are usually taken from the interface hardware tokens, such as MAC addresses. If such tokens are not available, system administrators may configure these manually.

Local-use unicast addresses. There are two types of local-use unicast addresses. Link-local addresses are for use on a single link, and the prefix identifier is FE80::/10 and the next 54-bits are all zeros. Site-local addresses are for use in a single site. They serve as private addresses to networks that do not connect to the Internet. The prefix for site local addresses is FEC0::/10 with the next 54-bits are for subnet identifier assignments.

Unspecified address. The address 0:0:0:0:0:0:0:0 is called the unspecified address. This address indicates the absence of an address and may not be assigned to any node.

Loopback address. The unicast address 0:0:0:0:0:0:0:1 is called the loopback address. It is used by a node to send packets only to itself, and must never be assigned to any physical interface.

IPv6 addresses with embedded IPv4 addresses. IPv6 nodes uses these addresses for transition from IPv4. These addresses have IPv4 address in its low-order 32-bits and have prefix 000..0 (80bits). There are two types of such address: 1. IPv4-compatible IPv6 address; and 2. IPv4-mapped IPv6 address.

Multicast address

An IPv6 multicast address is an identifier for a group of interfaces. IPv6 does not recognize broadcast addresses, and all broadcast address functionalities in IPv4 have been replaced by multicast addresses in IPv6. An interface may belong to any number of multicast groups. There are pre-defined multicast addresses, for example reserved addresses, All Node addresses, and Solicited-Node-Addresses, that serve for particular purposes.

Anycast address

An IPv6 anycast address is an address that is assigned to more than one interface. Packets destined to an anycast address are routed to the nearest interface having the anycast address. At this moment, anycast addresses may only be assigned to IPv6 routers. An IPv6 router must recognize a subnet-router anycast address for each subnet to which they have interfaces.

A node's addresses

An IPv6 node is required to recognize the following addresses in identifying itself:

- Its required Link-Local Address for each interface.
- Any additional Unicast and Anycast Addresses that have been configured for the node's interfaces (manually or automatically).
- The loopback address.

- The All-Nodes Multicast Addresses.
- The Solicited-Node Multicast Address for each of its unicast and anycast addresses.
- Multicast Addresses of all other groups to which the node belongs.

An IPv6 router must recognize the below addresses in addition to the above addresses:

- The Subnet-Router Anycast Addresses for all interfaces for which it is configured to act as a router.
- All other Anycast Addresses with which the router has been configured.
- The All-Routers Multicast Addresses.

IPv6 and ICMPv6 Packets

IPv6 structure

An IPv6 packet consists of an IPv6 header and a payload (Figure 1.1). The IPv6 header has a fixed size of 40 bytes, and it provides the information to forward the packet hop-by-hop. The header contents is displayed in Figure 1.2. The payload may be formed by a series of extension headers followed by data. Each extension header has a Next Header field as in an IPv6 header. The Next Header field in these header informs what header will follow after the current header. If the Next Header value shows that the next header is an upper-layer protocol data unit (PDU), such as TCP, UDP, or ICMPv6, then no more extension header follows the current header.

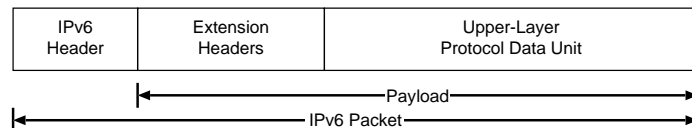


Figure 1.1: IPv6 packet structure

Version	Traffic Class	Flow label	
Payload Length		Next Header	Hop Limit
Source Address			
Destination Address			

Figure 1.2: IPv6 header structure

An IPv6 packet may have many extension headers. Extension headers is required to follow the order below:

1. Hop-by-Hop Options header
2. Destination Options header (for immediate destinations)
3. Routing header
4. Fragment header
5. Authentication header
6. Encapsulating Security Payload header
7. Destination Options header (for final destination)

An IPv6 node must support all these extension headers.

ICMPv6

The specifications of IPv6 header and extension headers are only for hop-by-hop packet processing and not for handling errors. ICMPv6 (Internet Control Message Protocol version 6) are used for these mechanisms. There are two types of ICMPv6 messages: error messages, and informational messages. Intermediate routers or the destination node send ICMPv6 error messages to report errors in forwarding or delivering IPv6 packets. In order to conserve bandwidth, error messages are rate limited, and not sent for every error. Informational messages are used to provide diagnostic functions and other host functionality, including Multicast Listener Discovery (MLD) and Neighbor Discovery (ND).

Below are the types of ICMPv6 error messages:

- Destination Unreachable (ICMPv6 Type 1)
- Packet Too Big (ICMPv6 type 2)
- Time Exceeded (ICMPv6 type 3)
- Parameter Problem (ICMPv6 type 4)

ICMPv6 has two types of informational messages for diagnostic and troubleshooting: Echo Request (ICMPv6 Type 128) and Echo Reply (ICMPv6 Type 129).

Neighbor Discovery

IPv6 Neighbor Discovery is a set of protocols related to interactions between nodes on the same link. Neighbor Discovery has five message types:

- Router Solicitation
- Router Advertisement

- Neighbor Solicitation
- Neighbor Advertisement
- Redirect

Router Solicitation and Router Advertisement messages are used for router discovery, prefix discovery, parameter discovery, and address autoconfiguration. Routers send Router Advertisement messages periodically to multicast-capable link to announce its availability. When an interface becomes enabled, it may send a Router Solicitation to request routers to send Router Advertisement messages immediately. This way, the interface may soon discover its prefixes and other related parameters.

Neighbor Solicitation messages are used to resolve link-layer address of a neighbor or to verify that a neighbor is still reachable, and to detect duplicate address. A node sends a Neighbor Advertisement in response to a Neighbor Solicitation message. In addition, a node may send an unsolicited Neighbor Advertisement to announce that its link-layer address has changed.

Routers send Redirect messages to inform hosts of a better first-hop router toward a destination. A router must ignore any Redirect messages.

Multicast Listener Discovery

MLD is a set of messages to enable each IPv6 router to discover the presence of multicast listeners on its directly attached link. MLD is an IPv6 equivalent of IGMPv2 of IPv4. It has three types of messages:

- Multicast Listener Query
- Multicast Listener Report
- Multicast Listener Done

A router uses Multicast Listener Query messages to learn which multicast addresses have listeners on an attached link and to learn if a particular multicast address has any listeners on an attached link. A listening node sends a Multicast Listener Report immediately to report its interest to a specific multicast address or to respond to a Multicast Listener Query. It also sends a Report message to an attached link periodically.

A listening node should send a Multicast Listener Done message when it ceases listening to a multicast address. When a router receives a Done message, it sends a Query message for that multicast address to confirm whether there are other listeners on the link. If there are no listeners for that multicast address, the router ceases to forward the corresponding multicast traffic to the link.

Exercise

Ex. 1 Enabling IPv6

FreeBSD

1. Log on as `root`, edit `/etc/rc.conf`, and add the below line to the file
`ipv6_enable="YES"`
2. Reboot your machine.
3. Log on as `root`.
4. At the command prompt, type:
`ifconfig -a`
 You should see that your machines' interfaces have IPv6 addresses, e.g. the ones starting with `inet6`.
5. Write the interfaces addresses below

Ethernet iface	MAC address	IPv6 address

Windows XP

Prior to Service Pack 1

1. Log on with a user account that has privileges to change network configuration.
2. Open a command prompt.
3. At the command prompt, type:
`ipv6 install`

Service Pack 1

1. Log on with a user account that has privileges to change network configuration.
2. Click **Start**, click **Control Panel**, and then double-click **Network Connections**.
3. Right-click any local area connection, and then click **Properties**.
4. Click **Install**.
5. In the **Select Network Component Type** dialog box, click **Protocol**, and then click **Add**.

6. In the **Select Network Protocol** dialog box, click **Microsoft IPv6 Developer Edition**, and then click **OK**.
7. Click **Close** to save changes to your network connection.

After installing IPv6, check your interfaces.

