Journal of Network and Computer Applications 80 (2017) 200–218

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Review

Quality of Service (QoS) in Software Defined Networking (SDN): A survey

Murat Karakus*, Arjan Durresi

Department of Computer and Information Science, Indiana University Purdue University Indianapolis, Indianapolis, IN 46202, USA

A R T I C L E I N F O

Keywords:
Quality of service
Software defined networking
OpenFlow
SDN
QoS
Survey

A B S T R A C T

Supporting end-to-end Quality of Service (QoS) in existing network architectures is an ongoing problem. Although researchers from both academia and industry have proposed many solutions to solve the QoS limitations of the current networking, many of them either failed or were not implemented. Software Defined Networking (SDN) paradigm has emerged in response to limitations of traditional networking architectures. Its main advantages are the centralized global network view, programmability, and separation of the data plane and control plane. These features have got attention of researchers to improve the QoS provisioning of today's various network applications. In this survey paper, we aim at making a picture of QoS-motivated literature in OpenFlow-enabled SDN networks by comprehensively surveying relevant research studies. We organize the related studies according to the categories that are the most prominent ways in which QoS can benefit from the concept of SDN: Multimedia flows routing mechanisms, inter-domain routing mechanisms, resource reservation mechanisms, queue management and scheduling mechanisms, Quality of Experience (QoE)-aware mechanisms, network monitoring mechanisms, and other QoS-centric mechanisms such as virtualization-based QoS provisioning and QoS policy management etc. In addition, we discuss QoS capabilities of OpenFlow protocol by reviewing its versions along with some well-known, open-source, and community-driven controller projects. Furthermore, we outline the potential challenges and open problems that need to be addressed further for better and complete QoS abilities in SDN/OpenFlow networks and lessons we have learned during preparation of this survey paper.

1. Introduction

With the growth of the Internet, new types of networking applications and services (e.g. web surfing, texting, VoIP, email, audio, video conferencing and streaming, online gaming, e-commerce etc.) have emerged for end users. These applications and services generate their own characteristic flows which need to be delivered by the Internet. However, all of these applications require different treatments for their own flows to make the delivery successful over a network (Liu et al., 2015). For example, some applications such as video conferencing require a certain bandwidth for its flows while applications like VoIP are more sensitive to the delay over a network (Yang et al., 2016). Addressing these requirements needs a well-defined Quality of Service (QoS) mechanism(s) in a network. However, today's *de facto* delivery model, *best-effort*, in the Internet is not capable of serving to all of the aforementioned services. In addition, proposed QoS solutions have not been successful enough to solve the QoS issues of the traditional networking paradigms.

The Internet Engineering Task Force (IETF) has defined various types of QoS architectures to support QoS provisioning. The *Integrated Services (IntServ)* model (Braden et al., 1994) It utilizes the resource reservation protocol (RSVP) (Zhang et al., 1997) to provide the QoS to end users. In IntServ model, resources are explicitly reserved through an end-to-end path and hence all routers store network states related to the service. Therefore, it suffers from the scalability and complexity issues. To mitigate that scalability issue, researchers have proposed the *Differentiated Services (DiffServ)* model (Blake et al., 1998), which is on flow-aggregation basis and exploits the hop-by-hop process. It classifies the incoming flows (using pre-configured classes) based on the Type of Service (ToS) field in the header of the packets. Since DiffServ treats packets in the same class identically, it is difficult to provide quantitative QoS to individual flows. It is strong on simplicity, but weak on guarantees. The *Multiprotocol Label Switching (MPLS)* (Rosen et al., 2001) is another technology that is used to reduce the complex routing table lookups by labeling techniques. All of these advantages and disadvantages show that the current QoS architectures are not truly successful at QoS support for service providers, enterprises and/or end users.

Software Defined Networking (SDN) (Software-Defined Networking: The New Norm for Networks, 2012; Li et al., 2016;

* Corresponding author.
E-mail addresses: mkarakus@iupui.edu (M. Karakus), durresi@cs.iupui.edu (A. Durresi).

http://dx.doi.org/10.1016/j.jnca.2016.12.019
Received 27 May 2016; Received in revised form 9 November 2016; Accepted 6 December 2016
Available online 09 December 2016

Akhunzada et al., 2016; Masoudi and Ghaffari, 2016) is a new emerging architecture in recent years. SDN is described in Open Networking Foundation's (Software-Defined Networking: The New Norm for Networks, 2012) definitions as "*In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications*". This separation provides network operators/administrators with efficient use of network resources and ease of resource provisioning. Also, SDN brings ease of programmability to change the characteristics of whole networks. This ability simplifies the management of the network since it is decoupled from the data plane. Therefore, network operators can easily and quickly manage, configure, and optimize network resources with dynamic, automated and proprietary-free programs written by themselves in SDN architecture (Sezer et al., 2013; Bakshi, 2013).

In addition, since the SDN is logically centralized, controllers have a global visibility of the whole network unlike conventional networking. Hence, they can dynamically optimize flow-management and resources. Furthermore, per-flow or application-level QoS provisioning becomes easier and feasible for network administrators. For these reasons, SDN is drawing attention of companies, universities, data centers, and service providers to be deployed in their networks. Google's private WAN (B4 (Jain et al., 2013)), connecting Google data centers across various geographical location over the world, is one of the examples for SDN adoption in a large-scale network with the aforementioned purposes.

1.1. Survey organization

In this survey paper, we aim at making a picture of QoS-motivated literature in OpenFlow-enabled SDN networks by surveying relevant research studies. The scope of this paper revolves around the term *QoS* characterized by network characteristics such as bandwidth, delay, jitter, and loss along with industry-wide set of standards and mechanisms for ensuring high-quality performance for critical applications. Focus of studies presented in and the scope of this paper are centered around aforementioned typical network characteristics.

As seen in Fig. 1, we organize the related studies into seven categories that are the most prominent ways in which QoS can benefit from the concept of SDN: Multimedia flows routing mechanisms, inter-domain routing mechanisms, resource reservation mechanisms, queue management and scheduling mechanisms, Quality of Experience (QoE)-aware mechanisms, network monitoring mechanisms, and other QoS-centric mechanisms such as virtualization-based QoS provisioning and QoS policy management etc. We should note that each category itself in our organization reflects a problem/challenge for QoS in SDN. Therefore, the organization is indeed a taxonomy of the problems for QoS in SDN at the same time. We explain these categories (i.e. problems) and related studies (i.e. solutions) in corresponding sections. In addition, we discuss QoS capabilities of OpenFlow protocol by reviewing its versions along with some well-known, open-source, and community-driven control platform projects. Finally, we outline the potential challenges and open problems that need to be addressed further for improved and complete QoS abilities in OpenFlow-enabled SDN networks.

In this paper, we give an overview of the relations between QoS and SDN. This survey paper may be a useful primer for a reader interested in studying QoS in/with SDN. After reading this survey paper, the reader will be familiar with:

- A lightweight overview of the SDN Architecture
- QoS capabilities of specific OpenFlow protocol versions
- QoS support of some well-known, active, and open-source SDN controller projects
- QoS problems in SDN and related solutions from researchers
- Some other potential challenges and critical points for QoS in SDN requiring attention of research community

To the best of our knowledge, this study is the first attempt for such a survey paper comprehensively examining QoS in SDN/OpenFlow networks.

The rest of the paper is organized as follows: Section 2 gives a lightweight overview of the SDN framework with OpenFlow protocol. In Section 3, we discuss the QoS capabilities of OpenFlow protocol in its different versions and that of (well-known) open-source SDN control platforms. Section 4 summarizes the role of SDN with relation to QoS. While Section 5 outlines multimedia flows-based routing mechanisms Section 6 presents inter-domain QoS routing frameworks. Section 7 introduces resource reservation based frameworks to provide QoS. Section 8 discusses frameworks focusing on queue management and packet scheduling. Section 9 states the QoE-oriented mechanisms. In Section 10, we present network monitoring frameworks. Section 11 discusses miscellaneous QoS-related mechanisms. Section 12 outlines few potential challenges and open problems for QoS support in OpenFlow networks along with lessons we have learned while preparation of this survey. Finally, Section 13 wraps the paper up with concluding remarks.

2. An overview of SDN architecture and OpenFlow Protocol

SDN architecture with OpenFlow protocol enables network operators to treat flows in a finer-granular way compared to the traditional networks by means of controllers. In a traditional network, flows (or packets) are mainly treated based on a single or a few attribute combinations of packet headers, such as longest destination IP prefixes, destination MAC addresses, or a combination of IP addresses and TCP/UDP port numbers etc. SDN allows us to manage flows based on more attributes of packet headers by means of a Controller-Data Plane Interface (C-DPI) such as OpenFlow protocol.

As shown in Fig. 2, Open Networking Foundation (ONF) vertically splits SDN architecture into three main planes (SDN architecture, 2014):

- *Data Plane*: Data plane is the bottom plane and consists of network devices such as routers, physical/virtual switches, access point etc. These devices are accessible and managed through C-DPIs (Controller-Data Plane Interfaces) by SDN controller(s). The network elements and controller(s) may communicate through secure

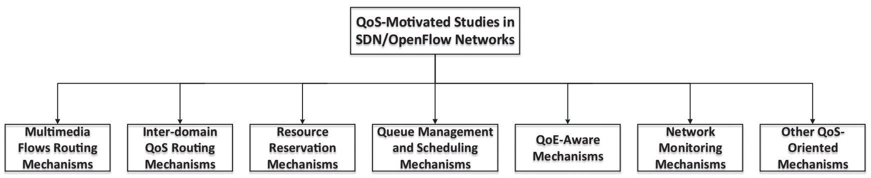
QoS-Motivated Studies in SDN/OpenFlow Networks
Multimedia Flows Routing Mechanisms
Inter-domain QoS Routing Mechanisms
Resource Reservation Mechanisms
Queue Management and Scheduling Mechanisms
QoE-Aware Mechanisms
Network Monitoring Mechanisms
Other QoS-Oriented Mechanisms

Fig. 1. The Organization of QoS-based studies in SDN/OpenFlow networks. The first two types of mechanisms are driven by the routing functionality. The third and fourth types of mechanisms are concentrated around resource reservation and queue management and packet scheduling for QoS support. The fifth type of the studies address the QoE of the system while the sixth group of the studies revolve around network monitoring frameworks. The last group of the mechanisms study miscellaneous QoS-related issues such as QoS policy management, QoS testbed extensions etc.

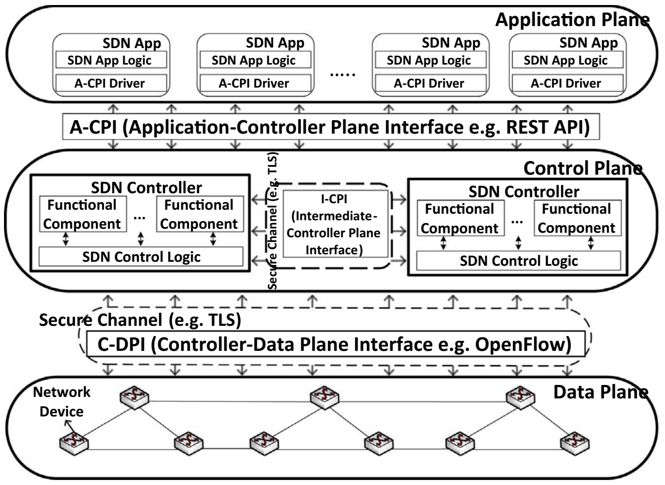
Application Plane
SDN App
SDN App Logic
A-CPI Driver
A-CPI (Application-Controller Plane Interface e.g. REST API)
Control Plane
SDN Controller
Functional Component
SDN Control Logic
Secure Channel (e.g. TLS)
I-CPI (Intermediate-Controller Plane Interface)
C-DPI (Controller-Data Plane Interface e.g. OpenFlow)
Network Device
Data Plane

Fig. 2. An Overview of SDN architecture with its main planes: Data plane, control plane, and application plane.

connections such as the TLS connection. OpenFlow protocol (McKeown et al., 2008; Vaughan-Nichols, 2011) is the most prevalent standard C-DPI used for communication between controller(s) and data plane devices.

- *Controller Plane*: An SDN controller plane comprises of one or more software-based SDN controller(s) to provide control functionality by supervising the network forwarding behavior through C-DPI. It has interfaces to enable communication among controllers in a control plane (Intermediate-Controller Plane Interface, i.e. I-CPI (Lin et al., 2015), optionally secured using the TLS), between controllers and network devices (C-DPI), and also between controllers and applications (Application-Controller Plane Interface, i.e. A-CPI). An A-CPI[1] renders the communication possible between network applications/services and controller(s) for network security, management etc. A controller consists of two main components: Functional components and control logic. Controllers include more than one functional components such as Coordinator, Virtualizer etc. to manage controller behavior. Furthermore, The control logic maps networking requirements of applications into instructions for network element resources (SDN architecture, 2014).
- *Application Plane*: An SDN application plane consists of one or more network applications (e.g. security, visualization etc.) that interact with controller(s) to utilize abstract view of the network for their internal decision making processes. These applications communicate with controller(s) via an open A-CPI (e.g. REST API). An SDN application comprises of an SDN App Logic and A-CPI Driver.

In an SDN network with OpenFlow-enabled switches, there are three main parts in a switch: *Flow Table*, *Secure Channel*, and *OpenFlow Protocol*. An OpenFlow switch maintains a number of flow tables containing a list of flow entries. Each flow entry consists of 3 parts: A "Rule" field to define the flow entry based on certain header attributes such as source/destination addresses, an "Action" field to apply on a packet matching the values in the "Rule" field, and a "Stats" field to maintain some counters for the entries (OpenFlow Switch Specification, 2014). A *Secure Channel* (e.g. TLS) is the interface that connects data plane elements to a remote controller. Switches are managed and configured by the controller over the secure channel. In addition, the controller receives events from the switches and sends packets out to switches through this channel.

In SDN, a controller can work in three operational modes to setup a new flow rule (a.k.a flow entry): reactive mode, proactive mode, and hybrid mode (Fernandez, 2013):

- *Reactive Mode*—In the reactive mode, when a new packet arrives to a network device (e.g. switch), the switch does a flow rule lookup in its flow tables. If no match for the flow is found, the switch forwards it to the controller using C-DPI so that the controller decides how to handle the packet. After the controller processes the packet according to the network policies, it creates and sends a flow entry to be installed in the network device. Future flows matching with this flow entry, based on packet header attributes, will be treated according to the corresponding matching rule.
- *Proactive Mode*—In the proactive mode, flow entries are setup in flow tables of the switches before new flows arrive at the switches. When a packet arrives at a switch, the switch already knows how to deal with that packet. In this case, the controller is not involved in any flow rule setup process.
- *Hybrid Mode*—In the hybrid mode, a controller benefits advantages of both reactive and proactive modes. It is quite possible that network administrators proactively install certain flow entries in data plane devices and the controller(s) reactively modify (delete/update) them or even add new flow entries based on incoming traffic.

