Paper

Traffic Engineering in Software Defined Networks: A Survey

Mohammad R. Abbasi[1], Ajay Guleria[2], and Mandalika S. Devi[1]

[1] *Department of Computer Science and Application, Panjab University, Chandigarh, India*
[2] *Computer Center, Panjab University, Chandigarh, India*

Abstract—An important technique to optimize a network and improve network robustness is traffic engineering. As traffic demand increases, traffic engineering can reduce service degradation and failure in the network. To allow a network to adapt to changes in the traffic pattern, the research community proposed several traffic engineering techniques for the traditional networking architecture. However, the traditional network architecture is difficult to manage. Software Defined Networking (SDN) is a new networking model, which decouples the control plane and data plane of the networking devices. It promises to simplify network management, introduces network programmability, and provides a global view of network state. To exploit the potential of SDN, new traffic engineering methods are required. This paper surveys the state of the art in traffic engineering techniques with an emphasis on traffic engineering for SDN. It focuses on some of the traffic engineering methods for the traditional network architecture and the lessons that can be learned from them for better traffic engineering methods for SDN-based networks. This paper also explores the research challenges and future directions for SDN traffic engineering solutions.

Keywords—application awareness, Software Defined Networking, traffic engineering.

1. Introduction

A major problem with underlying communication network is the dynamic nature of the network applications and their environment. This means that the performance requirements of the transferred data flows, like Quality of Service (QoS), can vary over time. The applications operate in a wide range of environments, i.e. wired and wireless with a variety of networking devices. For the applications to perform effectively, the underlying network should be flexible enough to dynamically change in response to any changes in the application requirements and their environment. The current approaches are either based on static or overprovisioned overlay networks, or require the applications to change in accordance with the network performance.

An important way to address this problem is through traffic engineering (TE). It is the process of analyzing the network state, predicting and balancing the transmitted data load over the network resources. It is a technique used to adapt the traffic routing to the changes in the network condition. The aim of traffic engineering is to improve network performance, QoS and user experience, by efficient use of resources, which can reduce operation cost too. The QoS techniques assign the available resources to the prioritized traffic to avoid congestion for this traffic. However, these techniques do not provide additional resources to the traffic that requires QoS. The traditional routing techniques do not provide any mechanism to allocate network resources in an optimal way.

To address this problem the research community started working on traffic engineering and proposed new ways to improve network robustness in response to the growth of traffic demands. Traffic engineering reduces the service degradation due to congestion and failure, e.g. link failure. Fault tolerance is an important property of any network. It is to ensure that if a failure exists in the network, still the requested data can be delivered to the destination.

Computer networks consist of numerous networking devices, such as switches, middle boxes (e.g. firewalls) and routers. Traditional network architecture is distributed, as shown in Fig. 1, where each networking device has both the control plane and the data plane. The control plane is the intelligent part of networking devices. It makes decision about forwarding and routing of data-flow. The data plane is the part of a networking device that carries user traffic. It executes the control plane's commands and forwards the data.

Network operators have to manually configure these multi-vendor devices to respond to a variety of applications and event in the network. Often they have to use limited tools such as command line interface (CLI) and sometimes scripting tools to convert these high-level configuration policies into low-level policies. This makes the management and optimization of a network difficult, which can introduce errors in the network. Other problems with this architecture can cause oscillations in the network, since control planes of the devices are distributed, innovation is difficult because the vendors prohibit modification of the underlying software in the devices.

To overcome these problems, the idea of network programmability was introduced, particularly with the introduction of Software Defined Networking (SDN) [1]. SDN allows a network to be programmed so that its behavior can

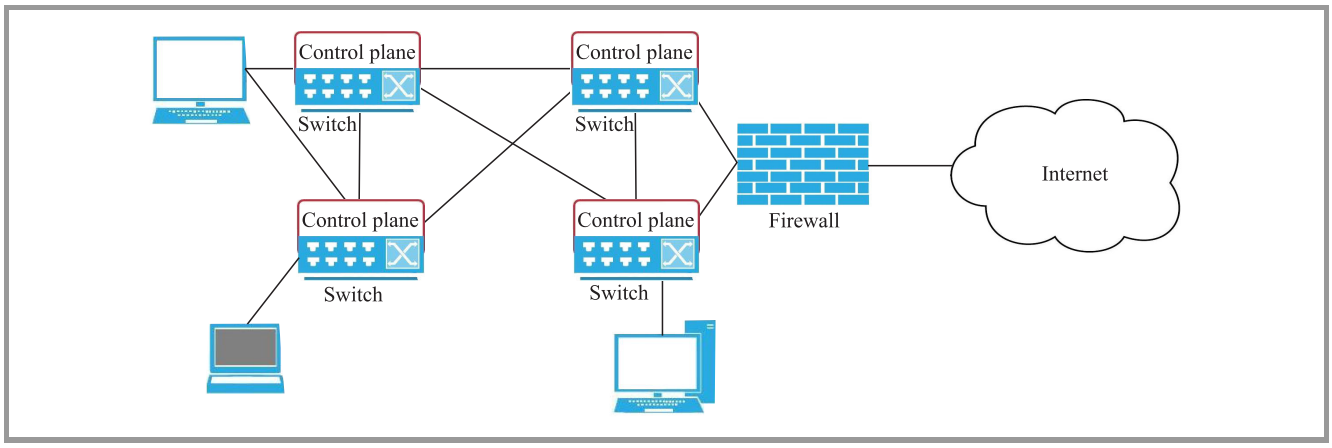

Fig. 1. Traditional network architecture.

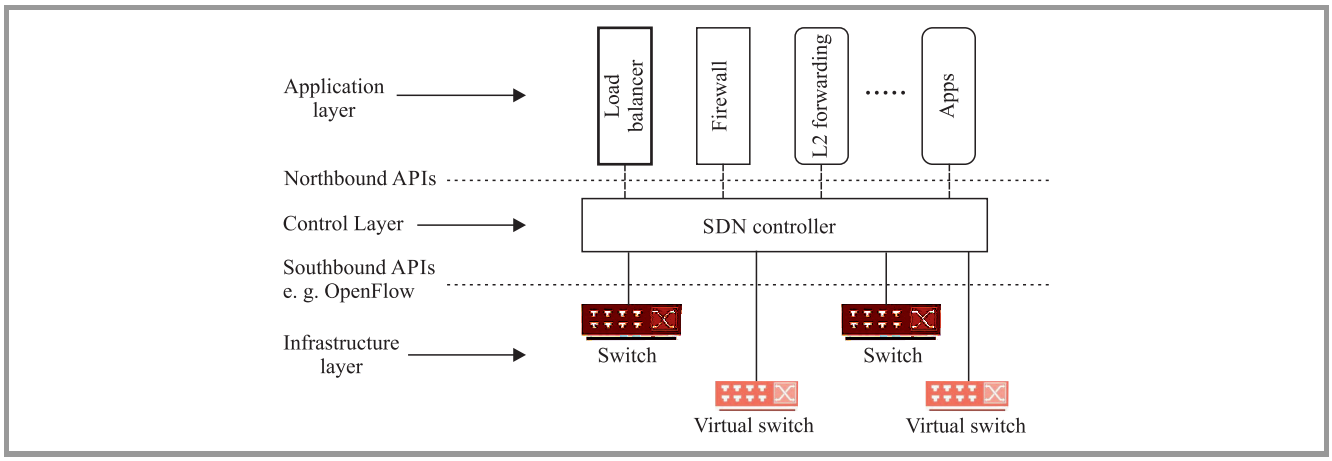

Fig. 2. An example of SDN architecture.

be changed actively on demand and in a fine-grained manner. It is a new networking model, where the control plane and the data plane are separated. The idea behind SDN is to simplify network management and enable innovation, i.e. to develop and deploy new network applications and services with ease, also to manage and optimize network performance through high-level policy enforcement.

