Implementation of RSVP Protocol in Quagga Software

Teodora Komazec, Aleksandra Smiljanić, Hasan Redžović, and Andreja Radošević

***Abstract* — Multimedia applications are already most popular on the Internet, and they are much better served with bandwidth and delay guarantees. Providing support for multimedia applications over the Internet is a difficult task because they require more bandwidth than standard applications. Aside from that, multimedia applications are real time so it is essential to guarantee certain capacity. RSVP protocol provides the basic quality of service on the Internet such as bandwidth reservations. This paper presents implementation of RSVP protocol that supports basic functionalities in Quagga Routing Suite software. Our implementation is meant to serve as a base for adding new functionalities. We have used modern concept of virtualization to provide environment for testing. In this paper, we present implementation and routing of Resv messages. Resv messages are routed based on IP lookup that uses the longest prefix matching. We validated our RSVP implementation through its testing in virtual environment. Guidelines for future improvements and upgrades to RSVP-TE protocol are recommended.**

***Keywords* — RSVP protocol, RSVP – TE protocol, Quagga Routing Suite software.**

I. Introduction

RSVP (Resource ReSerVation Protocol) is a protocol designed to support integrated services Internet. RSVP [1] allows routers and hosts to request a specific quality of service for its data flows. It is a receiver oriented protocol, which means that the receiver initiates and maintains reservations. Data is transmitted in the opposite direction, from a sender to possibly multiple receivers. Integrated services Internet is an architecture that includes support for best-effort and real-time services. One of the solutions for real-time services over Internet is to classify traffic, allocate specific priority for different data flows and make reservations for the high priority traffic.

In addition, RSVP is a protocol that allows setting up MPLS (Multiple Protocol Label Switching) tunnels. These tunnels use RSVP to set up, maintain and tear down connection. MPLS helps reduction of the lookup tables which are becoming critical with the increasing number of things on the Internet. Lookup based on MPLS labels is much faster and computationally less intensive than lookup based on IP addresses. LDP is another protocol for establishing MPLS tunnels, that uses paths provided by the underlying routing protocols, while RSVP could use an arbitrary given path.

RSVP can be extended to RSVP-TE [2] which provides support for traffic engineering [3]. Traffic engineering includes techniques for optimizing the performance of telecommunication networks. Importance of traffic engineering is rapidly growing because current IGPs (Interior Gateway Protocols) use the shortest path algorithm to calculate routes which can result in favorizing certain links and consequently their congestion.

RSVP alone is not sufficient to provide support for the quality of service across network. RSVP has to cooperate with other protocols, such as OSPF – TE or PCE, which are needed for the creation of traffic engineering routing topology. This topology can be significantly different from the conventional network topology. For example, OSPF – TE speaking router can calculate path from source to destination while applying a set of constraints. These constraints can be defined through bandwidth that should be available on links. RSVP requests are further used to establish reservations in network nodes along the calculated path.

RSVP is designed to work with current and future routing protocols. Routing protocols are responsible for correct forwarding of data packets while RSVP provides requested quality of service. RSVP consults local databases to obtain routes. Although RSVP occupies the place of transport protocol in the TCP/IP stack, it is considered as an Internet control protocol because RSVP does not transport application data.

RSVP uses the following types of messages: Path, Resv, PathTear, ResvTear, PathErr, ResvErr and ResvConfirm. Two basic types of RSVP messages are Path and Resv.

RSVP uses the "soft state" approach in establishing and managing RSVP states. Path and Resv messages create or periodically refresh RSVP states. Once created, RSVP state can be explicitly deleted with PathTear or ResvTear messages or it can expire if an appropriate refresh message is not received.

Paper received March 27, 2019; revised May 11, 2019; accepted June 02, 2019. Date of publication July 31, 2019. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Grozdan Petrović.

This paper is revised and expanded version of the paper presented at the 26th Telecommunications Forum TELFOR 2018 [8].