While the proactive mode brings some concerns regarding inefficient use of switch memory, the reactive mode provides more agile, flexible, and dynamic environment for both controllers and switches (Fernandez, 2013; Braun and Menth, 2014).

3. QoS Implementation in OpenFlow-Enabled SDN networks

Although SDN and OpenFlow couple support some limited QoS capabilities it allows us to obtain per-flow QoS control in a more scalable, flexible and finer-granular way compared to the above traditional architecture. In this section, we review the QoS capabilities of OpenFlow protocol by looking at its different versions and that of (well-know) open-source SDN control platforms.

3.1. QoS in openFlow protocol

Each OpenFlow specification version has brought some different features along with minor and major changes compared to their previous versions. In the following, we highlight the QoS-related features and changes implemented in the different versions of OpenFlow specification.

OpenFlow 1.0—In OpenFlow 1.0 OpenFlow Switch Specification (2009), there is an optional action called *enqueue*[2] which forwards packet through a queue attached to a port. An OpenFlow-enabled switch can have one or more queues depending on its ports. An OpenFlow controller can query an information about queues of a switch. However, the behavior of the queue is determined outside the scope of OpenFlow, which can be configured through the OF-CONFIG protocol (OpenFlow Management and Configuration Protocol, 2014) but requires OpenFlow 1.2 and later versions. Also, header fields can include VLAN priority and IP ToS, so packets can be matched against rules and their associated header fields can be rewritten.

OpenFlow 1.1—OpenFlow 1.1 OpenFlow Switch Specification (2011) performs matching and tagging of VLAN and MPLS labels and traffic classes. Prior versions of OpenFlow specification had limited VLAN support (only supported a single level of VLAN tagging with ambiguous semantic). The new tagging support has explicit actions to add, modify and remove VLAN tags, and can support multiple levels of VLAN tagging. This version also adds a similar support the MPLS shim headers.

OpenFlow 1.2—OpenFlow 1.2 OpenFlow Switch Specification (2011) has added an ability that enables a controller to query all

[1] An A-CPI is mostly called as "Northbound Interface (NBI)" by the SDN community.

[2] This action has been renamed to *set_queue* in OpenFlow 1.1 and later versions.

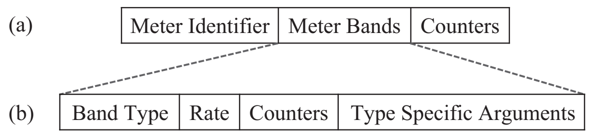

Fig. 3. Main components of a meter band (b) in a meter entry (a).

queues in a switch. It also has added experimenter queue property. Another QoS related improvement in this version is that it has added a max-rate queue property. In addition, this version specifies that queues can be attached to ports and be used to map flows on them.

OpenFlow 1.3—OpenFlow 1.3 OpenFlow Switch Specification (2012) introduces the rate-limiting functionality by means of meter tables consisting of meter entries. A meter entry consists of “Meter Identifier”, “Meter Bands”, and “ Counters”. A Meter Band, in turn, consists of “Band Type” (e.g. drop or remark DSCP etc.), “Rate” (e.g. kb/s burst), “Counters”, and optional “Type specific arguments”, such as drop and DSCP remark, as seen in Fig. 3. Counters may be maintained per-queue, per-meter, and per-meter band etc. They help controller collect statistics about the network. There may be one or more meter bands per meter table entry. Meters can be combined with the optional *set_queue* action, which associates a packet to a per-port queue in order to implement complex QoS frameworks such as DiffServ. These meters complement the queue framework already in place in OpenFlow by allowing for the rate-monitoring of traffic prior to output. More specifically, with meters, we can monitor the ingress rate of traffic as defined by a flow rule. Packets can be directed to a specific meter using the optional *meter*(meter_ id) instruction, where the meter can then perform some operations based on the rate it receives packets.

OpenFlow 1.4—OpenFlow 1.4 OpenFlow Switch Specification (2013) presents the flow monitoring framework that allows a controller to monitor the changes done by other controllers to any subsets of the flow tables in real time. To this end, a controller can define a number of monitors, each selecting a subset of the flow tables. Each monitor includes a table id and a match pattern that defines the subset monitored. When any flow entry is added, modified or removed in one of the subsets defined by a flow monitor, an event is sent to the controller to inform it about the change.

OpenFlow 1.5—OpenFlow 1.5 OpenFlow Switch Specification (2014) replaces the *meter* instruction, which was used for metering in previous versions, with a *meter* action. As a result, multiple meters can be attached to a flow entry, and meters can be used in group buckets.

3.2. QoS in SDN Controllers

Since OpenFlow does not currently provide support for queue configuration in its specification, queue configuration is handled by specific OF-CONFIG and OVSDB (Open vSwitch Database Management Protocol) (Pfaff and Davie, 2013) protocols. The former is currently being standardized by ONF and the latter is already standardized by the IETF. Although OVSDB is already implemented in OVS switches, there is no available controllers providing a standardized management of queues. Currently, there are many different SDN controller platforms offering various features for users. Although there are many commercial and proprietary SDN controllers from different vendors, there also exist some collaborative and open-source projects with active development support from research community and industry. Below, we discuss some of these active, open-source, and collaborative SDN controller projects with regards to their QoS support.

OpenDaylight—OpenDaylight (ODL) OpenDaylight Project is a community-led and open-source controller platform. It is a Linux Foundation collaborative project to promote use of SDN. The ODL community has come together to establish an open reference controller framework to freely program and control an SDN architecture. ODL project consists of many other sub-projects, such as southbound protocol plugins (e.g. OpenFlow, NetCONF, SNMP, and BGP) and applications (e.g. DDoS Protection and Virtualization Coordinator), complementing each other to compose a complete reference controller platform for heterogeneous networks. PCMM (PacketCable MultiMedia), presented in ODL-Lithium release in June 2015, plugin is another southbound plugin utilized to enable flow-based dynamic QoS for the DOCSIS infrastructure. Packet Cable MultiMedia (PCMM) provides an interface to control and management service flow for CMTS network elements. Also, OVSDB southbound plugin has been introduced in ODL-Lithium release, which can manage and configure queues in switches. In addition, the Reservation module in ODL also aims at providing dynamic low-level resource reservations so that users can get network services, connectivity or a pool of resources (ports, bandwidth) for a specific period of time.

ONOS—ONOS (Open Network Operating System) (ONOS Project) is a distributed SDN control platform aimed at improving scalability, performance and availability of networks for service providers. It is also an open-source platform with over 50 partners and collaborators that contribute to all aspects of the project. ONOS has limited QoS support currently. It supports OpenFlow metering mechanism, but this feature is rarely implemented in existing switches. The idea behind this support is based on implementation of OpenFlow *set_queue* functionality in ONOS. As another QoS support improvement attempt in ONOS, a new high-level instruction *SetQueueInstruction* has been implemented in org.onosproject.net.flow.instructions library and the corresponding references in ONOS libraries have been modified accordingly.

Floodlight—Floodlight (Floodlight Project) is a Java-based another open-source SDN controller that is supported by community developers including engineers from Big Switch Networks. There are community driven projects built on top of Floodlight proposing integrating/updating new/existing modules. QoS module (Wallner and Cannistra, 2013) implemented for Floodlight controller aims at providing an application that does burden of matching, classification, flow insertion, flow deletion, and policy handling for QoS. The module utilizes OpenFlow 1.0 *enqueue* action and the network ToS bits. It controls tracking and storing services with their DSCP values, applying policies for services class, and tracking of policies in switches. The QueuePusher (Palma et al., 2014) extension utilizes OVSDB protocol integrated with northbound API of Floodlight to generate appropriate queue configuration messages. The QueuePusher module uses a CRUD (Create, Read, Update, Delete) API, exposed by Floodlight, that allows external entities to manage Open vSwitch.

4. Relationship between SDN and QoS

QoS is typically defined as an ability of a network to provide the required services for a selected network traffic. The primary goal of QoS is to provide priority with respect to QoS parameters including but not limited to:

- bandwidth
- delay
- jitter
- loss

characteristics. In order to provide QoS, differentiating application flows is needed since they battle for available network resources. These network resources have to be allocated to ensure the precedence of the higher-priority traffic for the appropriate network resource distribution. This process often requires knowledge of the current network states, so that the right decisions with regard to packet forwarding can be made.

Today, QoS provisioning mostly relies on Service Level Agreements (SLAs) between end users and service providers. This approach works

Table 1
QoS models implemented in the techniques. The hard QoS approach guarantees the network resources for flows sent from source to destination. IntServ mechanism is an example for this approach. On the other hand, the soft QoS method does not guarantee the QoS requirements of the flows sent from source to destination throughout the entire session. DiffServ method is an example of soft QoS method.

	QoS Models							
	Hard QoS				**Soft QoS**			
Techniques	**Bandwidth**	**Delay**	**Jitter**	**Loss**	**Bandwidth**	**Delay**	**Jitter**	**Loss**
Wallner and Cannistra (2013)					✓	✓		
Civanlar et al. (2010)					✓	✓		✓
HiQoS - Jinyao et al. (2015)					✓	✓		
OpenQoS - Egilmez et al. (2012)					✓	✓		✓
VSDN - Owens and Durresi (2013)	✓	✓	✓					
RVSDN - Owens et al. (2014)	✓	✓	✓					
Tomovic et al. (2014)	✓							
Egilmez et al. (2011)					✓	✓		✓
Egilmez et al. (2013)					✓	✓	✓	✓
ARVS - Yu et al. (2015)							✓	✓
Yilmaz et al. (2015)							✓	✓
Egilmez et al. (2012)							✓	✓
Egilmez and Tekalp (2014)					✓	✓	✓	✓
Karakus and Durresi (2015)	✓	✓						
Marconett and Yoo (2015, 2015)	✓	✓		✓	✓	✓		✓
Wang et al. (2015, 2014)	✓	✓						
Miao et al. (2015)		✓		✓		✓		✓
CXP - Kotronis et al. (2016)	✓	✓			✓	✓		
Kim et al. (2010)	✓	✓		✓				
NCL - Bueno et al. (2013)	✓	✓	✓					
Duan (2014); Duan et al.	✓	✓			✓	✓		
FlowQoS - Seddiki et al. (2015, 2014)	✓	✓			✓	✓		
Afaq et al. (2015, 2015)					✓			
QoSFlow - Ishimori et al. (2013)					✓			
OpenQFlow - Nam-Seok et al. (2013)	✓				✓			
Xu et al. (2015)					✓	✓	✓	
J. Wang et al. (2015); W. Wang et al. (2015)	✓	✓			✓	✓		
Caba and Soler (2015)	✓	✓			✓	✓		
Huong-Truong et al. (2013)					✓	✓	✓	✓
Kumar et al. (2013)	✓				✓			
Yiakoumis et al. (2012)	✓	✓	✓		✓	✓	✓	
Kassler et al. (2012)					✓	✓		
Q-POINT - Dobrijevic et al. (2014)	✓	✓		✓	✓	✓		✓
QFF - Georgopoulos et al. (2013)					✓			
Gorlatch et al. (2014); Gorlatch and Humernbrum (2015)					✓	✓		
Jarschel et al. (2013)	✓				✓			
Ayadi et al. (2013)					✓	✓		
Q-Ctrl - Govindarajan et al. (2014)	✓				✓			
PolicyCop - Bari et al. (2013)					✓	✓		
OpenCache - Broadbent et al. (2015); Broadbent and Race (2012)					✓	✓		
Sonkoly et al. (2012)	✓	✓	✓		✓	✓	✓	
SoIP - Hu et al. (2015)					✓			
ACDPA - Desai and Nagegowda (2015)					✓	✓		

well for best-effort service and does not support finer-granular traffic control. However, there are other types of applications, such as VoIP, online-gaming, and video conferencing, whose flows are sensitive to delay, jitter, and bandwidth, thereby requiring special handling. Also, "hop-by-hop" decision architecture of the Internet is sometimes difficult to monitor, mainly because of the many different vendor-specific firmwares at use. There is no standardized way for specifying high level traffic control policies and restrictions with regard to the depth of traffic differentiation exist.

QoS is mainly implemented in two approaches: hard QoS and soft QoS. The hard QoS method guarantees the QoS requirements of connections but it suffers from resource limitations. IntServ method is an example of this type of QoS guaranteeing approach. On the other hand, the soft QoS method is not as strict as hard QoS methods regarding QoS requirements. DiffServ is an example of the soft QoS method. Table 1 illustrates the implemented QoS models (hard QoS vs. Soft QoS) with respect to QoS metrics considered in the survey studies. Certain metrics are considered target QoS metrics to be provided in the studies. Therefore, Table 1 also reveals a broad category of problems/challenges handled in the studies from the QoS metrics viewpoint.

SDN adopts separation of data plane and control plane for networks. This separation enhances the network controller with regard to control of the networks. Also, in SDN concept, the network applications are not forced to deal with low-level configurations of data plane devices and are provided with abstract view of the network by controllers. Controllers can obtain global network view and states, e.g. statistics, network resource availability, events, by sampling of packets. Using this information, control policies and SLAs can be specified (even dynamically be adjusted) by an administrator at a higher abstraction level without a need to reconfigure low-level settings at each of the forwarding devices. The set of policies and also the different flow classes are unrestricted, allowing for fine-grained tuning based on the needs of the user. The rules can, therefore, be defined per-flow (if necessary) and the controller has the task to apply them properly to the different network elements. Without a doubt, all of these mechanisms are crucial for QoS.

QoS can benefit from advantages of SDN concept in different network functions. Table 2 shows some main features of SDN, which

Table 2
Main features of SDN, which are used in the surveyed papers, and their relation with the organization of this survey.

SDN features	Organization						
	Routing	Architecture	Management			Modeling	
	Multimedia flows routing mechanisms	Inter-domain QoS routing mechanisms	Resource reservation mechanisms	Queue Management and scheduling mechanisms	Network monitoring mechanisms	QoE-aware mechanisms	Other QoS-related mechanisms
Flow-based forwarding	✓	✓				✓	
Dynamic Flow Rule Update	✓	✓	✓	✓		✓	✓
Flow/Packet Analysis	✓	✓	✓	✓	✓	✓	✓
Flow Path Analysis		✓	✓		✓		✓
Traffic Monitoring			✓	✓	✓	✓	✓
Queue Configuration	✓			✓		✓	✓

are used in the surveyed papers, and their relation with our organization. Flow based forwarding allows networks to route different application flows in different treatments (e.g. priorities). Dynamic flows rule update enables network operators to update rules installed in network devices on-the-fly without interrupting device operations. SDN also renders flow/packet analysis possible to acquire header fields of them. Since SDN provides global network view it is possible to maintain related states for a full path of a flow. Furthermore, monitoring network statistics based on different levels such per-flow, per-port, per-device and so on is achievable. Moreover, in OpenFlow-enabled SDN networks, queue management and scheduling operations are also possible by means of some other southbound plugins such as OF-CONFIG and OVSDB protocols.