To optimize these heterogeneous networks, both classic networks and SDN-based networks, a number of TE techniques have been introduced. Most are based on tweaking wide area TE and routing mechanism, such as Equal Cost Multi-Path routing (ECMP), Intermediate System to Intermediate System (IS-IS), and Multi protocol Label Switching (MPLS) [2], [3].

From traffic engineering point of view, even though these techniques perform well, they suffer from several limitations such as, they take routing decision locally, and it is difficult to change the link weights dynamically. In addition, while sending traffic these techniques consider few criteria, such as link capacity.

SDN separates the control plane and data plane of networking devices and introduces a well-defined interface, the OpenFlow protocol [4], between the two planes. The SDN architecture (Fig. 2) and the OpenFlow takes the intelligence, control functions, out of networking devices and place them in a centralized servers called controller, and provides centralized control over a network. The SDN/OpenFlow controller acts as an operating system for the network. It executes the control applications and services, such as routing protocols and L2 forwarding. This configuration abstracts the underlying network infrastructure. Therefore, it enables the applications and network services to treat the network as a logical entity.

One of the most widely used SDN enabler is the OpenFlow v.1.3 protocol. It allows the controller to manage the OpenFlow switches. The OpenFlow switches contain one or more flow tables, a group table, and a secure OpenFlow channel (Fig. 3). The flow tables and the group table are used for packet lookup and then to forward the packets. The OpenFlow channel is an abstraction layer. It establishes a secure link between each of the switches and the controller via the OpenFlow protocol. This channel abstracts the underlying switch hardware. As of OpenFlow version 1.5, a switch can have one or more OpenFlow channels that are connected to multiple controllers.

SDN is, generally, a flow-based control strategy. Through the OpenFlow a controller can define how the switches should treat the flows. In a SDN when a source node sends

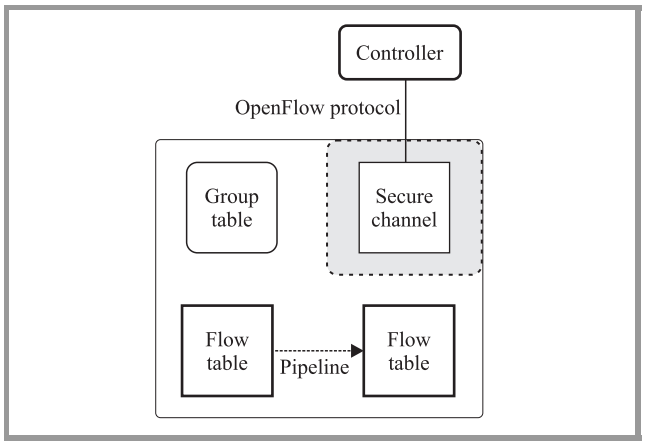

Fig. 3. Main OpenFlow switch components.

data to the destination, the switch sends the first packet to the controller, since it doesn't know how to treat this packet. The controller calculates the path for this packet and installs the appropriate rules in the switches on the packet's path. The new networking paradigm, SDN, has introduced new characteristics such as:

- separation of the control plane functionality, and the data plane functionality;
- centralized architecture allows the controller to have a central view of the deployed network. The controller has the global view of the network devices, servers, and virtual machines;
- network programmability, SDN provides an open standard, which allows external applications to program the network;
- facilitates innovation, new protocols and control applications can be introduced because OpenFlow provides the required abstractions, so we do not need to know the switch internals and configuration;
- flow management, through the OpenFlow a controller can define flows in different granularity, and how the switches should treat the flows.

The rest of the paper surveys some of the TE techniques, and it is organized as follows: Section 2 provides some of the TE mechanisms available for the classic network architecture and the assimilation from them. Section 3 describes an overview of SDN TE solutions. In Section 4, research challenges and future directions are discussed. Sections 5 and 6 conclude the paper.

2. Review of Classic Traffic Engineering Techniques

Classic traffic engineering techniques are based on tweaking wide area TE and routing mechanism such as ECMP or existing routing protocols such as IS-IS or MPLS [2], [3], [5], [6]. The Open Shortest Path First (OSPF) and IS-IS routing protocols do not adapt to the changes in the network condition because the link weights are static and these protocols lack any performance objectives while selecting the paths. The traffic engineering extensions to IS-IS and OSPF standard, extends these protocols by incorporating the traffic load while selecting a path. In these approaches during link state advertisements, routers advertise the traffic load along with link costs. After routers exchange link costs and traffic loads, then they calculate the shortest path for each destination. These standards require the routers to be modified to collect and exchange traffic statistics [5], [6].

Fortz *et al*. [7] propose a traffic engineering mechanism that monitors network wide view of the traffic pattern and network topology, then changes the link weights accordingly. This mechanism is based on the interior gateway protocols, like IS-IS. The authors says that classic inter-domain gateway protocols are effective traffic engineering tools in a network, and ensure robustness in terms of scalability and failure recovery. The introduced mechanism keeps the router and routing protocols unchanged. The mechanism is a centralized approach where it monitors the network topology and traffic, then optimizes the link costs to provide the best path possible to address the network goals. Routing protocols, like OSPF, select the path with minimum cost. If multiple paths with the same minimum cost are available then the traffic can be equally distributed among these paths. This is the concept behind ECMP. As depicted in Fig. 4, ECMP is a routing technique which balances the load over multiple paths by routing the packets to multiple-paths with equal cost. Various routing protocols such as OSPF and IS-IS explicitly support ECMP routing [8].

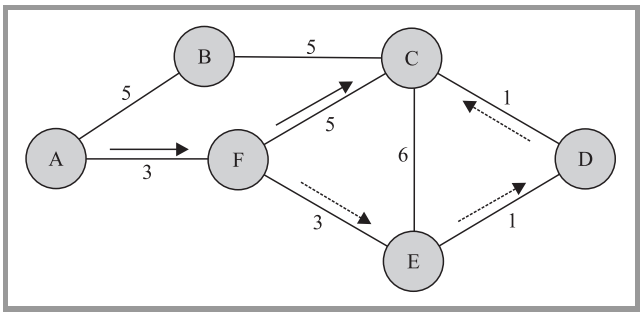

Fig. 4. An example of ECMP – there are two paths with equal cost to the destication node C, i.e. (A, F, C) and (A, F, E, D, C).