This work has been supported in part by the Serbian Ministry of Education, Science and Technology as a part of project TR32022.

Teodora Komazec is with the School of Electrical Engineering, University of Belgrade, Bul. kralja Aleksandra 73, 11120 Belgrade, Serbia (phone: 381-64-9676308, e-mail: tea.k94@gmail.com).

Aleksandra Smiljanić is with the School of Electrical Engineering, University of Belgrade, Bul. kralja Aleksandra 73, 11120 Belgrade, Serbia (e-mail: aleksandra@etf.rs).

Hasan Redžović is with the Innovation center of School of Electrical Engineering, University of Belgrade, Bul. kralja Aleksandra 73, 11120 Belgrade, Serbia (e-mail: hasan.redzovic@ic.etf.bg.ac.rs)

Andreja Radošević is with Stellar Labs, USA (e-mail: andreja.radosevic@sansresearch.com).

Sender host sends a RSVP Path message downstream along the routes learned from routing protocols. Path messages follow the path of the data and create "path state" in each network node along the way. Path state contains at least the information about IP address of a previous hop node, which is used for routing Resv messages hop – by – hop in the reverse direction. Receiver hosts send Resv messages as a response to the received Path messages. Resv messages create and maintain "resv state" in each node along their way. Resv messages travel in reverse direction of the data. They carry information about the requested quality of service. When sender host receives Resv message, reservation is established, so sender host can send application data.

Two local modules process RSVP quality of service requests. Those are "admission control" and "policy control". Admission control examines whether a network node has sufficient available resources while policy control determines if user has a permission to perform reservation.

RSVP – TE is an extension of RSVP protocol for traffic engineering. Five new objects are defined but only LABEL_REQUEST and LABEL are mandatory for RSVP – TE.

RSVP – TE provides the following additional capabilities in comparison with RSVP: establishing LSP tunnels with or without quality of service requirements, dynamically rerouting established LSP tunnels in case of network failures, congestion or bottlenecks and loop detection.

There are different open-source suites for routing protocols: Quagga [4], Vyos, Bird [5], Xorp. Quagga is used in data centers, while Bird is widely used in IXPs (Internet eXchange Points) as its BGP protocol is optimized [6]. Quagga can also be used with virtualization to achieve simulation of a network or in situations where we need more resources than standard routers can offer [7]. These suites do not provide RSVP protocol, OSPF–TE or PCE protocols, which they are needed for deterministic quality of service that is required, or desirable, for growing multimedia applications. This paper is an extension of the paper presented at Telfor conference [8]. This extended version includes further development and more elaborate testing of the RSVP protocol.

II. Quagga Routing Suite

Quagga routing suite is a package of Unix/Linux software that provides routing based on the TCP/IP stack. Quagga supports the following routing protocols: RIP (Routing Information Protocol), OSPF (Open Shortest Path First), BGP (Border Gateway Protocol) and IS – IS.

Quagga has one core daemon, zebra, which acts as a routing manager. Other daemons are protocol daemons and they implement routing protocols within Quagga. Every daemon has a configuration file and interface. Protocol daemons are controlled by zebra daemon. Zebra daemon communicates both with protocol daemons and kernel. One of the tasks of zebra daemon is to update kernel's routing table with the information obtained from protocol daemons. Therefore, zebra connects the control plane to the data plane. Quagga architecture is shown in Fig. 1 [7].

Today, market is controlled by big companies. Routing devices are constrained by design to meet specific customer needs. Quagga allows research to explore ideas that are not present on the market. It is a flexible solution that offers implementation of new protocols and extension of currently implemented protocols. Many companies have already integrated Quagga into commercial routing devices. As we have mentioned, RSVP is not implemented in Quagga.

