Evaluation of the Open-Source Implementation of PIM-SM

Jelena Veličković, Jelena Seović and Aleksandra Smiljanić

***Abstract*— PIM-SM is the routing protocol which describes the exchange of the network topology information in order to construct logical multicast trees. Multicast traffic is routed along these multicast trees in order to more efficiently utilize the network resources. In this paper, we evaluate capabilities of the open-source implementations which allow routing of multicast traffic: Quagga, and Pimd.**

***Index Terms*—Multicast, Quagga, PIM-SM, Pimd**

I. Introduction

Modern services that require high data flow, like IPTV, video on demand, video conferencing and distance learning are increasingly common on the Internet. Therefore, it is very important to use protocols which efficiently use the link capacity.

Multicast provides optimal utilization of network resources for communication in which a source host sends a data to a group of destination hosts. Even if we can separately send unicast packets to all destination hosts, there are many reasons why it is desirable to use multicast in this case. The first advantage of using multicast is the decrease of the network load [1]. The second advantage is that multicast sources do not have need to know topology of the entire network.

Multicasting is widely deployed in data centers, cable TV networks, and large corporations.

Multicast protocols use network resources optimally. Even if data needs to be delivered to a large number of receivers, using multicast protocols source needs to send only one copy of packets. The network nodes perform replication of the data packets in order to deliver it to all receivers. In this way, link loads close to a multicast source are significantly reduced. Source host generates multicast traffic which is defined by its multicast address. When the receiver join a multicast group which is defined with multicast group address, it actually begins receiving traffic. Transport of data is based on the multicast tree.

Today, there are several multicast protocols used. In case where multicast group members are densely distributed over the network, the most popular multicast protocols are DVMRP (Distance Vector Multicast Routing Protocol) [2] and PIM-DM (Protocol Independent Multicast-Dense Mode) [3]. PIM-SM (Protocol Independent Multicast-Sparse Mode) [4] multicast protocol performs better when group members are sparsely distributed.

DVMRP was derived from RIP (Routing Information Protocol). DVMRP creates multicast trees based on the information on the previous-hop back to the source. The first packet of multicast messages sent from a particular source to a particular multicast group is flooded across the network. If a router does not wish to be part of a particular multicast group, it sends a Prune message along the RIP path to the multicast source.

PIM-DM is very similar to DVMRP, but there are differences between these two protocols. PIM-DM requires the presence and cooperation with unicast routing protocol but the advantage is that is independent of any concrete unicast routing protocol. Unicast protocol is used for finding routes back to the source node. This is different from DVMRP which uses RIP exchange messages to create its unicast routing table. PIM-DM supports only source trees and creates a shortest path distribution tree. This is efficient in cases in which there are receivers on every subnet in the network.

Unlike PIM-DM protocol, in PIM-SM protocol routers need to explicitly announce their desire for receiving multicast messages of certain multicast groups [5]. Each multicast group has a single RP (Rendezvous Point) router at any given time. Every router that wants to receive multicast messages from a certain group needs to send a Join message to the RP of that group. Multicast packets of the group are transmitted via RP for that group. Each host has a DR (Designated Router) which is the router connected to the same subnetwork with the highest IP address. When a DR receives an IGMP message indicating the membership request of a host to a certain group, the DR finds the RP and forwards a unicast Join message to it. There are several proposed improvements of PIM-SM protocol [6-7].

In this paper we will describe our testing results of integration of the Quagga open source routing software for unicast packet forwarding and Pimd open source daemon which implements PIM-SM protocol for multicast packet forwarding. We created the test network environment using virtual machines which was built with LXC (Linux Containers) virtualization software.

This paper is organized as follows. The second section describes architecture of Quagga and Pimd. The third section describes the emulated network topology which is used for

Jelena Veličković is with the Inovation Center, School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia (e-mail: jelena.velickovic@ ic.etf.rs).

Jelena Seović is with the Inovation Center, School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia (e-mail: jelena.seovic@ ic.etf.rs).

Aleksandra Smiljanić is with the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia (e-mail: aleksandra@ etf.rs).

testing. Tested scenarios and test results are given in the fourth section. The paper is concluded in the last section.

