QUAGGA ROUTING PLATFORM: APPLICATION AND PERFORMANCE

Slavica Tomović[*], Milutin Radonjić[**], Igor Radusinović[***]

Keywords: *RIP, open-source routing, OSPF, Quagga.*

Abstract: Quagga is an open-source software platform that provides routing services. It supports the following IP routing protocols: RIPv1, RIPv2, RIPng, OSPFv2, OSPFv3, IS-IS, BGP4 and BGP4+. Quagga is unique in its design because it uses a separate process for each running protocol. This platform offers true modularity since each protocol module can be upgraded or configured independently of the others. In order to illustrate its capabilities, this paper investigates support of this platform for RIP and OSPF, two most commonly used routing protocols on the Internet. In addition, the results of convergence time measurements are presented. Considering that software defined networking (SDN) promise to be Future Internet platform, brief description of possible application scenario for Quagga in SDN networks is provided.

1. INTRODUCTION

Huge popularity of the information and communication technologies and expansion of broadband applications impose increasingly strict demands in terms of performance of the networking equipment and routing protocols on the Internet. Technological advances of microelectronic have enabled development of hardware routers that offer ever increasing switching speeds. The most commonly used equipment for routing in today's networks is product of the big companies (Cisco, Juniper, etc.). These routers are characterized by extremely sophisticated hardware that in most cases provides all the necessary functionalities and meets the needs even of the largest networks. However, the routing software market is closed, in the sense that standard router can only run software from its vendor. For this reason, it is almost impossible to experiment with new protocols in real network environment. Service providers are often forced to wait for network equipment vendors to consider implementation of desired functionalities in their systems, which

[*] S. Tomović, Faculty of Electrical Engineering, University of Montenegro, Montenegro (e-mail: slavicat@ac.me).

[**] M. Radonjić, Faculty of Electrical Engineering, University of Montenegro, Montenegro (e-mail: mico@ac.me).

[***] I. Radusinović, Faculty of Electrical Engineering, University of Montenegro, Montenegro (e-mail: igorr@ac.me).

require a lot of time and financial effort. This problem is possible to overcome with open-source routing platforms, based on freely available code that can be modified in any way to satisfy user requirements. Equipped with routing software, ordinary PC can act as a multiple-interface router, which forwards packets in the network. Although, in general, they cannot be compared with the hardware solutions in terms of performance, software alternatives have become very popular lately due to their substantially lower price. In addition, software platforms simplify the process of developing a project, because the upgrade, revision and testing are performed in far much easier and faster way. Routing software is typically based on Linux, although, recently open-source tools have been started to be developed. Also, these tools can be implemented on other operating systems as well.

The open-source routing has not yet been implemented in larger networks, and is usually viewed with skepticism by people who are not familiar with this technology. Therefore, the open-source platforms today are mostly used for research and educational purposes. Limited performance, instability of software, lack of the system support, scalability and lack of functionalities support criticism of software routers. However, performance limitations can be compensated by the natural evolution of PC architecture. Current PC-based routers offer switching capacities of up to a few gigabits per second. It is more than enough for large number of applications. On the other hand, due to the open-source nature, majority of problems related to scalability and availability of software functionalities are gradually being overcome.

As one of the most popular open-source routing platforms, Quagga [1] is in the focus of this paper. The rest of the paper is organized as follows. Section 2 provides an overview of basic Quagga features. Configuration procedures for RIP and OSPF protocols, as most commonly used routing protocols on the Internet, are described in Section 3 and Section 4, respectively. Section 5 explains the procedure of the performed convergence time measurements and presents obtained results. Section 6 describes possible application of Quagga for integration of SDN and traditional networks. The conclusions are drawn in Section 7.

2. QUAGGA SOFTWARE PLATFORM

Software routing platforms have gained a lot of attention from the research community in the past few years. Several solutions have been proposed, and following four stand out in terms of performance and flexibility: Quagga, XORP [2], Bird [3] and OpenBGPd [4]. These platforms are supported on a number of standard Unix/Linux systems, which allows them to be run on breadth of hardware. Quagga is chosen to be elaborated in this paper due to its massive popularity, modular architecture and a wide range of routing protocols it supports: RIPv1, RIPv2, RIPng, OSPFv2, OSPFv3, BGP4, BGP4+ and IS-IS. As the name implies, only BGP functionality is implemented on OpenBGPd platform. Bird and XORP lacks implementation of IS-IS, protocol that threatens to suppress OSPF from operators' networks. On the other hand, significant shortcoming of Quagga compared to XORP is lack of support for multicast protocols. This problem is expected to be overcome in the future versions of this software.