- One function that SDN can help networks improve is QoS-motivated routing. With SDN architecture, per-flow routing (both intra-domain and inter-domain) becomes viable through more scalable, simpler and less time-consuming mechanisms compared to traditional architectures. OpenFlow enables network operators to use various routing algorithms (rather than the typical shortest path) within the controller to generate forwarding tables that govern different isolated flows, such as the QoS flows, in the data plane (Tomovic et al., 2015). Also, dynamic routing of flows are viable by controllers due to decoupling of control and forwarding functions of devices. These abilities, per-flow and dynamic routing, allow network administrators to come up with more QoS-motivated routing mechanisms for their networks.
- Also, SDN can help network operators create powerful and easy-to-use automated QoS management frameworks by means of resource reservation and queue management and packet scheduling for their networks. QoS provisioning for network applications require well-defined control mechanism due to dynamic nature of network resources. As SDN brings capabilities to obtain global view of network controlling QoS configuration becomes easier compared to traditional network architectures.
- Furthermore, user QoE improvement can also benefit from SDN capabilities. User satisfaction cannot be guaranteed just by providing certain QoS parameters since these low-level network parameters represent the network states in terms of numbers. However, real user satisfaction (i.e. QoE) may require different QoS parameters which can dynamically change over the time. SDN's ability to manage network flows in a finer-granular way by flow rules through an automated control can help improve user QoE in a network.
- Moreover, network monitoring task is another function that SDN can help within a network. Monitoring task is crucial for a network since it helps detect and respond threads, performance issues in real time, and predicting future behaviors in a network. SDN allows network managers to monitor network dynamics through counters at very low levels such as per-packet, per-port, per-table, per-queue, and per-meter.
- Finally, SDN can be utilized to provide QoS in some miscellaneous ways such virtualization-based QoS provisioning, QoS policy management, and content delivery mechanisms due to some of its features such as per-flow control concept and multi-header field based routing.

The aforementioned network functions/tasks mainly form the underlying logic of the organization in which we present the surveyed papers. These categories are the most prominent ways in which QoS can benefit from the concept of SDN.

In the rest of the paper, Table 3 shows some features of studies such as queuing and/or scheduling mechanisms, scaling domain, simulation and/or emulation environment, and the controller(s) exploited in the development stages. Table 4 illustrates the corresponding SDN planes that the techniques impact. Table 5 illustrates the organization, based on the categories identified, of the studies surveyed in the paper along

Table 3
Some features of studies such as queuing and/or scheduling mechanisms, scaling domain, simulation/emulation environment, and the controller(s) exploited in the development stages of the techniques.

	Features			
Techniques	**Queuing/Scheduling**	**Scale (Domain)**	**Simulation/Emulation environment**	**Controller**
Wallner and Cannistra (2013)	Default	Single	Presentation of the architecture, no simulation or testbed implementation	Floodlight
Civanlar et al. (2010)	Default	Single	Simple 4-forwarder OpenFlow testbed	NOX
HiQoS - Jinyao et al. (2015)	Default	Single	Used Mininet tool and topology with 5 switches, 2 servers, and 11 clients	Floodlight
OpenQoS - Egilmez et al. (2012)	Default	Single	3 OpenFlow-enabled Pronto 3290 switches, 1 controller, 3 hosts	Floodlight
VSDN - Owens and Durresi (2013)	Token Bucket Shaping (TBS), Weighted Fair Queuing (WFQ)	Single	A topology of 6 nodes in NS-3 simulator	VSDN
RVSDN - Owens et al. (2014)	TBS and WFQ	Single	A topology of 6 nodes in NS-3 simulator	RVSDN
Tomovic et al. (2014)	HTB	Single	6 Open vSwitch (OVS), 4 clients, 4 servers	POX
Egilmez et al. (2011)	Default	Single	Used a simulator implemented using LEMON library, used 6 nodes	Any
Egilmez et al. (2013)	Default	Multi	Used a simulator implemented using LEMON library, used 15 domains with 300 nodes total	Any
ARVS - Yu et al. (2015)	Default	Single	Used Mininet and a topology with 30 nodes, 20 Mbps bandwidth, 10 ms and 20 ms delays randomly of each link	Floodlight
Yilmaz et al. (2015)	Default	Single	Used 2 servers, 2 switches, 1 controller and a traffic loader to test different scenarios	OpenDaylight
Egilmez et al. (2012)	Default	Multi	Used a simulator implemented using LEMON library, used 6 domains with 30 nodes each	Any
Egilmez and Tekalp (2014)	Default	Multi	Used a simulator implemented using LEMON library, used 6 domains with 30 nodes each	Any
Karakus and Durresi (2015)	Default	Multi	4 domains with 4 nodes each in simulation	Any
Marconett and Yoo (2015, 2015)	Default	Multi	Used Mininet to test 5 different scenarios	Floodlight
Wang et al. (2015, 2014)	WFQ, RED, Self-clocked Fair Queueing (SCFQ)	Multi	Used Google's IDC backbone network (G-WAN) and IBM's global data center network (SoftLayer) for extensive simulations	Central Traffic Manager
Miao et al. (2015)	Default	Single	Used virtual networks connected to each other with ToRs and Racks.	OpenDaylight
CXP - Kotronis et al. (2016)	Default	Multi	Used 5 IXPs data	Any
Kim et al. (2010)	Priority Queuing (PQ)	Single	3 ProCurve 5406zl switches	NOX
NCL - Bueno et al. (2013)	Default	Single/Multi	Presented a use case with description of the architecture	Floodlight
Duan (2014); Duan et al.	Default	Single/Multi	Numerical Results with examples	Any
Seddiki et al. (2015, 2014)	Default	Single	OpenWrt router with OVS integration, Raspberry Pi as controller hardware	POX
Afaq et al. (2015, 2015)	Default	Single	Used Mininet and a linear topology with 4 OVSes connected to a host each	Floodlight
QoSFlow - Ishimori et al. (2013)	HTB, Stochastic Fairness Queuing (SFQ), RED	Single	Up to 3 TPLink 1043ND switches	Any
OpenQFlow - Nam-Seok et al. (2013)	Rate Controlled Static Priority (RCSP), CETA based scheduling	Single	Used a data plane module Cavium OCTEON CN5650 with multicore processors, each assigned different tasks	Any
Xu et al. (2015)	Pre-defined queues w/ priority levels	Single	A topology w/ 8 switches and 6 hosts in Mininet and also a psychical network w/ 3 switches	RYU
J. Wang et al. (2015); W. Wang et al. (2015)	Weighted RED, PQ, Weighted Round-Robin (WRR) scheduling	Single	Used a testbed with 3 Dell R410 servers and 4 R710 servers	NOX
Caba and Soler (2015)	Priority Queuing (PQ)	Single	Used Distributed OpenFlow Testbed (DOT), Roy et al. (2014) and topology consisting VMs containing controller and 4 OVS switches and 3 hosts	Floodlight
Huong-Truong et al. (2013)	Default	Single/Multi	Implemented in a testbed infrastructure, consisting of 3 layers, set up in their lab	Any
Kumar et al. (2013)	FIFO and HTB	Single	Emulated a small home network with TP-LINK WR1043ND gateway router and DELL PowerEdge R620 OVS switch as ISP access switch	Floodlight
Yiakoumis et al. (2012)	Minimum-rate queuing and WFQ	Single	Implemented a minimal user-ISP with TP-LINK WR1043ND home gateway router and 48-port Pronto switch as ISP access switch	Any
Q-POINT - Dobrijevic et al. (2014)	WFQ	Single	Evaluated in a random topology with 4 nodes and CARnet-like topology with 9 nodes using IBM ILOG CPLEX Optimization Studio	Any
QFF - Georgopoulos et al. (2013)	Default	Single	A testbed recreating home network with TP-LINK WR1043ND home gateway router with Pantou and 3 clients	Any
Gorlatch et al. (2014); Gorlatch and Humernbrum (2015)	Default	Single	Numeric experimental study for low-level and application-level QoS metrics	Any
Jarschel et al. (2013)	Default	Single	A testbed with two Pronto 3290 switches and a Dell PowerEdge 860 server as controller platform	Floodlight

(continued on next page)

Table 3 (*continued*)

Techniques	Features			
	Queuing/Scheduling	**Scale (Domain)**	**Simulation/Emulation environment**	**Controller**
Ayadi et al. (2013)	Default	Single	Numerical evaluation	Any
Q-Ctrl - Govindarajan et al. (2014)	Default	Single	Real-time experimental setup with 2 PowerEdge T110 II servers, HP2920 and Pica8 switches and 6 VMs launched in servers	Floodlight
PolicyCop - Bari et al. (2013)	Default	Single	An experiment with 5 switches and 4 hosts	Floodlight
OpenCache - Broadbent et al. (2015?)	Default	Single	No complete experiments yet	NOX
SoIP - Hu et al. (2015)	Default	Single/Multi	Used 3 switches w/ 100 Mbps link capacities for edge network and 2 routers for core network	Any
ACDPA - Desai and Nagegowda (2015)	Default	Single	Used Mininet and a topology w/ randomly connected 20 switches and 30 hosts	OpenDaylight

with their short descriptions.

5. Multimedia flows routing mechanisms

With proliferation of different applications (e.g. video conferencing, VoIP etc.) on the Internet, more sophisticated and efficient routing mechanisms are needed for these types of application to meet their QoS demands. However, routing in today's traditional networking is an ossified issue due to some unsolved issues such as network's limited global view, per-hop decisioning, and limited QoS abilities for flows. The SDN and OpenFlow couple is considered a prospective solution architecture for the routing problems of the current networking. Decoupling of control plane and data plane in SDN brings many opportunities to routing functionalities. Supporting QoS in SDN/OpenFlow networks becomes more feasible owing to a logically centralized controller component of the SDN. With OpenFlow, it is possible to use various routing algorithms with different objectives such certain delay limit or packet loss (rather than just shortest path routing) within a controller and generate flow tables accordingly in forwarding devices. Flows can be dynamically routed in a per-flow basis with end-to-end QoS over the paths by means of the controller. Further, it allows to utilize the network resources in a more efficient way compared to today's architectures.

QoS-greedy multimedia applications such as video conferencing, distance learning, and interactive gaming are becoming prevalent in recent years. Efficient delivery of streaming media over the Internet presents many challenges. Flows of multimedia streaming require steady network resources with little or no packet drop and delay variation depending on the application. For example, while VoIP data is delay sensitive HTTP data requires reliable transmission. This indicates that different types of media may have different quality impairments under the same network condition. Therefore, designing multimedia flows routing frameworks that can cope with varying network conditions becomes important. Classification and prioritization of flows are the key points at designing such frameworks. QoS routing of video streaming over OpenFlow networks is studied in Civanlar et al. (2010). The authors introduce a formula based on linear programming aiming at reducing packet loss and keeping delay tolerable for Scalable Video Coding (SVC) base layer video flows while calculating routes for QoS flows. The idea is to keep the best-effort traffic on typical shortest paths and maintain a best-effort traffic table while video flows are routed on QoS-rich paths calculated by the proposed formula and maintaining QoS flows table for them. HiQoS application (Jinyao et al., 2015) exploits an SDN-based ECMP (Equal Cost Multipath Routing) algorithm, presented in Zhang et al. (2014), to find multiple paths between source and destination along with using queuing mechanisms to provide bandwidth guarantee for different classes of traffic. It differentiates different types of traffic and provide different bandwidth guarantees to different services through queuing mechanisms on the SDN switches. The multi path routing component finds multiple paths meeting certain QoS constraints between the source node and the destination node, and calculates the optimal path for each flow by real time monitoring of the network state.

An OpenFlow controller (OpenQoS) design for video streaming with QoS support is presented in Egilmez et al. (2012). The key concept in this architecture is the classification of the incoming flows as multimedia flows and data flows using packet header fields. These flows are dynamically routed on the QoS-supported paths while data flows are subject to best-effort routing. Another controller architecture and protocol (VSDN) for supporting QoS for video applications over SDN networks is presented in Owens and Durresi (2013). It allows video applications to request end-to-end guaranteed services (GS) from the network. They achieve guaranteed services by modifying limited switch capabilities provided by OpenFlow. The queue properties structure of OpenFlow, *ofp_ queue_ properties*, has been modified to support GS based queuing as *ofp_ queue_ prop_ gs_ rate* to contain

Table 4
Impact of the techniques on the SDN planes. Each study targets a plane in the SDN architecture to implement the idea presented in the technique. Most of the studies are conducted in control plane since it provides the control functions of the SDN paradigm. It is important to notice that the techniques do not solely rely on a specific plane of SDN architecture to implement their ideas due to cooperation among planes.

