QoSVisor: QoS Framework for SDN

Ronak Al-Haddad (1st)
Computing and Technology Department
Anglia Ruskin University
Cambridge, United Kingdom
Ronak.al-haddad@pgr.anglia.ac.uk

Erika Sanchez Velazquez (2nd)
Computing and Technology Department
Anglia Ruskin University
Chelmsford, United Kingdom
Erika.sanchez@anglia.ac.uk

Adrian Winckles (3rd)
Computing and Technology Department
Anglia Ruskin University
Cambridge, United Kingdom
Adrian.winckles@anglia.ac.uk

***Abstract*—The increasing demand for network services and quality across wide selections of digital applications in the internet era has caused growing congestion and raised questions about how to deal with prioritizing data in ways tailored to particular uses of applications and managing peak congestion times. Software Defined Network (SDN) in particular Slicing Strategy, seems the best solution due to its new constitution intelligently implemented through the SDN OpenFlow protocol. However, Slicing Strategies specifically "FlowVisor" are limited in certain mechanisms such as Traffic Engineering (TE), which make it a requirement to find new ways to deliver Quality of Service (QoS) for different applications. In this paper, QoSVisor presented as an SDN extension action QoS Slicer based as an enhancement to the standard FlowVisor operation slicing tools to ensure the QoS for each Slice-based class of application.**

Keywords—SDN; FlowVisor; QoS; OpenFlow; Traffic Engineering

I. Introduction

At the IEEE, the 802 LAN/MAN Standards Committee has recently started some activities to standardize SDN capabilities on access networks based on IEEE 802 infrastructure through the P802.1CF project for both wired and wireless technologies to embrace new control interfaces[1], [2]. This work is based on the recognized need to find new ways of delivering quality and reliability as the new era applications became more varied, more complex, connected to everything and accessible from everywhere. Also, nowadays applications such as voice and video conferencing, which are also delayed sensitive, are being more in demand in addition to other applications, such as file transfer, that are more concerned with the average transmission rate. Network performance faces a significant growth potential regarding dealing with network scalability, management, monitoring, security, and quality of service. On the other hand, the potential complexity of network equipment adds extra effort which impacts on user experience. Software Defined Networks (SDN) seems to be the best solution due to its new constitution intelligently implemented through its new concept separating the control plane and data plane, and allowing centralized Traffic Engineering (TE) [3]. SDN is clearly a new approach for network programmability (i.e., the ability to access the network via APIs and open interfaces), which refers to the ability to control, change, and manages network behavior dynamically through the SDN controller [4]. OpenFlow protocol comes as a communication protocol between the controller and the switch; it is defined as a set of flow instructions that comprise the forwarding behavior of the switch [5] suggested by Stanford University [6] as one of the most widely deployed testbeds which opened the gates to innovations in network architectures and protocols.

Software Defined Network-OpenFlow (SDN-OF) Traffic Management and Control, e.g., Centralized TE and control with OF 1.2+ compliant controllers and capable switches are the future of the Traffic Engineering evolution. Many TE tools for SDN-OpenFlow have been proposed, but some of them have limitations regarding TE and OpenFlow. At the moment, it does not meet the need to separate traffic in a controlled and isolated way by using Network Virtualization scheme as required for SDN approaches. FlowVisor is proposed to be deployed between OpenFlow controllers and switching devices to slice the network and enforce isolation from topology and traffic this tool is able to virtualize the modern network and enable distinct virtual machines to share the same hardware resources, [7]. Researchers use slicing tools in their proposals without thinking whether the tool enables satisfactory performance or not. Therefore, efficient new slicing tools are urgently needed, or existing tools must be enhancing to guarantee performance that is more precise. This paper presents QoSVisor the new enhanced FlowVisor tool as a model for the QoS, which deals with the new requirements and current system limitations.

The rest of this paper is structured as follows: related work, research background, and limitations were present in section II. Section III presents the proposed design of QoSVisor. Finally, Conclusion and Future work were present in section IV.

II. Related Work, Research Background and Current Limitations

A. FlowVisor and QoS

Network virtualization roots back in the 90's when it start been used, for example, the Tempest project is one of the first initiatives to introduce this technology [8]. In modern computer network platforms it has been adopted as a consolidated technology in the research community to refer to the process of inserting an abstract layer or virtual resources to enable sharing the same hardware resources, reflecting a huge cost reduction, greater agility and more flexibility than the physical network equipment; this is possible by the use of the virtual machines defined as a software implementation of a machine with a completely independent operating system [1], [9]. A network virtualization layer called "FlowVisor" has been proposed by Sherwood et al., at Stanford University [10] for slicing the network; they proposed the establishment of a hypervisor sitting between software and hardware of a PC.