Multi-protocol Label Switching, MPLS, provides a tunneling mechanism. It creates end-to-end connections between the nodes. MPLS can integrate short path labels with IP routing mechanism, where the ingress routers assign short fixed labels to the packets, instead of long network addresses. The networking devices use this label to forward the packets to the destination through label-switched path (LSP). This reduces the routing table lookup overhead. The MPLS based traffic engineering, MPLS-TE, first reserve the resources for end-to-end path and then transfer the data. It establishes a labeled switched path over links

with sufficient bandwidth. This technique assures that enough resources are available for a flow. Since MPLS-TE works on available bandwidth in one aggregated class, it does not support QoS [9]. To provide QoS capability DiffServe-aware MPLS-TE techniques have been introduced, which combine both the Differentiated Services (DiffServ) and MPLS traffic engineering techniques to provide QoS [10]. Dongli *et al*. [9] analyze the QoS performance of DiffServe-aware MPLS traffic engineering techniques. The experimental results show that DiffServ-aware MPLS-TE can provide good QoS for traffics such as VoIP and other data, but due to the variable bit-rate property of the video data, these techniques cannot guarantee QoS for video data. As compared with conventional routing protocol MPLS is more flexible in selecting paths, since it sets up virtual circuit paths to send the traffic. The disadvantage of MPLS is that network operators need to manage the resource allocation to each path, and change the network configuration to adjust the path to the traffic condition. Because MPLS-TE transfers the aggregated traffics along allocated LSPs, it suffers from scalability and robustness [11]. In MPLS-TE it is necessary to use backup links so that if any link fails the traffic can be transferred through different paths.

An important way to balance the traffic over network resources is to disseminate the traffic over multiple paths. Gojmerac *et al*. [12] introduce an adaptive multi-path routing, which allows dynamic traffic engineering. Unlike other solutions, using global network information, the proposed technique focuses on local information in each node. This means the routers exchange information about links only to their immediate neighbors. So the nodes only have the information regarding their neighbors. During multi-path routing any neighboring node which is closer to the destination has a smaller cost than the current node. This neighboring node is considered as a viable candidate for the next hop. The advantage of taking routing decision based on local information is that it can reduce the signaling and memory overhead. The downside to the approach is, since the nodes do not have the global knowledge of the network state, it may not result in optimum routing of the traffic. Also due to the inherent limitation of the traditional network architecture it cannot adapt to the rapid changes in the traffic pattern and it can cause oscillation in the network.

Frank *et al*. in a [13] propose a content-aware traffic engineering technique for content distribution/delivery networks. The content providers duplicate the contents over distributed server infrastructures to provide better services to the users in different locations. The authors argue that it is essential for the content providers to know network topology and measure network state before mapping user request to the servers, which can introduce new challenges such as assigning users to the servers and performing traffic engineering. ISPs have the knowledge of the individual links status and network topology. This information can separate the server selection task from content delivery task, and help the content providers to focus on mapping the user to a server that provides better user experience. The introduced traffic engineering uses the information provided by ISPs along with the user's location to dynamically adapt to the traffic demand for the contents on the servers. This framework focuses on the traffic demand rather than routing, and uses the knowledge of the content providers (e.g. server status), and ISPs' knowledge (e.g. the network state and the user's location). For this reason this framework can complement the existing routing protocols and traffic engineering because it emphasizes on traffic demand rather than on traffic routing. Routing protocol such as OSPF and IS-IS are used to produce a routing matrix. With this matrix it tries to adjust a set of flow demands to reduce the maximum link utilization. The results of the experiments show that this framework has improved the user experience while reducing maximum link utilization and traffic delay.

Several energy-aware traffic engineering solutions have been proposed in [14]–[16]. These solutions incorporate traffic engineering to reduce the energy cost while trying to keep the network performance unaffected.

Vasic *et al*. [16] introduce an online traffic engineering technique. It spreads the load among multiple paths to reduce the energy consumption without affecting traffic rate. It presumes that energy-aware hardware is used in the network. These devices are capable to adjust its operating rate to its utilization, also they can sleep whenever it is possible to save energy. To enhance energy saving and keep the transfer rate steady, it transfers the data over multiple paths. The authors propose a number of techniques where they shift data to the links with low energy consumption, or they try to remove the traffic from as many possible links to allow the links and routers to sleep.

Most of the discussed approaches agree on the point that to engineer traffic in an efficient way a network-wide approach is required. When short-term changes happen in traffic volume the traffic engineering solution should quickly decide on how to route the traffic to different paths to balance link utilization. Under such circumstances where traffic pattern changes frequently, it is important for the traffic engineering solution to be stable. Otherwise, it can cause oscillation. Traffic oscillation can have a number of undesirable effects on the network, for example, switch-buffer overflow, out-of-order packets, poor allocation of network resources to the users, traffic delay and service degradation [17]. The solutions that have the above characteristics are difficult to implement in the traditional network architecture since we need to have access to global information in real-time, which is a tedious work in this paradigm. To find an optimal solution, most of the proposed solutions are based on local measurements, i.e. require the networking devices to decide independently on how to send the packets. In the traditional networks, generally, the link costs are kept static for a long period. Since the link cost is fixed, the traffic is transferred through the same path for a long period, until the link costs are changed.

For a traffic engineering technique to have an optimum effect on the network, it should have the following characteristics:

- it should utilize multi-path diversity in the network,
- it should make routing decisions based on the global view of the network,
- it should consider the flow values.

3. Review of Traffic Engineering Techniques in SDN

In SDN-based networks the controller can dynamically change the network state, for example, in traditional networks the link cost for routing protocols such as IS-IS are kept static for a long period. If congestion happens in the network it may lead to poor delivery of data till the link costs are changed or the problem is resolved. However, in SDN these values can be changed more dynamically to adapt to the changes. More innovative routing mechanism can be implemented, or the existing routing protocols can be modified, so that they can change dynamically as per network state to enhance resource utilization, avoid failure and congestion, and improve QoS. With the advances in SDN several traffic engineering techniques have been introduced by the research community. Table 1 summarizes some of the TE techniques in SDN.

To connect their Data Centers across the world and meet their performance requirements, Google introduced a Software Defined WAN architecture called B4 [18]. B4 is designed to resolve the problems in Wide Area Network (WAN) such as reliability, failures, and performance. It assigns bandwidth to the competing services, dynamically shifts traffic pattern, and overcomes network failure. B4 is designed to allow rapid deployment of new or standard protocols and control functions. One of such introduced functionalities is a traffic engineering mechanism, which allows applications to dynamically adapt in response to changes in the network behavior or failure. This architecture employs the routing and traffic engineering as separate services. The TE is layered on top of the routing protocols. This enables the network with a fallback strategy. If the TE service faced with a serious problem, it would be stopped so that the packets are forwarded using short path forwarding mechanism. This architecture consists of 3 logical layers:

- global layer, allows centralized control of the entire WAN through logically centralized applications such as the Central Traffic Engineering server (CTE) and SDN gateway (it allows centralized control of the network);
- site controller layer which includes the OpenFlow controller and network control applications such as routing services;
- switch hardware layer includes the switches, and performs traffic forwarding.

