Testing IPv4/IPv6-Based Unicast/Multicast Routing Protocols Using Linux and FreeBSD

Tudor Blaga[1], Virgil Dobrota[1]

Abstract – This paper is focused on the study of routing protocols in IPv4 and IPv6. Their classification is based on type of updating (distance-vector versus link-state), working domains (intra- versus inter-domain), number of paths (single or multi-path), and type of traffic (unicast or multicast). The work is backed up by the practical experiments, using the Zebra and GateD routing software on RedHat Linux platform for IPv4, and pim6sd and pim6dd on FreeBSD for IPv6. The architecture used by the experiments revealed the operation of the following routing protocols: RIPv1/v2, RIPng, OSPFv2/v3, DVMRP, PIM-SM and PIM-DM in IPv4/IPv6. The deployment of a multicast testbed is more complex than for a unicast one. It involves a multicast traffic source, at least one member for that multicast group and the multicast routers.

Keywords: Zebra, GateD, KAME, multicast, routing protocols

I. INTRODUCTION

One of the basic functions of the Internet Layer is to forward packets from the source to the destination. This process of moving packets across an inter-network is called routing and the device performing it is an IP router.

Routing generally involves the (optimal) path determination and the packet switching [4]. The protocols involved use metrics to evaluate the best single or multiple paths to be followed by a packet in order to reach the destination. To perform the path determination process, the algorithms establish and maintain routing tables. According to this dynamically updated route information, the switching is able to move the packet from a router's interface to another, forwarding the datagram towards the next hop. The static and dynamic allocation of logical IP addresses is described in details in [1], [5].

During the current migration from IPv4 to IPv6, due to the lack of addresses and/or the exponential growth of the routing table size, additional mechanisms were implemented too. Among them, CIDR *(Classless InterDomain Routing),* VLSM *(Variable Length Subnet Mask)* and NAT *(Network Address Translation)* have a certain success. There are no doubts that IPv6 remains the complete solution, expecting to be generalized as soon as possible. For these reasons, the paper is focused on a major problem during the transition from one IP version to another: the routing protocols. Although the approach has taken into consideration the existing achievements in the field (including those from Cisco Systems, the world leader) the practical experiments were conducted in Linux, due to its availability for the new coming IPv6-based routing protocols.

IP multicasting is a Network Layer mechanism to support applications where data needs to be sent from a source to multiple receivers (point-to-multipoint). The applications based on this concept could be for instance conferencing systems, software updates, on-demand video distribution and resource discovery. A major benefit of using multicasting could be the considerable decrease of the network traffic (load). It optimizes the number of packets sent when there is a group of nodes as destination.

The problem of routing multicast packets efficiently to the group members is more complex than in unicast. As different approaches have been taken into consideration, a set of multicast protocols were designed: DVMRP (Distance Vector Multicast Routing Protocol), MOSPF (Multicast Extensions to OSPF), PIM (Protocol Independent Multicast) and CBT (Core Based Tree). Other challenging issues are how to implement reliable multicast, how to handle the flow control and how to secure a multicast session.

This paper focuses on multicast routing, involving DVMRP and PIM only, due to the lack of implementations of other protocols. The platforms were based on Linux Fedora Core 1 (FC1) for the IPv4 multicast routers and IPv4/IPv6 clients (multicast senders and receivers) and FreeBSD 5.2.1 for the IPv6 multicast routers.

II. ROUTING BASICS

II.1. Unicast Routing

The several existing routing algorithms may have different impacts on network but certain common properties are requested for all of them: correctness, simplicity, robustness, stability, fairness and optimality.

[1] Technical University of Cluj-Napoca, Communications Department, 26-28 Baritiu Street, 400027 Cluj-Napoca, Romania, Tel/Fax: +40-264-597083, E-mail: {Tudor.Blaga, Virgil.Dobrota}@com.utcluj.ro

There are two methods used to build a routing table: static and dynamic. *Static routing* does not involve an algorithm, as the network administrator manually configures the routes. On the other hand, the *dynamic* one utilizes algorithms that automatically discover, calculate and maintain paths through the network. These are classified into three main categories: *distance-vector*, *link state* and *hybrid*.

The multiplicity of routing protocols could be explained also by the domain they were designed to operate in. *Intra-domain protocols* work only within an AS (*Autonomous System),* whilst *inter-domain protocols* are applied between several routing domains. Note that an autonomous system is a collection of networks under a common administration. It might be possible to have an overlapping of routing domains within an AS, as the router may work with different routing protocols in the same time. However, an *administrative distance* (from 0 to 255) is a rating of the trustworthiness of a routing information source. For a given route, the protocol having the lowest administrative distance will be chosen. By default the distances for a connected interface is 0 and for a static route is 1. Table 1 presents some of the most used routing protocols, classified according to the criteria previously discussed.