While a single process usually provides all the routing functionalities in the other software platforms, Quagga uses different approach. Its work is based on several daemon processes, which are responsible for initialization and maintenance of kernel routing table, and exchange of routing information with the daemons running on the other systems. When it comes to IPv4, four daemons are associated with routing protocols, and they run independently, while the *zebra* daemon is the main process that controls the kernel routing table and performs route redistribution between different protocols (Figure 1). During configuration, it is necessary to run only those routing daemons associated with routing protocols that are going to be used. However, *zebra* daemon has to be always active since it is an interface for these protocols to communicate with the kernel. Each daemon has its own configuration file and terminal interface that can be accessed via Telnet. Terminal interface is similar to Cisco's routers. When a static route needs to be configured, it must be performed in *zebra* configuration file. On the other hand, configuration of RIP, OSPF, IS-IS and BGP protocols is performed in the configuration files associated with *ripd*, *ospfd*, *bgpd,* and *isisd* daemons, respectively.

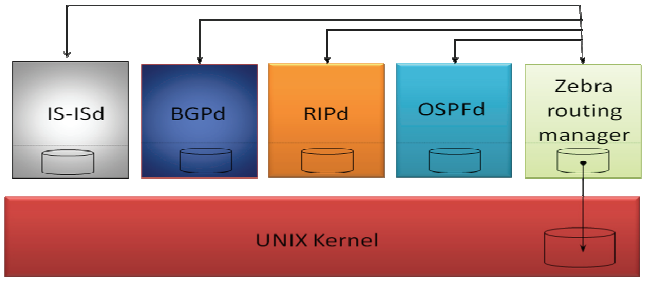

Fig. 1. Quagga software architecture

3. CONFIGURATION OF RIP PROTOCOL

RIP is the first routing protocol used in TCP/IP based networks. It has some significant limitations that have prompted the development of more complex and advanced routing protocols. Although, due to its stability, RIP is still widespread in networks with small and medium number of redundant paths. To run this protocol on Quagga platform it is necessary to activate *ripd* and *zebra* daemons and create configuration files for each of them. In order to illustrate possibilities of Quagga platform for research in the area of Future Internet we implemented testbed presented in Figure 2**.**

The *zebra* configuration file defines the interfaces that will participate in the forwarding process, and password for VTY (Virtual TeletYpe) connection that provides Telnet access to Quagga daemons. To configure *ripd* daemon on the same router, configuration file with the following content was used:

```
hostname ruter1
password example
key chain test
 key 1
   key string test
```

```
interface eth0
interface eth1
interface eth2
  ip rip authentication mode md5 auth-length old-ripd
  ip rip authentication key-chain test
interface eth3
  ip rip authentication mode md5 auth-length old-ripd
  ip rip authentication key-chain test
router rip
  version 2
  network eth0
  network eth1
  network eth2
  network eth3
  timers basic 30 180 120
  offset-list lista in 4 eth2
  distribute-list private in eth2
access-list lista permit 172.16.2.0/24
access-list private deny 172.16.1.0/24
access-list private permit any
```