1. Open a command prompt.
2. At the command prompt, type:
`ipconfig`
 You should see that your machines' interfaces have IPv6 addresses.
3. Write the interfaces addresses below

Ethernet iface	IPv6 address

Ex. 2 Discovering neighbors

FreeBSD

1. At the command prompt, type:
`ndp -an`
 You should see a NDP cache table with 7 columns:

Column	Description
Neighbor	IPv6 address of neighbor
Linklayer Address	link layer address of neighbor
Netif	the network interface toward neighbor
Expire	expire time for cache entry
St	neighbor cache state; the possible states are N : no state W : wait to delete I : incomplete R : reachable S : stale D : delay P : probe
Flgs	neighbor flags; R: router; P: proxy
Prbs	num. of sent Neighbor Solicitation messages

2. Write the displayed NDP cache below.

Neighbor	Linklayer Addr.	Netif	Expire	St	Flgs	Prbs

Windows XP

- At the command prompt, type:
`netsh interface ipv6 show neighbors`
 Use `<command> | more` if you want to be able to stop your command window from scrolling.
 This command displays the NDP cache table of your Windows XP.
- Write the displayed NDP cache of the local area network interface below.

Internet Address	Physical Address	Type

Ex. 3 Ping and NDP

FreeBSD

- At the command prompt, type:
`ndp -cn` followed by: `ndp -an`
 You should see a NDP cache table containing only your host entries.
- Run `ping6` to know your neighbor. For example, to know the neighbors of your host on the Ethernet interface named `fxp0`, type:
`ping6 -c 5 ff02::1%fxp0`
 You should see ICMPv6 echo replies from neighbors, if there are any.
- Check the NDP cache again. You should see the entries of your neighbors, if there are any.
- Get to know routers on `fxp0` by typing:
`ping6 -c 5 ff02::2%fxp0`
- Ping to several hosts on your neighbor.

Windows XP

1. At the command prompt, type:
`netsh interface ipv6 delete neighbor`
to clear NDP cache.
2. Ping to several hosts on your neighbor using
`ping6 <ipv6node address>`

Ex. 4 Routing table and traceroute**FreeBSD**

1. At the command prompt, type:
`netstat -nr -f inet6`
You should see the IPv6 routing table.
2. Check the path to another (random) host, for example to 2001::
`traceroute6 -n 2001::`
What are the results? Why such results appear?

Windows XP

1. At the command prompt, type:
`netsh interface ipv6 show routes`
to see the IPv6 routing table.
2. Check the path to another (random) host, for example to 2001::
`tracert6 -d 2001::`

Ex. 5 tcpdump**FreeBSD**

As root, at the command prompt, type:

```
tcpdump -vvn 'ip6'
```

What are the packets you see on your console?

Chapter 2

Name Resolution

Domain Name System

The Domain Name System is a distributed internet directory service. It's function is to translate a domain into an IP address. DNS is used for everything that's related to communicating with a web address be it email, browsing, FTP etc.

IPv6 DNS setting

RFC1886 defines the changes that need to be made the DNS to support IPv6. The changes include a new resource record type, AAAA record.

BIND9

BIND (Berkeley Internet Name Domain) is an implementation of the DNS protocols and provides an openly redistributable reference implementation of the major components of the Domain Name System.

BIND version 9 (BIND9) is a major rewrite of nearly all aspects of the underlying BIND architecture. BIND9 can answers DNS queries on IPv6 sockets.

BIND9 Configuration Files

The BIND configuration file (the directory is usually `/etc/namedb/`) instructs the BIND name server about the zone file it is serving.

In this chapter we show configuration and zone files to enable a DNS server to resolve the below hosts.

```
DNS-server    dns.example.net
host-name     host1.example.net
prefix       2001:d30:101:/64
IP address    2001:d30:101:1::1
```

```
host-name    host2.example.net
prefix      2001:d30:101:/64
IP address  2001:d30:101:1::2
```

```
host-name    host3.example.net
prefix      2001:d30:102:/64
IP address  2001:d30:102:1::1
```

Used files are

- /etc/namedb/named.conf
- /etc/namedb/example.net
- /etc/namedb/2001:0d30.rev

Address Lookup Setting

The address lookup zone file is the most straightforward to set up. For each host or domain carrying an IPv6 address, it is simply a matter of adding a AAAA resource record.

/etc/namedb/named.conf

```
options {
    directory "/etc/namedb";
    listen-on-v6{ any; };
};

zone "example.net" {
    type master;
    file "example.net";
};
```

Zone files

The SOA Resource Record defines a zone and lists all of the information for the machines in the zone. The SOA record matches all of the hostnames in the zone to their respective IP addresses. A sample SOA is shown below:

/etc/namedb/example.net

```

example.net  IN      SOA  dns.example.net. postmaster.example.net. (
                2004071401      ; serial
                3H              ; refresh
                1H              ; retry
                1W              ; expiry
                1D )            ; minimum
                IN      NS   dns.example.net.
host1        IN      AAAA  2001:d30:101:1::1
host2        IN      AAAA  2001:d30:101:1::2
host3        IN      AAAA  2001:d30:102:1::1

```

Restart named process and check the DNS server using dig command.

```

$ dig host1.example.net AAAA
;; ANSWER SECTION:
host1.example.net. 1D IN AAAA    2001:d30:101:1::1

```

Reverse Lookup Zone Files

Similar to the forward translation of names, the reverse lookups under IPv6 have to cope with the various scopes of the addresses.

There are a few major differences in the way that domain names are used to support IPv6 reverse address lookup compared to IPv4. The first one lies in the fact that the reverse lookup domain names for IPv6 addresses are listed under the IP6.INT domain. The second one is that each digit in the address make a domain token of its own.

Below is how to set the reverse lookup .

/etc/namedb/named.conf

```

zone "0.3.d.0.1.0.0.2.ip6.int."
{
    type master;
    file "2001:0d30.rev";
};
zone "0.3.d.0.1.0.0.2.ip6.arpa."
{
    type master;
    file "2001:0d30.rev";
};

```



```
/usr/local/sbin/named -c /etc/namedb/named.conf
```

5. Use nslookup.

```
nslookup
> server localhost
> localhost
```

Confirm how it is displayed.

Ex. 2 Setup BIND

1. Setup your DNS server so it can perform address lookup and reverse lookup of the following hosts.

```
host-name    pc1.exercise.net
prefix       fec0::1/64
IP address   fec0::1
```

```
host-name    pc2.exercise.net
prefix       fec0::2/64
IP address   fec0::2
```

```
host-name    pc3.exercise.net
prefix       fec0::3/64
IP address   fec0::3
```

2. Restart named and use nslookup.

```
killall named
/usr/local/sbin/named -c /etc/namedb/named.conf
nslookup
> set type=AAAA
> server localhost
> pc1.exercise.net
.....
> pc2.exercise.net
.....
> pc3.exercise.net
.....
> fec0::1
.....
> fec0::2
.....
> fec0::3
.....
```

Confirm the results.

Chapter 3

IPv6 Routing

Overview

Routing in IPv6 works just like in IPv4. An IPv6 router must advertise itself to its attached links that it can route IPv6 packets, while in IPv4, a router doesn't have to advertise itself. Router Advertisement messages are used for this purpose. When an IPv6 router advertises itself, it advertises several information, such as Default Router Preference, Router Lifetime, etc. It also may advertise optional information, e.g. MTU, Prefix Information.

In this chapter we discuss the how to install and operate a FreeBSD-based IPv6 router. The steps to build an IPv6 router are:

1. Enable IPv6 forwarding.
2. Assign site-local and/or global addresses to the interfaces.
3. Activate Router Advertisement.
4. Populate routing table statically and/or using routing protocols.

The basic configurations are added to the `/etc/rc.conf` file.

Enabling IPv6 forwarding is the first step to build an IPv6 router. The configuration lines are:

```
ipv6_enable="YES"
ipv6_gateway_enable="YES"
```

The next step is assigning site-local and/or global IPv6 addresses to a router interfaces. You can assign an address using one of these methods:

1. Assign the first 64 bits of an IPv6 address. The interface ID part of the address will be calculated automatically. For example for interface `fxp0`.

```
ipv6_prefix_fxp0="fec0:0000:0000:0001 fec0:0000:0000:0002"
```

2. Assign the whole address. If you assign the whole address, it is better to assign a unique interface identifier for each router interface in a site.

```
ipv6_ifconfig_fxp0="fec0:0:0:5::1 prefixlen 64"
```

An IPv6 router must send Router Advertisement messages to its link. You can enable this by activating the Router Advertisement daemon using the below configuration line.

```
rtadvd_enable="YES"
```

You can limit the Router Advertisement to certain links, e.g. only to the downstream links, using the following configuration.

```
rtadvd_interfaces="fxp1"
```

You may want to populate the routing table statically. Usually you should add a default route to the routing table. The following configuration sets the default route to an IPv6 router on the fxp0 interface of the router.

```
ipv6_defaultrouter="fe80::207:e9ff:fe05:ba6f%fxp0"
```

A site is better to use routing protocols to populate the routing table. The simplest routing protocol for IPv6 is RIPng, which can be activated using the following configurations.

```
ipv6_router_enable="YES"  
ipv6_router="/usr/sbin/route6d"
```

These are the basic steps to build an IPv6 router. Next, we will explain a better routing protocol, OSPF for IPv6, and how to operate the protocol using zebra routing protocol package.