The need for balanced utilization of the network resources has lead to complex routing protocols that support multiple paths to the same destination. This evolution requested a different approach. Within the initial *flat routing systems*, all routers were on the same level, all of them exchanging routing information and all being peers of all others. *Hierarchical systems* divide the network into routing areas. Some routers in an area can communicate with routers in other areas, while others can communicate only with routers within their area.

The use of hierarchical routing reduces the amount of routing update traffic and simplifies sometimes the algorithms. The process of bringing all routing tables to a state of consistency is called *convergence*. The time it takes a network to converge depends on the routing algorithm, the *convergence time* being one of the most important performances criteria. The continuous circling of traffic between two or more routers is referred to as a *routing loop*. The *count-to-infinity problem* is specific to distance vector protocols only and it consists on a continuous increment of the path cost up to infinity. Obviously, the routing loops and the count-to-infinity problem have to be carefully considered and eliminated.

Table 1. Classification of IP routing protocols

Routing Protocol	Working Domain		Type		Metrics		IP Version		Administrative Distance	Path		Unicast/ Multicast		Zebra Implementation
	Intra-	Inter-	Distance vector	Link-state	Single	Composite	IPv4	IPv6		Single	Multi	Uni-cast	Multi-cast	
RIP	X		X		X		X		120	X		X		X
RIP-2	X		X		X		X		120	X		X		X
RIPng	X		X		X			X	120	X		X		X
IGRP	X		X			X	X		100		X	X		-
EIGRP	X		X	X		X	X		90/170		X	X		-
OSPF-2	X			X	X		X		110		X	X		X
OSPF-3	X			X	X		X	X	110		X	X		X
MOSPF	X			X	X		X						X	-
IS-IS	X			X	X		X	X	115		X	X		-
DVMRP	X		X		X		X						X	-
PIM	X						X	X					X	-
EGP		X					X		140	X		X		-
BGP-4	X	X	X				X	X	20/200		X	X	X	X

A. RIPv1, RIPv2 and RIPng

RIP (Routing Information Protocol) is a distance vector routing protocol that uses hop count as a metric to determine the direction and distance to any link in the internetwork. It was designed for small domains, selecting the path with the lowest number of hops (not greater than 15).

Routing tables are exchanged through routing updates that are broadcasted periodically every 30 seconds. Split horizon or split horizon with poison reverse are used to prevent the count-to-infinity problem. RIPv1 (RIP Version 1) requires all devices in the network to use the same subnet mask, because it cannot include subnet mask information in its routing updates. This is a typical characteristic of a classful routing protocol.

RIPv2 (RIP Version 2) provides support for CIDR and VLSM, being a classless routing protocol [5]. It prefers multicasting instead of simple broadcasting of routing announcements. This reduces the processing load on hosts that are not listening for RIPv2 messages. Additional support for authentication and fully interoperability with RIPv1 are provided.

RIPng (RIP Next Generation) was developed to allow routers within an IPv6-based network to exchange

information. Although it uses the same algorithms, timers and logic as in previous version, there are two major differences. First, the RIPng does not include any native authentication support, relying on security features offered by IPv6. Second, the packets were updated to support the 128-bit IPv6 address format. RIPv1, RIPv2 and RIPng have the same limitation as any distance vector routing protocol, which is the slow convergence. An additional drawback such as the path cost restriction (maximum 15 hops) might be eliminated within a Cisco routers only, by choosing IGRP (Interior Gateway Routing Protocol) (up to 255 hops).

B. OSPFv2 and OSPFv3

OSPFv2 *(Open Shortest Path First Version 2)* is a link-state routing protocol developed to address the limitations of RIP. It provides immediate propagation of routing updates because they are events triggered, offering faster convergence and no cost limitations. OSPF was from the beginning designed as a classless routing protocol, by supporting CIDR and VLSM. Being very complex, it permits hierarchical routing. Thus OSPF networks are divided into a collection of areas (logical groups of networks and routers). Area 0 (known as backbone area) physically connects to all other areas. Each router has a view of the entire network (a routing loop-free approach) and it executes the SPF *(Shortest-Path First)* algorithm to determine the routes to the destination. Routing information is flooded by using LSA *(Link State Advertisement)* packets to the following IPv4 multicast addresses: 224.0.0.5 for OSPF routers and 224.0.0.6 for Designated/Backup Designated routers.