II. Quagga Software And Pimd Daemon

Quagga is an advanced open source routing software package that implements common TCP/IP based routing protocols [8]. In addition to traditional IPv4 routing protocols, Quagga also supports IPv6 routing protocols. Quagga provides routing functions so machine on which it is installed can exchange routing information with other routers using routing protocols. Quagga uses this information to update the kernel routing table and according to this table redirects data packets. Platforms which have support for Quagga are gnu/Linux and BSD (FreeBSD, NetBSD and OpenBSD). It is tested and confirmed that each of these platforms and Quagga work correctly together. Quagga also may work on Solaris and MAS OSX platforms, but with some effort. For correct work of Quagga it is also important which compiler is used. The compilers whose interoperability with Quagga is confirmed are GCC (GNU Compiler Collection), Clang and ICC (Intel C++ Compiler). It should be noted that only recent versions of this compilers are well-tested.

A common feature of all traditional routing softwares is that they are designed as a single process program which provides all of the routing protocol functionalities. Quagga takes a different approach, it is composed of several daemons, one per routing protocol and another one called Zebra acting as the kernel routing manager. Zebra provides kernel routing table updates and redistribution of routes between different routing protocols. System architecture of Quagga is shown in Fig. 1. It consists of Zebra, Ripd, Ripngd, Ospfd, Ospf6d, Bgpd, Isisd and Babeld daemons. Ripd and Ripngd implements RIP protocol for IPv4 and IPv6 respectively. These daemons are tested and they have full functionality. For implementation of OSPF protocol for IPv4 and IPv6 Ospfd and Ospf6d are used, respectively. Correct functionality of Ospfd is confirmed, but functionality of Ospf6d is not full, it is known that it has some errors. Bgpd daemon provides BGP protocol for IPv4 and IPv6 and its work is tested and confirmed. Isisd daemon implements IS-IS, and it has support for IPv4 and IPv6, but its functionality is not completely tested. Babeld daemon implements Babel protocol, its functionality is not fully developed.

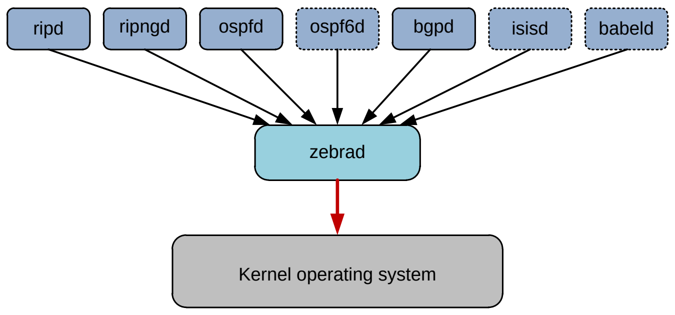


Fig. 1. Quagga architecture

This multi-process architecture provides extensibility, modularity and maintainability. Each daemon is independent from others and each daemon has its own configuration file. If we do not need all implemented protocols, we need to run only Zebra daemon and the protocol daemons associated with routing protocols which we want to use. Also, multi-process architecture of Quagga provides simple adding of new routing protocol daemons without affecting any other software. At the same time this architecture also brings many configuration files and terminal interfaces. In order to resolve this problem Quagga provides command-line tool called Vtysh. Vtysh is integrated user interface which connects to each daemon with UNIX domain socket and then works as a proxy for user input. On that way Vtysh allows that all processes can be monitored and their configuration modified from one place.

Currently Quagga supports only unicast routing protocols. However there are several daemons which implement multicast routing protocols. One of them is Pimd daemon. Pimd is a lightweight stand-alone PIM-SM v2 multicast routing daemon [9]. It implements PIM-SM protocol according to RFC 2362. Pimd provides IP multicasting, regardless of what unicast routing protocol is in use. Multicast protocols use existing routing tables which are created by unicast routing protocols. For Pimd, PIM support in kernel needs to be enabled.

III. Test Environment

In order to test functionality of Pimd daemon, the test network environment was created using the LXC virtualization software. LXC is method at the operating system level which enables the creation of multiple isolated virtual containers which act like regular hosts and have their own processes.