Routing protocols

Routers may run routing protocols for their operations. A routing protocol is a set of messages that defines how to exchange routing information between routers to form a routing table in routers. There are several routing protocols available for IPv6, e.g.:

- RIPng
- OSPF for IPv6 (OSPFv3)
- BGP

RIPng is a version of RIP (Routing Information Protocol) for IPv6. OSPFv3 modifies the existing OSPF for IPv4 to support IPv6. These two protocols are intra-domain gateway protocols. Border Gateway Protocol (BGP) is an inter-domain gateway protocol to exchange routing information between Autonomous Systems.

OSPF for IPv6

OSPF is a link-state routing protocol that operates between routers in a single Autonomous System. OSPF was designed to address the limitations of RIP in the supported network size. Several advantages of OSPF are: scalability, full subnetting support, and TOS routing.

OSPF works as follows. OSPF routers exchange Hello packets periodically to maintain neighbor relationships. The routers then flood the network with Link State Advertisement (LSA) using Link State Update packets. An LSA basically states who is connected with who. Using the received LSAs, each router builds a picture of the network, then calculate the shortest path to reach all subnets using the Dijkstra Shortest Path First algorithm, and the results will create a forwarding table. LSAs are sent to the network periodically and whenever there is a change on the network, e.g. link or router up and down. On such event, the routers that experience the event send Link State Update packets to the network so other routers can recalculate the shortest paths.

OSPF allows contiguous networks to be grouped together to form areas. Splitting an AS into areas is useful when there are many routers in the AS. When an AS is splitted into areas, each area has its own separate link-state database. LSAs are flooded only within an area, therefore routers in an area do not know the detailed network topology of other areas. A router may be connected to multiple areas. In this case, the router must have the same number of link-state database as the areas it is connected to. These routers are called Area Border Routers. For example, a router has an interface is in Area 0, while another is in Area 1. This router has two link-state databases. OSPF backbone is the special OSPF Area. It must be exists in a network, and other areas must be connected to the OSPF backbone.

OSPF uses link-local addresses and relies on IPv6 security (AH and ESP) for exchanging messages. Each OSPF packet contains a Router ID, a 32-bit identifier, to identify the sending router. OSPF protocol has five types of packets:

1. Hello To discover/maintain neighbors
2. Database Description To summarize database contents
3. Link State Request To download database
4. Link State Update To update database
5. Link State Ack To acknowledge database flooding

Routers send and receive Hello packets on a link to discover and maintain neighbor relationship with other routers on the link. A router sends a Hello packet periodically every HelloInterval. When a router doesn't hear a Hello packet from another router for RouterDeadInterval, then the router considers that the other router is dead.

A router will attempt to form adjacencies with some of the neighbors. On broadcast and NBMA link, routers elect a Designated Router and a Backup Designated Router. Routers on a broadcast and NBMA link should form adjacencies with these routers. Link-state databases are then synchronized between pairs of adjacent routers. The Database Description and Link State Request Packets are used in forming adjacencies. A router describe its LSA database by sending a series of Database Description packets to its neighbor. When a router sees that its neighbor has a more recent LSA, it sends a Link State Request packet to that neighbor. The neighbor will give the requested LSA using Link State Update packets, and a router will acknowledge the update by sending Link State Ack packets. Neighboring

routers are fully adjacent after their databases are synchronized.

Zebra Routing Daemon

Zebra (<http://www.zebra.org>) is a free routing software distributed under GNU General Public License. Zebra runs on several platforms, including FreeBSD. It supports IPv4 and IPv6, and several routing protocols: RIP, OSPFv2, BGP4+, RIPng, and OSPFv3. Zebra consists of routing daemons specific for each protocol and zebra the kernel routing manager. Zebra the kernel routing manager must be running for the operation of a router. Each zebra routing daemon (called zebra beast) runs independently from other daemons, so when we want to run OSPFv3, for example, we only need to run **zebra** daemon and **ospfd** daemon.

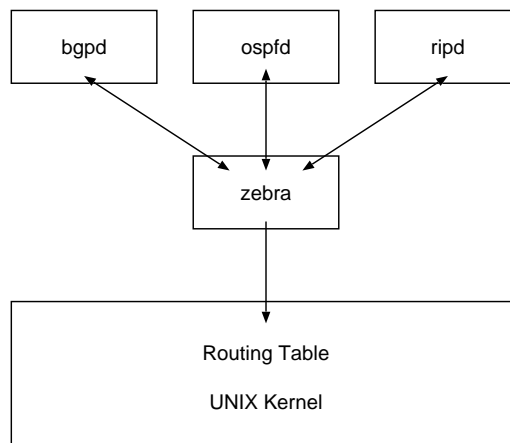


Figure 3.1: Zebra System Architecture

Zebra user interface is a command line interface (CLI). The commands are similar to those of Cisco, so people who are familiar with Cisco can easily configure Zebra. We access a zebra beast CLI by accessing a certain TCP port of zebra interface: `telnet localhost <port>`. The ports used by zebra beast are shown in Table 3.1. These ports have been added to the Well Known Port Numbers, thus we can also access the port not by the port number, but the port name.

Each Zebra beast stores its configuration in a file named according to the beast filename. For example, `ospfd.conf` is the configuration file for `ospfd`. The default directory for the configuration files is `/usr/local/etc`.

Configuring Zebra

Below is a sample of `zebra.conf` file with line numbers. This configuration is basic, but it is enough for configuring zebra to work for IPv6.

Table 3.1: Ports of Zebra beast CLI

Port name	Port number
zebrasrv	2600
zebra	2601
ripd	2602
ripngd	2603
ospfd	2604
bgpd	2605
ospf6d	2606

```

1 !
2 hostname Router
3 passwd 8 bJ0xh87QLiLbI
4 log syslog
5 service password-encryption
6 !
7 interface fxp0
8   description Ethernet to Upstream
9   multicast
10  ipv6 nd suppress-ra
11 !
12 interface fxp1
13  description Another Ethernet
14  multicast
15  ipv6 nd suppress-ra
16 !
17 ipv6 route ::/0 fe80::280:c8ff:feb9:4491 fxp0
18 !
19 access-list vty-access permit 127.0.0.1/32
20 access-list vty-access deny any
21 !
22 ipv6 access-list vty-access permit ::1/128
23 ipv6 access-list vty-access deny any
24 !
25 line vty
26  access-class vty-access
27  ipv6 access-class vty-access
28  exec-timeout 0 0
29 !

```

We now explain the above configuration file. Line 1 starts with `!`, which is just a comment line. Line 2 determines the hostname displayed when we access zebra CLI. Line 3 is the password to access zebra CLI. This line shows the encrypted password caused by Line 5. The password must not be encrypted when creating the configuration file for the first time. You can do this by removing Line 5.

Line 7–15 are for configuring interfaces. Here we have two interfaces, `fxp0` and `fxp1`, whose multicast flags are enabled. An interface should have a description for clarity. The

Router Advertisement is suppressed by `ipv6 nd suppress-ra` because we use `rtadvd` for this purpose.

Line 17 is a static route command for the default prefix (`::/0`). The routing table entry for the default prefix is usually configured in the `/etc/rc.conf` file. The next-hop for this route is `fe80::280:c8ff:feb9:4491` on the `fxp0` interface. Remember that the next-hop should be a link-local address.

Line 19–28 are to limit access to zebra CLI only from the router itself. This is an approach to secure access to the CLI.

Configuring OSPFv3

A basic OSPFv3 configuration using `ospf6d` of zebra is shown below. This is a configuration of an OSPF router having two interfaces and located in the backbone area. By default, this file name is `/usr/local/etc/ospf6d.conf`.

```
1 !
2 hostname ospf6router
3 password zebra
4 enable password zebra
5 log syslog
6 !
7 interface fxp0
8  ipv6 ospf6 cost 1
9  ipv6 ospf6 hello-interval 10
10 ipv6 ospf6 dead-interval 40
11 ipv6 ospf6 priority 10
12 !
13 interface fxp1
14 !
15 router ospf6
16  router-id 0.1.2.3
17  interface fxp0 area 0.0.0.0
18  interface fxp1 area 0.0.0.0
19 !
```

Line 2, 3, and 5 contain commands that are already shown in the sample zebra configuration. Line 4 shows the password to enter the configuration mode, which was left out in the sample zebra configuration. Both passwords on Line 3 and 4 are shown in plain text because the password encryption is not active.