CTE is responsible for tasks such as measuring the unoccupied network bandwidth for multi-path forwarding, assigning and adjusting resource demands among the services, and actively relocating traffic from failed links and switches. SDN gateway provides the network topology graph for CTE. CTE uses this graph to compute the aggregated traffic at site to site edges. Then, an abstract of the computed result is fed to TE optimization algorithm to fairly allocate resources among the competing application groups/services. To achieve fairness it allocates resources using Min-Max fairness technique. Based on the applications' priority it allocates bandwidth to the applications. It uses hashed-based ECMP to balance the load among multiple links.

Hedera [19] is introduced to make an effective use of the bandwidth in a data center. Hedera detects the elephant-flows at the edge switches. The Hedera implementation uses periodic pulling, where it collects statistics every five seconds to detect large flows. At first switches send a new flow using its default flow matching rules on one of its equal-cost paths, until the flow size grows and meet the threshold. Then, the flow is marked as elephant-flow. The default threshold is 10% of network interface controller (NIC). At this point Hedera's central scheduler uses its global view of the network and calculates a better path for the flow and route the traffic. To effectively use the bandwidth the scheduler calculates the path in a way that it is non-conflicting, and it can accommodate the flow. This method can improve the bandwidth utilization, but because it uses periodic pulling, it can cause high resource utilization and overhead.

The main design goal of DevoFlow [20] is to improve network scalability and performance by keeping the flows in the data plane as much as possible without losing the centralized view of the network. This reduces the interaction between control plane and data plane. DevoFlow uses aggressive use of wildcards to reduce the controller and switches interactions. Therefore, switches take routing decision locally, while controller manages the overall control of the network and routes the significant flows, i.e. elephant-flows. It uses techniques such as packets sampling to collect switch statistic and detect the elephant-flows. The flows that have transferred a certain number of bytes is marked as large flow. The suggested threshold is 1–10 MB. In the beginning DevoFlow forwards the traffic using DevoFlow's multi-path wildcard rules. When an elephant flow is detected the controller will calculate the path that is least congested, and re-route the traffic to this path.

The flow detection mechanisms used in Hedera and Devoflow have high resource overhead. To overcome this problem Mahout [21] modifies the end-hosts to detect elephant-flows. It uses a shim layer in the operating system to mark the significant flows. The shim layer monitors the TCP socket buffer and marks the flows when in a given period the buffer exceeds the rate threshold. It

Table 1
Overview of traffic engineering techniques in SDN

Technique	Description	Routing	Comments
B4 [18]	• it uses a centralized TE, layered on top of the routing protocols, • to achieve fairness it allocates resources using Min-Max fairness technique.	• it uses hashed-based ECMP to balance the load among multiple links.	• if TE service can be stopped so that the packets are forwarded using short path forwarding mechanism.
Hedera [19]	• detects the elephant-flows at the edge switches, • if threshold is met, i.e. 10% of NIC bandwidth, the flow is marked as elephant flow, • uses periodic pulling, every 5 s.	• uses the global view of network and calculate the better paths, which are non-conflicting, for the elephant flows.	• it achieves 15.4 b/s throughput, • achieves better optimal bisection of bandwidth of network, in comparison to ECMP, • periodic pulling can cause high resource utilization in switches.
DevoFlow [20]	• detects the elephant-flows at the edge switches, • if threshold is met, i.e. 1–10 MB, it marks the flow as elephant-flow.	• it uses aggressive use of wild carded OpenFlow rules, and a static multi-path routing algorithm to forward the traffic.	• it can improve throughput up to 32% in CLOS network.
Mahout [21]	• detects the elephant-flows at end-host using a shim layer, the default threshold is 100 k, and then the flow is marked as elephant-flow, • it uses in-band signaling to inform the controller about the elephant-flows.	• it computes the best path for elephant-flow; otherwise it forwards other flows using ECMP, • it calculates the path that is least congested by pulling the elephant-flow statistics and link utilization from switches.	• it can detect elephant flow, if threshold is 100 k, in 1.53 ms, • it has 16% better bisection than ECMP.
MicroTE [22]	• detect the elephant flows at end-host, • it calculates the mean of traffic matrix between ToR-ToR, if the mean and traffic is between δ of each other, default is 20%, then it is predictable.	• uses short term predictability to route the traffic on multiple paths, • the remaining flows are routed by the EMCP scheme with heuristic threshold.	• if traffic is predictable it perform close to optimal performance otherwise it performs like ECMP.
MiceTrap [23]	• it addresses the mice-flows, • uses end-host elephant-flow detection to distinguish between mice-flows, and elephant-flows.	• it aggregates the mice-flows to improve scalability, • it route the mice-flows using a weighted multi.	• N/A.
Rethinking Flow Classification in SDN [26]	• it is a tag-based classification, • source-edge switch tags the packets based on the application classes.	• the tag is also an identifier for matching & forwarding the packets	• it is 3 ms faster than the OpenFlow field matching, • it requires introduction of new API's to the data plane.
Atlas [25]	• it classifies each application uniquely. • it uses C5.0 machine learning tool to classify the applications, • it requires user to install agents on their mobile devices to collect information to train ML trainer.	• it routes the flow based on applications, and network requirements.	• it has about 94% accuracy, • requires extension to OpenFlow.
MSDN-TE [32]	• it gathers network state information and considers the actual path's load to forward the flows on multiple paths.	• it dynamically selects the best shortest path among the available paths.	• it has better performance over other forwarding mechanisms such as Shortest Path First, • it reduced download time by 48%.

uses an in-band signaling mechanism to mark the flows as elephant-flows and inform the controller about the significant flows. Mahout uses ECMP to route normal traffic. When an elephant-flow is detected the controller calculates the best path for this flow. To calculate the best paths the controller pulls the elephant-flow statistics and link utilization from the switches to select the least congested path. This method can detect the elephant-flows faster, with lower

processing overhead than other method. But, it requires modification of the end-hosts.

In [22] Benson *et al*. present a traffic engineering mechanism for data center network called MicroTE, which uses an end-host elephant flow detection to detect the elephant flows. It exploits short-term prediction, and quickly adapts to the changes in traffic pattern. To efficiently handle the network load, it takes advantage of multiple paths in the network and coordinates traffic scheduling by using global view of traffic across the available network paths. The authors argue that the traffic nature of data center networks is bursty, and during long-run time the traffic is unpredictable, above 150 s, but it is predictable in short-time scale of 1–5 s. The TE methods for ISPs do not perform well in data center environments because they work on the granularity of hours, but TE for Data Centers should work on granularity of seconds.