OSPFv3 *(OSPF Version 3)* was designed for both IPv4 and IPv6. Obviously, some modifications were needed due to the changes in protocol semantics or simply to handle the increased address size. New LSAs have been created to carry IPv6 addresses and prefixes. Authentication has been removed from the OSPF protocol itself, relying on IPv6's AH *(Authentication Header)* and ESP *(Encapsulating Security Payload)* [1]. Although the SPF algorithm was employed again, the multicasting is using to distinct IPv6 addresses: FF02::5 for all OSPF routers and FF02::6 for all Designated/Backup Designated routers. Note that the packets sent to these multicast addresses should never be forwarded because they are meant to travel a single hop only. This is the reason to have multicast addresses with link-local scope and packets sent to these addresses should have their IPv6 Hop Limit set to one.

II.2. Multicast Routing

Multicast data delivery requires a set of protocols and mechanisms at the Network Layer:

- Multicast addresses that designate a multicast group as the destination of a datagram
- A mechanism that allows a host to join and leave a multicast group
- Multicast routing protocols that set up paths, called distribution tree, from the sender to the members of a multicast group.

IPv4 and IPv6 multicast addresses have different structure. The range of IPv4 multicast addresses, from 224.0.0.0 to 239.255.255.255, corresponds to the class D addresses of IPv4 addressing scheme. Class D address is identified by the first four bits (1110) within the address. The remaining 28 bits contain the group ID of the multicast group [9]. IPv6 multicast addresses are divided in four different fields. The first field (8bits) identifies that the address is a multicast address. This is followed by a flag field (4 bits) showing whether the address is permanent or non-permanent. The next four bits contain the delivery scope of the multicast packet. The group ID makes up the rest of the address (112 bits) [11].

Table 2. IPv4 multicast address structure

1110	Multicast group ID
4	28 bits

Table 3. IPv6 multicast address structure

11111111	flags	Scope	Multicast group ID
8	4	4	112 bits

A group management protocol is used by host willing to join or leave a multicast group. This way, hosts inform neighboring routers that they are interested in receiving multicast packets. IGMP (Internet Group Management Protocol) is the IPv4 version of this protocol and the IPv6 version is called MLD (Multicast Listener Discovery).

There are three versions of IGMP. IGMPv1 has two types of messages, membership query and membership report. Hosts send IGMP membership reports corresponding to a particular multicast group to indicate that they are interested in joining that group.

Router periodically sends IGMP membership query messages to verify that at least one host on the subnet is interested in multicast traffic. IGMPv2 is an enhancement of the first version and it includes a few extensions. Among them there is a procedure for the election of the multicast query device for each LAN, explicit leave messages for faster pruning and group specific query messages. IGMPv3 permits a host to join a group and specify a set of sources of that group from which it wants to receive multicast. This feature is called source filtering. MLD protocol has two versions: MLDv1 (IPv6 version of IGMPv2) and MLDv2 (IPv6 version of IGMPv3).

To deliver traffic to all receivers, multicast-capable routers create distribution trees. The simplest way of providing multicast routing is by flooding. If a multicast router receives a multicast packet for the first time, it forwards this packet to all the outgoing interfaces except the one from which it receives it. This solution is very inefficient in terms of network bandwidth utilization. A better solution is to build a spanning tree where a multicast router could forward multicast packets to all the interfaces that are part of a multicast tree except the source. Multicast forwarding algorithms can be classified in two categories: source-based (shortest path tree) and core-based (shared tree).

A source-based tree has its root at the source and branches forming a spanning tree through the network to the receivers. The source-based tree has the drawback that it is dependent on the source of the multicast tree; it must be computed separately for each source.

Core-based tree algorithms need a single common root placed at a chosen point in the network, but the root is not at the source. This shared root is called a rendez-vous point (RP). The disadvantage of core-based trees is that under certain circumstances the paths between the source and receivers might not be the optimal. Therefore the placement of the RP must be carefully considered.

Source-based trees, as well as core-based trees can be constructed using RPF (Reverse Path Forwarding). The idea of RPF is the following: by given the address of the tree's root, a router selects as its upstream neighbor in the tree the router which is the next-hop neighbor for forwarding unicast packets to the root. The network interface used to reach this upstream neighbor is called the RPF interface.

RPF tells each router the upstream neighbor in the distribution tree, but not the downstream neighbors, so additional protocol mechanisms are needed to determine the outgoing interfaces. One method to achieve this is *flood-and-prune,* which starts by forwarding multicast packets on all its interfaces, and then deletes interfaces which are not part of the distribution tree. Another method, called *explicit join,* requires that multicast receivers initiate the process of getting connected to the distribution tree.

Multicast routing protocols minimize the paths from the receivers to the source, as opposed to minimizing the path from the source to the receiver.