Line 7–11 are the `fxp0` interface configuration. The cost to use this interface is 1. The time interval between sending Hello packets is 10 seconds, and if other routers do not hear a Hello packet from this router in 40 seconds, this router is assumed dead. These intervals must be same for all routers on a link. The priority for `fxp0` to become a Designated Router (DR) is 10; router with the highest priority will be elected as the DR. A router with priority 0 will never be a DR.

Line 13 shows the the simplest form of interface configuration for OSPFv3. Without any other parameter, `fxp1` uses the default configuration. Some of the defaults are: `hello-interval 10`, `dead-interval 40`, `cost 1`, and `priority 1`. Other parameters and the default values are available on the `ospf6d` documentation.

Line 15–18 are the OSPv3 routing configuration. Line 15 states that this router runs OSPFv3. The router ID in line 16 is a 32 bit number written in dotted-decimal notation. The router ID must be unique within an AS. Line 17–18 state that interface fxp0 and fxp1 are active, and they are in the backbone area (area ID 0.0.0.0).

Access control may also be applied to the ospf6d CLI by adding lines similar to the ones in zebra configuration.

Troubleshooting

You will certainly face with troubles in operating networks. This section provides a guide to basic routing troubleshooting for routers that use Zebra routing package. The general procedure to troubleshoot routing problems with Zebra is:

1. Direct zebra beast log to a file.
The command is: `log file <filename>`
2. Turn on related debugging messages.
For example, you want to debug packets received by zebra from zebra beasts. The command is:
`debug zebra packet recv detail`
3. Watch the log file.

Below are the outline of several symptoms and possible solution to the problems.

Routes are not installed in the kernel routing table.

1. Are the routes present in zebra routing table?
Yes: May be zebra problem; try to debug zebra or restart zebra. If the problem still persists, it may be a bug, find information from the Internet.
No: Zebra doesn't have the routes. If the routes are static, check zebra configuration. If from a routing protocol, check the corresponding zebra beast (step 2).
2. Are the routes present in zebra beast routing table?
Yes: May be an inter-process communication problem between zebra beasts. Restart the zebra beast. If fails, restart all zebra beasts.
No: Check the routing protocol states.

Problems in OSPF neighbor states.

1. No output.
Check the interfaces; the links may be disconnected or the interfaces are not enabled.
2. state = init
This state means that the router has seen Hello message from the neighbor, but the neighbor has not seen this router.
Check the firewall and the authentication type and key.

3. state = exstart or exchange
Neighbors in this state get stuck when trying to initiate database synchronization. Check the MTU and try to ping the neighbor with large packets.
4. state = loading
Router is exchanging LSA, but the packets may be corrupted. Debug the LSA packets.
5. state = two-way
Two routers that are not DR or BDR are in two-way state. If the router has Full state with the DR and BDR, then there is no problem. If there are no DR and BDR on the link, check the priority.

Missing routes in OSPF routing table.

1. Are all OSPF routes missing?
Check whether the router forms Full Adjacency with DR and BDR.
2. Are only External routes missing? Check the OSPF as-external database whether the advertising router is an AS border router.

Exercise

Ex. 1 IPv6 router basic configuration

In this exercise the instructor will show the topology and IP address assignment of the class network.

1. Write down the IP addresses assigned for your router.
ifname: _____ IP address: _____.
ifname: _____ IP address: _____.
2. Write down the default route next-hop for your router.
Next-hop: _____ ifname: _____.
3. Log on as `root`. Edit `/etc/rc.conf`. Configure to activate IPv6 router.
4. Assign the IP addresses to the network interfaces.
5. Activate the Router Advertisement daemon only to the downstream link.
6. Configure the default router.
7. Activate the RIPng routing daemon.
8. Confirm your configuration then save your changes.
9. Restart.
10. After the login prompt appears, log on as `root`.
11. Check the following on your router:

- IP addresses on interfaces
 - rtadvd and ripng process
 - ndp cache
 - routing table
12. Ping to other routers.
 13. On the Windows XP machine, check the following:
 - IP address
 - ndp cache
 - routing table
 14. On the Windows XP, ping and traceroute to other routers.

Ex. 2 Installing zebra

1. Download zebra source code.
You can download from the Zebra website <http://www.zebra.org/>
We provide a local copy in this workshop, please ask the URL to the instructor.
2. Untar the package. At the command prompt, type:

```
tar xzpf zebra-0.94.tar.gz
```
3. Run the commands below:

```
cd zebra-0.94
./configure
make
```
4. As root, install zebra by typing:

```
make install
```

Ex. 3 Configuring OSPF router with Zebra

Use the same network topology and the IP address as the Exercise 1.

1. As root, create zebra configuration file `/usr/local/etc/zebra.conf` with below configuration
 - plain text password, no enable password
 - suppress RA
 - no route
 - limit access from localhost only
2. Create ospf6d configuration file `/usr/local/etc/ospf6d.conf` with below configuration
 - plain text password, no enable password

- cost 50 for all interfaces
 - priority 1 for interface to upstream, 100 to downstream
 - router ID is the IPv4 IP address
 - area is backbone
 - limit access from localhost only
3. Confirm your configuration with others.

Ex. 4 Operating zebra

1. As root, run zebra the kernel routing manager.
`/usr/local/sbin/zebra -d`
If you don't see any error messages, zebra should be running.
2. Confirm that the zebra process is running.
If your configuration is syntactically correct, zebra should be running. If it doesn't, check the syslog for errors.
3. Access the zebra CLI, log on using the configured password.
`telnet localhost zebra`
4. At the zebra CLI, type a question mark (?).
Typing a question mark is the way to get information and help in completing your command in any zebra beast CLI.
5. See the interfaces.
`show interface`
6. See the IP and IPv6 routing tables.
`show ip route show ipv6 route`
7. Enter to the privilege mode.
`enable`
If you configure the enable password, you will be asked for a password.
8. See the start-up configuration.
`show startup-config`
9. See the running configuration.
`show running-config`
If you had changed the configuration without saving it, the running and start-up configuration would be different.
10. Enter the configuration mode.
`configure terminal`
11. Set enable password and activate password encryption.
`enable password <your password here>`
`service password-encryption`

12. Save your configuration.
`write memory`
13. Quit from zebra CLI.
14. Read the zebra configuration, see the change.

Ex. 5 Operating OSPFv3 with zebra package

1. As `root`, kill the RIPng routing daemon.
A network only need to run a single IGP unless there is a very good reason to run more then one routing protocol.
2. Run `ospf6d`.
`/usr/local/sbin/ospf6d -d`
If you don't see any error messages, `ospf6d` should be running.
3. Confirm that the `ospf6d` process is running.
If your configuration is syntactically correct, `ospf6d` should be running. If it doesn't, check the `syslog` for errors.
4. Access the `ospf6d` CLI, log on using the configured password.
`telnet localhost ospf6d`
5. Confirm that your configuration is OK.
`enable`
`show running-config`
When finished, turn off priviliged mode command.
`disable`
6. See the OSPFv3 interface information.
`show ipv6 ospf6 interface`
7. See the OSPFv3 neighbor list.
`show ipv6 ospf6 neighbor`
You should see a list of neighbors with their status. Write down the neighbor list.

RouterID	State/Duration	DR	BDR	I/F[State]

 Analyze and discuss the neighbor list.
8. See the OSPFv3 routing table.
`show ipv6 ospf6 route`
Analyze and discuss the routing table.
9. Quit `ospf6d` CLI.

10. Access zebra CLI.
11. See the IPv6 routing table.
What has been changed? Analyze and discuss.
12. Quit zebra CLI.
13. See the kernel's IPv6 routing table.