We implemented the test network environment using only one PC. PC uses Ubuntu 12.10. Linux operating system. Virtual network was emulated on them. Using LXC virtualization software we created nine virtual containers. We created required numbers of virtual Ethernet ports on each of virtual containers and connected them using bridges in order to make the network topology which is shown in Fig. 2.

As shown in Fig. 2. virtual network hosted by PC consist of five virtual routers, three client hosts and one server host. We gave routing functionality to some of virtual containers installing Quagga routing software on them. We ran Zebrad and Ospfd daemons in Quagga. In order to enable multicast functionality we installed Pimd daemon additionally. In order to send and receive multicast control and data traffic on server and client hosts, the Iperf open source networking software was installed. Iperf is often used for TCP and UDP packet streaming and measuring performance of the network. It has both, client and server functionality. Iperf can be used on various platforms, including Linux, Unix and Windows.

In the created virtual network environment, we were able to track forwarding of multicast control and data packets and to observe multicast routing table, which were essential for testing of the Pimd daemon.

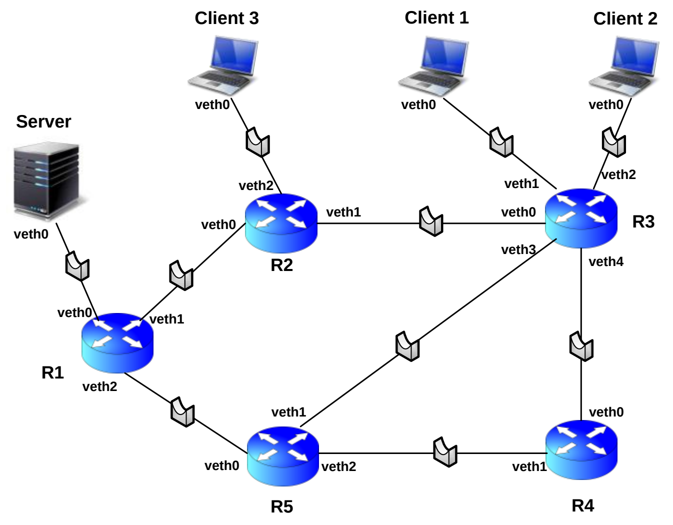


Fig. 2. Network topology

IV. Results of the Performed Tests

In this section we will describe scenarios that we tested in the created network and results of those tests.

The first scenario is when RP router is statically configured, server sends data and clients are interested in receiving these data. In each router, in the Pimd configuration file, we set the static address of a RP router. For correct network operation it is necessary that each router is configured with the same RP router static address. For the RP router we chose veth2 interface of the R1 router. Server sends data to the multicast address 224.0.2.3 using UDP protocol. Fig. 3. represents PIM Join messages which clients send in order to receive multicast data. These Join messages create a RP tree and this tree determines the paths along which data will flow.

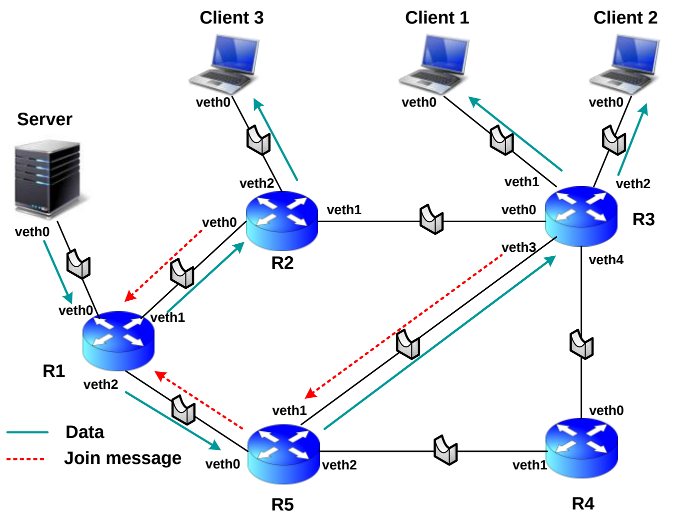


Fig. 3. Data and Join messages flow

Since client 1 is interested in traffic of the multicast group 224.0.2.3, it sends IGMP message of type Report for this group to the router R3. It sends this message to the R3 router because it is its DR router. As a response to this message, router R3 in multicast routing table creates (*,G) state, where G corresponds to the 224.0.2.3 multicast group. This multicast routing table entry is shown in Fig. 4. As we can see, this entry has WC and RP flags. WC flag denotes that any source may send data for the 224.0.2.3 multicast group. The RP flag denotes that the RP tree is created with this entry.