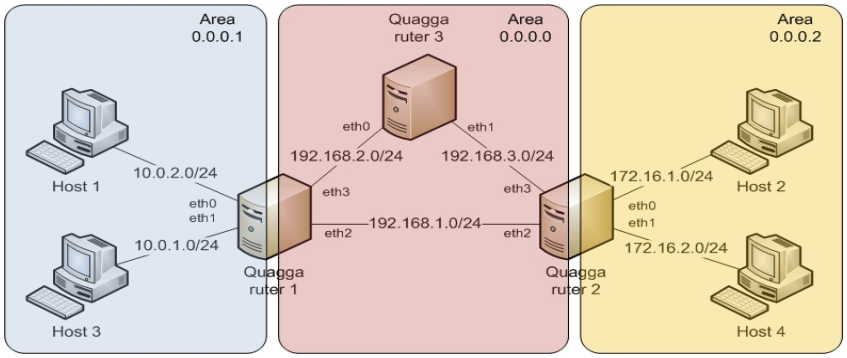

Fig. 2. Testbed configuration. Rounded rectangles represents OSPF areas

Basic *ripd* configuration includes definitions of the interfaces which the router will periodically use to inform its neighbors about the records in the local routing table. After receiving these notifications, routers determine the optimal paths to all other destinations in the network, according to the Bellman-Ford algorithm [5], and update their routing tables. Optimal path in RIP protocol is the shortest path or the path with the least number of hops. As such metric is often inconvenient because it prevents the use of paths with larger number of hops having unloaded links or a higher throughput, Quagga allows overcoming of this problem. In the above example this is performed with **offset-list** command which configures router to increase metric of route 172.16.2.0/24 by 4, when receiving information about it on the *eth2* interface. Quagga router supports both versions of RIP protocol. When specifying the protocol version in the configuration, it should be taken into account that RIPv1 does not support authentications mechanisms. On the other hand, within

RIPv2 standard two protection methods are defined: a simple string authentication and cryptographic authentication. Quagga has adopted both of them.

With **basic timers** command it is possible to change values of the basic RIP timers. The first argument of the command (*update-timer*) defines the interval at which routes that are learned by RIP are advertised to neighbors, and its default value is 30s. The second argument refers to the *timeout-timer* which determines expiration time of the route. If, during the interval defined by the *timeout-timer* value (typically 180s), router does not receive again information about the route, the route is considered to be invalid. The third timer (*garbage-collection timer*) counts down the time after which the route that is marked as invalid will actually be removed from the routing table. Its main role is to provide enough time for neighbors to be notified that the route has been dropped. Unless otherwise is specified, it is 120s. For avoiding routing loops, Quagga supports only the *split-horizon* [6] and it is activated by default. Route filtering can be made with **distribute-list** command, by defining an appropriate access list as it is shown in the example, but also in many other ways that offer more flexibility [1].

4. CONFIGURATION OF OSPF PROTOCOL

OSPF protocol implies that each router in autonomous system maintains a database with information about the network topology. Routers compute the shortest path tree by using Dijkstra's algorithm [7]. Optimal path is considered to be the path with the least cost, where the cost metric is usually based on the bandwidth of transmission media such that cost decreases as the link speed increases. However, administrator can also define these values in accordance with some other criteria.

In order to run OSPF on Quagga platform, *ospfd* and *zebra* daemons need to be activated. As already mentioned, the *zebra* daemon must run no matter which routing protocol is being used. Therefore, *zebra* configuration from previous example can be also exploited for configuration of OSPF protocol on the same topology. In a rest of the Section, basic capabilities of Quagga OSPF implementation will be discussed on the following configuration example for router 1:

```
hostname ruter1
password example
interface eth0
  ip ospf cost 3
interface eth1
  ip ospf cost 3
interface eth2
  ip ospf cost 7
  ip ospf dead-interval 60
  ip ospf hello-interval 10
  ip ospf authentication message-digest
  ip ospf message-digest-key 1 md5 ABCDEFGHIJK
interface eth3
  ip ospf cost 7
  ip ospf authentication message-digest
  ip ospf message-digest-key 1 md5 ABCDEFGHIJK
```

```
router ospf
 ospf router-id 1.1.1.1
 network 10.0.1.0/24 area 0.0.0.1
 network 10.0.2.0/24 area 0.0.0.1
 network 192.168.1.0/24 area 0.0.0.0
 network 192.168.2.0/24 area 0.0.0.0
 area 0.0.0.0 authentication message-digest
 area 0.0.0.1 stub no-summary
 default-information originate
 area 0.0.0.1 default-cost 1
 area 0.0.0.1 range 10.0.0.0/22
```

Perhaps the most important advantage of OSPF in comparison with RIP is the capability of hierarchical routing, i.e. the division of the autonomous system in multiple areas. Experimental network was organized in three areas, as illustrated in Figure 2. The areas are numerated, and each of them is controlled by the belonging router. Since the areas can exchange summarized routing information, the amount of control traffic can be significantly reduced. The type of routing information the area receives are determined by the attributes assigned to it. As shown in the example above, when defining interfaces which will participate in OSPF routing, the identifier of the area is also listed. When it comes to the area types, Quagga supports standard, stub and totally stub areas. If none of these attributes is explicitly assigned to the area, it is considered to be a standard area and thus it will accept all the types of routing information. The backbone area (area 0) must always exist. It connects all other areas in the autonomous system, and it is a standard area by definition. Stub areas do not accept information about routes that are external to the autonomous system they belonging to. If **no-summary** attribute is additionally used in the definition, as for an area 0.0.0.1 in the example above, the area is declared as totally stub. Therefore, it will not receive routing information from other areas of the same autonomous system. Access to the other network areas is achieved by defining default route to the area-border router.

