MSDN-TE: Multipath Based Traffic Engineering for SDN

Khoa Truong Dinh[1(✉)], Sławomir Kukliński[1,2], Wiktor Kujawa[1], and Michał Ulaski[1]

[1] Warsaw University of Technology, Warsaw, Poland
k.truongdinh@stud.elka.pw.edu.pl, slawomir.kuklinski@tele.pw.edu.pl, wkujawa@mion.elka.pw.edu.pl, michal.ulaski@gmail.com
[2] Orange Polska, Warsaw, Poland

Abstract. Software-Defined Networking is a new networking paradigm that has recently gained a lot of attention in the networking community. Its fundamental idea lies on the separation of the centralized control plane and the distributed forwarding plane. Due to global network view and the programmability, SDN is a better tool for solving traffic engineering problems than mechanisms, which exist in classical IP networks. In this paper a simple approach to SDN traffic engineering based on multipath forwarding is presented. The approach lies on dynamic selecting of the best path among several paths for forwarding of an incoming flow. The concept has been simulated using the OpenDaylight controller and Mininet simulator. Simulations have confirmed advantages of the proposed concept over the Shortest Path First (built-in the OpenDaylight controller) and Spanning Tree Protocol (Mininet simple forwarding techniques) approaches.

Keywords: SDN · Traffic engineering · Multipath · OpenDaylight

1 Introduction

In classical IP networks, IP packets are forwarded using the least (shortest) cost paths toward the destination. The shortest cost path algorithms (implemented as a part of IS-IS, CSPF or OSPF protocols) are usually based on simple, static metrics such as link weight or number of hops. The cost for the path is the sum of the weights of all the links belonged to this path. The available path bandwidth is typically not taken into account – even if taken, the time required to update or to gather the entire network state information is typically too long for fast network reaction. In case when the network has spare links, if a failure happens, the routers will compete for the best new path, which will likely be the shortest path. The moved traffic therefore will be forwarded through the new shortest path causing this path to become congested, whereas links on other paths remain under-utilized. This example shows the importance of traffic load balancing among multiple paths. It is worth to mention that the load balancing mechanism contributes to increased network efficiency (ability to handle more intensive traffic) and increased QoS for the best effort service by reduction of network congestion and packet loss rate.

N.T. Nguyen et al. (Eds.): ACIIDS 2016, Part II, LNAI 9622, pp. 630–639, 2016.
DOI: 10.1007/978-3-662-49390-8_61

There are already existing some approaches dealing with Traffic Engineering problem in the networks such as MPLS-TE, RSVP-TE, ECMP, etc. RSVP-TE extends RSVP with many features (such as LSP set up, Fast Reroute, GMPLS extension) in order to provide traffic engineering mechanisms [2]. MPLS-TE control plane is seen to be complex and makes dynamic operation difficult and expensive. Another traffic engineering solution, ECMP (Equal Cost Multipath) has some limitations, it cannot be used to split traffic between parallel paths with different capacities [3].

The creation of Software Defined Networking (SDN) that is based on the OpenFlow [7] protocol, used for controlling flows forwarding has recently gained enormous interest. SDN [8] decouples the programmable control plane from network forwarding plane. SDN consists of the forwarding plane, the control plane and the application plane [9]. SDN approach comes with multiple benefits. It has been proved that such centralized approach provides much shorter time of paths setup than using existing, distributed routing protocols [1]. Moreover, by limiting the number of control plane devices and simplifying data plane device architecture, the complexity and cost of the network can be significantly reduced. In addition, the traffic engineering (TE) mechanisms don't have to be implemented in every network element. The programmability of SDN, easy interface to applications and flow oriented operations are also important in the context of TE and creation of application-aware networks. The centralized controller becomes the brain of the entire network, and with network global overview, it can efficiently decide where to forward flows in order to fulfil application (end-user) demands and at the same time to optimize the forwarding plane resource utilization. The following properties of SDN show that an efficient TE based can be implemented:

- The whole network topology is known to the SDN controller, therefore all existing paths and their actual load can be easily identified and more than a single route can be used for traffic forwarding between any source-destination pair.
- The controller, using OpenFlow, can change the configuration of Flow Switching Tables (FST) of all switches in real-time.