A. DVMRP

DVMRP was the first multicast routing protocol developed. It utilizes a dynamic routing protocol for route exchange and routing table construction based on RIP (Routing Information Protocol). It employs RPF to prevent multicast traffic from circulating in the network until the TTL field within the IP header becomes. It can operate in an environment where not all routers in the network are capable of multicast forwarding and routing. This implies a tunnel between multicast capable routers using IP-IP encapsulation.
The basic operation of DVMRP consists of four processes [8]:

- Neighbor discovery, which is used to find other DVMRP capable routers attached to a common network
- Route exchange, similar to RIP
- Graft messages, used to add networks to the forwarding list
- Prune messages, used to remove networks from the forwarding list

In the case of several multicast routers connected to a multi-access network, DVMRP provides a mechanism to elect a designated router (DR) to be responsible for forwarding multicast traffic to the network, thus preventing duplicate packets. The router with the lowest cost to the source is elected to be the DR. In case of a tie, the one with the lowest IP address is chosen.

Routing information exchanged consist of three components: the netmask, the network, and the metric. DVMRP routes are sent in abbreviated format. Because the first octet of every subnet mask is assumed to be 255, it is not included in the route report. Only one netmask is listed for all networks having the same netmask. To reduce the packet size further, only the portion of the network that corresponds to a non-zero value of the netmask is reported.

DVMRP messages are sent using IP packets with the protocol field set to 2, identifying the packet type as an IGMP message, the destination IP address used is 224.0.0.4, ALL-DVMRP-ROUTERS

B. PIM

PIM is a multicast routing protocol that is independent of the mechanisms provided by any unicast routing protocol. It requires some unicast routing protocols (such as RIP or OSPF) to determine the network topology and the topology changes.

PIM is not a single multicast routing protocol, it has two different modes: PIM-DM (PIM Dense Mode) and PIM-SM (PIM Sparse Mode). PIM-DM builds source-based trees using flood-and-prune, and is intended for large multicast groups where most networks have a group member. PIM-SM builds core-based trees as well as source-based trees with explicit joins, and it is intended for environments where group members are distributed across many regions of the network.

PIM-DM is quite similar to DVMRP: they both use a flood-and-prune mechanism to build delivery trees. However there are some important differences between these two algorithms. The first is that PIM-DM uses an existing unicast routing protocol to adapt to topology changes, but at the same time is independent of the mechanisms of this unicast routing protocol.

The operation of PIM-DM is similar to DVMRP without the route exchange. To avoid duplicate multicast packets forwarding in multi-access networks, PIM-DM uses assert messages to determine a designated forwarder for the network.

Multicast forwarding is performed for the interfaces from the *oilist* (output interface list). The oilist is populated with those interfaces on which neighbors were discovered or on which multicast receivers have indicated their desire to receive traffic.

PIM-SM assumes that each receiver has to explicitly join a multicast tree if it wants to receive any multicast packet. It creates a core-based tree with a share root called RP. The RP is responsible for forwarding all packets destined for the multicast group. Each-group has a single RP at any given time.

PIM-SM operation consists in three processes:

- Neighbor discovery, which uses router query messages
- RP registering, accomplished with register and register-stop messages
- RP joining/pruning, with join/prune messages

During neighbor discovery for a multi-access network a query message is sent to the all-routers multicast address, 224.0.0.2, which serves as the DR (Designated Router) election mechanism.

When a source sends a multicast packet to a certain group, the DR of that source encapsulates the first message in register message and sends it to the RP of that group as a unicast message. After receiving this message, the RP sends back a join message to the DR of the source. This way a distribution tree is created from the DR to the RP so the next multicast message of this source can be forwarded to the RP. Until the distribution tree is created, all multicast messages will be forwarded as encapsulated unicast messages. When the RP detects that multicast packets from the source are received as normal IP multicast packets, the RP sends a register-stop message to the DR. Upon reception of register-stop message the DR will stop encapsulating the multicast traffic from the source.

Hosts wanting to receive multicast traffic for a certain group will send an IGMP join message to their DR. The DR sends a join message to the RP for that multicast group.

Using the shared tree is not the best option in all cases. PIM-SM provides a method for using shortest-path trees for some or all of the receivers. When a threshold on a leaf router is exceeded, the router will switch from the shared tree through the RP to the source tree. In these situations, the leaf router sends a join message to the source node, thus creating a shortest-path tree.

PIM messages are encapsulated in IP packets with protocol number 103 and are sent to the multicast group 224.0.0.13, ALL-PIM-ROUTERS. The type field from the PIM packet header identifies the operation mode (dense/sparse) and the message type.

The operation mode of PIM for IPv6 does imply any major changes, except the type of addresses used within the header. The ALL-PIM-ROUTERS address ff02::d is the destination address for most messages. A link-local address of the interface on which the message is being forwarded will be used as source address. A special case is the Register message which uses domain-wide reachable IPv6 addresses, for both source and destination [12].