Unlike MicroTE, MiceTrap [23] incorporates the end-host flow detection to handle mice-flows and uses OpenFlow group table (multi-path group type) to aggregate the incoming mice-flows for each destination. The authors believe that TE mechanism, which handles elephant-flows, can cause congestion to mice-flows, i.e. short-lived flows. Also the resources should be distributed according to flow values. Managing mice-flows using ECMP and giving preference to elephant flows can degrade QoS. MiceTrap architecture consists of end-host elephant flow detection module, multi-path aggregates implemented in OpenFlow switch, and a controller. It uses the kernel-level shim layer approach to mark the elephant flow detection. The shim layer method monitors the TCP socket buffer and marks the flows when in a given period the buffer exceeds the specified threshold. Multi-path Mice-flow Aggregator, aggregates the incoming mice-flows for each destination. This reduces the rules for traffic management because if each mice-flow is managed by an exact flow rule, it will cause a bottleneck and limit the scalability. The advantage of using group table is that it saves bandwidth since one single group message can update a set of flows when the traffic distribution is changed. It uses a weighted routing algorithm which forwards aggregated traffic into multiple paths by considering the current network load while calculating the paths.

These are effective solutions for data center networks, but they share the network resources based on the flow size and do not consider the flow-value. An important way to consider the flow-values is to classify the applications. Two promising techniques for application classification are Deep Packet Inspection (DPI) and Machine Learning (ML) classification method. In comparison to techniques such as port-based classification, these techniques have a high classification accuracy. DPI methods inspect the payload of packets and search for known patterns, keywords or regular expressions that are characteristic of a given application. These methods are more accurate, but with higher overhead (in terms of memory and processing). ML methods exploit several flow-level features to classify the traffic. To classify the flows these methods look for known flow behavior such as packet counts, data bytes, TCP flags, etc. [24].

In the work [25] Qazi *et al*. try to investigate how to integrate application awareness in SDN-based networks and how to classify traffics with high accuracy. A framework called Atlas is introduced, which is capable of classifying the traffic in the network and enforcing higher layer policies. The presented framework uses a ML tool called C5.0 to classify the flows based on the application types. It shows 94% accuracy. The Atlas framework can classify each specific application. It can classify each VoIP application uniquely rather than classifying them as a common VoIP flows. Such framework should be scalable so the application detection and enforcing application-aware policy is done in a smooth and uninterrupted manner. They have deployed the framework in the HP Lab wireless network. It requires the users to install software agents on their mobile devices. These agents collect information such as active network sockets, Netstat logs for each application. The agents send this information to the controller, where the controller runs machine learning trainer. The OpenFlow switch statistics are extended to store first n packet size of each flow and announce it to the controller. The controller collects such flow features along with the information sent by the agents to train the ML tool by using the ML trainer.

Hamid *et al*. in [26] introduce a tag-based classification architecture, where the source-edge switches tag the packets based on the application classes. This way the network operator can apply different policies for each of the application classes. The tag is also used as an identifier for matching the packets which reduce the matching overhead. After a tagged packet is delivered to the destination edge switch, the switch removes the tag and performs the required actions, if there is any action, and sends the packet to the destination host. The experimental result shows this tag-based approach is 3 ms faster than the hash-based field matching methods like OpenFlow field matching, and reduces processing overhead. To solve the backward compatibility, unlike MPLS, the tag is added to the end of packet instead of its middle. This way, if the variable length packet is supported, there is no need for whole packet parsing. Otherwise, the whole packet should be parsed. The downside of this approach is that it requires changes in the switch internal by introducing a new API to the switch data plane. This API is an application layer processor in the data plane.

A promising way to address challenges and problems in distributed environments, such as a computer network, is with the help of intelligent agents, i.e. Multi-Agent System (MAS) and mobile agents. Bieszczad *et al*. [27] describes how intelligent agents can be used to facilitate network management. It explains the potential of Intelligent Agents to tackle various difficulties in different network management areas such as fault management, security, performance management, accounting, etc. SDN provides a good platform for the agents to tackle such difficulties.

In [28] Skobelev *et al*. propose a task-scheduling system for SDN-based networks. This system incorporates MAS to overcome task-scheduling problems in the distributed systems, i.e. where the servers and computational resources are distributed. The MAS task scheduler associates the basic system entities with an agent. It consists of three main agents:

- task agent represents the task that should be processed with minimum cost by a server in the network,
- resource agent provides the system with a server to process tasks,
- commutator agent is responsible for providing information about network state and task allocation to the nodes.

This system is developed in C#, .NET framework, as a Windows application.
The research [29]–[31] show that by providing application-awareness and feedback from clients' machines to the network, the user Quality of Experience (QoE), and resource utilization can be improved. These works use agents on user side to collect information (like audio/video quality, waiting time, etc.) and send this information to the controller to adjust the network state accordingly to improve users' QoE.
To address traffic forwarding and traffic engineering in SDN, Dinh et al. [32] introduced a multipath-based forwarding traffic engineering mechanism called MSDN-TE. The goal of this mechanism is to forward the traffic in such a way that it avoids congestion on any link in the network. MSDN-TE dynamically selects the best available shortest paths and forwards the incoming traffic. This TE mechanism gathers network state information and considers the actual path's load to forward the flows on multiple paths. The MSDN-TE is a module which extends OpenDayLight controller. It consists of three components:

- a monitoring function which is used for gathering information about network states and flows in the network; for example, flow's static, link utilization, network topology, etc. The path matrices are refreshed every 10–15 s;
- the TE algorithm calculates n number of paths, which have the lowest traffic load, between the source and destination node. To select the shortest paths Eppstein [33] algorithm is used;
- the actuating function supports TE algorithm module. It takes certain actions and dynamically allots flows to the selected paths. Compared with the Shortest Path First and spanning tree, MSDN-TE shows better performance in regard to download time, delay and packet drops. For example, it reduced packet drops by 72.9% in AGIS [34] simulated topology and more than 90% in Abilene [34] simulated topology.

4. Traffic Engineering Research Challenges

As SDN becomes widely used, the research community and industry introduce new protocols and control applications like TE mechanisms. However, like any new technology the potential of SDN to simplify and improve network management comes with new challenges that need to be addressed. In this section, some of the challenges and future directions for traffic engineering in SDN are discussed, namely, fault-tolerance, energy-awareness, flow-update scheduling and consistency, and data-flow scheduling and dissemination.