The router R3 sends Join message to the router R5. With this message a (*,G) state with WC and RP flags in the router R5 is created. Further, the router R5 sends a Join message to the router R1 (which is the RP router). In the R1 router, a (*,G) state for the 224.0.2.3 multicast group with WC and RP flags is also created. In this way, the RP tree for client 1 and group 224.0.2.3 is created.

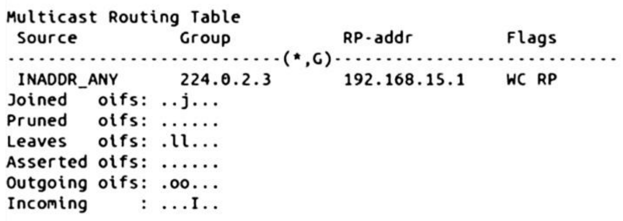

```
Multicast Routing Table
 Source           Group           RP-addr          Flags
---------------------------(*,G)---------------------------
 INADDR_ANY       224.0.2.3       192.168.15.1     WC RP
Joined   oifs: ..j...
Pruned   oifs: ......
Leaves   oifs: .ll...
Asserted oifs: ......
Outgoing oifs: .oo...
Incoming     : ...I..
```

Fig. 4. (*,G) entry in the multicast routing table of the router R3

Considering that the client 2 is connected to the network via the same router as the client 1, paths of Join messages and created states are identical to the paths and states of client 1.

DR router for client 3 is the R2 router. So, client 3 sends IGMP message of type Report for the 224.0.2.3 group to router R2. This initiates the creation of a (*,G) state in the R2 router for the 224.0.2.3 multicast group with WC and RP flags. After this, the R2 router sends Join message to the router R1, router R1 creates proper state and the RP tree for client 3 and multicast address 224.0.2.3 is formed.

It should be noted that routers store also (S,G) entries beside (*,G) entries. These entries correspond to the veth0 interface of the server (it is the address of an interface from which data is sourced) and 224.0.2.3 multicast address. They are created after the server sent the first packets of data and DR routers of clients discovered its identity. The purpose of these entries is the path optimization between server and clients. (S,G) states create a source-specific tree. If the source-specific tree offers more optimal path than the RP tree, the data will flow along the source-specific tree. In our case, the RP tree and source-specific tree offer identical paths.

Server sends only one data replica to the RP router. RP router replicates the data and sends one replica to the veth1 interface and another to the veth2 interface. Table I shows the traffic statistics of the router R1 which we obtained using Wireshark software. On the interface veth0, the average data rate is 1.0792 Mb/s. On the interfaces veth1 and veth2 the average data rates are 1.0791 Mb/s and 1.0792 Mb/s, respectively. This shows that the multicast traffic is successfully replicated.

TABLE I
ROUTER R1 TRAFFIC STATISTICS

Interface (traffic)	Veth0 (incoming)	Veth1 (outgoing)	Veth2 (outgoing)
Average number of packets per second	89.318	89.512	89.448
Average size of packet [B]	1510.329	1506.967	1508.09
Average data flow [Mb/s]	1.0792	1.0792	1.0792

The second scenario is similar to the first, but in this case, there are no clients who are interested in receiving multicast data. The traffic comes to the R1 router (which acts as a RP router), which did not receive any Join messages, and so it does not forward traffic to the other routers. In the R1 router, (S,G) state is created, where S represents the interface from which data is sent and G is the multicast address for which data is bound. The rest of the network routers do not have any multicast entries.

The third scenario represents the first scenario in which the failure has happened. We tested a case in which the link between router R1 and router R2 is broken. This failure does not have impact on the traffic of clients 1 and 2. However, the traffic for client 3 is temporarily stopped. It is stopped until the new RP tree is created for client 3 and multicast group 224.0.2.3. Now, the RP router sends data to the client 3 via the veth2 interface and data flows through the routers R5, R3 and R2. Fig. 5. represents a data flow for clients in the case of failure which we tested.