III. ROUTING TOOLS

To achieve our goal, i.e. to have an IPv4/IPv6 unicast and multicast routing testbed, the following software tools where used: Zebra 0.94 and NextHop GateD Enterprise 2.0 running on Fedora Core 1, the KAME IPv6 stack for BSD, code merged in FreeBSD 5.2 and a collection of programs developed to send and receive IPv4/IPv6 multicast traffic: *m4send/m4receive* and *m6send/m6receive*.

A. Zebra 0.94

Zebra is a Free Open Source package running under GNU/Linux, FreeBSD, NetBSD, OpenBSD, Solaris and providing (simultaneous) real routing services based on a collection of routing daemons, as you can observe in Table 4.

Table 4. Routing daemons in Zebra

Routing daemon	TCP port	Routing protocols
ripd	2602	RIPv1, RIPv2
ripngd	2603	RIPng
ospfd	2604	OSPFv2
ospf6d	2606	OSPFv3
bgpd	2605	BGP-4, BGP-4+

As a major advantage, Zebra is also ready to support IPv6-based routing protocols (which is not the case of the most Cisco routers currently available in our

academic network). At least two Linux boxes with Zebra are needed for tests. The first one starts an integrated shell (zebra vty) at TCP port 2601, allowing to change the configuration and to display the routing table. The second Linux box starts a dedicated routing daemon. The machine exchanges routing information with other routers using the previously mentioned protocols and updates the kernel routing table.

Note that the default commands for a Linux-based router configuration are *ifconfig* and *route*, whilst the status of routing table is displayed by *netstat*. These commands work only if the user has root privileges.

On the other hand, Zebra is administrated in a different way. Actually, there are two modes: *normal* and *enable*. Within the first one, the user can only view system status but within the *enable mode,* he/she can change system configuration (does not matter his/her rights in Linux). Another option for testing would have been gated, but unfortunately, this routing daemon originally coordinated by Cornell University is not free, currently being developed by Merit GateD Consortium.

B. GateD Enterprise 2.0

We used the free version for academic and research purposes provided by NextHop. GateD includes the following IPv4 unicast and multicast routing protocols: RIPv1/v2, OSPF, BGP, DVMRP, PIM-DM, PIM-SM, PIM-SSM (PIM Source Specific Multicast) [16].

GateD operation and configuration is similar to any Cisco router. During the installation of the software a special user, called cligated, is created. It gives us access to the command line interface. Configuration can be performed through XML sessions with telnet to the port 4242 on the machine running GateD.

C. KAME – FreeBSD

KAME Project is a joint effort of six companies from Japan to provide an IPv6 stack for different BSD variants. Several platforms contain KAME code in the source files, FreeBSD 4.0 and beyond, OpenBSD 2.7 and beyond, NetBSD 1.5 and beyond, BSD/OS 4.2 and beyond [17]. A separate KAME kit is available for each platform. The KAME kit includes more experimental protocols but is not as stable as the code merged in BSD.

D. IPv4/IPv6 multicast sender/receiver

The *m4sender/m6sender* programs periodically (1 second) transmit UDP datagrams to a given multicast group. They do not join the multicast group. On the other hand the *m4receive/m6receive* programs join the multicast group and display the payload of received multicast messages to the standard output.

The sender is started with the following command:

```
#./mXsend group_address port "text"
ttl/hoplimit
```

For example:

```
# ./m4send 224.5.5.5 5555 "IPv4 multicast" 4
# ./m6send ff15::5 5555 "IPv6 multicast" 4
```

The payload of the datagram contains besides the text given by the user a counter that allows us to track down which packets have been received.

The receiver is started with the following command:

```
# ./mXreceive group_address port
```

For example:

```
# ./m4receive 224.5.5.5 5555
# ./m6receive ff15::5 5555
```

This program sends the IGMP/MLD message to notify the DR that the host wants to receive multicast traffic for that multicast group.

IV. EXPERIMENTAL RESULTS

IV.1. Unicast Testbed

The practical experiments regarding the performance evaluations of the routing protocols are under progress. This paper is discussing first the unicast tests using Zebra 0.94 under Fedora Core 1, for both IPv6 and IPv4. Apparently, the network topology described in Figure 1 is not very complex but it is an excellent testbed for the study proposed. There are two routers, R1 and R2, actually two Linux machines, each running Zebra in two Linux boxes (as it was explained in section III).

The following subsections present the configuration files of for each routing daemon (*ripd, ripngd, ospfd* and *ospf6d*), as well some preliminary results. IPv6-based experiments are of a greater interest than those related to “classical” IPv4, so the comments are concentrated on the new coming routing protocols.