	SDN Planes			
Techniques	**Application Plane**	**Control Plane**	**Data Plane**	**Management Plane**
Wallner and Cannistra (2013)		✓		
Civanlar et al. (2010)		✓		
HiQoS - Jinyao et al. (2015)	✓			
OpenQoS - Egilmez et al. (2012)		✓		
VSDN - Owens and Durresi (2013)		✓		
RVSDN - Owens et al. (2014)		✓		
Tomovic et al. (2014)		✓		
Egilmez et al. (2011)		✓		
Egilmez et al. (2013)		✓		
ARVS - Yu et al. (2015)		✓		
Yilmaz et al. (2015)	✓			
Egilmez et al. (2012)		✓		
Egilmez and Tekalp (2014)		✓		
Karakus and Durresi (2015)		✓		
FlowBroker - Marconett and Yoo (2015, 2015)		✓	✓	
Wang et al. (2015, 2014)	✓	✓	✓	
Miao et al. (2015)		✓	✓	
CXP - Kotronis et al. (2016)		✓		
Kim et al. (2010)		✓		
NCL - Bueno et al. (2013)		✓		
Duan (2014); Duan et al. ()	✓	✓		
FlowQoS - Seddiki et al. (2015, 2014)	✓	✓		
Afaq et al. (2015, 2015)	✓	✓		
QoSFlow - Ishimori et al. (2013)			✓	
OpenQFlow - Nam-Seok et al. (2013)		✓	✓	✓
Xu et al. (2015)		✓	✓	
J. Wang et al. (2015); W. Wang et al. (2015)	✓		✓	✓
Caba and Soler (2015)		✓	✓	
Huong-Truong et al. (2013)	✓			
Kumar et al. (2013)	✓	✓		
Yiakoumis et al. (2012)	✓	✓		
Kassler et al. (2012)	✓	✓		
Q-POINT - Dobrijevic et al. (2014)	✓	✓		
QFF - Georgopoulos et al. (2013)		✓		
Gorlatch et al. (2014); Gorlatch and Humernbrum (2015)	✓			
Jarschel et al. (2013)	✓	✓		
Ayadi et al. (2013)		✓		
Q-Ctrl - Govindarajan et al. (2014)		✓		
PolicyCop - Bari et al. (2013)		✓		✓
OpenCache - Broadbent et al. (2015); Broadbent and Race (2012)		✓		
Sonkoly et al. (2012)			✓	
SoIP - Hu et al. (2015)		✓		
ACDPA - Desai and Nagegowda (2015)		✓	✓	
OpenNetMon - van Adrichem et al. (2014)	✓	✓		
PayLess - Chowdhury et al. (2014)	✓	✓		
Heleno Isolani et al. (2014)	✓	✓		
Jose et al. (2011)	✓	✓		
OpenSketch - Yu et al. (2013)	✓	✓		
OpenTM - Tootoonchian et al. (2010)	✓	✓		
OpenSAFE - Ballard et al. (2010)	✓	✓		

required fields for token bucket based traffic shaping. VSDN switch creates a token bucket shaping queue for each requested flow. The queuing process using GS regulates traffic per flow based on traffic specification provided by VSDN controller. The study in Owens et al. (2014) is an extension of the VSDN architecture to address reliable QoS support for video streaming by adding "reliability" constraint to the problem of path calculation for a requested QoS path. Classification of flows is exploited for different routing treatment in networks. Tomovic et al. (2014) also propose an SDN controller architecture that performs route calculations and resource reservations based on flow specifications for priority flows in an automated manner. It uses an algorithm that avoids highly utilized links even if traffic passing over them is best-effort.

Finding a route that provides best QoS for flows is not an easy task. Also, calculating such a route is not enough since network resources can dynamically change anytime. Therefore, a certain path may not be a good route for a flow all the time. To this end, frameworks taking into account these network changes are needed to keep flows under QoS guaranteed routes and provide optimized QoS. QoS routing should optimize a different cost function than simply the path length. For example, routes that have larger capacity even with longer distances may be more preferable to shorter routes that may cause packet loss. In Egilmez et al. (2011, 2013), the authors propose an optimization framework for video streaming with dynamic rerouting capability on the OpenFlow controller. To this end, they introduce two optimization problems along with their formulations. In the first problem, only lossless QoS flows (the base layer of the SVC encoded video) are routed under congestion conditions with an aim of no packet loss. In the second problem, both lossless QoS flows and lossy QoS flows (enhancement layers of the SVC encoded video) are routed with goals of no

Table 5
Organization and short descriptions of the studies surveyed in SDN/OpenFlow networks. These categories are the most prominent ways in which QoS can benefit from the concept of SDN.

	Description	
Organization	**Techniques**	**Description**
Multimedia flows routing mechanisms - (Section 5)	Civanlar et al. (2010)	A QoS-enabled routing architecture for scalable video streaming
	Jinyao et al. (2015)	Design of HiQoS application for multi path routing and queueing mechanisms
	Egilmez et al. (2012)	A controller design, OpenQoS", for QoS-enabled routing of multimedia traffic delivery
	Owens and Durresi (2013); Owens et al. (2014)	A QoS-enabled (reliable) routing architecture (R-VSDN) for video streaming
	Tomovic et al. (2014)	A QoS routing framework to provide resource-guaranteed paths for multimedia applications
	Egilmez et al. (2011, 2013); Yu et al. (2015)	A QoS-enabled dynamic optimization-based routing architecture for scalable video streaming
	Yilmaz et al. (2015)	Server load balancing application that reroutes flows of video streams
Inter-domain QoS routing mechanisms - (Section 6)	Egilmez et al. (2012); Egilmez and Tekalp (2014)	A distributed QoS routing architecture for scalable video streaming over multi-domain OpenFlow networks
	Karakus and Durresi (2015)	A hierarchic network architecture with an inter-AS QoS routing approach
	Marconett and Yoo (2015, 2015)	Design of Broker-based FlowBroker architecture for QoS support
	Wang et al. (2015, 2014)	Design of MCTEQ model proposing a joint bandwidth allocation for trafffic classes
	Miao et al. (2015)	Use of SDN and OPS nodes for QoS support
	Kotronis et al. (2016)	Design of Control Exchange Points (CXPs)" for QoS routing among ISPs
Resource reservation mechanisms - (Section 7)	Kim et al. (2010)	A network QoS control framework for management of converged network fabrics
	Bueno et al. (2013)	A Network Control Layer (NCL) based on SDN, OpenFlow, and NaaS for QoS requirements of applications
	Duan (2014); Duan et al. ()	A framework to apply NaaS in SDN/OpenFlow networks to enable network service orchestration for supporting inter-domain end-to-end QoS
	Seddiki et al. (2015, 2014)	A system, FlowQoS, enabling users to specify high-level application flow prioritization (e.g. VoIP etc.)
	Afaq et al. (2015, 2015)	A QoS provisioning mechanisms for elephant flows
Queue management and scheduling mechanisms - (Section 8)	Ishimori et al. (2013)	A QoS control framework (QoSFlow) using multiple packet schedulers
	Nam-Seok et al. (2013)	A QoS-motivated SDN architecture (OpenQFlow) for scalable and stateful SDN/OpenFlow networks
	Wallner and Cannistra (2013); Xu et al. (2015)	ToS/DSCP-based classification approach for QoS
	J. Wang et al. (2015); W. Wang et al. (2015)	A hierarchical autonomic QoS model by adopting SDN
	Caba and Soler (2015)	A QoS configuration API using OVSDB protocol
QoE-aware mechanisms - (Section 9)	Huong-Truong et al. (2013)	A QoE-Aware IPTV network architecture design over OpenFlow networks
	Kumar et al. (2013); Yiakoumis et al. (2012)	A system to improve user QoE by bandwidth allocation management framework at home networks
	Kassler et al. (2012); Dobrijevic et al. (2014)	Design of Q-POINT, a QoE-driven path optimization model
	Georgopoulos et al. (2013)	An OpenFlow-assisted QoE Fairness Framework (QFF) to maximize the QoE of clients in a shared network
	Gorlatch et al. (2014),Gorlatch and Humernbrum (2015)	A Northbound API design for online applications to increase QoE of users
	Jarschel et al. (2013)	A study investigating how different kinds of information such as per-flow parameters, application signatures etc. can improve network management
Network monitoring mechanisms - (Section 10)	van Adrichem et al. (2014)	Design and implementation of OpenNetMon monitoring framework
	Chowdhury et al. (2014)	Design and implementation of PayLess monitoring framework
	Heleno Isolani et al. (2014)	Design and implementation of an interactive network monitoring framework
	Jose et al. (2011)	Design and implementation of traffic measurement framework
	Yu et al. (2013)	Design and implementation of OpenSketch monitoring framework
	Tootoonchian et al. (2010)	Design and implementation of OpenTM monitoring framework
	Ballard et al. (2010)	Design and implementation of OpenSAFE monitoring framework
Other QoS-related mechanisms - (Section 11)	Ayadi et al. (2013)	A language to express QoS requirements of applications when placing virtual network components
	Govindarajan et al. (2014)	A QoS controller architecture, Q-Ctrl, for programmatically attaining requested QoS constraints by users in an SDN-based cloud infrastructure
	Bari et al. (2013)	Design of a QoS policy management framework called PolicyCop
	Broadbent et al. (2015),Broadbent and Race (2012)	A caching mechanism (OpenCache) to store content for VoD services
	Sonkoly et al. (2012)	An architectural extension for QoS-enabled experiments in Ofelia using OpenFlow
	Hu et al. (2015)	Design of SoIP architecture showing interoperability of SDN and IP for better QoS
	Desai and Nagegowda (2015)	Design of ACDPA architecture using SDN and Hadoop for better QoS support
	Middleton and Modafferi (2015)	Report of 2 years-running SDN network experiments on 3 different testbeds

packet loss and minimized loss, respectively. ARVS (Adaptive Routing Video Streaming) approach (Yu et al., 2015) also studies the same optimization problem for adaptive routing of video packets. In ARVS, if the shortest path does not satisfy the delay variation constraint, the base layer packets have the first priority to be rerouted to a calculated feasible path based on the available bandwidth of this path, and the enhancement layer packets will stay on the shortest path. However, if there is no available bandwidth in this path, the base layer packets will stay on the shortest path while the enhancement layer packets will be rerouted to this path.

Server load balancing can affect quality of video streaming for end users. Server load balance requires continuous monitoring of the load of each server and dynamically rerouting current or new service requests to available servers for lower delay and distortion in case of servers are overloaded. SDN can help mitigate this problem since it can provide global network view to users. For this problem, a load balancing application that reroutes flows of video streams is presented in Yilmaz et al. (2015). When the application detects server overloading, it calculates cost metrics (packet loss and delay) for each route connecting the user to each server. The old flows are deleted and new flows are pushed to all switches along the new least cost route.

Providing QoS-guaranteed paths for flows in networks is a challenging task for network operators. This objective requires taking many restrictions (e.g. bandwidth, delay etc.) into account before supplying such paths. Researchers anticipate that SDN and OpenFlow couple can help network administrators make flow-based routing easier compared to current state of it since it can provide centralized and finer-granular flow management along with global network view. Therefore, they propose various routing frameworks that exploit advantages of SDN and OpenFlow to make QoS provisioning easier for network paths.

6. Inter-domain QoS routing mechanisms

A single controller solution in the current OpenFlow specification is not scalable for large-scale multi-domain networks due to the limitation in processing power of the single controller, latency resulting from distant network devices, and huge amount of overhead because of messaging between controller and switches. Therefore, there is need for a distributed control plane with multiple controllers so that each controller is responsible for a part (domain) of the network. Routing end-to-end QoS flows between these networks requires collecting up-to-date global network state information, such as delay, bandwidth, and packet loss rate for each link. However, over a large-scale network, this is a difficult task because of problem dimension (size) and network operators' intent not to share internal precise network dynamics in detail. Therefore, distributed QoS routing models need to consider all these challenges to ensure optimal end-to-end QoS for applications.

A distributed control plane-based routing architecture for video streaming over OpenFlow networks is presented in Egilmez et al. (2012); Egilmez and Tekalp (2014). In this routing architecture, each domain controller aggregates internal network resource information for each border node pairs (called virtual links) and share with other domain controllers. In this way, each controller acquires a global view of whole network and becomes capable of calculating an end-to-end QoS optimized route for flows. Karakus and Durresi (2015) propose a similar QoS routing architecture but it utilizes a hierarchy-based network architecture in which network controllers compose hierarchy-levels along with another controller, called "Broker", on the top level. Each network controller shares its summarized network state information with the Broker instead of other controllers. The Broker keeps the global network state and view to share necessary information with certain controller when needed. FlowBroker Marconett and Yoo (2015, 2015) architecture also exploits Brokers for network performance enhancement and load balancing regarding flow coordination over multiple domains in SDN.

An important problem in inter-data center (IDC) traffic management is bandwidth allocation to competing applications while maximizing the overall network utilization and considering QoS metrics and fairness. MCTEQ (Wang et al., 2015, 2014) model proposes a joint bandwidth allocation to multiple traffic classes. It uses SDN concept to give preference to higher priority traffic in grabbing bandwidth by associating its utility with a larger weight while considering end-to-end delay requirement of interactive applications. Miao et al. (2015) exploit SDN paradigm's control plane to update the look-up-table (LUT) of OPS (Optical Packet Switching) nodes at data center networks by extending OpenFlow protocol. By this way, application flows are switched by the OPS at sub-*m*s hardware speed, decoupled from the slower (millisecond timescale) SDN control operation. Hence, with flows prioritization and faster speed, it is possible to guarantee QoS for flows for intra data center traffic.

Pathlets (i.e. partial paths) based models are also leveraged to provide inter-domain end-to-end QoS paths. In this model, pathlets with specific QoS properties from each autonomous domain are advertised to an independent external entity that manage them for an end-to-end route. Control Exchange Point (CXP) Kotronis et al. (2016) exploits abstracted network paths to orchestrate the end-to-end stitching of slices (a flow space associated with a specific service and a virtual topology (e.g. pathlets)) that the ISPs provide. The task of the CXP is to admit requests for QoS-guaranteed end-to-end paths, embed paths in the inter-domain virtual topology, and monitor the provided QoS guarantees.

7. Resource reservation mechanisms

This type of frameworks typically exploits flow classification and a rate-shaping through some modules implemented in controllers. A classifier module uses packet header fields to classify the packet and assign a priority to the corresponding flow based on network QoS policies. The rate-shapers then manage the flow rates to install corresponding rules in switches over the path in order to reserve resources for flows needing QoS.

The rate-limiters and priority queues can also be used with high level service requirements for resource reservation to provide QoS. The architecture in Kim et al. (2010) exploits extensions to the OpenFlow's QoS capabilities. The proposed QoS controller creates network slices[3] for different applications and feeds them with required performance requirements. The authors utilize a mechanism called "QoS APIs", an extension to OpenFlow, so as to control configuration and management of QoS parameters. The aggregated bandwidth usage is accomplished by the rate-limiter APIs and the queue mapping API is exploited to map flow(s) to priority queues in ports in order to cope with bandwidth and delay allocation.

Table 3 shows some features of studies such as queuing and/or scheduling mechanisms, scaling domain, simulation and/or emulation environment, and the controller(s) exploited in the development stages. Most of the studies target a single domain and do not modify (i.e. use available default queues) the queue mechanism(s) of associated data plane devices (e.g. switches) in their architectures. Moreover, albeit the most of the studies state that their frameworks work with any OpenFlow controller by little modification (if not necessary), the Floodlight controller has been also chosen due to its QoS support over other controllers, highly modular design, and rich set of APIs.