Ex. 6 Installing zebra to the FreeBSD start-up configuration

1. Remove RIPng daemon from startup configuration.
As root, edit `/etc/rc.conf`, remove the following lines.

```
ipv6_router_enable="YES"
ipv6_router="/usr/sbin/route6d"
```

Save `/etc/rc.conf`

2. Create the start-up file for zebra and ospf6d.
Create a file `/usr/local/etc/rc.d/zebra.sh`, whose content is as below.

```
#!/bin/sh

case "$1" in
start)
    /usr/local/sbin/zebra -d && echo -n ' zebra'
    /usr/local/sbin/ospf6d -d && echo -n ' ospf6d'
    ;;
stop)
    killall ospf6d && echo -n ' ospf6d'
    killall zebra && echo -n ' zebra'
    ;;
*)
    echo "Usage: 'basename $0' {start|stop}" >&2
    ;;
esac

exit 0
```

3. Change the start-up file mode into executable.
`chmod +x /usr/local/etc/rc.d/zebra.sh`
4. Confirm all your changes.
5. Reboot the FreeBSD.
6. After the FreeBSD has restarted, confirm that zebra and ospf6d are working correctly.

Chapter 4

Multicast Routing

Overview

Multicast is a many-to-many communication model in the Internet where a host send packets to a group of receivers. The intermediate routers between the sending host and receivers are responsible in forwarding packets to reach all member of the group. The main advantage of multicast over unicast is a host only needs to send a single copy of data to many receivers. The network will duplicate the data when necessary so that each receiver receives a copy of the data. The main disadvantage is multicast communication uses UDP therefore it is a best effort delivery. Reliability, congestion control, and such functions have to be handled by multicast applications.

A host join to a multicast address in order to receive multicast traffic being sent to the address. When a host wants to send traffic to a multicast address, it does not have to join to the multicast address. It just sends packets to the multicast address, and routers on the same link with the host are responsible to forward the traffic toward the receivers.

Multicast is assigned special address prefix in IPv4 and IPv6 networks. The prefix in IPv4 is 224.0.0.0/4, and in IPv6 is ff00::/8. These prefixes can only be used as the destination address in IP packets.

Multicast traffic flows from a source to receivers according to the multicast distribution tree for the traffic. The multicast distribution tree is formed by routers running a multicast routing protocol. There are two types of distribution trees:

- Source or Shortest Path Tree
The root of the tree is the multicast source. This tree is abbreviated as SPT.
- Shared Tree
Distribution trees share a root, which is called the Rendezvous Point (RP). This tree is also called Rendezvous Point Tree (RPT).

The SPT is the most efficient tree because the traffic flows from the source to the receivers using the shortest path. Meanwhile, if using RPT, multicast traffic from a source first goes to the RP then flows from the RP to the receivers using the shortest path.

The process to forward multicast traffic along the distribution tree is called multicast forwarding. Multicast forwarding is backward from unicast routing, i.e. it concerns about where a packet comes from. When a router receives a multicast packet, first it checks its

routing table whether the packet arrives at the correct interface toward the source address of the packet. This process is called Reverse Path Forwarding (RPF) Check. If the packet fails the RPF check, it is dropped. If the packet passes this check, the router forwards the packet to the outgoing interfaces, i.e. interfaces that have downstreams, not including the interface where the packet is received.

Routers use multicast routing protocols to build multicast distribution trees. The available multicast routing protocols are:

- Protocol Independent Multicast (PIM)
uses the existing unicast routing protocol to determine RPF and uses join-prune mechanism to build tree.
- Distance Vector Multicast Routing Protocol (DVMRP)
exchanges routing information to built its own routing table and uses flood-prune mechanism.
- Multicast Open Shortest Path First (MOSPF)
extends OSPF for multicast trees.
- Core Based Tree (CBT)
uses the existing unicast routing protocol to determine RPF and uses join-prune mechanism to build tree.

PIM-SM (PIM Sparse Mode) is currently de-facto standard on the Internet.

PIM-SM

PIM-SM is specified in RFC 2362. Currently PIM-SM version 2 is in the standardization process. PIM-SM is good for wide scale network, thus it is considered as de-facto standard for multicast routing protocol on the Internet.

PIM relies on the underlying system to populate its routing table, called Multicast Routing Information Base (MRIB). Routes from unicast routing table such as OSPF or from multicast address family of MBGP (Multiprotocol Border Gateway Protocol) are typically used to populate the MRIB. PIM uses the MRIB to perform RPF check and to send Join/Prune messages to create/destroy multicast distribution tree. PIM-SM builds distribution tree in three phases.

1. RP tree
2. Register stop
3. Shortest path tree

Phase one: RP tree

When a host wants to receive multicast traffic on G, the DR of the host issues a (*,G) Join message to the next hop toward the RP. A DR is a router on a link that will act on behalf of hosts on the link. An RP is a router that is configured to be used as the root of shared trees in PIM-SM. These routers must be available before PIM-SM can build multicast distribution trees. Each PIM routers along the path from the receiver to the RP

send a $(*,G)$ Join message to the correct next-hop. The RP tree is complete when $(*,G)$ Join message reaches the RP or reaches a router that already has $(*,G)$ Join state. This shared tree is rooted at the RP. A Join state has a time out, and Join messages have to be resent periodically to keep the state in the upstream routers.

When hosts on a leave network no longer want to receive traffic to the group, the DR will issue a Prune message to the upstream router. If the upstream router no longer has any downstream, it leaves the Join state and issues a Prune message to its upstream router. Prune messages are sent upward to the root of the tree, and are stopped by the first router that still has a downstream for the corresponding group.

When a source S sends traffic to a multicast group G, the Designated Router (DR) on the same link with S encapsulates the multicast packets from S, and sends the encapsulated packets to the RP as unicast packets. The encapsulated packet is called PIM Register message. The RP decapsulates PIM Register messages back to the native multicast packets and forward the packets onto the shared tree.

Phase two: Register stop

Sending PIM Register packet is an inefficient process because packets must be encapsulated at DR and decapsulated at RP. An RP may choose to stop receiving Register messages and switch to native multicast forwarding. If it does so, it initiates a (S,G) Join toward S. Routers along the path between RP and S send (S,G) Join to their RPF next hop. When (S,G) Join arrives at the DR of S, the DR starts to forward packets from S using both native multicast and Register-encapsulated messages. RP then starts receiving two copies of packets from S; and when that happens it sends a Register-Stop message to S's DR. The DR stops sending Register-encapsulated packets to RP, completing the phase two.

Phase three: Shortest path tree

After the phase two is complete, native multicast packets are flowing from S to RP, removing the inefficiencies of Register packets. The path delay from S to a receiver may be reduced if the receiver doesn't use shared tree. The router on the same subnet with a receiver, typically DR, may optionally switch to the source tree. If it switches to the source tree, it issues an (S,G) Join to the RPF next hop toward S. The (S,G) Join will eventually arrives at the subnet of S or at a router that already has (S,G) Join state. The routers then starts forwarding packets onto the source tree to the receiver. When the first traffic begin arriving at S's subnet, the DR or the upstream router sends a (S,G) Prune toward the RP to the upstream router. This message is known as (S,G,rpt) Prune, which says that RP should not forward traffic coming from S to G. (S,G,rpt) Prune message travels hop-by-hop until it reaches RP or a router that still needs traffic from S.

PIM Assert

Multiple copies of a multicast traffic may arrive on a transit LAN from different routers. PIM-SM doesn't prevent this from happening, instead it provides a mechanism to elect a single forwarder, using PIM Assert messages. Routers elect the single forwarder in favor of the router that has the best metric toward the source, then toward RP.

RP discovery

PIM-SM uses a Rendezvous Point (RP) for each multicast group. An RP serves two purposes:

- as the root of shared trees; and
- for senders and receivers to learn the existens of each others.

Routers on the network should know the RP for a group, otherwise it cannot form the multicast distribution tree for the group. The RP for each group may be statically configured at each router. PIM-SM provides a mechanism to discover the RP dynamically using Bootstrap Router mechanism. A router in a PIM domain is elected as the Bootstrap Router. Routers that are configured to become RP candidate inform their candidacy to the Bootstrap Router (BSR). The BSR gathers the candidacy information and picks an RP-set then send the RP-set in Bootstrap messages. The Bootstrap messages are flooded throughout the PIM domain until all routers know the RP-set.

Multicast Routing on FreeBSD

Multicast-routing enabled kernel

The FreeBSD's generic kernel does not support IPv4 multicast routing, but IPv6 multicast routing support is already enabled. To enable multicast routing, the `MROUTING` kernel option must be activated. Also, if you want to run PIM-SM multicast routing protocol, the `PIM` kernel option must be activated. The kernel configuration lines to enable PIM-SM multicast routing are:

```
options MROUTING
options PIM
```

Multicast routing cache table

Multicast routing cache table and the virtual interface table can be displayed with the command `netstat -ng`. Below is an example of the result for IPv4.