A hybrid attempt based on MPLS and SDN has been already proposed in [4], which uses the standard MPLS data plane and an OpenFlow based control plane. This is seen as one of the first step to use SDN to handle the problem of TE, despite the fact that only bandwidth reservation mechanism combined with the shortest path algorithm in this approach was proposed. In the B4 solution [10], Google deployed TE in a hybrid, commercial network that includes SDN switches and classical IP routers. It has been claimed that the B4 approach drives most of links (ca. 70 %) to near 100 % utilization, providing 14 % increase in average network throughput due to forwarding the data flows among multiple paths in comparison to previous, MPLS based solution. In B4 the splitting of the traffic applies to new coming flows only, for the on-going flows no actions are taken. In [11] it has been shown that SDN network significantly outperforms OSPF routing in terms of network throughput, delay and packet loss ratio.

The goal of this paper is to exploit and evaluate the properties of SDN and the ability to manipulate flows in real-time for TE. Section 1 (this section) introduces TE issues in IP networks and benefits of SDN in that context. Section 2 presents multipath

routing and dynamic routing issues. The proposed MSDN-TE approach is described in Sect. 3. The simulation environment and results of MSDN-TE approach are included in Sect. 4. Finally, Sect. 5 summarizes the paper.

2 Traffic Engineering Overview

The main goal of TE is to use effectively network resources and to avoid congestion of links or paths, while serving multiple network users. In order to achieve a high level of users' satisfaction, all CBR (constant bitrate) flows should obtain sufficient average bitrate in a short-term whereas the TCP traffic can adapt to changing network conditions and if source or destination provides high capacity a TCP section can saturate an existing path. The traffic properties between two end-devices in most cases are asymmetrical. In case of the dynamic routing, the routing decision can be taken according to path's cost that is based on the actual path's metric. In order to achieve that goal, the state of all network paths has to be evaluated and the information about the overall network's load and other statistics need to be collected.

In case when multiple paths are found, there are two different possibilities of traffic forwarding:

- The best path can be selected and only this path is used for traffic forwarding whereas the second path serves as a backup and is used in case of network failure. Most IP networks TE nowadays are applying this approach.
- The traffic can be distributed among the existing paths. There are several possibilities of such distribution, e.g. per packet or per flow basis. The first approach provides higher granularity of traffic distribution for the sake of packets disordering, whereas the second one is simpler, but less flexible.

The main problems with dynamic routing lies on:

- Incorrect decision related to path swapping, which may lead to the ping-pong effect due to increased load of the switched paths. As a result, routing instability can be observed.
- Unpredictable traffic volume in the switched paths caused by intrinsic TCP mechanism that can grab all the available bitrate therefore leading to the new path congestion.
- In case of per packet splitting there is a need to reorder packet at the destination. Such operation has to be supported by appropriate receiver buffer used for restoring the proper order of packets, which introduces additional transmission delay.
- In case of per flow traffic splitting, the granularity of switched traffic is potentially low, and the knowledge about the flow properties can be beneficial for this process. However, not much information about incoming flow can be obtained until it actually occurs.

3 MSDN-TE Description

In this paper, the multipath traffic engineering approach for SDN (MSDN-TE) is proposed. This approach is based on the usage of multiple paths to forward flows taking into account the actual path's load for the forwarding decisions. The approach relies on computing k-paths available to forward flows between any Source-Destination pair (S-D) and to select the least loaded path to handle an incoming flow. The main goal of the MSDN-TE algorithm is to avoid congested points in the network, moreover more distributed traffic among the existing paths contributes to easier handling of faults.