SDN and Network as a Service (NaaS) paradigms can be cooperated to address the problem of providing QoS parameters for application requirements while providing end-to-end service provisioning. NCL (Network Control Layer) (Bueno et al., 2013) framework supports the low-level network QoS provisioning for requirements of different types

[3] These network slices are set of services defined by certain QoS performance requirements such as max bandwidth, min delay, etc. for each network slice.

of data flows by means of resource reservation. While SDN brings the ability to flexibly manage and program the underlying network, the NaaS paradigm supply users secure and isolated access to the network. In addition, the NaaS paradigm provides ability to easily expand or shrink the network services. The proposed NCL architecture has two main parts: The *QoS SDN Application (SDNApp)* and the *Monitor Module*. The *SDNApp* accounts for adaptation of control plane to the providers' requirements and configures the data plane accordingly. while the *SDN Monitor* component is responsible for monitoring the network states and collecting statistics from switches by means of OpenFlow counters. Duan (2014) also present a NaaS-applied framework in SDN that enables network service orchestration for supporting inter-domain end-to-end QoS. A high-level abstraction model for network service capabilities is proposed and a technique for determining required bandwidth in network services to achieve QoS guarantee is developed. Network calculus is exploited in the proposed modeling and analysis which makes the developed techniques general and applicable to networking systems consisting of heterogeneous autonomous domains. In Duan et al., the authors extend the study presented in Duan (2014) to develop the idea of NaaS-SDN integration to propose a framework of a NaaS-based Service Delivery Platform (SDP) for a multi-domain SDN environment. This platform provides a high-level abstraction of each SDN domain as a network service and enables network service orchestration for end-to-end service delivery. They investigate two key technologies for achieving end-to-end QoS guarantee through this SDP, an abstract model for network service capabilities and a technique for end-to-end bandwidth allocation.

Making per-flow and application-based QoS allocation hassle-free is an important task in home networks using an SDN-based approach because home networking devices have less processing power than typical networking devices and the users are not skilled. FlowQoS (Seddiki et al., 2015, 2014) is a system in which users of the broadband access network simply specify the high-level applications that should have higher priority (e.g., adaptive video streaming, VoIP) compared to others. The FlowQoS controller performs the appropriate application identification and QoS configuration for both upstream and downstream traffic to implement a user's preferences. For each flow, FlowQoS performs on-the-fly application identification. It also installs rules in the data plane that forward individual flows according to user-specified priorities for those applications. The system creates links in a virtual topology in the home router, configures each of these links with a user-specified rate, and assigns flows to these links to provide rate shaping per application.

Long-lived flows are mostly called elephant flows and are large transfer such as backups. These elephant flows can affect the performance of the network since network resources are consumed by them and they fill buffers end-to-end. Other flows may be affected from this tendency because they also use the same buffers with elephants. Therefore, detecting elephant flows and satisfying their QoS needs is needed for a better network performance. In Afaq et al. (2015, 2015), a QoS provisioning mechanism is proposed for elephant flows after their detection. In the proposed approach, flows over a specified threshold value, called elephant flows, are subject to QoS module application that routes them to rate-limited queues (e.g. max or min bandwidth) for traffic shaping QoS technique. The QoS module application enables the network to define a queuing policy which exploits the *enqueue* action in OpenFlow to enqueue certain types of flows in the network.

8. Queue management and scheduling mechanisms

The order of some packets in a queue may have more priority than other packets which are ahead of them in the queue. This idea has impact on QoS along with the traffic shaping. Hence, the QoSFlow (Ishimori et al., 2013) model manipulates the multiple packet schedulers, i.e. not only FIFO, in Linux kernel in order to provide more flexible and manageable QoS control mechanisms in OpenFlow-enabled networks. The QoSFlow combines the Linux packet schedulers along with OpenFlow networks and supports the *Hierarchical Token Bucket (HTB)*, *Random Early Detection (RED)*, and *Stochastic Fairness Queuing (SFQ)* schedulers. The QoSFlow enriches the software switches of OpenFlow. The authors state that they use OpenFlow 1.0 because of its stability and ability to let users make use of different schedulers. The QoS module of QoSFlow has three components: *Traffic Shaping*, *Packet Schedulers*, and *Enqueueing*. The *Traffic Shaping* and *Packet Schedulers* are responsible together for manipulation of bandwidth size in queues. On the other hand, the *Enqueueing* component administrates the flow table messages of OpenFlow protocol and mapping flows to queues.[4]

“OpenQFlow” architecture (Nam-Seok et al., 2013) is a variant of OpenFlow architecture that provides microflow-based QoS in a scalable manner. It divides classic flow table framework to three tables: flow state table, forwarding rule table, and QoS rule table. The flow state table entries are used to maintain 128-byte micro-flow state information including forwarding, QoS, and statistics information. It is used to find the forwarding and QoS information base without rule table lookups. Therefore, this increases the scalability of OpenQFlow architecture. Each entry of forwarding rule table maintains a pointer to a forwarding information base that comprises of forwarding information such as forward and drop. Similarly, each QoS table entry has a pointer to a QoS information consisting of the traffic type, bandwidth, and priority information. OpenQFlow brings two packet scheduling schemes, BETA and CETA, that provide max-min fairness without the need of output queues per flow.

Queue-based classification techniques are used in Wallner and Cannistra (2013) to achieve the QoS support in Floodlight-controlled SDN networks. To this end, traffic shaping (rate limiting) and DiffServ DSCP (Differentiated Services Code Point) approaches are exploited for QoS support in Floodlight-based SDN networks. The authors describe different class of services along with rate limiting paths between switches. In their approach, the main player is the “SDN module” that is responsible for packet matching, classification, and flow operations like insertion, deletion etc. This QoS component tracks and stores service classes with their DSCP values. The QoS module allows the network to define two different main policies: Queue-based policy and ToS/DSCP-based policy. The Queue-based policy exploits enqueueing mechanisms for flows while the ToS/DSCP-based policy uses class of services with a name (e.g. Expedited Forwarding, Best Effort etc.) and a corresponding DSCP value. An IPv4 ToS-based QoS mechanism is also proposed in Xu et al. (2015). It classifies flows as QoS flows and best flows and then assign them queues based on their priorities.

Another software defined automatic QoS management model is introduced in J. Wang et al. (2015); W. Wang et al. (2015). The proposed model includes certain QoS functions such as packet marking, queue management, and queue scheduling. It utilizes Weighted Random Early Detection (WRED) queue management algorithm, Priority Queuing (PQ), and Weighted Round-Robin (WRR) queue scheduling algorithms. It also proposes a Collaborative Borrowing Based Packet-Marking (CBBPM) algorithm to improve the utilization rate of network resource.

OpenFlow alone is not enough to build more complex SDN services that require complete control and management of the data plane in terms of configurations of ports, queues, and so on. OVSDB protocol has been exploited to configure QoS capabilities of OVS switches in data plane in Caba and Soler (2015). The proposed QoS Config API allows applications to configure priority queues on the ports of data plane devices by adding OVSDB at the D-CPI of a network controller. Hence, services and applications built on top of an SDN controller using the proposed QoS API can make use of the full set of QoS features available in OVS devices.

[4] Maximum 8 queues per switch port

9. QoE-aware mechanisms

The requirements for network applications are diverse and today's networks try to support them based on QoS parameters. However, user satisfactions are not necessarily always met by just providing QoS for some applications like IPTV, real-time online interactive gaming, e-learning etc. since QoS is not powerful enough to express all features involved in a communication service (Fiedler et al., 2009). Therefore, the performance of a specific application cannot be determined by simply relying on QoS metrics. Instead, user QoE is an alternative measurement of user satisfactions for those applications over the network. Therefore, a major challenge for future networks is to dynamically adapt QoE demands of the users to QoS parameters in the network. However, mapping user QoE to network QoS parameters is a challenging issue over the networks. This is especially true for networks with limited resources like today's access networks. To this end, there are some researches aiming to maximize QoE of users while providing required QoS in SDN/OpenFlow networks.

Table 4 illustrates the corresponding SDN planes that the techniques impact. Each study targets a main plane in the SDN architecture to implement the idea presented in the studies. Most of the techniques are conducted in control plane since it provides the control functions of the SDN paradigm. We should note that it is important to notice that the techniques do not solely rely on a specific plane of SDN architecture to implement their ideas due to cooperation among planes.

IPTV is an emerging application recently in networking world. Controlling and implementing QoS policies on a network is an issue for IPTV services. The QoE-aware IPTV network architecture presented in Huong-Truong et al. (2013) combines IP Multimedia Subsystem (IMS) and OpenFlow-based network to optimize the network resources and service characteristics according to user satisfactions. In this design, users are able to rate the services that they are receiving and the proposed architecture maps and provisions the network QoS parameters accordingly. The architecture consists of three layers. The *Application Layer* includes the IMS IPTV Client and QoS engine to predict the user satisfaction. The *IMS Core Layer* is responsible for signaling and session/service control. Finally, the *Media Layer* is the data plane consisting of OpenFlow switches for transportation of traffic in the unicast, multicast or broadcast manners.

Enabling users to gain some controls over bandwidth allocation of access links for their devices and applications at home networks can be used to improve user QoE in such networks. The study in Kumar et al. (2013) leverages the SDN paradigm in ISP network to make such control delegation possible for users. The authors state that such a control by users not only improve user QoE but also allows ISP to monetize their services and powerfully compete with other ISPs in the market. They design a GUI that allows a typical user to specify their requirements on a per-device and per-application basis. The GUI then translates these requests into the appropriate API calls exposed by the SDN controller hosted in the ISP network. Finally, the ISP's SDN controller determines an appropriate resource allocation for the request, which it then configures into the switching hardware associated with that user's access link. Yiakoumis et al. (2012) also present a very similar idea that proposes allowing users to choose the relative priority of their applications, and indicate their preference to the ISP that then enforces the preference by an OpenFlow controller.

Optimized path assignment while improving the QoE level of user perception for multimedia services is studied in Kassler et al. (2012). The proposed system aims to enable negotiation of service and network communication parameters between users and to find a path for delivering flows for corresponding communication. The system leverages OpenFlow to set up the networking paths for users in order to maximize QoE while considering network resources such as link capacities, delay etc. and network topology. The two principle components of the proposed system are QMOF (QoS Matching and Optimization Function) and PAF (Path Assignment Function). QMOF resides in the SDN application layer and conducts an initial parameter matching process to produce feasible service configurations. PAF is located in the SDN control layer and executed on an OpenFlow controller. It optimizes the network paths to meet the resource requirements of a currently active service configuration. In Dobrijevic et al. (2014), the authors propose the “Q-POINT”, a QoE-driven path optimization model, built on Kassler et al. (2012) by formulating and solving the multi-user domain-wide QoE optimization problem. Their aim is to find a best path for each media flow while maximizing the aggregated user-expected QoE value over all users and service flows in an SDN network domain, subject to resource constraints and network topology. They present the problem as a mathematical model, which is formulated as a mixed integer linear program.

Dynamic adjustment of bit rate has been used to reduce pauses and buffering times in video playbacks in recent researches. This idea brings its own advantages for overall user experiences. However, that model has some issues such as unstable and bursty flows, network congestion owing to independent adoption strategy as well. Furthermore, user requests to maximize their satisfactions without knowledge of others on the network is another drawback of variable bit rate idea. The “OpenFlow-assisted QoE Fairness Framework (QFF)” (Georgopoulos et al., 2013) architecture aims at mitigating aforementioned problems. The QFF framework improves the QoE for all network and video streaming devices, thereby users, along with network resources and requirements. The QFF framework watches video streams in the network so that it can dynamically adapt the flow parameters to fairly increase the QoE for users. The QFF exploits the idea of sharing resources (particularly bandwidth) evenly among users because a user or device may have a very low bandwidth rate than another one although its resolution is much higher than the latter. This results in reduced QoE of users. An OpenFlow-enabled controller takes a place in the heart of the QFF framework to control its functionalities.

Real-Time Online Interactive Applications (ROIA) such as Real-Time Strategy (RTS) games (e.g. StarCraft) require highly dynamic QoS characteristics from a network. ROIA currently use the network on a best-effort basis, because of the lack of control over QoS in traditional networks. However, this results in a sub-optimal QoE by the end-user. Use of SDN technology to meet the dynamic network demands of ROIA, therefore improving QoE, is studied in Gorlatch et al. (2014); Gorlatch and Humernbrum (2015). The study propose a Northbound API consisting of two parts, *Base API* and *Application-level API*, in order to differentiate and map application-oriented QoS metrics to network-oriented ones. The *Base API* is a bridge between SDN controller and SDN modules. It receives applications' high-level QoS metrics and translates them to low-level network QoS metrics so that the controller can provide. The *Application-level API* is responsible for applications' high level QoS metrics and prevents developer from low-level details.

Application information, such as per-flow parameters, and application signatures, and related QoS levels offer greater flexibility in terms of supporting QoE than hard QoS parameters. However, using them may require an overhead of signaling effort compared to management at the network level. The study in Jarschel et al. (2013) investigates how different kinds of information or application quality parameters can support a more effective network management in an SDN-enabled network. The authors examine the trade-off between the QoE improvement due to more detailed application information and corresponding signaling overhead in an SDN-enabled testbed for the application of YouTube streaming.

10. Network monitoring mechanisms

One of the benefits that SDN promises is efficient use of network resources and ease of resource provisioning. SDN renders these features possible by decoupling of data plane and control plane. This separation simplifies the management of the network. Network operators maintain a global view of a network from a central control mechanism (i.e. controller). They can dynamically optimize flow management and resources. Moreover, per-flow, and/or application-level QoS provisioning becomes easier and feasible for network administrators. However, making all these features possible requires well-designed network monitoring frameworks. Network monitoring is employed for many different applications such as QoS management, resource utilization, anomaly detection, traffic engineering and so on. It helps collect data from network components like switches, routers (through southbound APIs such as OpenFlow), and controllers (through west/eastbound APIs from other controllers). Monitoring frameworks should be able to gather, process and deliver monitored data at requested aggregation levels (such as per flow, port, table etc.) and frequency without introducing too much monitoring overhead into the network. In addition, they should pay attention to the accuracy and timeliness of measurements.

"OpenNetMon" van Adrichem et al. (2014) is a network QoS metrics monitoring module written for POX controller. It is used to monitor per-flow QoS metrics by polling flow ingress and egress switches at an adaptive rate. It utilizes querying flow counters to obtain per-flow throughput. They subtract the increase of the packet counter at destination switch from the increase of the source switch packet counter in order to calculate per-flow packet loss. The idea to calculate the path delay is to inject probe packets traveling the same path (i.e. links, nodes, buffers etc.). However, as a disadvantage, injecting such probes can bring extra message overhead to the controller.

"PayLess" Chowdhury et al. (2014) is a network statistics gathering framework. The PayLess framework works as a moderator between network applications and controller. It translates the high-level monitoring requirements of network applications for controllers and prevents applications from low-level details of statistics collection and storage management. The authors of PayLess also propose an adaptive monitoring algorithm which takes into consideration polling frequency to reduce the monitoring message overhead as well as accuracy of monitored statistics by only monitoring important switches.