Another interesting option is route aggregation or consolidation of multiple routes in one *super route*. Route aggregation directly affects network throughput and router resources used by OSPF process, since it results with reduced amount of control traffic and thus the lower frequency of route recalculations. Aggregation can only be performed on the area-border routers. In our example, it is achieved with command:

```
area 0.0.0.1 range 10.0.0.0/22
```

which summarizes routes 10.0.1.0/24 and 10.0.2.0/24 in *super route* 10.0.0.0/22.

Quagga also allows manipulations with link weights by **ip ospf cost** command. Authentications mechanisms are implemented in a similar way as in RIP protocol, whereby the authentication key functions as a password between OSPF routers in the same area. There is also possibility of adjusting *hello* and *dead* intervals, the two essential timers in OSPF protocol. *Hello* interval defines periods when router sends *hello* packets to establish or maintain adjacency state with its neighbors (typically 10s). If router stops receiving *hello*

packets from some neighbor, after expiration of the *dead* interval (typically 40) that neighbor is declared as dysfunctional.

5. CONVERGENCE TIME OF RIP AND OSPF PROTOCOLS ON QUAGGA PLATFORM

A network is said to be converged when all routers have consistent routing tables. Convergence time is therefore a measure of how fast the network converges, after a change in topology. Since network is not fully functional during this period, the convergence time is one of the key criteria for selection of routing protocol. As shorter this time is, the routing protocol is considered to be more optimal regarding this issue.

In order to examine convergence performances of RIP and OSPF implementations on Quagga platform, network topology from Figure 2 was used. The routers were using only basic protocol configurations which do not include any filtering, metric modifications or splitting of the autonomous system in multiple areas (in OSPF scenario). In the test, **ping** command was utilized to send packets from host 1 to host 2, and after deactivation of the *eth2* interface of router 1, convergence time was calculated based on the number of packets lost during update of the routing tables and establishment of alternative path through router 3. As, by default settings, **ping** sends ICMP packets in intervals of one second, number of lost packets approximately corresponds to the time during which the network converges.

Results of the measurements for RIP protocol are shown in Figure 3. The slow convergence of RIP protocol is mainly consequence of large default values of RIP timers. Reducing these values can significantly accelerate convergence process. For example, after setting update, timeout and garbage-collection timers on 10, 30 and 20 seconds respectively, measured convergence time was 30 second in average. However, these values need to be carefully selected, depending on the topology. If routing tables are large, small values of update timer can lead to congestion of slow links. On the other side, high values of timeout timer cause a slow reaction to changes, while too small values lead to instability.

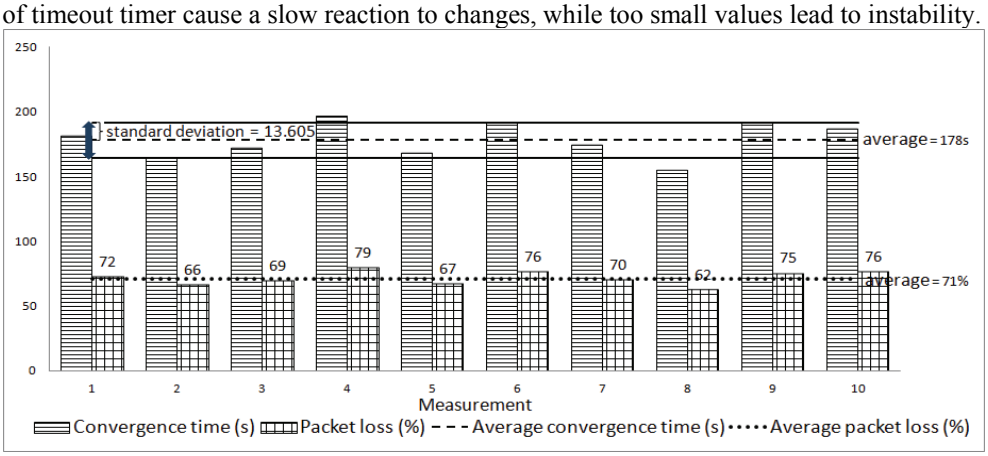

Fig. 3. Results of convergence time measurements for RIP protocol. Percent of packets lost during convergence is rounded to the closest integer value.