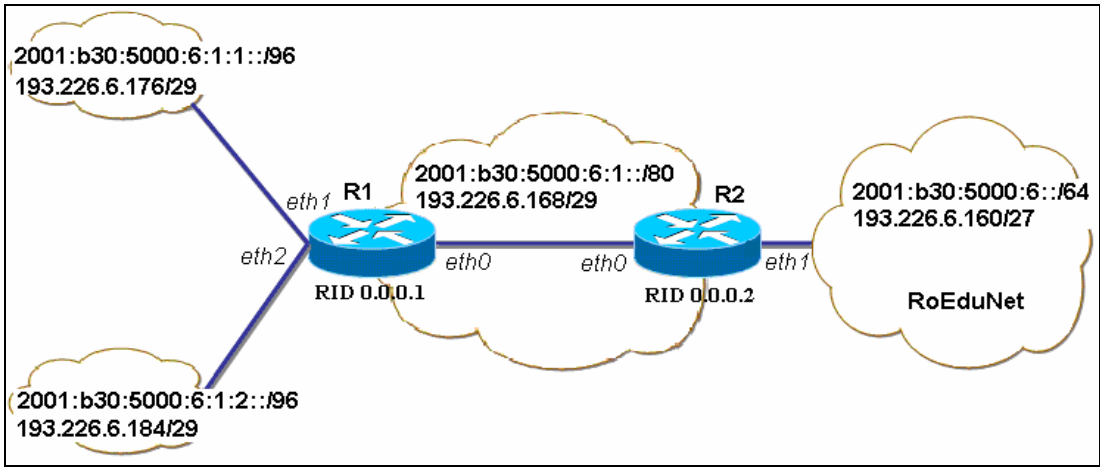

Figure 1. Unicast routing testbed

A. *Unicast Routing in IPv6*

The configuration file for the Zebra daemon is called `zebra.conf`, containing the router setup (hostname, password, enable password, IPv4/IPv6 interfaces addresses). Figure 2 presents an example for router R1 that has three interfaces (eth0, eth1, eth2).

```
hostname R1
password zebra
enable password zebra
!
interface eth0
 ip address 193.226.6.169/29
 ipv6 address 2001:b30:5000:6:1::169/80
!
interface eth1
 ip address 193.226.6.177/29
 ipv6 address 2001:b30:5000:6:1:1::177/96
!
interface eth2
 ip address 193.226.6.185/29
 ipv6 address 2001:b30:5000:6:1:2::185/96
```

Figure 2. R1's *zebra.conf* file

The routing daemons involved were *ripngd* and *ospf6d*. Each daemon had its own configuration file called **.conf*. Supposing the case of a single area OSPF (area 0.0.0.0), with the router-id for R1 being 0.0.0.1 (R2 has a router-id equal to 0.0.0.2), the configuration files applied to router R1 are presented in Figure 3 and 4.

```
hostname ripngd
password zebra
!
interface eth0
!
interface eth1
!
interface eth2
!
router ripng
 network eth0
 network eth1
 network eth2
 redistribute connected
```

Figure 3. R1's *ripngd.conf* file

```
hostname ospf6d
password zebra
!
interface eth0
 ipv6 ospf6 cost 1
 ipv6 ospf6 hello-interval 10
 ipv6 ospf6 retransmit-interval 5
 ipv6 ospf6 priority 1
!
router ospf6
 router-id 0.0.0.1
 interface eth0 area 0.0.0.0
 interface eth1 area 0.0.0.0
 interface eth2 area 0.0.0.0
 redistribute kernel
```

Figure 4. R1's *ospf6d.conf* file

The proper operation within zebra could be seen by analyzing the entries from the routing table or the results of the command *ping6*. For instance, the information got from R2 must contain the routes learned from R1 (either by RIPng, either by OSPFv3). No static routes have been previously configured, so if the routing protocol does not work properly the networks connected to eth1 and eth2 at R1 cannot communicate with the network connected to eth1 at R2.

There are two ways of analyzing the routing table: a) from the zebra daemon with the command: *show ip route;* b) from the Linux shell with the commands: *netstat -r* or *route –A inet6*. For a better understanding, we can capture the packets with the software packet analyzer called Ethereal. A failure situation must be provoked for further testing of our IPv6 testbed. Suppose one of the networks connected to router R1 will be disconnected either by unplugging the network cable, either by shutting down the interface. The time it takes for the new routing information to reach router R2 is actually the convergence time for the given routing protocol.