4.1. High-availability

Fault tolerance is an important feature of any computer network. It means if an unexpected error or problem happens, like the failure of a link or a switch, the services in the network will continue to be accessible. The faults in a network can cause congestion and packet loss. These conditions can last for seconds due to the time it takes for TE mechanism to respond to the faults and update the network, i.e. update the topology and switches. In a SDN-based network two types of failure are control plane failure and data plane failure, like failure of links and switches. Besides physical/logical failure of control plane, control plane failure can also refer to a situation when the controller fails to update switches in the right time, so the switches continue to forward the traffic with the old rules. This can lead to congestion because the link capacity is not considered. There are two approaches to address the faults in the network: proactive where the paths are calculated and reserved beforehand, and reactive where the resources are not reserved until failure happens. The paths are calculated dynamically or decided in advance. Proactive approach has faster fault recovery since the paths are already calculated. When a fault occurs there is minimum interaction between the controller and the switch. This approach is about 5 times faster than reactive approach [35], [36]. But, reactive approach has a lower cost because the link capacity is not reserved, so it requires less memory in the control plane.
In [37] Liu *et al*. have introduced a proactive fault management TE mechanism, known as forward fault correction (FFC), which handles both data plane and control plane faults. In this approach if the number of faults is smaller than a configurable bound f, it can ensure protection against failure and congestion in the network. Depending on the value of f, and the traffic distribution flowing in the network, FFC provisions a certain amount of link capacity to avoid failure in the network. However, since the link capacity is pre-provisioned it can have lower throughput. Kim *et al*. in [38] introduced a fault-tolerance framework called CORONET, which uses a centralized controller to forward packets and can work with any network topology. CORONET recovers from switch or link failures in a short period. It uses multi-path routing methods. Its architecture

consists of modules to discover network topology, route planning, shortest-route path calculation, and traffic assignment. To simplify packet forwarding, and minimize the number of flow-rules CORONET uses VLANs mechanism in the switches. Therefore, it is also scalable.
A traffic engineering framework should detect faults in the network and re-route the sensitive applications' traffic by avoiding the failed areas to allow these applications to work seamlessly and avoid service degradation. SDN characteristics such as failover mechanism introduced in OpenFlow protocol, global view of network, and its capability to dynamically change network state facilitate failure recovery. However, it is still challenging since the controller needs to calculate the new paths and install the flow-rules in the switches. The TE should achieve low communication overhead with a trade-off between the latency and memory usage.

4.2. Energy-awareness

To guarantee QoS in a network, high-end networking devices that have a high power consumption are used. To reduce delay and increase reliability, these resources are usually over-provisioned to increase network capacity. However, this leads to concerns about greenhouse gas emission and power wastage. A number of researches show that non-negligible percentage of world power consumption and CO_2 emission is due to information and communication technologies [39]. This motivated the researchers to propose new algorithms and devices to address these difficulties [14]–[16]. These techniques adapt the network elements' active time according to the traffic load.
The techniques proposed for classical network architecture are not as effective as they can be. There are few studies on energy-aware techniques for SDN-based networks. In [40] Giroire *et al*. have introduced an energy-aware routing technique for SDN that gathers traffic matrix, calculate the routing paths to guarantee QoS, and put the idle links and nodes into sleeping mode. This technique considers both memory limitation of routers and link capacity. SDN features such as centralized view of network and network programmability can help to introduce new efficient centralized energy-aware TE techniques that allow a network to dynamically adapt to the traffic load and network condition with the goal of achieving good performance and use network resources effectively to reduce power consumption. The centralized TE mechanism can shut down a set of switches and links, when the traffic demand is low to reduce power utilization while satisfying user experience.

4.3. Data-flow Scheduling

After the rules are installed in the switches, a switch will match and send the incoming packets to the destination. An important way to ensure QoS in a network is flow scheduling, where the packets that require better QoS are scheduled and transferred first. Flow priority is the main scheduling method [41], [42]. In this method, the flows with higher priorities are sent first. SDN provides a good platform to introduce new software-controlled flow schedulers that are capable of flow-oriented multi-policy scheduling. This abstraction can help to introduce advanced network configurability.
Bo-Yu *et al*. [43] have introduced an Iterative Parallel Grouping Scheduling Algorithm, IPGA-scheduling. It is designed to address the prioritized flow scheduling problem, which is required for QoS differentiation among different prioritized flows, and also energy saving in data center networks. This system is a Compute Unified Device Architecture (CUDA) system within SDN controller. CUDA is a parallel computing architecture developed by Nvidia for graphics processing.
In [42] Rifai *et al*. proposes coarse-grained scheduling. The authors argue that the data center networks and the Internet traffic nature mostly consist of short flows and most of the flows are carried by TCP. Therefore, the emphasis of this scheduling system is on flow-size scheduling. This system uses switch's OpenFlow flow statistics to identify the flow-size along with multiple queues per port to implement 802.1p QoS. The 802.1p standard delegates 8 queues per port. Two size-based schedulers are introduced. Both of these schedulers have two queues per port, and it is assumed that they are managed by strict priority scheduler.
Using a scheduler can improve the network performance. The majority of the schedulers are developed around the idea of "one size fits all", or consider only the flow size, and the flow value. The type of the application is ignored. These approaches examine, mostly, packet's priority and port workload while assigning the flows to a port. For example, in a VoIP network, the VoIP applications need to have the highest priority to ensure QoS. Even though priority-based solutions can address these requirements, they require precise configuration of the network which is time-consuming and error-prone.

4.4. Flow-update

In an operating network, the controller may change the configuration of the switches several times through flow-updates. Flow-update refers to updating the current switch configurations, forwarding rules, with new configurations. Flow-updates are important for various tasks such as fault management, adapting to changes in traffic pattern, etc. Flow-update is a challenging task since improper update of multiple flows can cause problems such as congestion, service degradation, and inconsistency in the network. Hence, flow-update scheduling is an important issue to be addressed. If a new rule is assigned for each flow it can increase the resource cost (e.g. processing and memory) in both the data plane and control plane. Also, the time that it takes for a flow to be added in a switch adds to the latency. There should be a tradeoff between load-balancing and latency.
The most common approaches for flow-update scheduling problem is the two-phase update mechanism, where controller first installs the new forwarding-rules into the

switches, if all packets that require the old rules are transferred, then the new installed rules will be used and the old forwarding-rules are removed from the switches. Compared to the one-phase approach, the advantage of this approach is that the chance of the controller to fail in updating the switches is lower. However, Li *et al*. [44] argue that the two-phase update mechanism is not effective, since it does not consider the switch's flow-table size. Thus, to address the multi-flow update problem, a step-by-step approach is introduced. This problem is formulated as an optimization problem to minimize the maximum link utilization, which is an important network performance metric. In this approach the controller schedules the flow updates and then updates each flow step by step, i.e. the path of a flow is changed to the new one in a step, so if there are *n* flow updates then the process is completed in *n* steps. This method considers both the link capacity and the flow-table size.
In [45] Mahajan believe that flow-update, to ensure consistency, has a number of properties such as loop free, packet drop free, switch memory limitation, load balancing, etc. Depending on the type of a network, different consistency, or combination of the consistency properties are needed, for example load-balancing or loop free network.