As opposite to RIP, whose convergence can take few minutes since each router copies and distributes its routing table to directly connected neighbors, OSPF converges much faster because only information about change is transmitted to other routers in the network. For the purpose of experiment, this time interval between consecutive packet transmissions was set on 100ms. In ten performed measurements, after deactivation of *eth2* interface of router 1, only about 200ms was necessary for network to converge with two lost packets. Fast convergence of OSPF protocol is one of the key advantages of Quagga over other software routing platforms.

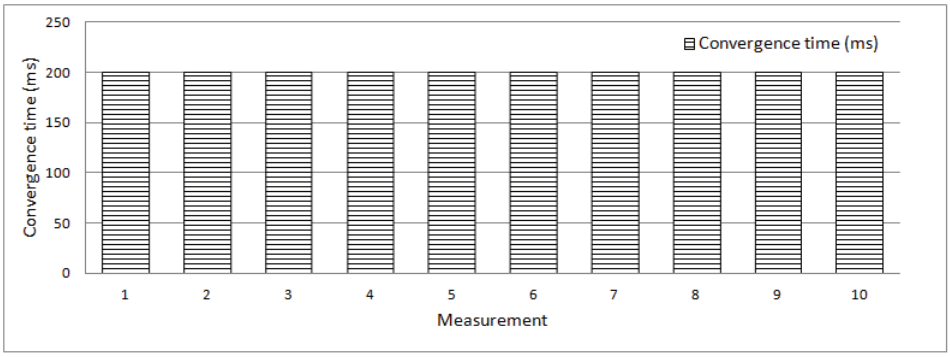

Fig. 4. Results of convergence time measurements for OSPF protocol.

6. QUAGGA IN SDN NETWORKS

Beside the standard application in traditional IP networks, Quagga has been recently considered as important tool that can facilitate migration towards SDN networks [8]. SDN is relatively new technology that promises to overcome many limitations of traditional network architecture by decoupling control functions from forwarding platform. In contrast to today's networks where control and data/forwarding plane are combined on the same device, in SDN, network intelligence is shifted to logically centralized controller, which maintains global view of the network, interacts with simple switching devices and provides a programming interface for user-written network management applications. Consequently, applications and policy engines have abstracted view on the topology, which appears as a single logical entity. Management of such networks is not limited anymore to configuration options that equipment vendor provides. Operator is free to configure devices and change the network behavior in real-time, by automated programs. However, although SDN architecture offers significantly improved network performances, as each new technology it also imposes a lot of risk, and hence needs to be gradually implemented. Since Quagga is software routing engine, it can be utilized for integration with legacy network equipment. Based on this, RouteFlow project [9] proposed control framework which offers IP routing as a service in SDN network. The architecture of RouteFlow is illustrated in Figure 5. Each device in SDN network is mapped to the virtual machine with Quagga, which provides IP control plane. These virtual machines are dynamically interconnected, and form a logic topology that mirrors a physical infrastructure. Virtual environment runs on an external

server and communicates with physical devices through the OpenFlow [10] control application, which gets information from RouteFlow server about routing decisions. After processing of the received information, RouteFlow controller installs set of rules for traffic flow management in forwarding tables of SDN devices. Although the control of physical networks is performed from a single point, in the virtual environment, control is distributed like in traditional IP networks.

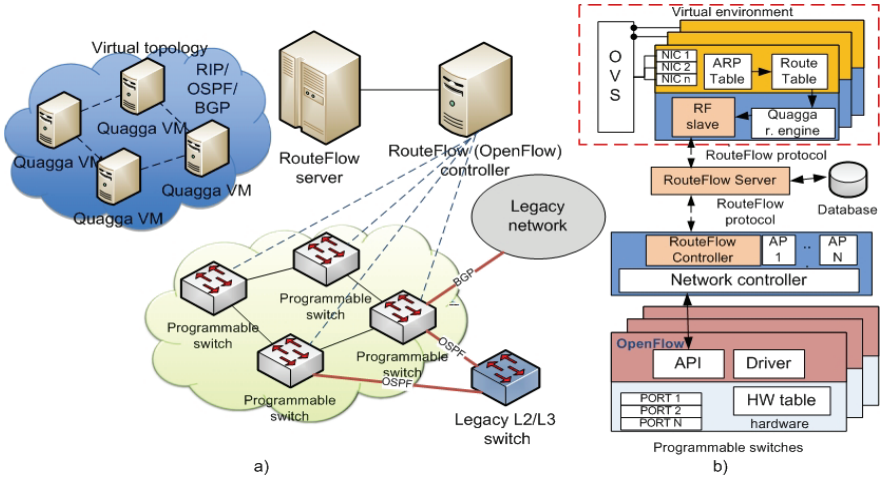

Fig. 5. RouteFlow architecture conceptual design: a) Overview of a RouteFlow-controlled network; b) RouteFlow components