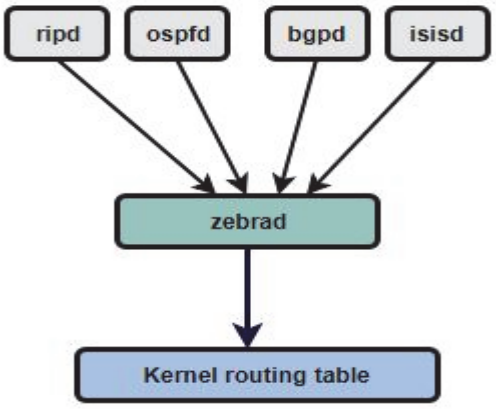

Fig. 1. Quagga architecture [7].

III. Implementation

Three phases are planned in our implementation. The first phase includes generating and receiving messages, defining and establishing states. The second phase represents integration of RSVP with routing protocols. The third phase is implementation of an extension needed for RSVP – TE. In this paper, we implement and present the basic functionalities of RSVP.

Three groups of data structures were used for implementation of the RSVP protocol. The first group represents RSVP (and RSVP-TE) objects that are defined by [1] (and [2]). The second group includes RSVP (and RSVP-TE) messages while the third group represents RSVP states.

All RSVP messages have a common header. A common header has a field that carries information about message type. Each type of RSVP messages consists of a specific set of variable length objects. Some objects are mandatory while others are optional. RSVP – TE messages include some modified and some new objects in comparison with RSVP.

The most important objects that we implemented are the following: SESSION, RSVP_HOP, FLOWSPEC and SENDER_TEMPLATE. The main fields of a SESSION object are: IP destination address of a session, destination port and protocol identifier for data flow. RSVP_HOP object carries the IP address of a node that sent the message. In the case of a Path message, name of the field that carries this address is PHOP and in the case of Resv message it is NHOP. SENDER_TEMPLATE, among other fields, has a field that carries IP address of a port from which data will be sent. The FLOWSPEC object is always included in Resv message and its purpose is to define QoS requirements.

RSVP states are defined through different fields. The path state consists of the following fields: destination IP address, sender's IP address, sender's port number and the IP address of a previous hop node. The IP address of a previous hop node is necessary for routing Resv messages. Sender's IP address can be used if an error occurs during processing of a Path message. PathErr message is sent on

that address. In the case of RSVP – TE, path state has the following additional data: tunnel ID and identifier of the layer 3 protocol using the path. Tunnel ID remains constant over the life of the tunnel. Tunnel ID along with source's IP address and destination IP address uniquelly identifies TE tunnel.

Resv state consists of the following fields: sender's IP address, sender's port and requested capacity for data flow. Information about sender's IP address can be used to determine whether user has a permission to request resources. In the case of RSVP – TE, resv state has additional tunnel ID and label field.

RSVP (and RSVP - TE) messages are sent as raw IP datagrams with protocol number 46. Therefore they are generated using raw network sockets. Raw network sockets enable applications to directly access protocols at lower layers of TCP/IP stack.

RSVP (and RSVP-TE) messages have: MAC header, IP header, RSVP header and payload. This kind of data, that is wrapped in various headers, is called network packet. Upon the arrival of IP packets, network node starts to process received packets. If the protocol number is not set to be 46, network node skips further processing of packet. If network node concludes that the received packet is indeed RSVP, processing will continue. Node, then, examines whether message is Path or Resv and applies appropriate processing. After that node extracts information from the received message, which is needed to establish the RSVP state. When the state is established, node creates txt file that stores the extracted information.

Fig. 2 illustrates the process of generating RSVP messages, their reception and processing to create RSVP states.

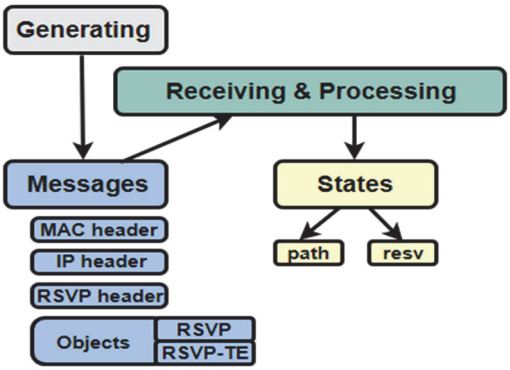

Fig. 2. Implementation of RSVP protocol.