FlowVisor uses OpenFlow as a hardware abstraction layer to sit logically between control and forwarding paths on a network device, it creates virtual slices in wired and wireless networks, offering transparency to the controller, strong isolation between slices and a modifiable slice policy [11]. As mentioned above FlowVisor introduces a new mechanism as a software Slicing layer between the forwarding and control planes on network devices [12], to this end, the main contributions of FlowVisor are:

Possibility to Slice any control plane message format implemented with OpenFlow (OF); it is the first slicing mechanism that allows a user-defined control plane to control the forwarding in deployed production hardware. Network resources are sliced according to bandwidth, topology, forward table entries, and device CPU.

A policy language that maps flows to slices giving the user flexibility to try new services so that users can precisely decide their level of involvement in an experiment. Also, the network users can signal to the network administrator to randomly add (opt-in) and remove (opt-out) their flows from a slice's flowspace at any time.

Transparency to both data and control planes and can easily transparent inter-ability between the traditional network and sliced network.

Ability to blocks and rewrite control messages as they cross the slicing layer. Thus, it enforces strong isolation between slices; this allows the experimenters to get along safely without affecting real production traffic.

To successfully operate on deployed networks in Stanford University with 20+ users, 40+ network devices, a production traffic slice, and four standing experimental slices [12].

Fig. 1: Shows the internal operation of FlowVisor and the communication between the controller and the switch. The operation starts when the command is sent by the controller to the OpenFlow switch. The controller commands are first received by a Slicer element (1), which is responsible for managing commands and messages from/to the OpenFlow controller. There is one Slicer for each virtual network controller. The Slicer then confirms if the received command corresponds to the virtual network definition (2), using its flow space rules, and amends the command when necessary. The output command is then sent to the OpenFlow switch (3) using the respective Classifier element, responsible for managing commands and messages to/from the OpenFlow switch. There is one Classifier for each OpenFlow switch.

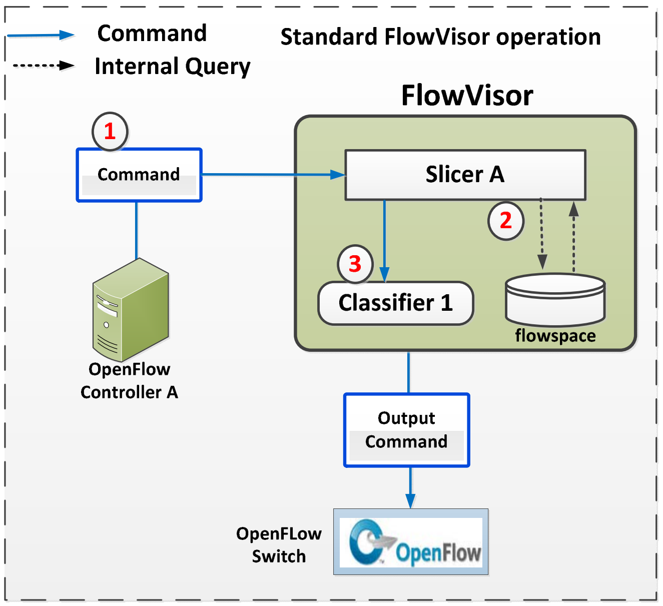

Fig. 1. The current internal operation of FlowVisor. [13]

B. Improvements and Contributions to FlowVisor

A number of researchers have made useful contributions to address the issues of networks functions, but there are limitations to the current approaches, for example, FlowVisor has been extended to include an action slicing mechanism, that allowed it to limit which actions can be used by each virtual network controller [13]. Similarly, in [14], another enhancement has been done to FlowVisor to implement admission control and minimum bandwidth guarantee scheme, their tests focused only on the transmission performance of QoS on Video streams. Another proposal is GiroFlow [15] a tool model to manage slices and policies within the network infrastructure focusing on the properties of the application running on the controller.

ADVisor [16] is an architecture that builts on top of FlowVisor to include additional functions such as Virtual links and Virtual ports management. The major purpose of ADVisor is the ability to establish complex and bandwidth-guaranteed virtual topologies completely decoupled from the underlying physical topology, providing flexibility in the adoption of an adequate L2 header space to identify virtual topologies in the network. However, this proposal makes no changes to the OpenFlow protocol which would enable FlowVisor to configure the data paths, such as defining schedules and the allocation queue. In [10], [14] both propose traffic scheduling by using VLAN (Priority Code Point) PCP field with relating databases which varies from (0 to 7) for marking packets to solve the bandwidth allocation needs. As shown in Fig. (2) The command for QoS slice creation is extended by adding required bandwidth at the basic slice creation command.