As illustrated in Figure 5(b), RouteFlow controller is nothing more than application running on the SDN controller that translates routing decisions in OpenFlow rules. The core of control logic is placed on RF-server, which receives and processes information about all events in both, virtual and real network environment. It initiates virtual machine for each detected device and communicates with RF-slave process, which monitors Quagga routing tables. In order to enable integration with legacy network, RouteFlow uses special entries in forwarding tables of SDN devices that direct all control traffic originated in traditional networks to the corresponding virtual entities. In a similar way, relevant routing information generated inside virtual topology is forwarded to the legacy network devices.

7. CONCLUSION

Routing protocols are critical for performance of IP networks. However, due to proprietary nature of network devices, routing protocols are hard to evolve and follow rapid development of the Internet. For this reason, as an alternative to dedicated hardware, open-source routing platforms are being increasingly used in combination with a general-purpose computer. By installing one of these platforms, ordinary PC with multiple network interfaces can be transformed into software router with performance acceptable for many applications. Beside this cost-effective solution, software routing platforms can be exploited as control plane for high-capacity routers, which use specialized hardware for

packet forwarding. These platforms are especially convenient for the research purposes, since their open-source nature enables experimentation with new protocols in working network environment. Quagga is one of the most popular open-source routing platforms. In order to illustrate capabilities of Quagga platform, this paper describes configuration procedures for RIP and OSPF protocols, having in mind their importance for the Internet. The results of convergence time measurements show undoubted advantage of OSPF over RIP in this respect. Very fast convergence of OSPF protocol confirms quality of the code base and makes Quagga convenient for application in commercial edge networks. The economic advantages of open solutions provide users a cost-effective way to increase performance that is unattainable with proprietary routing solutions. Taking into account potential importance of SDN for the future of networking, we briefly described application of Quagga for providing IP routing as a service in such networks. This relatively new application scenario shows that Quagga will play significant role in the development of Future Internet, regardless on approach to networking, SDN or traditional, that will prevail.

ACKNOWLEDGMENT

This paper is partly supported by the Ministry of Science of Montenegro and the Ministry for Information Society and Telecommunications under grant 01-451/2012 (FIRMONT)

REFERENCES

[1] K. Ishiguro, "Quagga 0.99.18, A routing software for TCP/IP networks," March 2011. Available: http://www.nongnu.org/quagga/docs/quagga.pdf. [Accessed: May, 05, 2014].
[2] M. Handley, O. Hodson, E. Kohler, "XORP: an open platform for network research," *ACM SIGCOM,* vol. 33, issue: 1, pp. 53-57, January 2003.
[3] Bird Internet routing daemon project. Available: http://bird.network.cz/
[4] OpenBGPd software routing engine. Available: http://www.openbgpd.org/
[5] R. Bellman, "On a routing problem," in *Quarterly of Applied Mathematics*, SAD: Brown univesity, 1958, pp. 87–90.
[6] *RIP Version 2*, IETF, RFC 2453, November 1998.
[7] E. W. Dijkstra, "A Note on Two Problems in Connection with Graphs," in *Numerische Math*, vol. 1, pp. 269-271, 1959.
[8] Open Networking Foundation, "Software Defined Networking: the new norm for networks," Web. White Paper, Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf. [Accessed: May, 17, 2014].
[9] C. E. Rothenberg, M. R. Nascimento, M. R. Salvador, C. Corrêa, S. Lucena, A. Vidal, and F. Verdi, "Revisiting IP Routing Control Platforms with OpenFlow-based Software-Defined Networks," *Future Internet Experimental Research Workshop* (*WPEIF*), Ouro Preto, Brazil, May 2012.
[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovations in campus networks,"*ACM SIGCOMM*, vol. 38., no. 2, pp. 69-74, 2008.
[11] *OSPF Version 2*, IETF, RFC 2328, April 1998.