Virtual Interface Table

Vif	Thresh	Rate	Local-Address	Remote-Address	Pkts-In	Pkts-Out
0	1	0	10.20.1.1		18871	5255
1	1	0	10.1.1.100		5255	12740
2	1	0	10.20.1.1		0	2532

IPv4 Multicast Forwarding Cache

Origin	Group	Packets	In-Vif	Out-Vifs:Ttls
10.20.1.20	239.18.100.100	208	0	2:1

In this example, the FreeBSD router has three virtual interfaces (Vif), each is identified by a number. Each Vif may forward multicast packets whose TTL is at least same as **Thresh** value. Each Vif has a rate limit, where 0 represents no limit. If the Vif is a tunnel,

then the remote address is also displayed. The number of incoming and outgoing packets are also displayed. In the IPv4 multicast forwarding cache you can see there is a multicast source sending packets from the Vif 0 and the packets are forwarded to Vif 2 if the TTL is more than 1.

The Vif 2 is the interface to send PIM Register packets to PIM Rendezvous Point. This is not clearly displayed in the netstat results, however you can see it using the PIM-SM multicast routing protocol daemon.

An example for the IPv6 is displayed below. You can see the difference between the interface tables of IPv4 and IPv6. The IPv6 interface table uses the physical interface name instead of the IP address of the interface. The reg0 interface in the table is the interface to send PIM Register packets. The IPv6 multicast forwarding cache is similar to that of IPv4.

IPv6 Multicast Interface Table

Mif	Rate	PhyIF	Pkts-In	Pkts-Out
0	0	fxp0	23834	10097
1	0	fxp1	10097	12066
2	0	reg0	0	9833

IPv6 Multicast Forwarding Cache

Origin	Group	Packets	Waits	In-Mif	Out-Mifs
2001:d30:13:e001:7172:2b1c:36	ff85::100	209	0	0	2

XORP

The eXtensible Open Router Platform (XORP) is an open source router platform. It runs on several operating systems, including FreeBSD. XORP (<http://www.xorp.org>) release 1.0 is just released on July 8, 2004. XORP will support unicast and multicast routing protocols for IPv4 and IPv6.

The XORP process model is displayed in Figure 4.1. XORP can be divided into two subsystems: the "user-space", which handles management processes and routing protocols, and the "kernel-space", which handles the forwarding path and provides API to the user-space.

Installing XORP

To compile XORP release 1.0, you must have GNU make (gmake) installed. XORP also needs Net-SNMP package for installation and operation. GNU make and Net-SNMP may be installed using the package or port installation. The details on how to install XORP is available in the XORP tarball. By default, XORP will be installed in `/usr/local/xorp`. Besides this, you also have to create `xorp` group.

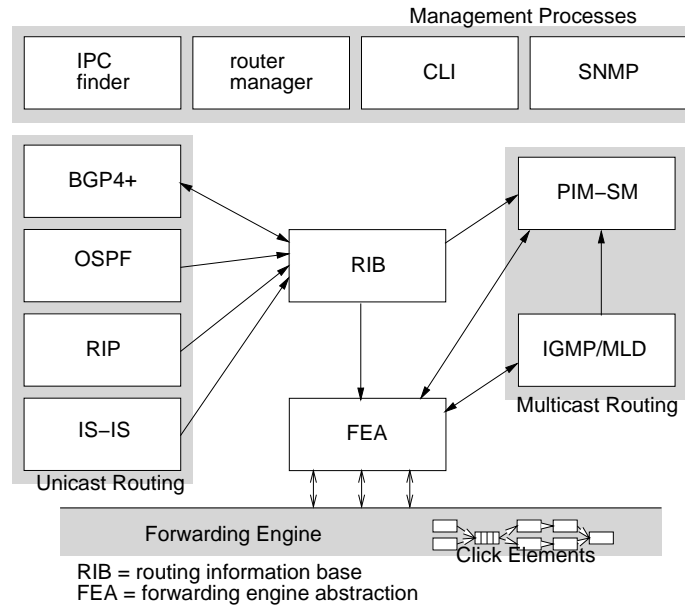


Figure 4.1: XORP process model

Configuring XORP for PIM-SM

XORP uses a configuration file whose name is `/usr/local/xorp/config.boot` by default. A sample configuration is available in `rtrmgr` directory inside the XORP source tree.

A basic configuration of XORP to enable PIM-SM for IPv4 and IPv6 is quite long, therefore we explain the configuration part-by-part.

```

interfaces {
  interface fxp0 {
    description: "upstream interface"
    enabled: true
    default-system-config
  }
  interface fxp1 {
    description: "downstream interface"
    enabled: true
    default-system-config
  }
}

```

Above is the network interfaces part. This configuration enables two physical interfaces. An interface must be explicitly enabled for XORP operation. An interface statement refers to a physical interface of the host. XORP may have several virtual interface for an interface. In majority of cases, the virtual interface name and the physical interface name is identical. The `default-system-config` statement tells XORP to use the IP addresses etc. that is set by the system, i.e. the kernel on FreeBSD.

Below is the Multicast Forwarding Engine Abstraction (MFEA) for both IPv4 and IPv6. The `mfea4` and `mfea6` statements are MFEA statement for IPv4 and IPv6, respectively.

The virtual interface `register_vif` should always be enabled for PIM-SM operation. The `traceoptions` statement is for debugging purposes.

```
plumbing {
  mfea4 {
    enabled: true
    interface fxp0 {
      vif fxp0 {
        enabled: true
      }
    }
    interface fxp1 {
      vif fxp1 {
        enabled: true
      }
    }
    interface register_vif {
      vif register_vif {
        /* Note: this vif should be always enabled */
        enabled: true
      }
    }
    traceoptions {
      flag all {
        enabled: true
      }
    }
  }
}

plumbing {
  mfea6 {
    enabled: true
    interface fxp0 {
      vif fxp0 {
        enabled: true
      }
    }
    interface fxp1 {
      vif fxp1 {
        enabled: true
      }
    }
    interface register_vif {
      vif register_vif {
        enabled: true
      }
    }
    traceoptions {
      flag all {
        enabled: true
      }
    }
  }
}
```

The next part is the protocol part for IGMP and MLD.

```

protocols {
  igmp {
    enabled: true
    interface fxp0 {
      vif fxp0 {
        enabled: true
      }
    }
    interface fxp1 {
      vif fxp1 {
        enabled: true
      }
    }
    traceoptions {
      flag all {
        enabled: true
      }
    }
  }
  mld {
    enabled: true
    interface fxp0 {
      vif fxp0 {
        enabled: true
      }
    }
    interface fxp1 {
      vif fxp1 {
        enabled: true
      }
    }
    traceoptions {
      flag all {
        enabled: true
      }
    }
  }
}

```

The next part is for PIM-SM configuration. You can see the three interfaces are enabled. This router has a static RP set, and also it is set to become a BSR and an RP candidate. Which RP will be used by this router, whether the static RP or the RP from Bootstrap mechanism, depend on the priority. Switching to the SPT is set by `switch-to-spt-threshold` statement, where in this case it will happen if total 10240 bytes arrive within 100 seconds. You can also see that the traceoption flags are enabled for this protocol.

The `fib2mrib` protocol statement at the end of this part tells XORP to populate MRIB with unicast routing entries.

```

protocols {
  pimsm4 {
    enabled: true
    interface fxp0 {
      vif fxp0 {

```

```
        enabled: true
    }
}
interface fxp1 {
    vif fxp1 {
        enabled: true
    }
}
interface register_vif {
    vif register_vif {
        /* Note: this vif should be always enabled */
        enabled: true
    }
}

static-rps {
    rp 10.10.1.1 {
        group-prefix 224.0.0.0/4 {
            /* rp-priority: 192 */
        }
    }
}

bootstrap {
    enabled: true
    cand-bsr {
        scope-zone 224.0.0.0/4 {
            cand-bsr-by-vif-name: "fxp0"
            bsr-priority: 1
            hash-mask-len: 30
        }
    }

    cand-rp {
        group-prefix 224.0.0.0/4 {
            cand-rp-by-vif-name: "fxp0"
            rp-priority: 192
            rp-holdtime: 150
        }
    }
}

switch-to-spt-threshold {
    /* approx. 1K bytes/s (10Kbps) threshold */
    enabled: true
    interval-sec: 100
    bytes: 102400
}

traceoptions {
    flag all {
        enabled: true
    }
}
}
```

```
pimsm6 {
  enabled: true
  interface fxp0 {
    vif fxp0 {
      enabled: true
      dr-priority: 1
    }
  }
  interface fxp1 {
    vif fxp1 {
      enabled: true
      dr-priority: 1
    }
  }
  interface register_vif {
    vif register_vif {
      enabled: true
    }
  }
}

static-rps {
  rp 2001:d30:13:f001::f1 {
    group-prefix ff00::/8 {
      rp-priority: 192
      hash-mask-len: 126
    }
  }
}

bootstrap {
  enabled: true
  cand-bsr {
    scope-zone ff00::/8 {
      cand-bsr-by-vif-name: "fxp0"
      bsr-priority: 1
      hash-mask-len: 30
    }
  }

  cand-rp {
    group-prefix ff00::/8 {
      cand-rp-by-vif-name: "fxp0"
      rp-priority: 192
      rp-holdtime: 150
    }
  }
}

switch-to-spt-threshold {
  enabled: true
  interval-sec: 100
  bytes: 102400
}

traceoptions {
  flag all {
    enabled: true
  }
}
```

```

        }
    }
}

protocols {
    fib2mrib {
        enabled: true
    }
}

```

Running XORP

XORP must be run as root by invoking the router manager, i.e.