FlowVisor stores this configuration information in an XML file, which is used to update the database. On the other hand, the database of FlowVisor has been enhanced and implemented for slice and topology information; the information in this database is required for admission control and OpenFlow GUI, but this proposal was a temporary solution and required a specific QoS control.

```
# Slice creation called KISTI-A requiring 300Mbps
/root/fv_test/bin/fvctl --passwd-file=/root/shmin/cnu_passwd
createBWSlice KISTI-A tcp:134.75.85.xxx:6633 minsh@cnu.ac.kr
300M
```

Fig. 2. Command for QoS slice creation [14]

C. *Other Slicing Technologies*

Architecturally FlowVisor can slice any data plane/control plane communication channel. Therefore, FlowVisor is built on top of OpenFlow protocol [12]. Similarly to FlowVisor there is OpenVirteX [17], [18]. A Network Virtualization Platform, the main aim of the proposal is to provide virtual SDNs (vSDNs). Each vSDN is customizable in terms of topology as well as addressing scheme, and control function virtualization, using multiple controllers, one per slice, and in terms of slicing using virtual flow table per slice, but it does not support QoS in the current OpenFlow protocol.

Another proposal is AutoSlice [19], which targets scalability aspects of network hypervisors by optimizing resource utilization and by mitigating the flow-table limitations through a precise monitoring of the flow traffic statistics, having a single third party to control the mapping of vSDN topologies. It has multiple controllers one per slice, the proposal used VLAN tags for slicing, but also doesn't provide QoS guarantees. AutoSlice has been used over OpenFlow protocol only. FlowN [20], [21] is analogous to a container-based virtualization, i.e., a lightweight virtualization approach.

FlowN was also primarily conceived to address multitenancy in the context of cloud platforms. FlowN lets tenants write arbitrary controller software that has full control over the address space and can target an arbitrary virtual topology. They used a shared controller platform (NOX3) rather than running a separate controller for each tenant whilst also using a modern database technology to perform the mapping between the virtual and physical address space. The solution proposed uses VLAN tags for slicing and no QoS guarantees, and it has been used over OpenFlow protocol only.

From comparing the available slicing tools as shown in table (1) FlowVisor seems to be the ideal tool which will help to improve the performance of the production network, because FlowVisor mainly virtualizes the network control and separates the traffic, be able to have access to resources for the experimental network without affecting the network in a parallel way with the real user within the flow space [22] the argument in [22] goes to compare the FlowVisor to a full virtualization technology solutions such as FlowN [21], they conclude that FlowN has higher Latency due to the database but scales better than FlowVisor, therefore *QoSVisor* proposal the main aim is considering enhancing the performance of the FlowVisor to ensure more precise Quality of Service in Centralized architecture.

TABLE I. SUMMARY OF THE AVAILABLE SLICING TECHNOLOGIES

Proposals	**Slicing Technology**	**QoS Architecture**	**Multiple Controllers**
Open VirteX	To provide (vSDNs) and virtual Flow table per slice.	No	Yes, one per slice
Auto Slice	Target the scalability aspects of network hypervisors using VLAN tags.	No	Yes, one per slice
Flow N	Shared controller platform (NOX3).	No	No, shared controller
FlowVisor	Slice data/control plane communication channel.	Yes, can guarantees QoS by using VLAN Priority Code Point bits (PCP)	yes
Enhanced FlowVisor -QoSVisor (proposal)	Slice data/control plane communication channel.	Provide Soft QoS by using (DiffServ) architecture; this includes packet classification functionality based class of application (Voice, video, and data transfer)	yes

D. *OpenFlow*

OpenFlow [6], [23], [24], [25] is an open standard based on an Ethernet switch, with an internal flow-table, and a standardized interface to add and remove flow entries to enable the researchers to control directly the packets are directed within the network, as shown in Fig. (3). When a packet arrives at a switch or router, the device checks the packet against the flow table. Each flow entry contains a set of instructions that are executed when a packet matches the entry, but if the packet does not match any entry, the packet is queued, and a new flow event is sent across the network to the OpenFlow controller. The controller responds by adding a new rule to the flow table to handle the queued packet. The following packet in the same flow will be treated without contacting the controller, in other words, the external controller is contacted only for the first packet in a flow; the following packets are forwarded at the switch's full line rate. However, the controller itself can be implemented as a distributed system, which enables rapid network application development [26], [27], [28], [29], [30].

Another advantage of OpenFlow is that it enables researchers to experiment with new network protocols on deployed hardware, but only a single researcher can use/control an OpenFlow-enabled network at a time. As a conclusion, without using the FlowVisor, the OpenFlow-based research is limited to isolated testbeds, limiting its scope and realism. Accordingly, FlowVisor's capability to slice a production network is an orthogonal and independent contribution to OpenFlow-like software-defined networks [12].