In the fourth scenario, the RP router is chosen via bootstrap mechanism, server sends data and clients are interested in that data. Routers which are not candidates for the BSR (BootStrap Router) and the RP router do not have additional configuration related to bootstrap mechanism.

We chose the R1 and R5 router as potential candidates for the BSR and RP routers. In router R1, we set priority 1 for the RP candidature and priority 15 for the BSR candidature on its interface veth2 (its address is 192.168.15.1). We also chose interface veth1 of the R5 router (its address is 192.168.16.1) for the RP and BSR candidatures. On this interface, we set priority 20 for the RP candidature and priority 5 for the BSR candidature. The router with the highest priority is chosen for the BSR router, and the router with lowest priority is chosen to be the RP router. So, in our case, the router R1 is chosen to act as a BSR router. The R5 router sends its candidature for the RP router to the R1 router. Router R1 advertise the set of the RP candidates via a bootstrap message. This bootstrap message is sent to all routers in network and each router locally runs algorithm for the RP router election. The RP priority of R1 is lower than R5 priority and so the router R1 is chosen to be the RP router.

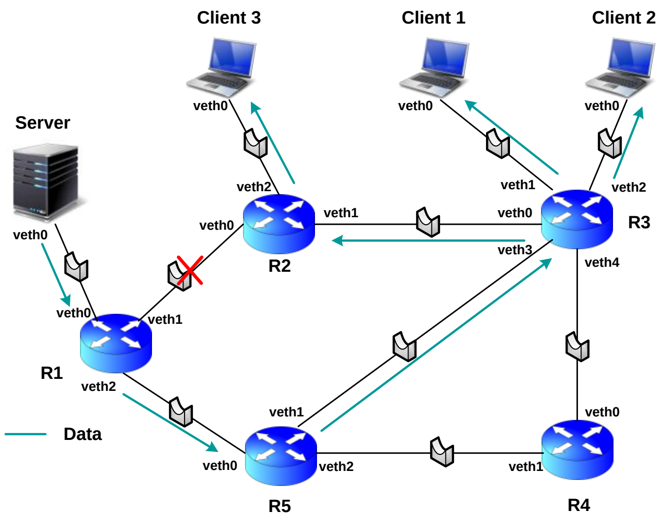
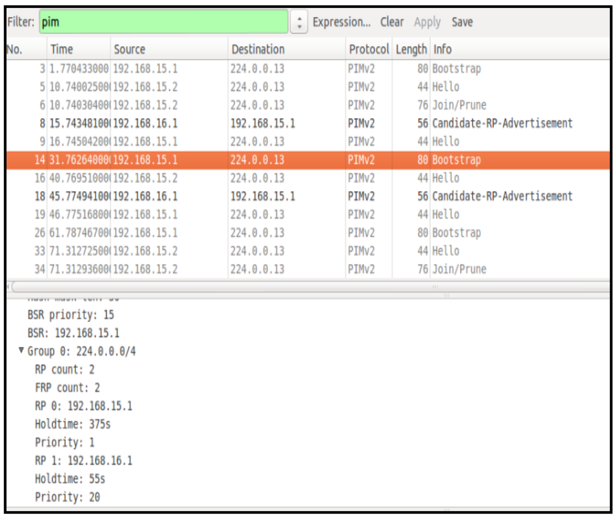