`/usr/local/xorp/bin/xorp_rtrmgr` while to access the CLI you can run the shell:

`/usr/local/xorp/bin/xorpsh`

To enable XORP running at boot time, create a startup file `/usr/local/etc/rc.d/xorp.sh` and make it executable. The contents are

```

#!/bin/sh

case "$1" in
start)
    /usr/local/xorp/bin/xorp & && echo -n ' xorp'
    ;;
stop)
    killall xorp && echo -n ' xorp'
    ;;
*)
    echo "Usage: 'basename $0' {start|stop}" >&2
    ;;
esac

exit 0

```

Operating PIM-SM Multicast Routing

We will use the network topology in Figure 4.2 to explain how to operate multicast routing with PIM-SM. In this section we only explain about PIM-SM for IPv4 because the IP addresses are shorter. The PIM-SM operation for IPv6 is similar to the IPv4. Both protocols will be included in the Exercise section.

The PIM-SM states that have to be watched are:

1. interface
2. neighbor
3. MRIB
4. bootstrap
5. Rendezvous Points
6. Join

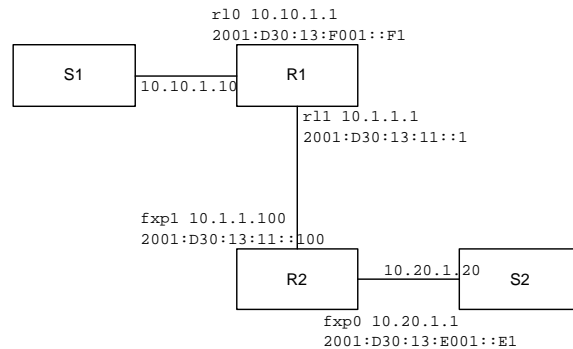


Figure 4.2: Sample network topology

Interface states

PIM router interface state can be displayed using the following command at XORP CLI.

```
show pim interfaces
```

Below are the results for the sample network topology. From these results you can see that both r10 and r11 of router R1 are assigned with Priority 255, and they are the Designated Router for their links. Meanwhile fxp1 has Priority 0, and it does not become a DR. The DR address is also shown in the results. The `register_vif` interface is the interface to send PIM Register packets to the Rendezvous Point. You can also see how many neighbors are there for each interface.

R1

Interface	State	Mode	V	PIMstate	Priority	DRaddr	Neighbors
r10	UP	Sparse 2	DR		255	10.10.1.1	0
r11	UP	Sparse 2	DR		255	10.1.1.1	1
register_vif	UP	Sparse 2	DR		1	0.0.0.0	0

R2

Interface	State	Mode	V	PIMstate	Priority	DRaddr	Neighbors
fxp0	UP	Sparse 2	DR		1	10.20.1.1	0
fxp1	UP	Sparse 2	NotDR		0	10.1.1.1	1
register_vif	UP	Sparse 2	DR		1	0.0.0.0	0

Neighbor states

You can display PIM neighbor states using

```
show pim neighbors
```

command. For each neighbor you can see its priority to become DR, how much time is left before the router deletes the neighbor if the router doesn't hear a Hello message from the neighbor, and several other information.

R1

Interface	DRpriority	NeighborAddr	V Mode	Holdtime	Timeout
rl1	0	10.1.1.100	2 Sparse	105	101

R2

Interface	DRpriority	NeighborAddr	V Mode	Holdtime	Timeout
fxp1	255	10.1.1.1	2 Sparse	105	83

MRIB

You can display the multicast routing information base using `show pim neighbors` command.

R1

DestPrefix	NextHopRouter	VifName	VifIndex	MetricPref	Metric
10.1.1.0/24	10.1.1.1	rl1	1	0	0
10.10.1.0/24	10.10.1.1	rl0	0	0	0
10.20.1.0/24	10.1.1.100	rl1	1	254	65535

R2

DestPrefix	NextHopRouter	VifName	VifIndex	MetricPref	Metric
10.1.1.0/24	10.1.1.100	fxp1	1	0	0
10.10.1.0/24	10.1.1.1	fxp1	1	254	65535
10.20.1.0/24	10.20.1.1	fxp0	0	0	0

Bootstrap states

Issuing

`pim show bootstrap`

at the XORP CLI shows Bootstrap Routers and their states. XORP keeps three databases: Active, Expiring, and Configured zones. BSRs in Active zones are the ones that are currently active. Expiring zones shows the BSRs that have not been heard of for a while and will be deleted soon. Configured zones shows the locally configured BSRs.

From the results below, you can see that R1 is configured to be a BSR candidate, while R1 is not. Also, there is only one Bootstrap Router whose IP address is 10.10.1.1 (R1), and R1 is the elected Bootstrap Router. There is no BSR that will be deleted from the XORP database.

R1

```
Active zones:
BSR          Pri LocalAddress  Pri State          Timeout SZTimeout
10.10.1.1    1 10.10.1.1        1 Elected          49      -1
Expiring zones:
BSR          Pri LocalAddress  Pri State          Timeout SZTimeout
Configured zones:
BSR          Pri LocalAddress  Pri State          Timeout SZTimeout
10.10.1.1    1 10.10.1.1        1 Init              -1      -1
```

R2

```

Active zones:
BSR          Pri LocalAddress  Pri State      Timeout SZTimeout
10.10.1.1    1 0.0.0.0      0 AcceptPreferred  73     1243
Expiring zones:
BSR          Pri LocalAddress  Pri State      Timeout SZTimeout
Configured zones:
BSR          Pri LocalAddress  Pri State      Timeout SZTimeout

```

RP states

The RP for each multicast group prefix are displayed by invoking `show pim rps`

command at the XORP CLI. The below results show that 10.10.1.1 is the RP for 224.0.0.0/4 multicast group prefix, and the RP is learned from the Bootstrap mechanism.

R1

```

RP          Type      Pri Holdtime Timeout ActiveGroups GroupPrefix
10.10.1.1   bootstrap 192    150    -1          3 224.0.0.0/4

```

R2

```

RP          Type      Pri Holdtime Timeout ActiveGroups GroupPrefix
10.10.1.1   bootstrap 192    150    120         3 224.0.0.0/4

```

Join states

To view the PIM Join states of a router, the command is `show pim join`

In this example, both C1 and C2 join to group 239.18.100.100. PIM Join states at R1 and R2 are shown below. Because the results of this command are long, we skip some of the details and we put line numbers for explanations.

Line 1–6 shows the (*,G) state. In this case, group *G* is 239.18.100.100. The Flag of this state is WC, meaning wildcard, i.e. for all source address (0.0.0.0). The RP is itself, thus the Upstream interface toward RP (Line 2) is register_vif. Line 3 and 4 entries are UNKNOWN because the RP is itself. The UNKNOWN values in the Join state results are actually 0.0.0.0, which means self. This is a XORP implementation problem. The (*,G) state is Joined, and the remaining time before it expect to have another PIM Join message is 33 seconds.

Line 7–12 show the (S,G,rpt) state, which is in the Pruned state. Line 8 and 9 show the upstream interfaces toward the source and the RP. Same as in the (*,G) state, the upstream MRIB next hop and RPF' are UNKNOWN because this router is the RP.