5. Conclusion

In this paper we have reviewed literature in the field of traffic engineering for both traditional network architecture and SDN, and examined some of the TE challenges and future directions. SDN is a new networking paradigm which simplifies the network management and enables innovation. It tries to address many problems in the traditional network architecture by simplifying network management through centralized management of a network, introducing network programmability, and providing a global view of a network and its state. New traffic engineering techniques are required to exploit these features for better control and management of traffic. Different TE mechanism should be included in SDN to control congestion and manage traffic for different applications in various QoS-sensitive scenarios such as video or business data, and to provide required QoS while balancing the load among the available resources in a network. To improve the network load handling, a traffic engineering mechanism should enable a network to react in real-time and classify a variety of traffic types from different applications. Routing optimization is one of the main techniques in TE. It should take advantage of multiple paths in the network and coordinate traffic scheduling by using global view of traffic across the available network paths. Beside load-balancing and optimization of resources, other components of TE are QoS and resilience form failure. SDN is currently capable of enforcing policy for lower layers, i.e. Layer 2-4, but not many studies have explored the higher layer policy enforcement. By identifying the packets sent by the applications to the network, it is possible to enforce higher layer, application layer, policies. Higher layer policy enforcement can help to engineer resilient and flexible network. Such networks can be optimized for each application to provide a good QoS and improve user experience. The authors described how an end-host flow detection mechanism and Multi-Agent System can improve network performance and scalability while reducing complexity.

6. Future Work

In terms of future work, authors plan to propose an efficient traffic engineer framework, which makes the SDN-based networks more application-aware. In this work a multi-agent based software framework consisting of a number of algorithms for application classification, and data scheduling and dissemination will be developed. The agents are responsible for application classification of user's traffic. Then, the data scheduling and dissemination algorithms will calculate the best path and order to process and forward the flow to the destination. All these modules will work together to ensure high-availability, load-balancing and optimizing resource utilization, and also to ensure high-QoS rating for QoS sensitive applications. This framework can help to automate the network configuration to achieve high QoS for the desired applications. By combining techniques such as scheduling, application classification, and MAS, a network can deliver better services.

References

[1] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks", *IEEE Commun. Surv. & Tutor.*, vol. 16, no. 3, pp. 1617–1634, 2014.

[2] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights", in *Proc. 19th IEEE Ann. Joint Conf. of the IEEE Comp. & Commun. Soc. INFOCOM 2000*, Tel Aviv, Israel, 2000, vol. 2, pp. 519–528.

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

[4] O. N. Foundation, "OpenFlow – open networking foundation" [Online]. Available: https://www.opennetworking.org/sdn-resources/openflow (accessed Aug. 23, 2016).

[5] K. Ishiguro, A. Lindem, A. Davey, and V. Manral, "Traffic engineering extensions to OSPF Version 3", RFC 5329, IETF Trust, 2008 [Online]. Available: https://tools.ietf.org/html/rfc5329

[6] T. Li and H. Smit, "IS-IS extensions for Traffic Engineering", RFC 5305, IETF Trust, 2008 [Online]. Available: https://tools.ietf.org/html/rfc5305

[7] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols", *Commun. Mag.*, vol. 40, no. 10, pp. 118–124, 2002.

[8] D. Thale and C. Hopps, "Multipath issues in unicast and multicast next-hop selection", RFC 2991, IETF Trust, 2000 [Online]. Available: https://tools.ietf.org/html/rfc2991

[9] D. Zhang and D. Ionescu, "QoS performance analysis in deployment of DiffServ-aware MPLS Traffic Engineering", in *Proc. 8th ACIS Int. Conf. on Software Engin., Artif. Intell., Netw., & Parallel/Distrib. Comput. SNPD 2007*, Qingdao, China, 2007, vol. 3, pp. 963–967.

[10] F. Le Faucheur *et al*., "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services", RFC 3270, IEFT Trust, 2002 [Online]. Available: https://tools.ietf.org/rfc/rfc3270.txt

[11] I. F. Akyildiz *et al*., "A new traffic engineering manager for DiffServ/MPLS networks: design and implementation on an IP QoS Testbed", *Computer Commun*., vol. 26, no. 4, pp. 388–403, 2003.

[12] I. Gojmerac, T. Ziegler, F. Ricciato, and P. Reichl, "Adaptive multipath routing for dynamic traffic engineering", in Proc. Global Telecommun. Conf. GLOBECOM'03, San Francisco, CA, USA, 2003, vol. 6, pp. 3058–3062.

[13] I. Poese, B. Frank, G. Smaragdakis, S. Uhlig, A. Feldmann, and B. Maggs, "Enabling content-aware traffic engineering", *ACM SIGCOMM Comp. Commun. Rev.*, vol. 42, no. 5, pp. 21–28, 2012.

[14] M. Zhang, C. Yi, B. Liu, and B. Zhang, "GreenTE: Power-aware traffic engineering", in *Proc. 18th IEEE Int. Conf. on Netw. Protocols ICNP 2010*, Kyoto, Japan, 2010, pp. 21–30.

[15] E. Amaldi, A. Capone, L. G. Gianoli, and L. Mascetti, "A MILP-based heuristic for energy-aware traffic engineering with shortest path routing", in *Network Optimization*, J. Pahl, T. Reiners, and S. Voß , Eds. *LNCS*, vol. 6701, pp. 464–477. Springer, 2011.

[16] N. Vasić and Dejan Kostić, "Energy-aware traffic engineering", in *Proc. of the 1st Int. Conf. on Energy-Effic. Comput. & Netw. e-Energy'10*, Passau, Germany, 2010, pp. 169–178.

[17] L. Zhang and D. Clark, "Oscillating behavior of network traffic: A case study simulation", *Internetworking: Res. and Exper.*, vol. 1, no. 2, pp. 101–112, 1990.

[18] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN", *ACM SIGCOMM Comp. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013.

[19] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic Flow Scheduling for Data Center Networks", in *Proc. 7th USENIX Symp. on Netw. Syst. Design & Implemen. NSDI'10*, San Jose, CA, USA, 2010, vol. 10, pp. 19–19.

[20] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: scaling flow management for high-performance networks", *ACM SIGCOMM Comp. Commun. Rev.*, vol. 41, no. 4, pp. 254–265, 2011.

[21] A. R. Curtis, W. Kim, and P. Yalagandula, "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection", in *Proc. 30th IEEE Int. Conf. Comp. Commun. IEEE INFOCOM 2011*, Shanghai, China, 2011, pp. 1629–163.

[22] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers", in *Proc. 7th Conf. on Emerg. Networking Experim. & Technol. Co-NEXT'11*, Tokyo, Japan, 2011, p. 8.

[23] R. Trestian, G.-M. Muntean, and K. Katrinis, "MiceTrap: Scalable traffic engineering of datacenter mice flows using OpenFlow", in *IFIP/IEEE Int. Symp. on Integr. Netw. Managem. IM 2013*, Ghent, Belgium, 2013, pp. 904–907.

[24] S. Valenti, D. Rossi, A. Dainotti, A. Pescapè, A. Finamore, and M. Mellia, "Reviewing traffic classification", in *Data Traffic Monitoring and Analysis*, E. Biersack, C. Callegari, and M. Matijasevic, Eds. *LNCS*, vol. 7754, pp. 123–147. Springer, 2013.