An interactive approach to SDN monitoring, visualization, and configuration is studied in Heleno Isolani et al. (2014). The proposed monitoring manager retrieves information about the network and stores it in a local database through a module called "Infrastructure Synchronizer". This module gathers control and data information such as traffic statistics and network topology information and stores a history of these changes along with SDN-related configurations performed by the network administrator.

A traffic measurement framework for online large traffic aggregates based on an OpenFlow approach is introduced in Jose et al. (2011). The proposed model works on commodity OpenFlow switches and can be used for various measurement tasks. The hierarchical heavy hitters (HHH) traffic problem is exploited to understand the trade-off between accuracy and overhead in the proposed framework.

OpenSketch Yu et al. (2013) is a measurement architecture that provides a three stage packet processing pipelines (hashing, filtering, and counting) in SDN. It helps operators by making understanding the complex switch implementations and parameter tuning easier in diverse sketches. It proposes a measurement library configuring the pipelines for different sketches and allocating switch memory across tasks to maximize accuracy. OpenTM Tootoonchian et al. (2010) concentrates on measuring traffic matrix estimation by periodically polling one switch on each flow's path and then combining the measurements. In OpenTM, after a switch has been chosen it is constantly queried for gathering flow statistics. Polling a single switch does not impose significant load on the network but may affect accuracy if the switch is not carefully chosen. A disadvantage of OpenTM is that it is limited to generating traffic matrices for offline use and does not capture packet loss and delay. OpenSAFE Ballard et al. (2010) uses OpenFlow to enable flexible monitoring of network traffic for security problems. It directs spanned network traffic towards predefined sinks (e.g., IDS) according to pre-specified policies. While such an approach could be used to compute network utilization (by analyzing the redirected traffic), the overhead it creates by copying all network traffic is prohibitive. OpenSAFE requires hardware investments to perform the actual monitoring that network operators are reluctant to do.

11. Other QoS-related mechanisms

QoS in SDN/OpenFlow networks is not bounded just by routing, queue management, and QoE-aware mechanisms. Studies have been conducted in many broad areas of networking by taking advantage of SDN concept. Virtualization-based QoS providing, QoS policy management, content delivery mechanisms, and testbed QoS extension are some of the other ongoing studies in the SDN/OpenFlow networks.

Virtualization-based QoS provisioning—In recent years, many research efforts focus on effectively virtualization of computation, storage, server and network resources that are provided as a service over a network. Although server and storage virtualization show a great success regarding efficiency and performance, the network resource virtualization do not achieve the same success due to restricted access of network operators to control plane of network devices. Therefore, SDN brings capabilities that pave the way for virtualizing network resources in an on demand manner by abstraction of the underlying network infrastructure to the applications. Ayadi et al. (2013) exploit the network virtualization and SDN paradigm to meet applications' QoS requirements. The proposed VNOS (Virtual Network Operating System) plane is the fundamental layer for the virtualization of the network. In terms of the SDN, they use a distributed approach which manages the flows for QoS requirements in each network. The "Network as a Service" framework is used on top of the SDN controllers for management of virtual network flows. NaaS framework brings the management of aggregated flows and creation of a logical virtual network. To manage aggregation of flows, a mechanism called "solver 1" is leveraged to categorize the flows regarding their QoS criteria such as availability, delay, capacity, and reliability. These criteria are associated with a degree of high, medium, and low for flows and then each flow is classified as a pre-defined class of service (CoS) for aggregation. After classification and aggregation of the flows, a logical virtual network is created by interaction of management, control, and data planes. Q-Ctrl Govindarajan et al. (2014), QoS Controller, is an architecture for programmatically attaining requested QoS constraints by users in a SDN-based cloud infrastructure. The Q-Ctrl system is able to execute in a virtual overlay network via Open vSwitch (OVS), physical network infrastructure equipped with an SDN controller, or a simulated SDN environment via Mininet. It regulates and allocates the bandwidth for the virtual machines running on the Cloud infrastructure.

QoS policy management—In general, service level agreements (SLAs) are used to establish QoS parameters for traffic management of QoS-greedy applications such as online interactive gaming, video streaming, and video conferencing. Each SLA consists of a set of Service Level Objectives (SLOs) that are used to derive network level policies, which are in turn translated into device level primitives (e.g. forwarding rules, queue configurations, packet dropping rate, and traffic shaping policies). In traditional network architectures like DiffServ and MPLS, managing these QoS related policies are difficult due to static traffic classes with a coarse granularity of QoS levels and installation requirement of specialized software or hardware components in the network. On the other hand, SDN promises a rich northbound API possibility and global network view, it enables network operators to implement wide-range of network policies and rapid service deployment. PolicyCop Bari et al. (2013) project aims at bringing a flexible, easy-to-control, and vendor-independent management of QoS policies by means of SDN Northbound APIs in SDN/OpenFlow networks. PolicyCop fairly benefits the features of OpenFlow to make itself a good management framework. It provides per-flow control and on-demand aggregation thanks to OpenFlow. Traffic definition is easier by PolicyCop, compared to DiffServ and MPLS, due to no need of shutting down the network devices. It promises reduced operational overhead and is easy to deploy in a network because of its vendor-independent feature. PolicyCop architecture has 3 planes: data plane, control plane, and management plane. The data plane and control plane are classic SDN planes. The management plane is the heart of the PolicyCop framework. It is divided into two parts as well: *Policy Validator* and *Policy Enforcer*. The *Policy Validator* tracks and detects the policy violations while the *Policy Enforcer* component takes charge in case of any policy violations and maintains network policy rules.

Content delivery mechanisms—The ability to shape and control data traffic is one of the primary advantages of SDN. Being able to direct and automate data traffic makes it easier to implement QoS for certain applications such as Video-on-Demand (VoD). The increase in the use of VoD services brings a huge demand on servers of content networks. However, this demand load floods network resources such as bandwidth, latency etc. in response to user requests. OpenCache Broadbent et al. (2015); Broadbent and Race (2012) mitigates the duplication of traffic in cases if a user from network A gets a content from another network B and another user from network A requests the same content from network B. Therefore, the content needs to traverse the operator's network again. By OpenCache, the VoD content is cached within the network (i.e. network A) to avoid this duplication. Therefore, it reduces not only congestion and inefficiency but also increases throughput and response time to user requests. It still keeps the unicast delivery fashion so that existing infrastructure can be maintained. To this end, OpenCache exploits SDN's data plane and control plane separation philosophy in order to redirect user requests for the same content to a local cache. In the OpenCache framework, there is another controller called “Cache Controller” that is intermediary (i.e. connected) between SDN controller and cache instances. The cache controller allows connections of redirected requests. It also maintains a full global state of underlying network so that it can modify and manage the cache instances.

Testbed QoS extension—Vendor-dependent implementations of composite device structures regarding hardware and software requires more attention to QoS support of testbeds. Hence, QoS support in OpenFlow-based testbeds like OFELIA[5] will contribute and encourage to SDN/OpenFlow QoS research from both academia and industry. Therefore, Sonkoly et al. (2012) extend the Ofelia's architecture to support more QoS features for OpenFlow experiments in a more flexible, user friendly, and easy-to-manage way by a comprehensive study of different QoS settings and use-cases. The authors study the QoS features of diverse devices used in the Ofelia project for a comprehensive QoS performance analysis. Also, they extend the OpenFlow switches by defining vendor specific queue properties to selected queue types. Middleton and Modafferi (2015) present their experience over 2 years running SDN network experiments on three classes of testbed facilities: commercial Amazon EC2,[6] pre-commercial federated testbed of FIWARE Lab[7] instances, and experimental OFELIA. They focus on measuring how testbed features limit the ability to perform an idealized experiment, and how effectively that experiment can be executed using the testbed support apparatus provided. We compare and give results for three testbeds regarding some qualitative metrics such as QoS Monitoring, external IP addresses, network slice isolation reliability and so on.

[5] http://www.fp7-ofelia.eu/
[6] http://aws.amazon.com/ec2
[7] https://www.fiware.org/lab/

SDN over IP for QoS—Although IETF has proposed a series of architectures QoS in IP networks, none of them has been successful to be a unified adoption due to their deficiencies such as complex structures or lack of fine-grained control over flows. It is becoming clear that a future architectures such as SDN can be a solution for aforementioned problems. However, use of such a new architectures over existing networks requires some long-term and fundamental changes such as equipment, training of network operators etc. Therefore, interoperation of legacy networks and SDN is being researched to overcome such changes in short-term. SoIP (SDN over IP) Hu et al. (2015) approach promises providing better QoS guarantee for end users and applications using SDN over IP concept. The basic idea of SoIP is to update or reconstruct the network edge and build SDN-based overlay networks to take advantage of its per-flow control over flows while the network core maintains the existing differentiated services based on the ToS field of IP protocol header. This approach not only preserves the existing infrastructure and network devices but also enhance resource utilization and QoS guarantee.

SDN and Hadoop for QoS—Advanced Control Distributed Processing Architecture (ACDPA) Desai and Nagegowda (2015) takes advantage of both SDN and Hadoop[8] software framework to provide better QoS for flows. It uses SDN for network abstraction and control and Hadoop for processing large amount of data coming from data plane. In ACDPA, Wireshark[9] packet sniffer is used to capture packets from the network. Hadoop is then used to process the captured packets regarding classification and the results are given to the controller. The SDN controller gives corresponding priorities to the flows and propagate associated flow rules to switches to provide QoS.

Table 5 illustrates the organization, based on the categories identified, of the studies surveyed in the paper along with their short descriptions.

12. Discussion

12.1. Research challenges

While SDN matures, QoS provisioning in SDN/OpenFlow networks deserves more research efforts from both academia and industry. In this subsection, we explain few main issues that need further attentions to complete QoS abilities of SDN/OpenFlow environments.

- *I* nter-AS QoS Provisioning: Most of the current research studies have been focused on providing QoS in intra-domain. While single-domain problem is important, supporting QoS for flows at inter-domain level is arguably more crucial and difficult owing to two obvious reasons among others: Firstly, majority of the traffic in the Internet is between hosts which are part of different autonomous networks (i.e. inter-AS traffic). Secondly, network administrators eschew sharing their internal network-related configurations since they are proprietary. SDX (Software Defined Internet Exchange) project (Gupta et al., 2014; Feamster et al., 2013) tries to realize the use of SDN for inter-domain routing in IXPs (Internet Exchange Points) for more expressive, flexible and destination-independent forwarding. It aims at curing the deficiencies of today's *de facto* inter-AS routing protocol BGP (Rekhter and Li, 1995) by utilizing SDN features. SDX can enable some applications that are difficult and complex (if not impossible) in today's routing infrastructure: domain-based or application specific peering, enforceable inter-domain routing policies, remote traffic control, preventing free-riding, time-based routing, wide-area server load balancing etc. SDX faces some challenges as well such as developing proper isolation mechanisms for AS route selection processes, backward compat-

[8] http://hadoop.apache.org/
[9] https://www.wireshark.org/

ibility, and resolution of potential policy conflicts among ASes. A similar study (Kotronis et al., 2016) demonstrates an architectural model, "Control Exchange Point (CXP)", which dynamically stitch partial paths (called *pathlets*) provided by ISPs and provisions end-to-end QoS for services. CXPs leverage SDN principles such as the clean decoupling of the routing control plane from the data plane and the consequent centralization of control. The task of the CXP is to admit requests for QoS-guaranteed end-to-end paths, embed paths in the inter-domain virtual topology and monitor the provided QoS guarantees.

Another approach used in SDN case to mitigate the inter-domain routing is to utilize a more powerful controller(s), mostly called "Broker", with a full global network view over different ASes. This controller(s) is connected to domain controllers of other ASes. FlowBroker Marconett and Yoo (2015, 2015); Yoo (2014) architecture proposes use of brokers connected with network controllers. These brokers collect network state updates from each associated domains and forms corresponding local and global link state tables. When an inter-as flow requests comes to a broker, the broker calculates a path satisfying network metrics (e.g. packet loss ratio, delay etc.) of the request and sends this path information to controllers over the path. The main concern with this broker approach is that network operators consider their internal configurations proprietary and are not willing to share them with a third party control mechanisms.

The eXtensible Session Protocol (XSP) (Kissel et al., 2012) supports application-driven configuration of network resources across domains. XSP provides mechanisms that enable the configuration of dynamic networks services in support of applications such as GridFTP. The XSP libraries and APIs consolidate applications with a standard interface to define parameters determining network paths. The realization of these paths is then managed by the XSP Daemon (XSPd) that signals the underlying provisioning service while providing feedback to the application.

- *Q*oS Signaling Overhead: SDN is a logically centralized architecture. This structure results in gathering all QoS-related signaling messages (i.e. overhead) at control mechanism of the network (i.e. controller) by means of statistics messages from data plane elements to controller(s). OpenFlow enables network operators to collect statistics at different level of flows such as per-flow or aggregation of flows. However, each of these collection approaches comes at a cost. While per-flow approach brings finer granularity regarding QoS-related states, it suffers from the scalability issue. On the other hand, aggregation of statistics mitigates the scalability problem yet restrains the OpenFlow fine-granular flow independence semantics. Also, in an SDN/OpenFlow environment, a controller can poll a switch to collect statistics on the active flows. Alternatively, it can request a switch to push flow statistics (upon flow timeout) at a specific frequency (i.e. periodically). Moreover, one important issue is how often the QoS information should be sent from network elements to controller. Even though pulling statistics frequently from data plane help controller maintain up-to-date global vision of network states, it brings extra overhead to be handled by the controller owing to processing information. Therefore, this is process is a trade-off between measurement accuracy, timeliness and signaling overhead and thereby resulting in control plane scalability issue for controller (Moshref et al., 2013). The PayLess (Chowdhury et al., 2014) framework provides different flow aggregation levels by a RESTful API for flow statistics collection. It uses an adaptive statistics collection algorithm that delivers highly accurate information in real-time without incurring significant network overhead. The algorithm can achieve an accuracy close to constant periodic polling method while having up to 50% reduced messaging overhead compared to periodic poling strategy.

12.2. Lessons learned

This survey experience has showed us several important points that require more attention from researchers to provide QoS in SDN networks.

QoS support for applications and service provisioning have been difficult tasks to achieve for quite a while even though newer applications such as video conferencing, VoIP etc. demand performance guarantees. Despite a large volume of work, QoS has not been completely deployed in today's networks. A primary reason for this is the complexity of proposed QoS solutions and largely manual per-device configuration of QoS knobs by network administrators. Supporting QoS for services and applications requires a well-defined automated QoS control and network management mechanisms in order to maintain the requested QoS performance over a network. A QoS control mechanism should provide an automated but fine-grained control for flow configurations. Also, it should be adaptive to dynamic workloads for dynamic QoS configurations based on network states. Furthermore, it should support legacy devices and large-scale networks like WANs. In addition, it should provide network-wide optimization in resource allocation by utilizing a global view of the network.