RSVP is meant to work alongside routing protocols. Zebra daemon uses Zebra Protocol to communicate with protocol daemons. Zebra Protocol is a streaming protocol that has a common header. A common header has a command field that carries identifier of one of Zebra Protocol commands. These commands initiate actions such as adding, deleting and configuring interfaces, adding and deleting addresses, redistributing routes to and from interface. Therefore, protocol daemons can request and send information about interfaces, routing states, nexthop – validations to and from zebra daemon [4]. They communicate their best routes to zebra, which, then, calculates the best route across protocols for each prefix. This information is stored in RIB (Routing Information Base) which resides inside zebra. RIB specifies in which way routers are connected to one another [9]. Zebra, then, communicates with FIB (Forwarding Information Base) which is in the kernel, so that kernel could forward packets according to the Quagga calculated routes.

Resv messages are routed hop by hop using information from path state. Before intermediate router starts to generate Resv messages, it has to look at the information that is written in path_state.txt and resv_state.txt files. IP lookup is, then, performed to determine the interface through which, the packet should be forwarded from the IP address of the previous hop node toward the router in question. In this paper, we used the longest prefix match algorithm for the IP lookup. Name of the interface is written into the path_interface.txt file. This information is used to generate Resv message from that interface. Sender's address of the Resv message is the IP address of a node that has sent the message, while capacity **is** the same as capacity from resv_state.txt file.

IV. Testing

We will use virtual routers for testing RSVP implementation. Virtualization provides a solution for companies to increase productivity while reducing IT infrastructure expenses. Virtualization is a method that allows to create multiple simulated systems from a single, physical hardware system. In this way companies reduce energy consumption and costs that would otherwise be spent on additional hardware. Physical hardware is called host, while systems that use its resources are called guests.

There are many different types of virtualization but for this paper was used OS (Operating System) virtualization. OS virtualization is a technology that alters a standard operating system so that it can run different applications handled by multiple users at a same time. Capability of virtualization is a part of host's operating system. Fig. 3 illustrates the concept of OS virtualization. The downside of this approach is that all guests have to install the same operating system, however it is a fast and efficient solution. OS virtualization is used by: Linux-VServer, FreeBSD jails, OpenVZ, Docker, LXC (Linux Containers) etc [10]-[11].

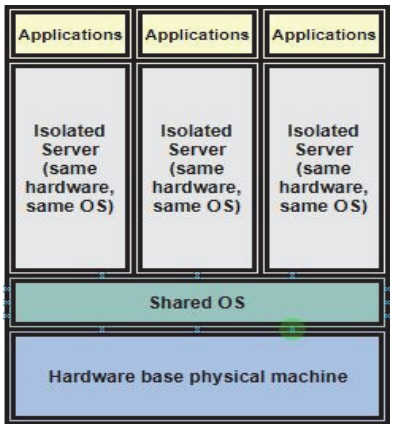

Fig. 3. Concept of OS virtualization.

LXC [12] was used for creating virtual containers. It is an open source software. LXC containers are considered to be a transition between chroot and virtual machine. While chroot on Unix operating systems changes root directory for the current running process and its children, LXC, although similar, offers more isolation and provides virtual environment that has its own process. Network for testing

was established by creating six virtual containers, configuring the containers, and installing Quagga and other programs on them. All software routers in Fig. 4 are using the OSPF protocol. IP addresses that are assigned to router interfaces are presented in table 1. Routers use the OSPF protocol in order to communicate with each other and learn network topology. OSPF is a link–state routing protocol which means that routers exchange information about state of their links. Because of this, OSPF can quickly detect changes in network topology.