Network devices generate OpenFlow protocol messages, which go to the FlowVisor special controller and are then routed by the network slice to the appropriate researcher(s) OpenFlow messages from researcher controllers are checked by the FlowVisor to ensure that the isolation between slices is maintained before being forwarded to switches [31].

At the time of writing this paper there are a number of improvements that are beening proposed for FlowVisor; the 0.10 version treated the type enqueue messages, enabling the creation of queues along the flowspaces, by defining new parameters for input streams. Output type actions can also be resetted as enqueue type actions. However, still a clear limitation is that the data path queue definitions must be manually configured by external applications [32].

The authors of GiroFlow [15] considered a new model in their architecture for the management of the slices focusing on the properties of the application running on the controller, using automated interfaces to the network controller, and FlowVisor to create and adjust the slices and policies inside a network managed by FlowVisor. However, this model still uses manual configuration to deploy the rule for routing the packets together with the OpenFlow controller and datapaths.

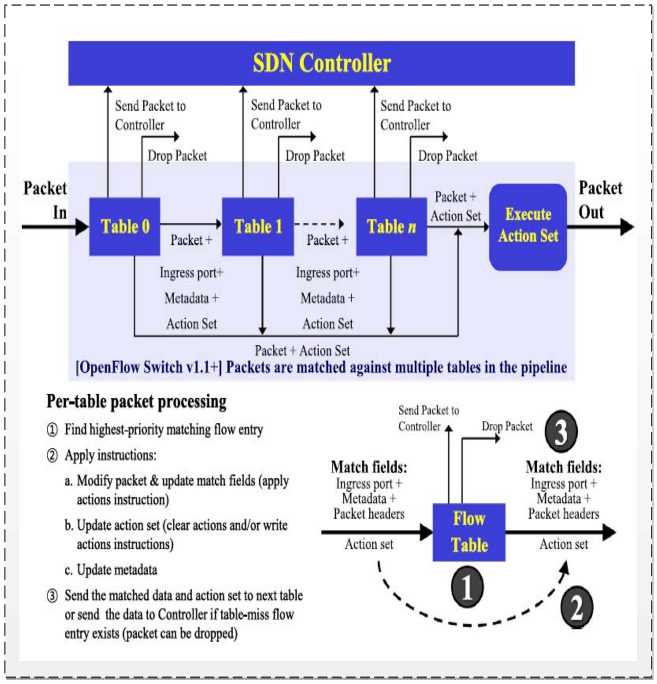
SDN Controller
Send Packet to Controller
Drop Packet
Send Packet to Controller
Drop Packet
Send Packet to Controller
Drop Packet
Packet In
Table 0
Packet +
Table 1
Packet +
Table n
Packet + Action Set
Execute Action Set
Packet Out
Ingress port+ Metadata + Action Set
Ingress port+ Metadata + Action Set
Packet + Action Set
[OpenFlow Switch v1.1+] Packets are matched against multiple tables in the pipeline
Per-table packet processing
① Find highest-priority matching flow entry
② Apply instructions:
a. Modify packet & update match fields (apply actions instruction)
b. Update action set (clear actions and/or write actions instructions)
c. Update metadata
③ Send the matched data and action set to next table or send the data to Controller if table-miss flow entry exists (packet can be dropped)
Send Packet to Controller
Drop Packet
3
Match fields: Ingress port + Metadata + Packet headers
Action set
Flow Table
Match fields: Ingress port + Metadata + Packet headers
Action set
1
2

Fig. 3. Packet flow over multiple flow table pipelines [41]

E. QoS in OpenFlow

The TE Techniques to enhance the performance and guarantee the Quality of Service (QoS) are the main objectives of long lists of research proposals based on SDN, and highlighted by the IEEE in the P802.1CF. The explosion of the internet and the different use of its applications made this fundamental challenge crucial, more complex to achieve and made the search for more solutions more urgent; many researches, for example (Kreutz et al., 2015)(Lara et al., 2014) (Jarraya et al., 2014) summarise and discuss the capabilities, applications, deployments and challenges of SDN/OpenFlow-based networks, they also researched the advantage of using OpenFlow-based applications such as, Traffic Engineering, simplify network management, adding security and dependability features, virtualized networks and data centers networking, mobility and wireless and also proposed solutions in measurement and monitoring applications [33], [23], [3].

These applications run on top of networking operating systems such as Nox, Beacon, Maestro, Floodlight, Trema or NodeFlow, FlowVisor, POX, Ethan controller, NMS. These central controllers have full visibility of the network and its vendors to enable the network traffic management [34]. QoS frameworks are needed within the operation systems to handle the applications requirement and slice the network accordingly. Authors Egilmez et al. [35] proposed OpenQoS, a novel OpenFlow controller design for multimedia delivery with end-to-end QoS based on dynamic QoS routing for multimedia applications using video delivery, the research is based on Floodlight controller and OpenFlow protocols; their contribution minimizes adverse effects like latency and packet loss on other types of flows while other flows (data) remain on their shortest path. Although they successfully grouped the incoming traffic as data flows and multimedia flows, where the multimedia flows are dynamically placed on QoS traditional shortest-path, their research did not show support for traffic shaping.