In recent years, some emerging applications, such as distance learning, video conferencing and so on, are becoming prevalent in networking world. Despite the advantages of these QoS-dependent applications for users, they still suffer from some issues regarding QoS or QoE requests of their users/customers. Firstly, today's QoS based applications take into account only the network parameters as a QoS performance. However this approach does not reflect the user's real satisfaction of provided services. Secondly, even if the user's satisfaction, i.e. QoE, is provided, converting this QoE indicators to network-based QoS parameters is another issue. Also, this conversion needs to be in a dynamic and optimized way. Thirdly, controlling and implementing QoS policies on the network is another issue for IPTV services.

An A-CPI enables applications to communicate with the controller to express their needs including dynamically specifying the QoS parameters of applications. Since they provide crucial tasks between applications and controller, network operators should consider certain points while designing A-CPIs. An A-CPI should be able to tolerate slow modifications of networks such as resource allocation for applications. It should also allow for determining the requirements beforehand using the application if possible. Defining different kinds of network parameters for different data types should be possible by an A-CPI. An interface should make sense for application developers while providing application metrics. A desired interface should not involve any application-related metrics such as response time. Instead, it should be able to convert these application-oriented metrics to network-based metrics such as delay, bandwidth etc.

13. Conclusions

Providing QoS is still a hot research problem in existing networking architectures. The emerging applications in the Internet (e.g. video streaming, VoIP etc.) generate diverse flows which require different treatments for each one. However, providing QoS needs of these flows is not easy with today's networking models. Therefore, researchers has started exploiting the SDN paradigm and OpenFlow protocol since they bring centralized global network view, and more fine-granular flow management opportunities in networks. These features of SDN make it a better candidate in order to provide QoS for applications in easier and more flexible ways compared to traditional network architectures. In this survey paper, we have made a picture of QoS in OpenFlow-enabled SDN networks by surveying the current QoS-motivated studies in the field. We have organized the related studies according to the categories that are the most prominent ways in which QoS can benefit from the concept of SDN: Multimedia flows routing mechanisms, inter-domain routing mechanisms, resource reservation mechanisms, queue management and scheduling mechanisms, Quality of Experience (QoE)-aware mechanisms, network monitoring mechanisms, and other QoS-centric mechanisms. We have also outlined the potential challenges and open problems that need to be addressed further for better and complete QoS abilities in SDN/OpenFlow networks and lessons we had learned during preparation of this survey paper.

Acknowledgements

This work was partially supported by National Science Foundation under Grant No. 1547411.

References

Afaq, M., Rehman, S.U., Song, W.-C., 2015. A framework for classification and visualization of elephant flows in sdn-based networks. Procedia Comput. Sci. 65 (2015), 672–681. http://dx.doi.org/10.1016/j.procs.2015.09.011, URL ⟨http://www.sciencedirect.com/science/article/pii/S1877050915028410⟩.
Afaq, M., Rehman, S., Song, W.-C., 2015. Visualization of elephant flows and qos provisioning in sdn-based networks. In: 17th Asia-Pacific Network Operations and Management Symposium (APNOMS), pp. 444–447 http://dx.doi.org/10.1109/APNOMS.2015.7275384.
Ayadi, I., Diaz, G., Simoni, N., 2013. Qos-based network virtualization to future networks: An approach based on network constraints. In: Proceedings of the Fourth International Conference on the Network of the Future (NOF), pp. 1–5 http://dx.doi.org/10.1109/NOF.2013.6724515.
Akhunzada, A., Gani, A., Anuar, N.B., Abdelaziz, A., Khan, M.K., Hayat, A., Khan, S.U., 2016. Secure and dependable software defined networks. J. Netw. Comput. Appl. 61, 199–221.
Bakshi, K., 2013. Considerations for software defined networking (sdn): Approaches and use cases. In: IEEE Aerospace Conference, pp. 1–9.http://dx.doi.org/10.1109/AERO.2013.6496914
Ballard, J.R., Rae, I., Akella, A., 2010. Extensible and scalable network monitoring using opensafe. In: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, INM/WREN'10, USENIX Association, Berkeley, CA, USA, pp. 8–8 URL ⟨http://dl.acm.org/citation.cfm?Id=1863133.1863141⟩.
Bari, M., Chowdhury, S., Ahmed, R., Boutaba, R., 2013. Policycop: An autonomic qos policy enforcement framework for software defined networks. In: Future Networks and Services (SDN4FNS), 2013 IEEE SDN for, pp. 1–7 http://dx.doi.org/10.1109/SDN4FNS.2013.6702548.
Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W., 1998. RFC 2475: An Architecture for Differentiated Service . URL ⟨www.ietf.org/rfc/rfc2475.txt⟩
Braden, R., Clark, D., Shenker, S., 1994. RFC 1633: Integrated Services in the Internet Architecture: an Overview .URL⟨www.ietf.org/rfc/rfc1633.txt⟩
Braun, W., Menth, M., 2014. Software-defined networking using openFlow: protocols. Appl. Archit. Des. Choices, Future Internet 6 (2), 302. http://dx.doi.org/10.3390/fi6020302, URL ⟨http://www.mdpi.com/1999-5903/6/2/302⟩.
Broadbent, M., Race, N., 2012. Opencache: exploring efficient and transparent content delivery mechanisms for video-on-demand. In: Proceedings of the 2012 ACM conference on CoNEXT student workshop, CoNEXT Student '12, pp. 15–16.
Broadbent, M., King, D., Baildon, S., Georgalas, N., Race, N., 2015. Opencache: A software-defined content caching platform. In: Proceedings of the 1st IEEE Conference on Network Softwarization (NetSoft), pp. 1–5 http://dx.doi.org/10.1109/NETSOFT.2015.7116129.
Bueno, I., Aznar, J., Escalona, E., Ferrer, J., Antoni, J., 2013. Garcia-Espin, An opennaas based sdn framework for dynamic qos control. In: IEEE SDN for Future Networks and Services (SDN4FNS), 2013 pp. 1–7 http://dx.doi.org/10.1109/SDN4FNS.2013.6702533.
Caba, C., Soler, J., 2015. Apis for qos configuration in software defined networks. In: Proceedings of the 1st IEEE Conference on Network Softwarization (NetSoft), pp. 1–5 http://dx.doi.org/10.1109/NETSOFT.2015.7116157.
Chowdhury, S., Bari, M., Ahmed, R., Boutaba, R., 2014. Payless: A low cost network monitoring framework for software defined networks. In: IEEE Network Operations and Management Symposium (NOMS), pp. 1–9 http://dx.doi.org/10.1109/NOMS.2014.6838227.
Civanlar, S., Parlakisik, M., Tekalp, A., Gorkemli, B., Kaytaz, B., Onem, E., 2010. A qos-enabled openflow environment for scalable video streaming. In: IEEE GLOBECOM Workshops (GC Wkshps), pp. 351–356 http://dx.doi.org/10.1109/GLOCOMW.2010.5700340.
Desai, A., Nagegowda, K., 2015. Advanced control distributed processing architecture (acdpa) using sdn and hadoop for identifying the flow characteristics and setting the quality of service(qos) in the network. In: IEEE International Advance Computing Conference (IACC), pp. 784–788http://dx.doi.org/10.1109/IADCC.2015.7154814.
Dobrijevic, O., Kassler, A.J., Skorin-Kapov, L., Matijasevic, M., 2014. Q-point: Qoe-driven path optimization model for multimedia services. In: Wired/Wireless Internet Communications. Springer International Publishing, pp. 134–147.
Duan, Q., Wang, C., Li, X., End-to-end service delivery with qos guarantee in software defined networks, arXiv preprint arXiv:1504.04076 URL ⟨http://adsabs.harvard.edu/abs/2015arXiv150404076D⟩
Duan, Q., 2014. Network-as-a-service in software-defined networks for end-to-end qos provisioning. In: Wireless and Optical Communication Conference (WOCC), 2014 23rd, pp. 1–5 http://dx.doi.org/10.1109/WOCC.2014.6839919.
Egilmez, H., Tekalp, A., 2014. Distributed qos architectures for multimedia streaming over software defined networks. IEEE Trans. Multimed. 16 (6), 1597–1609. http://dx.doi.org/10.1109/TMM.2014.2325791.
Egilmez, H., Gorkemli, B., Tekalp, A., Civanlar, S., 2011. Scalable video streaming over openflow networks: an optimization framework for qos routing. In: Proceedings of the 18th IEEE International Conference on Image Processing (ICIP), pp. 2241–2244 http://dx.doi.org/10.1109/ICIP.2011.6116083.
Egilmez, H., Civanlar, S., Tekalp, A., 2012. A distributed qos routing architecture for scalable video streaming over multi-domain openflow networks. In: Proceedings of the 19th IEEE International Conference on Image Processing (ICIP), pp. 2237–2240 http://dx.doi.org/10.1109/ICIP.2012.6467340.
Egilmez, H., Dane, S., Bagci, K., Tekalp, A., 2012. 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), Asia-Pacific, pp. 1–8.
Egilmez, H., Civanlar, S., Tekalp, A., 2013. An optimization framework for qos-enabled adaptive video streaming over openflow networks. IEEE Trans. Multimed. 15 (3), 710–715. http://dx.doi.org/10.1109/TMM.2012.2232645.
Feamster, N., Rexford, J., Shenker, S., Clark, R., Hutchins, R., Levin, D., Bailey, J., 2013. SDX: a software defined internet exchange. Open Netw. Summit, 1.
Fernandez, M., 2013. Comparing OpenFlow Controller Paradigms Scalability: Reactive and Proactive. In: IEEE Proceedings of the 27th International Conference on Advanced Information Networking and Applications (AINA), pp. 1009–1016.http://dx.doi.org/10.1109/AINA.2013.113.
Fiedler, M., Kilkki, K., Reichl, P., 09192 executive summary – from quality of service to quality of experience. In: Fiedler, M., Kilkki, K., Reichl, P. (Eds.), From Quality of Service to Quality of Experience, no. 09192 in Dagstuhl Seminar Proceedings, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, Dagstuhl, Germany URL ⟨http://drops.dagstuhl.de/opus/volltexte/2009/2235⟩.
Floodlight Project.URL ⟨http://www.projectfloodlight.org/floodlight/⟩
Georgopoulos, P., Elkhatib, Y., Broadbent, M., Mu, M., Race, N., 2013. Towards network-wide qoe fairness using openflow-assisted adaptive video streaming. In: Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking, FhMN'13, pp. 15–20.
Gorlatch, S., Humernbrum, T., 2015. Enabling high-level qos metrics for interactive online applications using sdn. In: 2015 International Conference on Computing, Networking and Communications (ICNC), pp. 707–711 http://dx.doi.org/10.1109/ICCNC.2015.7069432.
Gorlatch, S., Humernbrum, T., Glinka, F.,2014. Improving qos in real-time internet applications: from best-effort to software-defined networks. In: International Conference on Computing, Networking and Communications (ICNC), pp. 189–193 http://dx.doi.org/10.1109/ICCNC.2014.6785329.
Govindarajan, K., Meng, K.C., Ong, H., Tat, W.M., Sivanand, S., Leong, L.S., 2014. Realizing the quality of service (qos) in software-defined networking (sdn) based cloud infrastructure. In: Proceedings of the 2nd International Conference on Information and Communication Technology (ICoICT), pp. 505–510 http://dx.doi.org/10.1109/ICoICT.2014.6914113.
Gupta, A., Vanbever, L., Shahbaz, M., Donovan, S.P., Schlinker, B., Feamster, N., Rexford, J., Shenker, S., Clark, R., Katz-Bassett, E., 2014. SDX: a Software Defined Internet Exchange. In: Proceedings of the 2014 ACM Conference on SIGCOMM, SIGCOMM '14, pp. 551–562.
Heleno Isolani, P., Araujo Wickboldt, J., Both, C., Rochol, J., Zambenedetti Granville, L., 2014. Interactive monitoring, visualization, and configuration of openflow-based sdn. In: IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 207–215 http://dx.doi.org/10.1109/INM.2015.7140294.
Hu, C., Wang, Q., Dai, X., 2015. Sdn over ip: enabling internet to provide better qos guarantee. In: Proceedings of the Ninth International Conference on Frontier of Computer Science and Technology (FCST), pp. 46–51 http://dx.doi.org/10.1109/FCST.2015.17.
Huong-Truong, T., Thanh, N.H., Hung, N.T., Mueller, J., Magedanz, T., 2013. Qoe-aware resource provisioning and adaptation in ims-based iptv using openflow. In: Proceedings of the 19th IEEE Workshop on Local Metropolitan Area Networks (LANMAN), pp. 1–3 http://dx.doi.org/10.1109/LANMAN.2013.6528284.
Ishimori, A., Farias, F., Cerqueira, E., Abelem, A., 2013. Control of multiple packet schedulers for improving qos on openflow/sdn networking. In: Proceedings of the Second European Workshop on Software Defined Networks (EWSDN), pp. 81–86 http://dx.doi.org/10.1109/EWSDN.2013.20.
Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., Zolla, J., Hölzle, U., Stuart, S., Vahdat, A., 2013. B4: Experience with a globally-deployed software defined wan. In: Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, SIGCOMM '13, ACM, New York, NY, USA, pp. 3–14 http://dx.doi.org/10.1145/2486001.2486019URL ⟨http://doi.acm.org/10.1145/2486001.2486019⟩.
Jarschel, M., Wamser, F., Hohn, T., Zinner, T., Tran-Gia, P., 2013. Sdn-based application-aware networking on the example of youtube video streaming. In: Proceedings of the Second European Workshop on Software Defined Networks (EWSDN), pp. 87–92 http://dx.doi.org/10.1109/EWSDN.2013.21.
Jinyao, Y., Hailong, Z., Qianjun, S., Bo, L., Xiao, G., 2015. Hiqos: an sdn-based multipath qos solution. China Commun. 12 (5), 123–133. http://dx.doi.org/10.1109/CC.2015.7112035.
Jose, L., Yu, M., Rexford, J., 2011. Online measurement of large traffic aggregates on commodity switches. In: Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, Hot-ICE'11, pp. 13–13.

Karakus, M., Durresi, A., 2015. A scalable inter-as qos routing architecture in software defined network (sdn). In: IEEE Proceedings of the 29th International Conference on Advanced Information Networking and Applications (AINA), pp. 148–154 http://dx.doi.org/10.1109/AINA.2015.179.

Kassler, A., Skorin-Kapov, L., Dobrijevic, O., Matijasevic, M., Dely, P., 2012. Towards qoe-driven multimedia service negotiation and path optimization with software defined networking. In: Proceedings of the 20th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pp. 1–5.