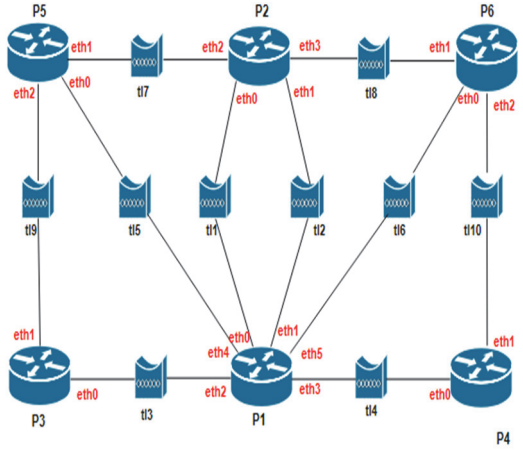

Fig. 4. Network topology.

TABLE 1: IP ADDRESSES ASSIGNED TO ROUTERS INTERFACES.

ROUTER	eth0	eth1	eth2	eth3	eth4	eth5
P1	192.168.1.1	192.168.2.1	192.168.3.1	192.168.4.1	192.168.5.1	192.168.6.1
P2	192.168.1.2	192.168.2.2	192.168.7.1	192.168.8.1	-	-
P3	192.168.3.2	192.168.8.1	-	-	-	-
P4	192.168.4.2	192.168.9.1	-	-	-	-
P5	192.168.5.2	192.168.7.2	192.168.8.2	-	-	-
P6	192.168.6.2	192.168.8.2	192.168.9.2	-	-	-

Fig. 5 shows the screen after generation of Path and Resv messages. It shows how many RSVP packets are sent and received through each interface. Interfaces are represented with their indices. Upon the arrival of a packet, the RSVP module examines whether a received packet is RSVP. If a received packed is the RSVP packet, the module continues with processing, if not, it skips processing of that packet. Fig. 6 shows that a received packet has a protocol number equal to 89, which is different from the RSVP protocol number. Therefore, the router RSVP module will skip processing of that packet.

```
Tx total = 324777 and Rx total = 0

501.257219 main [198] Tx Mpps 322525 and Rx Mpps 0, on port Tx 0 and Rx 0
501.257239 main [202]

Tx total = 322525 and Rx total = 0

502.257335 main [198] Tx Mpps 319635 and Rx Mpps 0, on port Tx 0 and Rx 0
502.257361 main [202]

Tx total = 319635 and Rx total = 0
```

Fig. 5. Generation of Path/Resv messages.

```
root@P3:/# sudo ./quagga/rsvp4d/packet_processor_te
Opening device: eth0
Total packet available: 82 bytes
Expected packet size: 82 bytes
IP header length (IHL) in bytes: 20
Protocol number of next encapsulated protocol is: 89
Not an RSVP packet. Skipping...
root@P3:/#
```

Fig. 6. Receiving a network packet.

In the test shown in Fig. 7 router P1 was using eth0 interface to communicate with router P2. Router P2 received a message and after concluding that the received message is RSVP, it proceeded with processing. Router P2 examined whether a message is Path or Resv and it found out that the received message is Path. Fig. 8 shows extracting of information that is necessary for establishing path state. Content of path_state.txt file is shown in Fig. 9. Fig. 9 shows that path_state.txt file has the following information: destination IP address, tunnel ID, sender's IP address, sender's port number, IP address of a previous hop node and identifier of layer 3 protocol.

```
root@P2:/# sudo ./quagga/rsvp4d/packet_processor_te
Opening device: eth0
Total packet available: 396 bytes
Expected packet size: 396 bytes
IP header length (IHL) in bytes: 20
Protocol number of next encapsulated protocol is: 46
Type is 1
Path packet length in bytes: 257

0: 11
1: 01
2: 12
```