Another approach was used in PolicyCop [36], which is a QoS policy management framework. The authors used a Floodlight controller-based OpenFlow protocol too; it provides an interface for specifying QoS-based Service Level Agreements (SLAs) and enforces them using the OpenFlow API; the major purpose is to monitor the network using an autonomic QoS policy enforcement framework for SDN by implementing three planes: data/control/management planes. They developed a few control applications that provide different control functions to the management plane, which consists of the Policy Validator and Policy Enforcer, which are responsible for validating and enforcing QoS policies and are the foundation of the route decision in the control layer. They showed the possibility of developing new proposals been benefit from the new architecture of SDN.

The authors Ishimori, A., et al. proposed QoSFlow to improve QoS which performs with bandwidth guarantees and by a well-known FIFO scheduling; QoSFlow uses multiple packet schedulers of Linux kernel to perform Routing and Traffic Engineering to extend the standard software switch (datapath) of OpenFlow version 1.0.; this new extension included Traffic Shaping, Packet Schedulers, and Enqueueing. Thus, the QoS module opens a channel with the kernel through Netlink and Packet socket families to connect both user and kernel space. The authors indicate that the solution supports only eight queues per switch port as maximum using the slicing mechanism, whereas FlowVisor can slice the flow space into an any desired number of separate slices [37], [38].

III. Proposed Design of QoSVisor

To build on the work already done and manage the limitations outlined above, QoSVisor proposed as an architecture to develop a solution that enhances the QoS in FlowVisor. This research focuses on developing and enhances the FlowVisor and produce new FlowVisor controller, which will ensure slicing the production network with guaranteed and more precise QoS for different applications to continue running even during peak congestion times with agreed priorities. The proposed design is based on gathering four main technologies together: Traffic Engineering (TE), SDN, Network Hypervisors and Network Slicing strategies. To address the limitations outlined in section II, a new modification model proposed to enhance the current FlowVisor to meet the requirements of improving the QoS classification within a sliced SDN. The objective of the proposed QoS model, which we have called "Action QoS Slicer," is to extend the current internal operation of FlowVisor. As shown in Fig. (4)

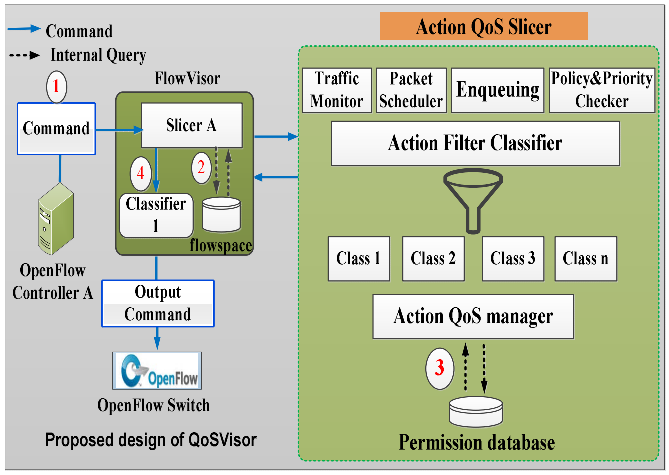

Fig. 4. (Proposed design of QoSVisor)

The new model will map the data path to different multimedia applications traffic such as voice, video, and data to identify and route each application individually into classes with a relational database; as shown in Fig. 4. Currently, there is no QoS architecture successfully implements and guarantees 99.9% of the required specifications of the networks the recent developments of the various types of applications proved that [39]. Moreover, in QoSVisor model Differentiated Services (DiffServ) architecture has chosen to provide soft QoS guarantees by the use of scheduling/priority queueing to enable routers to have packet classification functionality, then the core routers can forward the packets based on their priorities [40]. The main components and the workflow of the proposed QoS model are shown in Fig. (5):