[25] Z. A. Qazi *et al.*, "Application-awareness in SDN", *ACM SIGCOMM Comp. Commun. Rev.*, vol. 43, no. 4, pp. 487–488, 2013.

[26] H. Farhadi and A. Nakao, "Rethinking flow classification in SDN", in *Proc. IEEE Int. Conf. on Cloud Engin. IC2E 2014*, Boston, MA, USA, 2014, pp. 598–603.

[27] A. Bieszczad, B. Pagurek, and T. White, "Mobile agents for network management", *Commun. Surveys*, vol. 1, no. 1, pp. 2–9, 1998.

[28] P. O. Skobelev, O. N. Granichin, D. S. Budaev, V. B. Laryukhin, and I. V. Mayorov, "Multi-agent tasks scheduling system in software defined networks", *J. of Physics: Conf. Series*, vol. 510, no. 1, p. 012006, 2014 (doi: 10.1088/1742-6596/510/1/012006).

[29] M. Jarschel, F. Wamser, T. Hohn, T. Zinner, and P. Tran-Gia, "SDN-based application-aware networking on the example of YouTube video streaming", in *Proc. 2nd Eur. Worksh. on Softw. Defined Netw. EWSDN 2013*, Berlin, Germany, 2013, pp. 87–92.

[30] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race, "Towards network-wide QoE fairness using openflow-assisted adaptive video streaming", in *Proc. ACM SIGCOMM Worksh. on Future Human-Centric Multim. Netw. FhMN 2013*, Hong Kong, China, 2013, pp. 15–20.

[31] H. Nam, K.-H. Kim, J. Y. Kim, and H. Schulzrinne, "Towards QoE-aware video streaming using SDN", in *Proc. Global Commun. Conf. GLOBECOM 2014*, Austin, TX, USA, 2014, pp. 1317–1322.

[32] K. T. Dinh, S. Kukliński, W. Kujawa, and M. Ulaski, "MSDN-TE: Multipath Based Traffic Engineering for SDN", in *Intelligent Information and Database Systems. Asian Conference on Intelligent Information and Database Systems*, N. T. Nguyen, B. Trawiński, and R. Kosala, Eds. Springer, 2016, pp. 630–639.

[33] D. Eppstein, "Finding the *k*-shortest paths", *SIAM J. Comput.*, vol. 28, pp. 652–673. 1999.

[34] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet topology zoo", *IEEE J. on Selec. Areas in Commun.*, vol. 29, no. 9, pp. 1765–1775, 2011.

[35] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Enabling fast failure recovery in OpenFlow networks", in *Proc. 8th Int. Worksh. on the Des. of Reliable Commun. Netw. DRCN 2011*, Kraków, Poland, 2011, pp. 164–171.

[36] D. Staessens, S. Sharma, D. Colle, M. Pickavet, and P. Demeester, "Software defined networking: Meeting carrier grade requirements", in *Proc. 18th IEEE Worksh. on Local & Metropolitan Area Networks LANMAN 2011*, Chapel Hill, NC, USA, 2011, pp. 1–6.

[37] H. H. Liu, S. Kandula, R. Mahajan, M. Zhang, and D. Gelernter, "Traffic engineering with forward fault correction", *ACM SIGCOMM Comp. Commun. Rev.*, vol. 44, no. 4, pp. 527–538, 2014.

[38] H. Kim, J. R. Santos, Y. Turner, M. Schlansker, J. Tourrilhes, and N. Feamster, "Coronet: Fault tolerance for software defined networks", in *Proc. 20th IEEE Int. Conf. on Network Prot. ICNP 2012*, Austin, TX, USA, 2012, pp. 1–2.

[39] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy efficiency in the future Internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures", *Commun. Surveys & Tutor.*, vol. 13, no. 2, pp. 223–244, 2011.

[40] F. Giroire, J. Moulierac, and T. K. Phan, "Optimizing rule placement in software-defined networks for energy-aware routing", in *Proc. Global Commun. Conf. GLOBECOM 2014*, IEEE, Austin, TX, USA, 2014, pp. 2523–2529.

[41] F. Pop, C. Dobre, D. Comaneci, and J. Kołodziej, "Adaptive scheduling algorithm for media-optimized traffic management in software defined networks", *Computing*, vol. 98, no. 1-2, pp. 147–168, 2016 (doi: 10.1007/s00607-014-0406-9).

[42] M. Rifai, D. Lopez-Pacheco, and G. Urvoy-Keller, "Coarse-grained scheduling with software-defined networking switches", in *Proc. 2015 ACM Conf. on Spec. Interest Group on Data Commun. SIGCOMM'15*, London, UK, 2015, pp. 95–96. 2015.

[43] B. Y. Ke, P.-. Tien, and Y.-L. Hsiao, "Parallel prioritized flow scheduling for software defined data center network", in *Proc. 14th Int. Conf. on High Perform. Switch. & Rout. IEEE HPSR 2013*, Taipei, Taiwan, 2013, pp. 217–218.

[44] Y. Liu, Y. Li, Y. Wang, and J. Yuan, "Optimal scheduling for multi-flow update in Software-Defined Networks", *J. of Network & Computer Applications*, vol. 54, no. C, pp. 11–19, 215 (doi: 10.1016/j.jnca.2015.04.009).

[45] R. Mahajan and R. Wattenhofer, "On consistent updates in software defined networks", in *Proc. 12th ACM Worksh. on Hot Topics in Netw. HotNets-XII*, College Park, MD, USA, 2013, p. 20.

Mohammad Reza Abbasi received his MCA degree in Computer Science and Applications from Panjab University, Chandigarh, India, in 2013. He is currently pursuing his Ph.D. in Panjab University. His research interests include software defined networking, network management, and network virtualization.

E-mail: mabbasi@pu.ac.in
Department of Computer Science & Application
Panjab University
160014 Chandigarh, India

Ajay Guleria received his Ph.D. degree in Computer Science and Engineering from National Institute of Technology Hamirpur. Presently he is working as Senior System Manager in Panjab University Chandigarh. His current research areas of interest include software defined networking, information centric networking, network security and vehicular ad hoc networks. He is a member of IEEE, ISTE.

E-mail: ag@pu.ac.in
Computer Center
Panjab University
160014 Chandigarh, India

Mandalika S. Devi is a Professor in the Department of Computer Science and Applications, Panjab University, Chandigarh. She received her Ph.D. degree in Computer Science and Systems Engineering from Andhra University, Visakhapatnam and M.E. in Computer Science and Engineering, from NIT, Allahabad. She has completed M.Sc. in Applied Mathematics from Andhra University, Visakhapatnam. Before joining Panjab University, she served Indian Space Research Organization, Sriharikota, and National Institute of Technical Teachers' Training and Research, Chandigarh. Her areas of expertise include algorithms, image processing, distributed artificial intelligence and educational computing.

E-mail: syamala@pu.ac.in
Department of Computer Science & Application
Panjab University
160014 Chandigarh, India