Fig. 7. Receiving a Path message (router P2).

```
Length of a SESSION object in bytes: 16
Length of a PREVIOUS_HOP object in bytes: 12
Length of a TIME_VALUES object in bytes: 8
Length of a LABEL_REQUEST object in bytes: 8
Length of a SENDER_TEMPLATE object in bytes: 12
Length of a SNEDER_TSPEC object in bytes: 8
Destination address of a session is: 192.168.8.2
Tunnel ID is: 1
Sender address is: 192.168.1.1
Senders port is: 8
IP address of a previous hop is: 192.168.3.2
L3PID is: 90
root@P2:/#
```

Fig. 8. Processing a Path message (router P2).

```
 GNU nano 2.2.6                                File: path_state.txt

192.168.8.2 1 192.168.1.1 8 192.168.3.2 90
```

Fig. 9. Path state txt file.

Similary to the previous test, Fig. 10 shows how router P1 processed a received Resv message. Router P1 concluded that the message is Resv based on the type field in RSVP header. Fig. 11 shows extraction of information from message and checking whether all optional objects are present. After that, resv state was established and content of

resv_state.txt file is shown in Fig. 12. Resv txt file has the following information: tunnel ID, sender's IP address, sender's port, requested capacity and associated label.

```
root@P1:/# sudo ./quagga/rsvp4d/packet_processor_te
Opening device: eth2
Total packet available: 396 bytes
Expected packet size: 396 bytes
IP header length (IHL) in bytes: 20
Protocol number of next encapsulated protocol is: 46
Type is 2
Resv packet length in bytes: 257

0: 11
1: 02
2: 06
```

Fig. 10. Receiving a Resv message (router P1).

```
Length of a SESSION object in bytes: 16
Length of a NEXT_HOP object in bytes: 12
Length of a TIME_VALUES object in bytes: 8
Length of a RESV_CONFIRM object in bytes: 8
SCOPE object is absent
Length of a STYLE_2 object in bytes: 6
Length of a FLOWSPEC_2 object in bytes: 8
Length of a FILTERSPEC_2 object in bytes: 10
Length of a LABEL_2 object in bytes: 8
Sender address is:192.168.3.2
Senders port is: 0
Capacity is: 80.000000
Label is: 1
root@P1:/#
```

Fig. 11. Processing a Resv message (router P1).

```
 GNU nano 2.2.6                                File: resv_state.txt

1 192.168.3.2 0 80.000000 1
```

Fig. 12. Resv state txt file.

The trajectory of Path messages is: P4, P6, P2, P1 and P5. The destination IP address of a session is 192.168.5.2, while the source IP address is 192.168.9.1. Each router generates Path message which is set to have destination IP address and sending port according to the given trajectory. Path messages are then received, processed and path_state.txt files are created.

Resv messages travel hop – by – hop in the reverse direction with respect to Path messages. The process of sending Resv messages starts at router P5. Router P5 examines path_state.txt file, which is shown in Fig. 13, and concludes that IP address of a previous hop node is 192.168.5.1.

```
 GNU nano 2.2.6                                File: path_state.txt

192.168.5.2 1 192.168.9.1 8 192.168.5.1 90
```

Fig. 13. Path state txt file in router P5.

Fig. 14 shows a routing table, or RIB, that is stored in router P5. P5 compares the IP address of a previous hop node from path_state.txt file with IP addresses it has in RIB. In this way, P5 determines through which interface it has to forward next Resv message to reach the previous hop node address. Router P5 sends Resv message using interface eth0. Destination address of Resv message is 192.168.5.1, tunnel ID is set to be 1, capacity to 80.

```
root@P5:/# netstat -nr
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.1.0     192.168.5.1     255.255.255.0   UG        0 0          0 eth0
192.168.2.0     192.168.5.1     255.255.255.0   UG        0 0          0 eth0
192.168.3.0     192.168.5.1     255.255.255.0   UG        0 0          0 eth0
192.168.4.0     192.168.5.1     255.255.255.0   UG        0 0          0 eth0
192.168.5.0     0.0.0.0         255.255.255.0   U         0 0          0 eth0
192.168.6.0     192.168.5.1     255.255.255.0   UG        0 0          0 eth0
192.168.7.0     0.0.0.0         255.255.255.0   U         0 0          0 eth1
192.168.8.0     0.0.0.0         255.255.255.0   U         0 0          0 eth2
192.168.9.0     192.168.5.1     255.255.255.0   UG        0 0          0 eth0
root@P5:/#
```

Fig. 14. FIB in P5.