For finding multiple paths between the S-D, the k-shortest paths i.e. the Eppstein algorithm [12] has been used. At present, for the sake of simplicity, only the load of links contributed to paths is taken into account as a parameter to evaluate the path's cost. The number of paths used (k-number) is dependent on the network size. In a small network used during experiments, all available loop-free paths were used. However, in big networks, the number of paths can be dependent on network topology and is expected to be in range 2–5.

The MSDN-TE procedure is following:

1. *When new flow is coming the existence of an appropriate path is checked*
   - *If k-path between S-D doesn't exist such k-path is created*
2. *The cost of all paths (in this version load only) in k-path set is calculated and the flow is assigned to the lowest cost path*
3. *Network overall statistics are read and compared with previous values*
4. *Users' resource consumption is recorded (aggregated user's throughput)*
5. *All links (and paths) metrics are updated (available bitrate, packet loss rate, cost)*
6. *Go to 3 (or 1 if a new flow is detected).*

3.1 MSDN-TE Implementation

The MSDN-TE approach has been implemented as an additional module (OSGi bundle) to the OpenDaylight controller. In the implementation MSDN-TE consists of three main functional blocks: MONFUN, ACTFUN and TE Algorithm. MONFUN is a set of functions responsible for collecting information about the network. This information is input to the TE Algorithm, an algorithm which is responsible for decisions regarding flow to path assignment. This process is supported by ACTFUN, a block responsible for taking certain actions based on decision of TE algorithm. The overview of the MSDN-TE implementation is depicted in Fig. 1.

Monitoring Functions (MONFUN) retrieve monitoring parameters (such as network topology, flow's statistics, link utilization, etc.) from the underlying network elements. The set of monitoring data includes: PLR (*packet_loss_ratio*), *link_load*, *packet_delay*. MONFUN also provides information about flows and their assignments to paths. The total number of active flows in network is retrieved with their *idle_timeout/hard_timeout* (if no packet has matched the rule for such period of time, since the flow was inserted, the switch removes the entry and sends *FLOW_REMOVED*

message to controller). Besides, the number of flows per host is also obtained to analyse how much traffic each host generates in the network. The refresh rate of path metrics is in range 10–15 s and is dependent on the ability of the controller to collect this information from switches. The network topology update is also retrieved in such time period. Three synthetic network parameters are collected: the *Link_load is* an averaged load of each link over certain period of time; the *Average_network_load* refers to the averaged load of all network links; and the *total_network_dropped* is the number of all dropped network packets.

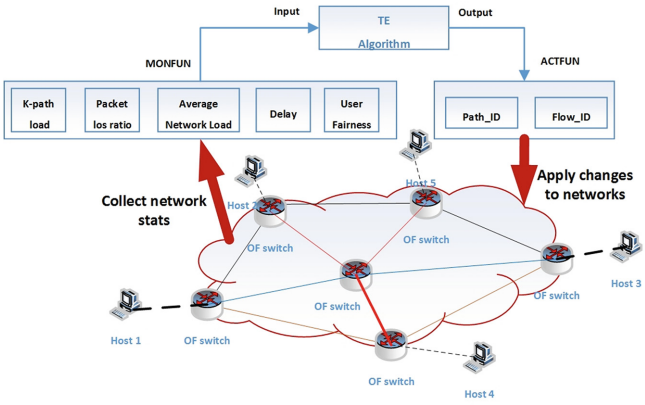

Fig. 1. Overview of MSDN-TE architecture

The monitoring of links is natively provided by OpenDaylight. On that basis there is possible to evaluate the path's quality and calculating quality parameters of all links constituting the path. The link parameters are *link_load, packet_loss_rate* and *delays*.

Actuating Functions (ACTFUN) are used to make changes to the network forwarding plane in a simple way. Controller using this functions can dynamically adjust the acting parameters *(path_ID and flow_ID)* to assign flows to the best available path. The *path_ID* is the path where flow is to be assigned and the *flow_ID* is the identification of the flow.