Kim, W., Sharma, P., Lee, J., Banerjee. S. , Tourrilhes, J., Lee, S.-J., Yalagandula, P., 2010. Automated and scalable qos control for network convergence. In: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, INM/WREN'10, pp. 1–1.

Kissel, E., Fernandes, G., Jaffee, M., Swany, M., Zhang, M., 2012. Driving software defined networks with xsp. In: Proceedings of the International Conference on Communications (ICC), pp. 6616–6621 http://dx.doi.org/10.1109/ICC.2012.6364805.

Kotronis, V., Klöti, R., Rost, M., Georgopoulos, P., Ager, B., Schmid, S., Dimitropoulos, X., 2016. Stitching inter-domain paths over ixps. In: Proceedings of the Symposium on SDN Research, SOSR '16, ACM, New York, NY, USA, pp. 17:1–17:12 http://dx.doi.org/10.1145/2890955.2890960 URL ⟨http://doi.acm.org/10.1145/2890955.2890960⟩.

Kumar, H., Gharakheili, H., Sivaraman, V., 2013. User control of quality of experience in home networks using sdn. In: IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS), pp. 1–6 http://dx.doi.org/10.1109/ANTS.2013.6802847.

Li, W., Meng, W., Kwok, L.F., 2016. A survey on openflow-based software defined networks: security challenges and countermeasures. J. Netw. Comput. Appl. 68, 126–139. http://dx.doi.org/10.1016/j.jnca.2016.04.011, URL ⟨http://www.sciencedirect.com/science/article/pii/S1084804516300613⟩.

Lin, P., Bi, J., Wolff, S., Wang, Y., Xu, A., Chen, Z., Hu, H., Lin, Y., 2015. A west-east bridge based sdn inter-domain testbed. IEEE Commun. Mag. 53 (2), 190–197. http://dx.doi.org/10.1109/MCOM.2015.7045408.

Liu, Y., Li, Y., Wang, Y., Yuan, J., 2015. Optimal scheduling for multi-flow update in Software-Defined Networks. J. Netw. Comput. Appl. 54 (C (August 2015)), 11–19, org/10.1016/j.jnca.2015.04.009.

Marconett, D., Yoo, S.J.B., 2015. Flowbroker: market-driven multi-domain sdn with heterogeneous brokers. In: Optical Fiber Communications Conference and Exhibition (OFC), pp. 1–3 http://dx.doi.org/10.1364/OFC.2015.Th2A.36.

Marconett, D., Yoo, S.J., 2015. Flowbroker: a software-defined network controller architecture for multi-domain brokering and reputation. J. Netw. Syst. Manag. 23 (2), 328–359. http://dx.doi.org/10.1007/s10922-014-9325-5, URL ⟨http://dx.doi.org/10.1007/s10922-014-9325-5⟩).

Masoudi, R., Ghaffari, A., 2016. Software defined networks: a survey. J. Netw. Comput. Appl. 67, 1–25. http://dx.doi.org/10.1016/j.jnca.2016.03.016, (URL ⟨http://www.sciencedirect.com/science/article/pii/S1084804516300297⟩).

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J., 2008. Openflow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. 38 (2), 69–74.

Miao, W., Agraz, F., Peng, S., Spadaro, S., Bernini, G., Perello, J., Zervas, G., Nejabati, R., Ciulli, N., Simeonidou, D., Dorren, H., Calabretta, N., 2015. Sdn-enabled ops with qos guarantee for reconfigurable virtual data center networks. IEEE/OSA J. Opt. Commun. Netw. 7 (7), 634–643. http://dx.doi.org/10.1364/JOCN.7.000634.

Middleton, S., Modafferi, S., 2015. Experiences monitoring and managing qos using sdn on testbeds supporting different innovation stages. In: 2015 Proceedings of the 1st IEEE Conference on Network Softwarization (NetSoft), pp. 1–5 http://dx.doi.org/10.1109/NETSOFT.2015.7116136.

Mininet. URL ⟨http://mininet.org/⟩

Moshref, M., Yu, M., Govindan, R., Resource/accuracy tradeoffs in software-defined measurement. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13, pp. 73–78.

Nam-Seok, K., Hwanjo, H., Jong-Dae, P., Hong-Shik, P., 2013. Openqflow: Scalable openflow with flow-based qos. IEICE Trans. Commun. 96 (2), 479–488.

ONOS Project.URL⟨http://onosproject.org/⟩

OpenDaylight Project.URL⟨https://www.opendaylight.org/⟩

OpenFlow Management and Configuration Protocol 1.2 (OF-Config 1.2), 2014. Technical Report, Open Networking Foundation (ONF).URL ⟨https://www.opennetworking.org/sdn-resources/onf-specifications/openflow-config⟩

OpenFlow Switch Specification, 2009. 1.0.0 (December).URL ⟨http://archive.openflow.org/documents/openflow-spec-v1.0.0.pdf⟩

OpenFlow Switch Specification, 2011. 1.1.0 (February).URL ⟨http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf⟩

OpenFlow Switch Specification, 2011. 1.2.0 (December).URL ⟨https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.2.pdf⟩

OpenFlow Switch Specification, 2012. 1.3.0 (June).URL⟨https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf⟩

OpenFlow Switch Specification, 2013. 1.4.0 (October).URL⟨https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf⟩

OpenFlow Switch Specification, 2014.1.5.0 (December).URL ⟨https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf⟩

Owens, H., Durresi, A., 2013. Video over software-defined networking (vsdn). In: Proceedings of the 16th International Conference on Network-Based Information Systems (NBiS), pp. 44–51 http://dx.doi.org/10.1109/NBiS.2013.10 URL ⟨http://www.computer.org/csdl/proceedings/nbis/2013/2510/00/2510a044.pdf⟩.

Owens, H., Durresi, A., Jain, R., 2014. Reliable video over software-defined networking (rvsdn). In: 2014 IEEE Global Communications Conference, pp. 1974–1979 http://dx.doi.org/10.1109/GLOCOM.2014.7037097.

Palma, D., Gonçalves, J.a., Sousa, B., Cordeiro, L., Simoes, P., Sharma, S., Staessens, D., 2014. The queuepusher: enabling queue management in openflow. In: 2014 Proceedings of the IEEE Third European Workshop on Software Defined Networks, EWSDN '14, Computer Society, Washington, DC, USA, pp. 125–126.http://dx.doi.org/10.1109/EWSDN.2014.34 URL⟨http://dx.doi.org/10.1109/EWSDN.2014.34⟩

Pfaff, B., Davie, B., 2013. RFC 7047: the Open vSwitch Database Management Protocol (2013). URL⟨www.ietf.org/rfc/rfc7047.txt⟩

Rekhter, Y., Li, T., 1995. A border gateway protocol 4 (bgp-4).

Rosen, E., Viswanathan, A., Callon, R., 2001. RFC 3031: multiprotocol Label Switching Architecture.URL⟨www.ietf.org/rfc/rfc3031.txt⟩

Roy, A.R., Bari, M.F., Zhani, M.F., Ahmed, R., Boutaba, R., 2014. Dot: distributed openflow testbed. In: Proceedings of the 2014 ACM Conference on SIGCOMM, SIGCOMM'14, ACM, New York, NY, USA, pp. 367–368 http://dx.doi.org/10.1145/2619239.2631457 URL ⟨http://doi.acm.org/10.1145/2619239.2631457⟩.

SDN architecture, 2014. Tech. rep., Open Networking Foundation (ONF) (June 2014). URL ⟨https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf⟩

Seddiki, M.S., Shahbaz, M., Donovan, S., Grover, S., Park, M., Feamster, N., Song, Y.-Q., 2014. Flowqos: Qos for the rest of us. In: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, HotSDN'14, ACM, New York, NY, USA, pp. 207–208 http://dx.doi.org/10.1145/2620728.2620766URL ⟨http://doi.acm.org/10.1145/2620728.2620766⟩.

Seddiki, M.S., Shahbaz, M., Donovan, S., Grover, S., Park, M., Feamster, N., Song, Y.-Q., 2015. Flowqos: per-flow quality of service for broadband access networks. Georgia Institute of Technology.

Sezer, S., Scott-Hayward, S., Chouhan, P., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M., Rao, N., 2013. Are we ready for sdn? Implementation challenges for software-defined networks. IEEE Commun. Mag. 51 (7), 36–43. http://dx.doi.org/10.1109/MCOM.2013.6553676.

Software-Defined Networking: the New Norm for Networks, 2012. Technical Report, Open Networking Foundation (ONF) (April).

Sonkoly, B., Gulyas, A., Nemeth, F., Czentye, J., Kurucz, K., Novak, B., Vaszkun, G., 2012. On qos support to ofelia and openflow. In: European Workshop on Software Defined Networking (EWSDN), pp. 109–113 http://dx.doi.org/10.1109/EWSDN.2012.26.

Tomovic, S., Prasad, N., Radusinovic, I., 2014. Sdn control framework for qos provisioning. In: 22nd Telecommunications Forum Telfor (TELFOR), pp. 111–114 http://dx.doi.org/10.1109/TELFOR.2014.7034369.

Tomovic, S., Radusinovic, I., Prasad, N., 2015. Performance comparison of qos routing algorithms applicable to large-scale sdn networks. In: IEEE EUROCON 2015 – International Conference on Computer as a Tool (EUROCON), pp. 1–6 http://dx.doi.org/10.1109/EUROCON.2015.7313698.

Tootoonchian, A., Ghobadi, M., Ganjali, Y., 2010. Opentm: traffic matrix estimator for openflow networks. In: Proceedings of the 11th International Conference on Passive and Active Measurement, PAM'10, pp. 201–210.

van Adrichem, N., Doerr, C., Kuipers, F., 2014. Opennetmon: network monitoring in openflow software-defined networks. In: Proceedings of the IEEE Network Operations and Management Symposium (NOMS), pp. 1–8 http://dx.doi.org/10.1109/NOMS.2014.6838228.

Vaughan-Nichols, S.J., 2011. Openflow: the next generation of the network? IEEE Computer, 44, 8, 13–15.URL(URL ⟨http://dblp.uni-trier.de/db/journals/computer/computer44.html#Vaughan-Nichols11⟩

Wallner, R., Cannistra, R., 2013. An sdn approach: Quality of service using big switchs floodlight open-source controller. In: Proceedings of the Asia-Pacific Advanced Network, vol. 35, pp. 14–19 http://dx.doi.org/10.7125/APAN.35.2.

Wang, J., Wang, Y., Dai, X., Bensaou, B., 2014. Sdn-based multi-class qos-guaranteed inter-data center traffic management. In: Proceedings of the IEEE 3rd International Conference on Cloud Networking (CloudNet), pp. 401–406 http://dx.doi.org/10.1109/CloudNet.2014.6969028.

Yang, S.-N., Ho, S.-W., Lin, Y.-B., Gan, C.-H., 2016. A multi-RAT bandwidth aggregation mechanism with software-defined networking. 61 (C (February 2016)), 189–198, http://dx.doi.org/10.1016/j.jnca.2015.11.003.

Wang, J., Wang, Y., Dai, X., Benasou, B., 2015. Sdn-based multi-class qos guarantee in inter-data center communications. IEEE Trans. PP Cloud Comput. 99, 1. http://dx.doi.org/10.1109/TCC.2015.2491930.

Wang, W., Tian, Y., Gong, X., Qi, Q., Hu, Y., 2015. Software defined autonomic qos model for future internet. J. Syst. Softw. 110, 122–135. http://dx.doi.org/10.1016/j.jss.2015.08.016, (URL ⟨http://www.sciencedirect.com/science/article/pii/S0164121215001776⟩.

Xu, C., Chen, B., Qian, H., 2015. Quality of service guaranteed resource management dynamically in software defined network. In: Journal of Communications, Vol. 10, pp. 843–850 http://dx.doi.org/10.12720/jcm.10.11.843-850.

Yiakoumis, Y., Katti, S., Huang, T.-Y., McKeown, N., Yap, K.-K., Johari, R., 2012. Putting home users in charge of their network. In: Proceedings of the 2012 ACM Conference on Ubiquitous Computing, UbiComp'12, pp. 1114–1119.

Yilmaz, S., Tekalp, A., Unluturk, B., 2015. Video streaming over software defined networks with server load balancing. In: Proceedings of the 2015 International Conference on Computing, Networking and Communications (ICNC), pp. 722–726. http://dx.doi.org/10.1109/ICCNC.2015.7069435.

Yoo, S., 2014. Multi-domain cognitive optical software defined networks with market-driven brokers. In: Proceedings of the 2014 European Conference on Optical Communication (ECOC), pp. 1–3. http://dx.doi.org/10.1109/ECOC.2014.6964144.

Yu, M., Jose, L., Miao, R., 2013. Software defined traffic measurement with opensketch.

In: Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13), Lombard, IL, pp. 29–42.

Yu, T.-F., Wang, K., Hsu, Y.-H., 2015. Adaptive routing for video streaming with qos support over sdn networks. In: 2015 International Conference on Information Networking (ICOIN), pp. 318–323 http://dx.doi.org/10.1109/ICOIN.2015.7057904.

Zhang, L., Berson, S., Herzog, S., Jamin, S., 1997. RFC 2205: Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. URL ⟨www.ietf.org/rfc/rfc2205.txt⟩

Zhang, H., Guo, X., Yan, J., Liu, B., Shuai, Q., 2014. Sdn-based ecmp algorithm for data center networks. In: Proceedings of the Computing, Communications and IT Applications Conference (ComComAp), 2014 , pp. 13–18 http://dx.doi.org/10.1109/ComComAp.2014.7017162.

Murat Karakus received the BS Degree in Mathematics from Suleyman Demirel University, Turkey, in 2009, and the MS Degree in Computer Science and Information Systems from the University of Michigan-Flint, in 2013. He is currently working through his PhD in the Department of Computer and Information Science at Indiana University Purdue University - Indianapolis. He is the recipient of the Best Paper Award at ACM SIGITE 2011 conference. His current research interests include new network architectures (particularly Software-Defined Networking (SDN)), scalability, Quality of Service (QoS), routing, and introducing programming to non-CS majors.

Arjan Durresi is a Professor of Computer Science at Indiana University Purdue University in Indianapolis, Indiana. In the past, he held positions at LSU, and The Ohio State University. His research interest include network architectures, security and trust management. He has published over eighty papers in journals and over 200 papers in conference proceedings, and seven book chapters. He also has over thirty contributions to standardization organizations such as IETF, ATM Forum, ITU, ANSI and TIA. His research has been funded by NSF, the States of Ohio and Louisiana, as well as university and industry sources.