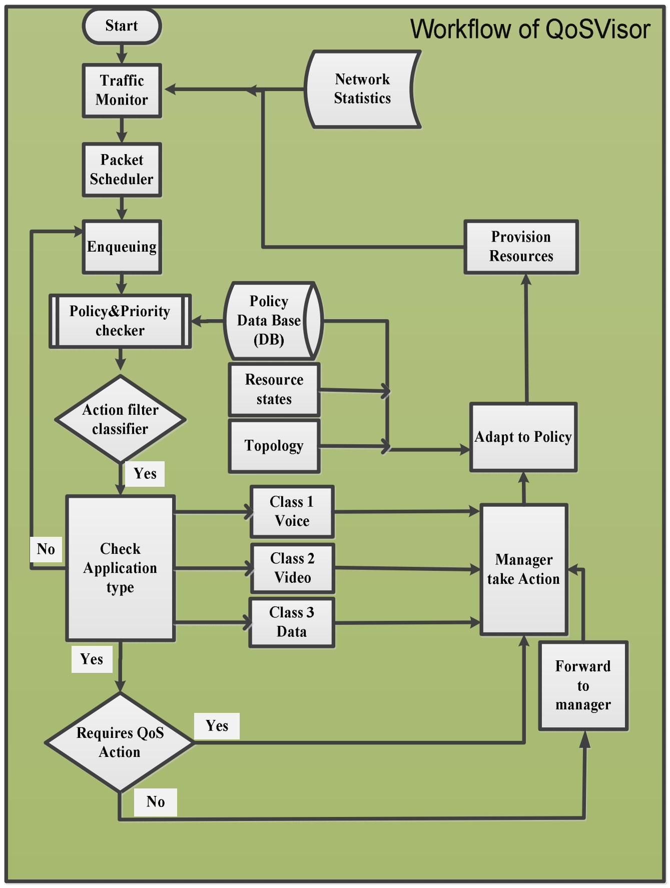

Fig. 5. Workflow of QoSVisor

A. *Traffic Monitoring:* This module represents the gate to monitor the network traffic and does the operation of analyzing, reviewing, and managing network traffic for any oddity or process that can affect network performance, availability and/or security. This operation can be done using different tools and techniques to examine the computer network-based communication, data and packet traffic. In order to measure performance and to provide an house mechanism to provide the ability to assess availability and security, the use of IPFIX flow collection would be useful. The ability to use the internal and external element nature of IPFIX templates [42] would allow more streamlined monitoring functions through an IPFIX exporter/collector embedded in the traffic monitoring function.

B. Packet Schedulers: This component is responsible for enforcing resource allocation to individual flows, and when queues start to build up in the routers the packet scheduler will decide which packet should get the resources.

C. Enqueueing: This component is responsible for operating messages of the OpenFlow protocol and will modify the state of the flow table. In this stage, each entry contains header field, counters, and actions for matching flow packets, these following component mechanisms maps flows to queues of kernels data structures, before the next component which is called, the Policy & Priority checker.

D. Policy checker: Policy checker will deal with the Traffic Monitor and the Policy DataBase (DB) for each queue to collect data and identify the Policy violation.

E. Action filter classifier: In the data path, the classifier divides an incoming packet stream into multiple groups on predefined rules. In the proposed model, Behavior Aggregate (BA) is going to be used, which is the simplest Differentiated Service classifier in the Behavior Aggregate (BA) classifier, and this will select packets based solely on the DSCP values in the packet header.

F. Check Application Type: This component will put the packet headers into classes: Class1, for Voice application, Class2, for Video application, and Class3, for data application. When each class has been identified then, each class will have the required QoS Action needs to be forwarded to Action Quality Manager (AQM) to check the action individually against the designed-in policies and then forward to the DataBase (DB) for user access according to control rules in Permission Data Base.

After treating the network traffic in this Action QoS Slicer each class is send to the FlowVisor FlowSpace as a (Slice 1), (Slice 2), and (Slice 3) to be ready to be send as an output command for the OpenFlow Switch, as in figure (4) and (5). Currently, the QoSVisor model is in the implementation stage, the set of QoSVisor performance tests and evaluation will be addressed in the next version of QoSVisor research paper.

IV. CONCLUSION AND FUTURE WORK

Due to the increasing use of internet of things/applications and the congestions caused by its demand, currently, there is no QoS architecture successfully implements and guarantees 99.9% of the required specifications of the networks, therefore in this paper, QoSVisor is a contribution to the search for ways of maintaining and improving quality of service and gives users the ability to control, specify and prioritize traffic management in an increasingly congested and complex information world. The proposal uses QoS Framework for SDN based on enhancing the current FlowVisor to provide a special-purpose controller to ensure QoS for each Slice-based class of application and bring more precise Quality of Service (QoS). Also, the proposal provides a Soft QoS by using (DiffServ) architecture; this includes packet classification functionality based class of application (Voice, Video, and Data transfer). The new model will map the data path to different multimedia applications to identify and route each application individually into classes with a relational database. The proposed enhancement represented by adding the following extensions to the current FlowVisor:

1) The Action QoS Slicer: which contains Traffic Monitor, Packet Scheduler, Enqueueing, and Policy & Priority Checker.