4 Simulation Environment and Results

In order to check the behavior of the proposed approach the testbed and testing scenarios have been created. The testbed was composed of:

- OpenFlow network emulator: Mininet (OpenVirtualSwitch with OF v1.0).
- OpenDayLight [13] Hydrogen version based SDN controller, enhanced by additional OSGi bundles. These new bundles implemented MONFUN, ACTFUN and the traffic engineering algorithm of MSDN-TE.
- I-DTG (Internet – Distributed Traffic Generator) used to generate network traffic and to monitor different network parameters (*jitter*, *packet_delay*) [14].

4.1 Testing Scenarios

Two kinds of benchmarks of different network topology types were used. The first one (Benchmark 1) is applied to simple network topologies (ring topology and small mesh network) in order to validate proper behaviour of the implemented functions. The second one (Benchmark 2) is using more realistic network topologies, i.e. network topologies taken from the Internet Topology Zoo database (AGIS and Abilene topology as shown in Figs. 5 and 6 respectively) [15]. In order to evaluate the efficiency of the MSDN-TE algorithm different testing scenarios have been performed for each benchmark.

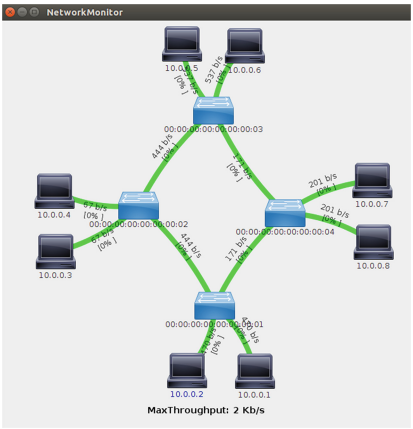

Fig. 2. Scenario 1: Ring topology

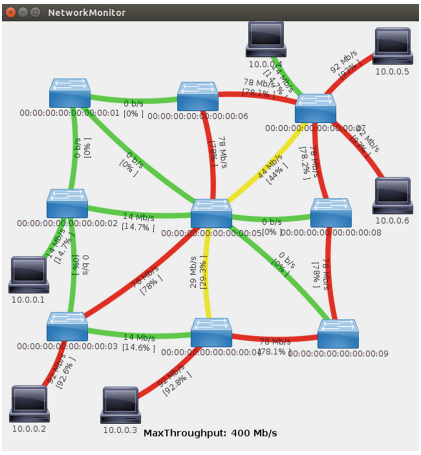

Fig. 3. Scenario 2: Traffic distribution in case of MSDN-TE (k = 5)

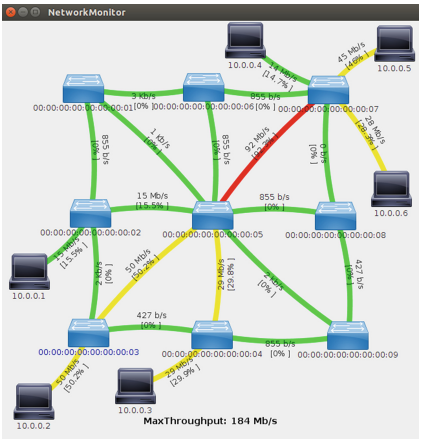

Fig. 4. Scenario 2: Traffic distribution in case of SPF

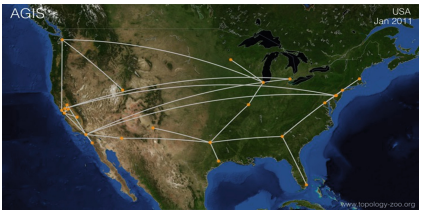

Fig. 5. AGIS topology

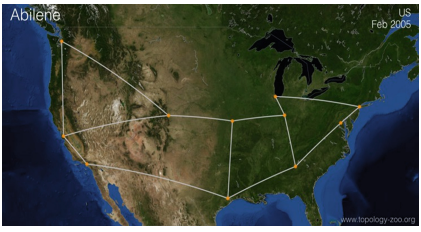

Fig. 6. Abilene topology

Benchmark 1. The description of testing scenarios is given below:

- Scenario 1: a ring network topology as shown in Fig. 2. Hosts are downloading a 100 Mb files from other hosts using the HTTP protocol.
- Scenario 2: mesh network (9 nodes, see Figs. 3 and 4), no background traffic provided. Hosts are downloading a 100 Mb files from other hosts using the HTTP protocol.
- Scenario 3: a modified Scenario 2 with the background traffic added.

Three data forwarding mechanisms were compared in Benchmark 1:

- the spanning tree protocol (STP) built-in natively in OpenVirtualSwitch;
- the shortest path first (SPF) protocol built-in OpenDaylight (ODL);
- multipath forwarding provided by MSDN-TE, implemented as an extension to OpenDaylight controller, with one and five paths used for data forwarding.

Benchmark 2. In this benchmark the MSDN-TE algorithm is compared with the STP protocol in terms of average delay and number of packet dropped. AGIS and Abilene topologies are used. The traffic is generated by I-DTG for the duration of 4 min. In case of Abilene topology, 7 hosts generate randomly 18 flows while in AGIS case, 25 hosts generate randomly 70 flows to different destination hosts.

4.2 Simulation Results

Figure 7 shows the comparison of downloading time of different scenarios in case of Benchmark 1. In this case the MSDN-TE algorithm with k = 5 in comparison to STP reduces significantly the downloading time: by more than 56 % in Scenario 1, about 33 % in Scenario 2 and more than 55 % in Scenario 3. In Scenario 1, due to the limited paths between every source-destination pair as a result of ring topology, the downloading times observed in case of SPF, MSDN-TE (k = 1) and MSDN-TE (k = 5) are similar. Even in Scenario 2, the MSDN-TE algorithm with k = 1 provides similar result in comparison with SPF, because only one shortest path is used to forward the traffic.

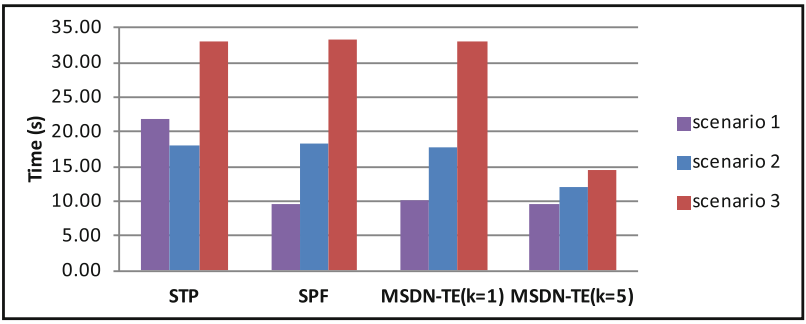

Fig. 7. Time of downloading files (using HTTP)

Figures 3 and 4 show that MSDN-TE is able to increase more than two times total network throughput in comparison to SPF (400 Mbps vs. 184 Mbps for mesh topology).

Figures 8 and 9 present the average network delay and the total number of dropped packets for STP and MSDN-TE algorithm in case of the Abilene topology whereas Figs. 10 and 11 present the same results but for the AGIS topology (Benchmark 2). For such networks, MSDN-TE reduces remarkably the average network delay and the number of dropped packets in comparison to STP. Table 1 shows that the overall delay was reduced by 35 % for the AGIS topology and 65 % for the Abilene topology; whereas the total number of dropped packets was reduced by 72.9 % in case of AGIS and more than 90 % in Abilene case.