Line 13–21 display the (S,G) state for source 10.10.1.10 and group 239.18.100.100. This state is in Joined state. The flags show that this source is directly connected to the router (DirectlyConnectedS). The upstream MRIB next hop toward source and RPF' are UNKNOWN because the source is directly connected. Line 20 shows that this router does not send PIM Register packets because it is the RP.

Line 22–29 show the (S,G) state for the same group but for source 10.20.1.20. This state is in Joined state, which means that R1 uses the shortest path to receive traffic from

10.20.1.20 for group 239.18.100.100. The upstream MRIB next hop toward source and RPF' are R2, thus the multicast traffic from 10.20.1.20 is expected to come from R2. The upstream MRIB next hop toward RP is irrelevant for (S,G) state, and displayed as UNKNOWN.

R1

```

Group          Source          RP          Flags
1 239.18.100.100 0.0.0.0      10.10.1.1   WC
2   Upstream interface (RP): register_vif
3   Upstream MRIB next hop (RP): UNKNOWN
4   Upstream RPF'(*,G):      UNKNOWN
5   Upstream state:          Joined
6   Join timer:              33
...

Group          Source          RP          Flags
7 239.18.100.100 10.20.1.20    10.10.1.1   SG_RPT
8   Upstream interface (S):  r11
9   Upstream interface (RP): register_vif
10  Upstream MRIB next hop (RP): UNKNOWN
11  Upstream RPF'(S,G,rpt):  UNKNOWN
12  Upstream state:          Pruned
...

Group          Source          RP          Flags
13 239.18.100.100 10.10.1.10    10.10.1.1   SG SPT DirectlyConnectedS
14  Upstream interface (S):  r10
15  Upstream interface (RP): register_vif
16  Upstream MRIB next hop (RP): UNKNOWN
17  Upstream MRIB next hop (S): UNKNOWN
18  Upstream RPF'(S,G):      UNKNOWN
19  Upstream state:          Joined
20  Register state:          RegisterNoinfo RegisterNotCouldRegister
21  Join timer:              10
...

Group          Source          RP          Flags
22 239.18.100.100 10.20.1.20    10.10.1.1   SG SPT
23  Upstream interface (S):  r11
24  Upstream interface (RP): register_vif
25  Upstream MRIB next hop (RP): UNKNOWN
26  Upstream MRIB next hop (S): 10.1.1.100
27  Upstream RPF'(S,G):      10.1.1.100
28  Upstream state:          Joined
29  Join timer:              14

```

Next you can see the Join states at R2. Line 1–6 show that R2 joins the (*,G) state. Line 4 shows that the upstream interface toward RP is fxp1, and the RPF' to RP is 10.1.1.1.

Line 7–12 state that R2 pruned the (S,G,rpt) state for the corresponding source, group, and RP. For this source, R2 joins the (S,G) state (Line 21–29), and it already stop sending PIM Register packets to RP, as seen by the RegisterPrune state.

Line 13–20 shows the (S,G) state of R2 for source 10.10.1.10 and group 239.18.100.100. You can see that the upstream MRIB next hop toward RP and the RPF' is 10.1.1.1. The upstream MRIB toward RP is irrelevant for this state.

R2

```

    Group          Source          RP          Flags
  1 239.18.100.100  0.0.0.0      10.10.1.1   WC
  2   Upstream interface (RP):  fxp1
  3   Upstream MRIB next hop (RP): 10.1.1.1
  4   Upstream RPF'(*,G):         10.1.1.1
  5   Upstream state:              Joined
  6   Join timer:                  41
...

    Group          Source          RP          Flags
  7 239.18.100.100 10.20.1.20   10.10.1.1   SG_RPT DirectlyConnectedS
  8   Upstream interface (S):      fxp0
  9   Upstream interface (RP):     fxp1
 10   Upstream MRIB next hop (RP): 10.1.1.1
 11   Upstream RPF'(S,G,rpt):     10.1.1.1
 12   Upstream state:              Pruned
...

    Group          Source          RP          Flags
 13 239.18.100.100 10.10.1.10   10.10.1.1   SG SPT
 14   Upstream interface (S):      fxp1
 15   Upstream interface (RP):     fxp1
 16   Upstream MRIB next hop (RP): 10.1.1.1
 17   Upstream MRIB next hop (S):  10.1.1.1
 18   Upstream RPF'(S,G):         10.1.1.1
 19   Upstream state:              Joined
 20   Join timer:                  56
...

    Group          Source          RP          Flags
 21 239.18.100.100 10.20.1.20   10.10.1.1   SG SPT DirectlyConnectedS
 22   Upstream interface (S):      fxp0
 23   Upstream interface (RP):     fxp1
 24   Upstream MRIB next hop (RP): 10.1.1.1
 25   Upstream MRIB next hop (S):  UNKNOWN
 26   Upstream RPF'(S,G):         UNKNOWN
 27   Upstream state:              Joined
 28   Register state:              RegisterPrune RegisterCouldRegister
 29   Join timer:                  38

```

Exercise

Ex. 1 Installing XORP

1. Create group named `xorp`. Chose any available GID.
2. Download XORP source code.
You can download from the XORP website <http://www.xorp.org/>
We provide a local copy in this workshop, please ask the URL to the instructor.
3. Untar the package. At the command prompt, type:

```
tar xzpvf xorp-1.0.tar.gz
```
4. Run the commands below:

```
cd xorp-1.0
./configure
gmake
```
5. As `root`, install XORP by typing:

```
gmake install
```

Ex. 2 Configuring XORP

In this exercise the instructor will show the topology and IP address assignment of the class network.

1. Write down the configuration requirement for your router.
2. As `root`, create the XORP configuration file.

```
/usr/local/xorp/config.boot
```
3. Write the configuration based on the configuration requirement.
Enable all `traceoptions` for debugging.
4. Confirm your configuration.
5. Start running XORP router manger by running the command

```
/usr/local/xorp/bin/xorp_rtrmgr
```


Check that the router manager process is running. If you still find a configuration error, fix your configuration and run it again.

Ex. 2 Running PIM-SM with XORP: initial states

1. Log on using another vty and run the XORP CLI.

```
/usr/local/xorp/bin/xorpsh
```
2. At the XORP CLI, check the interfaces.

```
show interfaces
```

3. Read and analyze PIM interface state.
`show pim interface`
4. Check and analyze PIM neighbor state.
`show pim neighbors`
5. Check and analyze PIM MRIB.
`show pim mrib`
6. Check and analyze PIM bootstrap mechanism.
`show pim bootstrap`
If you do not see any BSR, wait for a while, then check again.
7. Check and analyze PIM Rendezvous Points.
`show pim rps`
8. Check and analyze PIM Join states.
`show pim join`
9. Check and analyze PIM MFC.
`show pim mfc`
10. Repeat the above procedures for PIM-SM for IPv6.

```
show pim6 interface
show pmi6 neighbors
show pim6 mrib
show pim6 bootstrap
show pim6 rps
show pim6 join
show pim6 mfc
```

11. Discuss your findings.

Ex. 3 Running PIM-SM with XORP: Join states

1. On your Windows XP, run a program to receive IPv4 multicast traffic as instructed.
2. Check and analyze the PIM Join state of your router.
`show pim join`
3. The instructor's PC start sending multicast traffic. Check and analyze the PIM Join state and MFC of your router.
`show pim join`
`show pim mfc`
4. Discuss your findings.
5. Close the multicast program.

6. Check and analyze the PIM Join state and MFC of your router.
`show pim join`
`show pim mfc`
7. Discuss your findings.
8. Now run a program to receive IPv6 multicast traffic on your Windows XP.
9. Check and analyze the PIM Join state and MFC of your router.
`show pim6 join`
`show pim6 mfc`
10. The instructor's PC start sending multicast traffic. Check and analyze the PIM Join state and MFC of your router.
`show pim6 join`
`show pim6 mfc`
11. Discuss your findings.
12. Close the multicast program.
13. Check and analyze the PIM Join state and MFC of your router.
`show pim6 join`
`show pim6 mfc`
14. Discuss your findings.

Ex. 4 Installing XORP to the FreeBSD start-up configuration

1. Edit the XORP configuration file, disable all traceoptions.
2. Create the start-up file: `/usr/local/etc/rc.d/xorp.sh`.
3. Make it executable.
`chmod +x /usr/local/etc/rc.d/xorp.sh`
4. Reboot FreeBSD.
5. Confirm that XORP is running properly.

Ex. 5 Troubleshooting PIM-SM

You are asked to apply your knowledge for troubleshooting in this exercise. The instructor will provide the problems.