2) The Action Filter Classifier.

3) Action QoS Manager.

The work presented in this paper is based on the preliminary research. The major issues regarding how QoSVisor performs and how it is best implemented will be addressed in our ongoing and future research.

REFERENCES

[1] Kreutz, D., Ramos, F.M., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S. and Uhlig, S., 2015. Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 103(1), pp.14-76.

[2] IEEE.2014.Cooperation for OmniRAN P802.1CF. [ppt]: Publisher: IEEE Available at: http://slidegur.com/doc/5953095/presentation

[3] Jarraya, Y., Madi, T. and Debbabi, M., 2014. A survey and a layered taxonomyofsoftware-defined networking. Communications Surveys & Tutorials,IEEE,16(4),pp.1955-1980

[4] Hakiri, A., Gokhale, A., Berthou, P., Schmidt, D.C. and Gayraud, T., 2014. Software-defined networking: Challenges and research opportunities for future internet. Computer Networks, 75, pp.453-471.

[5] Marschke, D., Doyle J., and Moyer P., 2015. Software Defined Networking (SDN): Anatomy of OpenFlow. UK:Lulu Publishing Service.

[6] Hu, F., Hao, Q. and Bao, K., 2014. A survey on software-defined network and openflow: from concept to implementation. IEEE Communications Surveys & Tutorials, 16(4), pp.2181-2206.

[7] Sherwood, R., Chan, M., Covington, A., Gibb, G., Flajslik, M., Handigol, N., Huang, T.Y., Kazemian, P., Kobayashi, M., Naous, J. and Seetharaman, S., 2010. Carving research slices out of your production networks with OpenFlow. ACM SIGCOMM Computer Communication Review, 40(1), pp.129-130.

[8] Van der Merwe, J.E., Rooney, S., Leslie, I. and Crosby, S., 1998. The Tempest-a practical framework for network programmability. Network, IEEE, 12(3), pp.20-28.

[9] Hill, R., Hirsch, L., Lake, P. and Moshiri, S., 2012. Guide to cloud computing: principles and practice. Springer Science & Business Media.

[10] Sherwood, R., Gibb, G., Yap, K.K., Appenzeller, G., Casado, M., McKeown, N. and Parulkar, G., 2009. Flowvisor: A network virtualization layer. OpenFlow Switch Consortium, Tech. Rep, pp.1-13.

[11] Blenk, A., Basta, A., Reisslein, M. and Kellerer, W., 2015. Survey on Network Virtualization Hypervisors for Software Defined Networking.

[12] Sherwood, R., Gibb, G., Yap, K.K., Appenzeller, G., Casado, M., McKeown, N. and Parulkar, G.M., 2010, October. Can the production network be the testbed?. In OSDI (Vol. 10, pp. 1-6).

[13] Costa, V.T., and Costa, L.H.M., 2015. Vulnerabilities and solutions for isolation in FlowVisor-based virtual network environments. Journal of Internet Services and Applications, 6(1), pp.1-9.

[14] Min, S., Kim, S., Lee, J., Kim, B., Hong, W. and Kong, J., 2012, February. Implementation of an OpenFlow network virtualization for multi-controller environment. In Advanced Communication Technology (ICACT), 2012 14th International Conference on (pp. 589-592). IEEE.

[15] Do Nascimento Araujo, T. and Moreira Salles, R., 2014, November. GIROFLOW: Openflow virtualized infrastructure management tool. In Network and Service Management (CNSM), 2014 10th International Conference on (pp. 434-437). IEEE.

[16] Salvadori, E., Doriguzzi Corin, R., Broglio, A. and Gerola, M., 2011, December. Generalizing virtual network topologies in OpenFlow-based networks. In Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE (pp. 1-6). IEEE.

[17] Al-Shabibi, A., De Leenheer, M., Gerola, M., Koshibe, A., Snow, W. and Parulkar, G., 2014. Openvirtex: A network hypervisor. In Open Networking Summit 2014 (ONS 2014).

[18] Al-Shabibi, A., De Leenheer, M., Gerola, M., Koshibe, A., Parulkar, G., Salvadori, E. and Snow, B., 2014, August. OpenVirteX: Make your virtual SDNs programmable. In Proceedings of the third workshop on Hot topics in software defined networking (pp. 25-30). ACM.

[19] Bozakov, Z. and Papadimitriou, P., 2012, December. Autoslice: automated and scalable slicing for software-defined networks. In Proceedings of the 2012 ACM conference on CoNEXT student workshop (pp. 3-4). ACM.

[20] Drutskoy, D., Keller, E. and Rexford, J., 2013. Scalable network virtualization in software-defined networks. Internet Computing, IEEE, 17(2), pp.20-27.