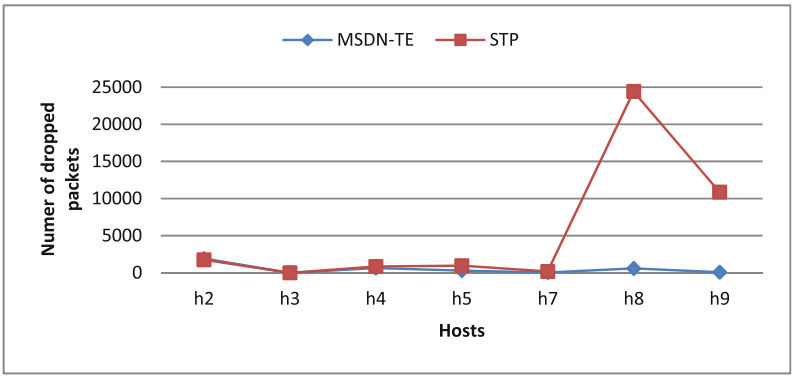

Fig. 8. Number of dropped packets in case of Abilene

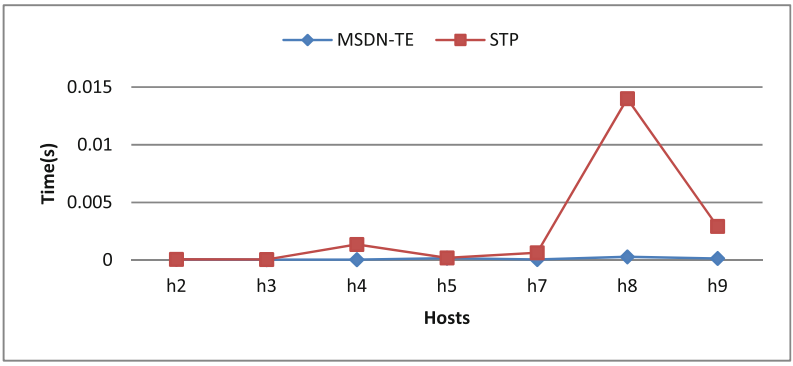

Fig. 9. Average network delay in case of Abilene

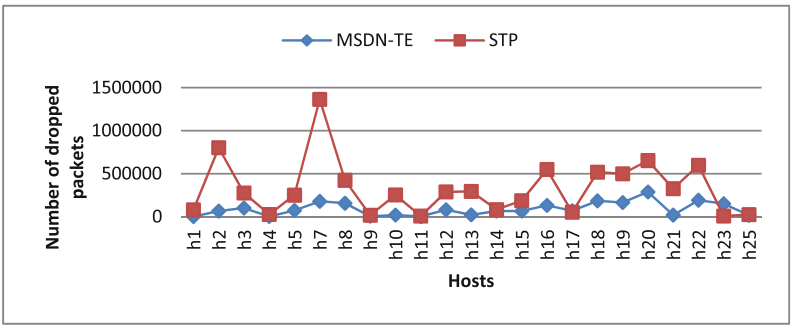

Fig. 10. Number of dropped packets in case of AGIS

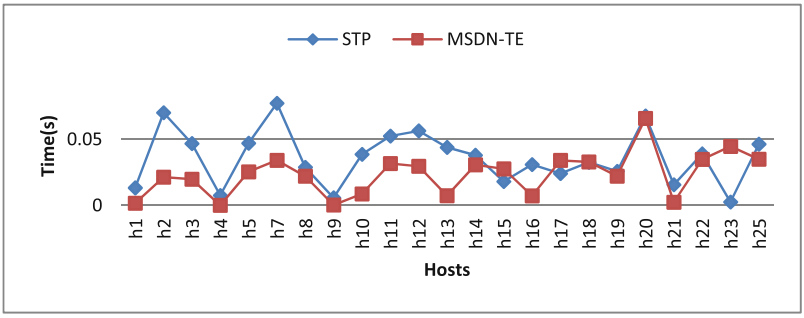

Fig. 11. Average network delay in case of AGIS

Table 1. Average delay and number of dropped packets comparison between STP and MSDN