B. Unicast Routing in IPv4

The same procedures were applied for IPv4, except the types of daemons started, which were in this case *ripd* and *ospfd*. Figures 5 and 6 show the requested configuration files.

```
hostname R1
password zebra
!
interface eth0
 ip rip send version 1 (2)
 ip rip receive version 1 (2)
!
[…]
!
router rip
 version 1 (2)
 redistribute connected
 network eth0
 network eth1
 network eth2
```

Figure 5. R1's *ripd.conf* file

```
hostname ospfd
password zebra
!
interface eth0
 ip ospf cost 10
 ip ospf priority 10
!
[…]
!
router ospf
 ospf router-id 0.0.0.1
 redistribute connected
 network 193.226.6.160/29 area 0
 network 193.226.6.176/29 area 0
 network 193.226.6.184/29 area 0
```

Figure 6. R1's *ospfd.conf* file

Note that the two version of RIP (RIPv1 and RIPv2) can be configured thus resulting two routing scenarios. The differences presented in section II.1 are easy to be notices from the packets captured. First, the destination address is 255.255.255.255 (local broadcast) for RIPv1 and 224.0.0.9 (multicast) for RIPv2. Second, the version field and the routing information carried are different.

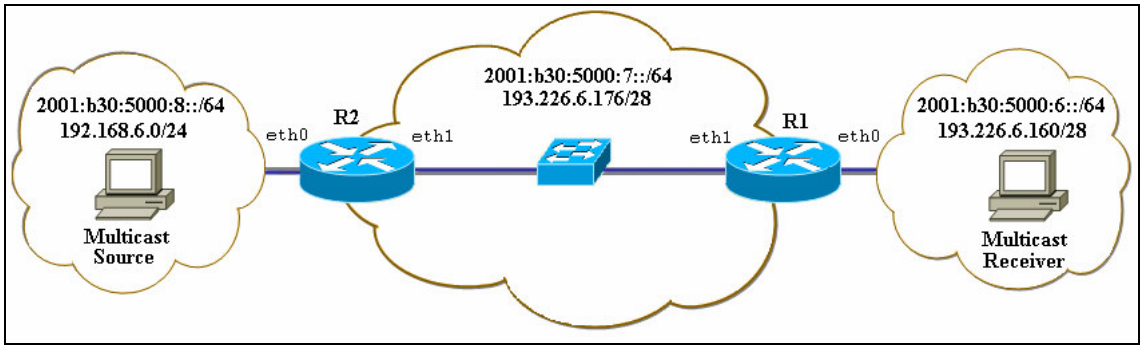

Figure 7. Multicast routing testbed

IV.2. Multicast Testbed

The topology presented in Figure 7 is not very complex, but it is an excellent testbed for the study of multicast routing. There are two multicast routers, R1 and R2 which operate under Fedora Core or FreeBSD depending on the experiment. We have also two machines which serve as multicast source and multicast receiver. The following subsections present the operations performed on the routers and hosts in order to enable multicasting and the problems that may arise.

A. Multicast Routing in IPv6

IPv6 multicast testing was performed using PIM because no other implementations were available. To operate one of the two PIM daemons, *pim6dd* or *pim6sd*, we first need to enable and configure IPv6 on the routers. Later we need to choose the unicast routing protocol and finally we can start multicast routing. The file */etc/rc.conf* is responsible for the IPv6 configurations and the starting of unicast-multicast routing daemons. Figure 8 below presents the R2 configuration file:

```
ipv6_enable="YES"
ipv6_gateway_enable="YES"
ipv6_network_interfaces="xl0 xl1"
ipv6_ifconfig_xl0="2001:b30:5000:8::2
prefixlen 64"
ipv6_ifconfig_xl1="2001:b30:5000:7::186
prefixlen 64"
ipv6_router_enable="YES"
ipv6_router="/usr/sbin/route6d"
mroute6d_enable="YES"
mroute6d_program="/usr/local/sbin/pim6dd"
```

Figure 8. R2's */etc/rc.conf* file

The IPv6 unicast routing daemon implementing RIPng protocol is called *route6d*. No configurations are needed for this daemon; the defaults assure the route exchange between R1 and R2.

The configuration files for the two multicast routing daemons are */etc/pim6dd.conf* and */etc/pim6sd.conf*. These files describe how the corresponding daemon

treats each interface on the system. When using *pim6dd* the defaults (or no configuration file present) activate PIM-DM on all interfaces, thus enabling multicast routing. Operating *pim6sd* requires some configuration, given by the special operation of the protocol.

```
phyint x10 mld_version any;
phyint x11 mld_version any;
cand_rp;
group_prefix ff15::5/128;
```