[21] Drutskoy, D.A., 2012. Software-Defined Network Virtualization with FlowN. Master's Thesis, Princeton University.

[22] Benamrane, F. and Benaini, R., 2015. Performances of OpenFlow-based software-defined networks: an overview. Journal of Networks, 10(6), pp.329-337.

[23] Lara, A., Kolasani, A. and Ramamurthy, B., 2014. Network innovation using OpenFlow: A survey. Communications Surveys & Tutorials, IEEE, 16(1), pp.493-512.

[24] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. and Turner, J., 2008. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 38(2), pp.69-74.

[25] Hegr, T., Bohac, L., Uhlir, V. and Chlumsky, P., 2013. OpenFlow deployment and concept analysis. Advances in Electrical and Electronic Engineering, 11(5), p.327.

[26] Bari, M.F., Roy, A.R., Chowdhury, S.R., Zhang, Q., Zhani, M.F., Ahmed, R. and Boutaba, R., 2013, October. Dynamic controller provisioning in software defined networks. In Network and Service Management (CNSM), 2013 9th International Conference on (pp. 18-25). IEEE.

[27] Hassas Yeganeh, S. and Ganjali, Y., 2012, August. Kandoo: a framework for efficient and scalable offloading of control applications. In Proceedings of the first workshop on Hot topics in software defined networks (pp. 19-24). ACM.

[28] Heller, B., Sherwood, R. and McKeown, N., 2012, August. The controller placement problem. In Proceedings of the first workshop on Hot topics in software defined networks (pp. 7-12). ACM.

[29] Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T. and Shenker, S., 2010, October. Onix: A Distributed Control Platform for Large-scale Production Networks. In OSDI (Vol. 10, pp. 1-6).

[30] Tootoonchian, A. and Ganjali, Y., 2010, April. HyperFlow: A distributed control plane for OpenFlow. In Proceedings of the 2010 internet network management conference on Research on enterprise networking (pp. 3-3).

[31] Sherwood, R., Chan, M., Covington, A., Gibb, G., Flajslik, M., Handigol, N., Huang, T.Y., Kazemian, P., Kobayashi, M., Naous, J. and Seetharaman, S., 2010. Carving research slices out of your production networks with OpenFlow. ACM SIGCOMM Computer Communication Review, 40(1), pp.129-130.

[32] Liao, L., Shami, A. and Leung, V.C., 2015. Distributed FlowVisor: a distributed FlowVisor platform for quality of service aware cloud network virtualisation. IET Networks, 4(5), pp.270-277.

[33] Kreutz, D., Ramos, F.M., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S. and Uhlig, S., 2015. Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 103(1), pp.14-76.

[34] Haldar, K., and Agrawal, D., 2014. QoS ISSUES IN OPENFLOW/SDN In F.FEI, ed. 2014. Network Innovation through OpenFlow and SDN/ Principles and Design. United State: CRC Press.Ch.11.

[35] Egilmez, H.E., Dane, S.T., Bagci, K.T. and Tekalp, A.M., 2012, December. OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks. In Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific (pp. 1-8). IEEE.

[36] Bari, M.F., Chowdhury, S.R., Ahmed, R. and Boutaba, R., 2013, November. PolicyCop: an autonomic QoS policy enforcement framework for software defined networks. In Future Networks and Services (SDN4FNS), 2013 IEEE SDN for (pp. 1-7). IEEE.

[37] Ishimori, A., Farias, F., Cerqueira, E. and Abelém, A., 2013, October. Control of multiple packet schedulers for improving QoS on OpenFlow/SDN networking. In Software Defined Networks (EWSDN), 2013 Second European Workshop on (pp. 81-86). IEEE.

[38] Raza, M., Samineni, V. and Robertson, W., 2016, November. Physical and logical topology slicing through SDN. In Electrical and Computer Engineering (CCECE), 2016 IEEE Canadian Conference on (pp. 1-4). IEEE.

[39] FU,X., HU,F., 2014.QoS-ORIENTED DESIGN IN OPENFLOW In F.FEI, ed. 2014. Network Innovation through OpenFlow and SDN/ Principles and Design. United State: CRC Press.Ch.12

[40] Wang, Z., 2001. Internet QoS Architectures and Mechanisms for Quality of Service. United States of America: MORGAN KAUMANN PUBLISHERS

[41] Akyildiz, I.F., Lee, A., Wang, P., Luo, M. and Chou, W., 2014. A roadmap for traffic engineering in SDN-OpenFlow networks. Computer Networks, 71, pp.1-30.

[42] Graham, M., Winckles, A. and Sanchez-Velazquez, E., 2016. Practical Experiences of Building an IPFIX Based Open Source Botnet Detector. Le journal de la cybercriminalité & des investigations numériques, 1(1), pp.21-28.