Mechanisms / Parameters	STP		MSDN-TE (k=5)	
	Abilene	AGIS	Abilene	AGIS
Average network delay (s)	0.02	0.83	0.007 (65%)	0.54 (35%)
Number of dropped packets	39037	7571268	3542 (91%)	2047934 (72.9%)

5 Conclusions

In this paper a simple concept of multipath based forwarding approach for SDN networks has been described and evaluated. By taking advantage of SDN properties (global view of all links load), this approach, i.e. MSDN-TE, significantly outperforms simple traffic forwarding mechanisms based on STP and SPF in terms of packet dropped level, average network delay and the total network throughput. At present only the mechanism of initial assignment of flows to paths was implemented and assessed in MSDN-TE. However, during the flow assignment to path the flow's properties are not

known yet. It is therefore desirable to add a mechanism to dynamically redirect the on-going flows to less loaded paths after their initial assignment. Such extensions to MSDN-TE has been already implemented and simulations are on-going.

Acknowledgement. This work has been partially conducted as part of the CoSDN (Cognitive Software Defined Networks) project, which is funded by FNR Luxembourg and NCBiR Poland.

References

1. Matsui, K., Kaneda, M., Matsuda, K.: Evaluation of a server-based traffic engineering architecture suitable for large-scale MPLS networks. In: 2010 8th Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT), pp. 1–6. IEEE, June 2010
2. Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., Swallow, G.: RSVP-TE: extensions to RSVP for LSP tunnels (No. RFC 3209) (2001)
3. Farrel, A., Ayyangar, A., Vasseur, J.P.: Inter-Domain MPLS and GMPLS Traffic Engineering–Resource Reservation Protocol-Traffic Engineering (RSVP-TE) Extensions (No. RFC 5151) (2008)
4. Das, S., Sharafat, A., Parulkar, G., McKeown, N.: MPLS with a simple OPEN control plane. In: Optical Fiber Communication Conference, p. OWP2. Optical Society of America, March 2011. [RL] Sutton, R.: Reinforcement learning: An introduction. MIT Press (1998)
5. Doria, A., Salim, J., Haas, R., Khosravi, H., Wang, W., Dong, L., Gopal, R., Halpern, J.: "Forwarding and Control Element Separation (ForCES) Forwarding Element Model," Internet Engineering Task Force (IETF) (2010)
6. Vasseur, J., Roux, J.: "Path Computation Element (PCE) Communication Protocol (PCEP)," Internet Engineering Task Force (IETF) (2009)
7. McKeown, N., Anderson, T., Balakrshman, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Tuner, J.: OpenFlow: enabling innovation in campus networks. Sigcomm Comput. Commun. **38**(2), 69–74 (2008)
8. McKeown, N.: Software-defined networking. INFOCOM Keynote Talk **17**(2), 30–32 (2009)
9. ONF, Software-Defined Networking: The New Norm for Networks (2012). https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf
10. Jain, S., Kumar, A., Mandal, S., Ong, J., et al.: B4: experience with a globally deployed software defined WAN. In: SIGCOMM 2013, 12–16 August 2013, Hong Kong, China (2013)
11. Agarwal, S., Kodialam, M., Lakshman, T.: Traffic engineering in software defined networks. In: Proceedings of the 32nd IEEE International Conference on Computer Communications, INFOCOM 2013, pp. 2211–2219, April 2013
12. Eppstein, D.: Finding the k shortest paths. SIAM J. Comput. **28**, 652–673 (1999)
13. https://www.opendaylight.org/downloads
14. Botta, A., Dainotti, A., Pescapè, A.: A tool for the generation of realistic network workload for emerging networking scenarios. Comput. Netw. **56**(15), 3531–3547 (2012). Elsevier
15. Knight, S., Nguyen, H.X., Falkner, N., Bowden, R., Roughan, M.: The internet topology zoo. IEEE J. Sel. Areas Commun. **29**(9), 1765–1775 (2011)