Fig. 15 shows that router P1 receives a Resv message from router P5. P1 analyzes a received message and creates a resv state. Content of resv_state.txt file is shown in Fig. 16. As an intermediate router on trajectory, P1 looks at both path_state.txt and resv_state.txt file to generate Resv message. Fig. 17 shows the content of path_state.txt file which is created during establishing trajectory. P1 generates Resv message with destination address 192.168.2.2 which is the IP address of router's P2 interface eth1, and the capacity is 80.

```
Length of a SESSION object in bytes: 16
Length of a NEXT_HOP object in bytes: 12
Length of a TIME_VALUES object in bytes: 8
Length of a RESV_CONFIRM object in bytes: 8
Both optional objects are present
Length of a SCOPE_1 object in bytes: 12
Length of a STYLE_1 object in bytes: 6
Length of a FLOWSPEC_1 object in bytes: 8
Length of a FILTERSPEC_1 object in bytes: 10
Length of a LABEL_1 object in bytes: 8
Tunnel ID is:1
Sender address is:192.168.5.2
Senders port is: 0
Capacity is: 80.000000
Label is: 1
root@P1:/#
```

Fig. 15. P1 receives Resv message from P5.

```
 GNU nano 2.2.6                                File: resv_state.txt

1 192.168.5.2 0 80.000000 1
```

Fig. 16. Resv state txt file in router P1.

```
 GNU nano 2.2.6                                File: path_state.txt

192.168.5.2 1 192.168.9.1 8 192.168.2.2 90
```

Fig. 17. Path state txt file in router P1.

```
Length of a SESSION object in bytes: 16
Length of a NEXT_HOP object in bytes: 12
Length of a TIME_VALUES object in bytes: 8
Length of a RESV_CONFIRM object in bytes: 8
Length of a SCOPE_1 object in bytes: 12
Length of a STYLE_1 object in bytes: 6
Length of a FLOWSPEC_1 object in bytes: 8
Length of a FILTERSPEC_1 object in bytes: 10
Length of a LABEL_1 object in bytes: 8
Tunnel ID is:1
Sender address is:192.168.2.1
Senders port is: 0
Capacity is: 80.000000
Label is: 1
root@P2:/#
```

Fig. 18. P2 receives Resv message from P1.

Upon receiving a Resv message and creating resv state, router P2 looks into path_state.txt and resv_state.txt files and generates a new Resv message. This message for destination address has 192.168.8.2, so, it will be sent to the router P6. Processing of a Resv message from P1 is shown in Fig. 18, while the content of path_state.txt and resv_state.txt files is shown in Fig. 20 and Fig. 21, respectively.

Fig. 19. Resv state txt file in router P2.

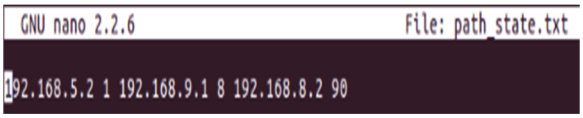

Fig. 20. Path state txt file in router P2.

Fig. 21, Fig. 22 and Fig. 23 show analyzing a received Resv message, content of resv_state.txt and path_state.txt file in router P6, respectively. P6 generates a Resv message and sends it to 192.168.9.1 which is IP address of router P4.