Figure 9. R2's *pim6sd.conf* file

The experiment is considered successful if the *m6receiver* program receives the packets from the source. Despite of all previous configuring the experiments have failed. The multicast daemons were started in debug mode, all the debug messages were examined, and it turned out that the router did not accept or did not receive the MLD messages sent by the receiver. Using Ethereal tool all the IPv6 packets from the network were captured. We discovered that the receiver was sending MLDv2 messages to the ALL-MLDv2-CAPABLE-ROUTES address ff02::16, in order to join the desired multicast group. The multicast router did not accept MLDv2 messages despite the *mld_version any* configuration was correct. There were two possible solutions: either to have an MLDv2 enabled router, either to have a receiver that uses MLDv1. The KAME code merged in FreeBSD supports only MLDv1, so the first solution would have been to compile a FreeBSD kernel and usertools (PIM daemons) from the original KAME kit. The other solution would have been to force the receiver to use MLDv1. We chose the later one, which meant compiling a 2.6.4 Linux kernel and configuring the *net.ipv6.conf.all.force_mld_version=1* in the sysctl.conf file, thus forcing MLDv1.

B. Multicast Routing in IPv4

The following IPv4 multicast routing protocols were tested: DVMRP, PIM-DM and PIM-SM. Figures 10 and 11 present the GateD configuration files for each protocol used on router R1

```
dvmrp on {
  interface "eth0";
  interface "eth1";
      };
igmpv3 on {
  interface
    "eth0"
      };
```

Figure 10. R1's *gated.conf* file

```
igmpv3 on {           x igmpv3 on {
  interface "eth0";   x   interface "eth0";
          };          x           };
pim on {              x pim on {
  interface "eth0";   x   interface "eth0"
  interface "eth1";   x     {mode dense; };
  static-rp 224.5.5.5x   interface "eth1"
    mask-length 32    x     {mode dense; };
    193.226.6.185;    x        };
        };            x
       ospf on {
         instance 2 {
             defaults { multicast-rib on;
                      };
             area 1.1.1.1 {
                   match 193.226.6.160
                   wildcard 0.0.0.15;
                   match 193.226.6.176
                   wildcard 0.0.0.15;
                         };
             router-id 10.0.0.1;
                   };
           };
```

Figure 11. R1's *gated.conf* file

We used OSPF as the underlying unicast routing protocol. The routes learned by OSPF are also placed in the multicast RIB (Routing Information Base), so that PIM could use them. Otherwise PIM cannot build the delivery tree from the source to the receiver.

V. CONCLUSIONS AND FURTHER WORK

This paper presented the setup of unicast routing protocols and of new coming multicast routing protocols for IPv6-based networks.

PIM was analyzed by comparing its implementations in both versions of IP. GateD Enterprise 2.0 running under Fedora Core 1 was used in IPv4 and FreeBSD's KAME supported IPv6-based trials. Other multicast routing protocol, such as DVMRP, was tested in IPv4 only, whilst MOSPF was not available at all due to lack of implementations.

Following the trials presented herein, it is for further work to determine the performances of multicast networks, according to new IETF recommendations. One of the parameters to be measured is join latency, the time it takes a host to receive the first multicast packet.

REFERENCES

[1] V. Dobrota, *Digital Networks in Telecommunications. Volume 3: OSI and TCP/IP.* Second Edition, Mediamira Science Publishers, Cluj-Napoca, 2003 (in Romanian)

[2] A. Rodriguez, J.Gatrell, J.Karas and R. Peschke, *TCP/IP Tutorial and Technical Overview*, IBM, August 2001

[3] ***, "Redistributing Routing Protocols", *Cisco Systems*, 2002

[4] ***, "Internetworking Technologies Handbook", *Cisco Systems,* 2002

[5] A. Tanenbaum, *Computer Networks,* Fourth Edition, Prentice Hall, 2003

[6] W. Parkhurst, *Cisco Multicast Routing & Switching,* McGraw-Hill, 1999

[7] B. Adams, E. Cheng, T. Fox, *Interdomain Multicast Solutions Guide,* Cisco Press, 2002
[8] C. Waitzman, *Distance Vector Multicast Routing Protocol* RFC 1075, November 1988
[9] S. Deering, *Host extensions for IP multicasting*, RFC 1112, August 1989
[10] D. Estrin, D. Farinacci, *Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification*, RFC 2362, June 1998
[11] R. Hinden, S. Deering, *IP Version 6 Addressing Architecture*, RFC 2373, July 1998
[12] B. Fenner, M. Handley, H. Holbrook, I. Kouvelas, *Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised),* draft-ietf-pim-sm-v2-new-09.txt, February 2004
[13] E. Nemeth, G. Snyder, T.R. Hein, *Linux Administration Handbook*, Prentice Hall 2002
[14] M. Lucas, *Absolute BSD The Ultimate Guide to FreeBSD,* No Starch Press 2002
[15] www.zebra.org
[16] www.nexthop.com
[17] www.kame.net