Fig. 5. Data flow in the case of link failure

Fig. 6. shows sending and receiving PIM messages on the R5 router interface veth0. These messages are observed with Wireshark. It can be seen that router R5 sends its candidature to address 192.168.15.1 (which is the address of the veth2 interface of the router R1). Also, the content of the bootstrap message that the R5 router received can be seen in Fig. 6. This bootstrap message comprises the information about RP candidates (address, priority, hold time).



```
No.  Time         Source        Destination   Protocol Length Info
  3  1.770433000  192.168.15.1  224.0.0.13    PIMv2       80  Bootstrap
  5  10.74002500( 192.168.15.2  224.0.0.13    PIMv2       44  Hello
  6  10.74030400( 192.168.15.2  224.0.0.13    PIMv2       76  Join/Prune
  8  15.74348100( 192.168.16.1  192.168.15.1  PIMv2       56  Candidate-RP-Advertisement
  9  16.74504200( 192.168.15.1  224.0.0.13    PIMv2       44  Hello
 14  31.76264000( 192.168.15.1  224.0.0.13    PIMv2       80  Bootstrap
 16  40.76951000( 192.168.15.2  224.0.0.13    PIMv2       44  Hello
 18  45.77494100( 192.168.16.1  192.168.15.1  PIMv2       56  Candidate-RP-Advertisement
 19  46.77516800( 192.168.15.1  224.0.0.13    PIMv2       44  Hello
 26  61.78746700( 192.168.15.1  224.0.0.13    PIMv2       80  Bootstrap
 33  71.31272500( 192.168.15.2  224.0.0.13    PIMv2       44  Hello
 34  71.31293600( 192.168.15.2  224.0.0.13    PIMv2       76  Join/Prune

   BSR priority: 15
   BSR: 192.168.15.1
 ▾ Group 0: 224.0.0.0/4
    RP count: 2
    FRP count: 2
    RP 0: 192.168.15.1
    Holdtime: 375s
    Priority: 1
    RP 1: 192.168.16.1
    Holdtime: 55s
    Priority: 20
```

Fig. 6. PIM messages which are exchanged via veth0 interface of the router R5

We observed the flow of control messages and data and it was identical as in the first scenario. We modified the fourth scenario so the RP priority of the R5 router is lower than the

RP priority of the R1 router, and, consequently, router R5 was chosen to be an RP router. In this case, multicast traffic does not reach all the clients which expressed their interest to receive it. Actually, only client 3 receives the multicast traffic, since it receives directly from server along the source-specific tree. According to the PIM-SM v2 protocol, if a server is not directly connected to the RP router, multicast packets should be encapsulated and sent as unicast traffic to the RP router. By observing traffic with Wireshark, we noticed that the data sent from server did not reach the RP router.

V. Conclusion

In this paper, we tested interoperability between the Pimd multicast routing daemon and the Quagga routing software. It has been shown that Pimd correctly exchanges information with Quagga. Simulation of sending data to multicast address from a server and sending requests for receiving data from clients, confirmed exchange of control packets in a way that is prescribed by the PIM-SM protocol. We also confirmed proper creation of multicast routing tables in network routers. Efficiency of multicasting has been demonstrated as well - server sends only one replica of each multicast packet, and the routers of the multicast tree replicate the packet only if it is needed. We also validated the functionality of PIM-SM in the case of link failures in network. When a link that belongs to a multicast tree fails, the multicast tree is reconfigured so that it does not include the failed link.

In the case when a multicast source is not connected directly to the RP router, the Pimd daemon did not work correctly. In that case, multicast packets are not forwarded to the RP, and, they, consequently did not reach the clients. We believe that the problem lies in encapsulation of multicast traffic by server, and we plan to solve this problem in our future work.

Acknowledgment

We express gratitude to the Serbian Ministry of Education, Science and Technology, Telekom Srbija and Informatika for their financial support.

References

[1] S. Ueno, T. Kato, K. Suzuki, “Analysis of Internet multicast traffic performance considering multicast routing protocol”, *Proceedings 2000 International Conference on Network Protocols*, Osaka, Japan, November, 2000

[2] https://tools.ietf.org/html/rfc1075

[3] https://tools.ietf.org/html/rfc3973

[4] https://tools.ietf.org/html/rfc2362

[5] T. Bartczak, P. Zwierzykowski, “Validation of PIM DM and PIM SM protocols in the NS2 network simulator”, *AFRICON 2009*, Nairobi, Kenya, September 2009.

[6] T. Čičić, S. Gjessing, O. Kure “An Improved PIM-SM Tree Recovery Algorithm”, *High Performance Switching and Routing, 2001*, Dallas, USA, May 2001.

[7] Weicheng Xiong, LibingWu, Shengchao Ding, Chanle Wu, “Research on PIM-SM Multicast Routing Improvement”, Computer Design and Applications (ICCDA) 2010, Qinhuangdao, China, June 2010.

[8] http://www.nongnu.org/quagga/

[9] http://troglobit.com/multicast-howto.html