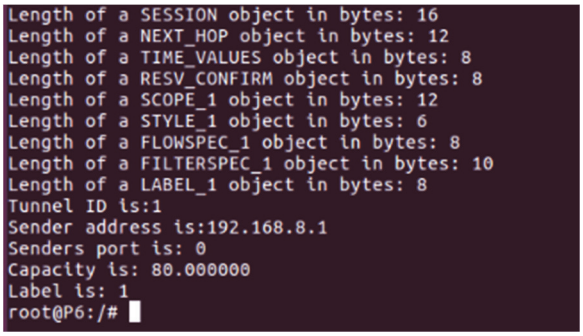

Fig. 21. P6 receives Resv message from P2.

Fig. 22. Resvstate txt file in router P6.

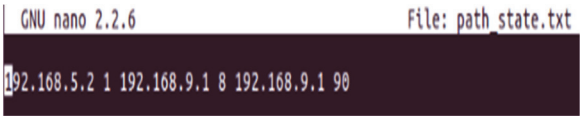

Fig. 23. Path state txt file in router P6.

Finally, router P4 received a Resv message. P4 analyzed the received message and established resv state which is shown in Fig. 24. Content of resv_state.txt file is presented in Fig. 25. Resv message is routed to its final destination.

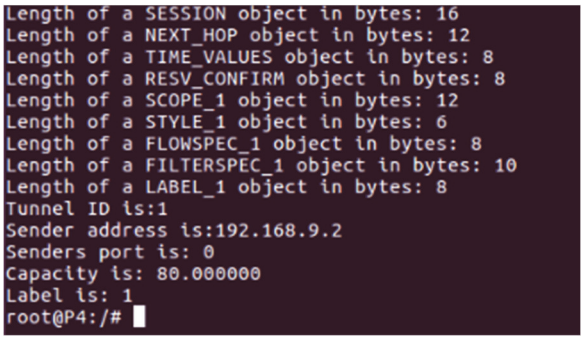

Fig. 24. P4 receives Resv message from P6.

Fig. 25. Resv state txt file in router P4.

V. Conclusion

In this paper, we have presented implementation that can serve as a base for adding new functionalities. We used OS virtualization for creating virtual containers which gained the functionality of routers by installing Quagga in them. IP lookup is performed using the longest prefix match algorithm. Further work will include writing the rsvp protocol daemon for Quagga.

References

[1] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource reservation protocol (RSVP) – Version 1 functional specification," No. RFC 2205, 1997.

[2] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, G. Swallow, "RSVP-TE: extensions to RSVP for LSP tunnels," No. RFC 3209, 2001.

[3] X. Xiao, A. Hannan, B. Bailey and L. M. Ni, "Traffic Engineering with MPLS in Internet," *IEEE Network*, vol. 14, no. 2, pp. 28-33, 2000.

[4] "Quagga Routing Software Suite," [Online]. Available: https://www.quagga.net/ , last access in October 2018.

[5] O. Filip, L. Forst, P. Machek, M. Mares and O. Zajicek, "BIRD internet routing daemon", NANOG-48, Austin, TX, 2010.

[6] S. Spies, "Analyzing and Testing Route Servers for Reliability and Scalability", PhD Thesis, 2010.

[7] P. Jakma and D. Lamparter, "Introduction to the Quagga Routing Suite," *IEEE Network*, vol. 28, no. 2, pp. 42-48, 2014.

[8] T. Komazec, A. Smiljanic, H. Redzovic and A. Radosevic, "Implementation of RSVP in Quagga software", *Proceedings of Telfor 2018*., Belgrade, Serbia, November 2018.

[9] P. Tang and H.Y.Liu, "Systems and methods for updating routing and fowarding information", U.S Patent No. 7, 209, 449, Washington, DC, 2007.

[10] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, "Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisors," *ACM SIGOPS Operating System Review*, vol. 41, no. 3, pp. 275-287, 2007.

[11] R. Morabito, J. Kjällman and M. Komu, "Hypervisors vs. Lightweight Virtualization: a Performance Comparison," Proc. 2015 IEEE International Conference on Cloud Engineering, Tempe, AZ, USA, 2015.

[12] "LXC Documentation," [Online]. Available: https://linuxcontainers.org/ , last access in October 2018.