A Congestion Contribution-based Traffic Engineering Scheme using Software-Defined Networking

Dongjin Hong*, Jinyong Kim*, and Jaehoon (Paul) Jeong†
* Department of Electrical and Computer Engineering, Sungkyunkwan University, Republic of Korea
† Department of Interaction Science, Sungkyunkwan University, Republic of Korea
Email: {dong.jin, timkim, pauljeong}@skku.edu

***Abstract*—This paper proposes a Traffic Engineering (TE) scheme for routing in a target network topology that supports OpenFlow using a metric called congestion contribution and Software-Defined Networking (SDN). According to the network traffic volume has been increasing dramatically, it is clear that the congestion, which is caused by the imbalance of the traffic flow, can be occurred. Furthermore, an Interior Gateway Protocol (IGP) that is used for an Autonomous System (AS) also can cause congestion because it uses snapshot information of network topology to make the shortest path-based routing table. Due to the reasons mentioned above, much research on TE has been conducting to enhance the network performance. However, as an example of TE, the Multi-Protocol Label Switching (MPLS) using Resource Reservation Protocol (RSVP) is not flexible in reconfiguring the labeled path, and it has resource overhead that cannot be ignored. Thus, we propose a TE scheme using SDN to overcome the limitations in the target network. In order to show the feasibility of the proposed scheme, TE application predicts the bandwidth utilization for network devices in SDN networks. It also determines the near-optimized routing path for the maximization of the average bandwidth utilization with avoiding the congestion nodes. Our scheme can configure network devices (e.g., SDN switches) along the paths of traffic flows so that the SDN network can have maximum bandwidth utilization. Through experiments, it is shown that our proposed scheme can utilize the average bandwidth of the whole topology up to 66%.**

Keywords*—Routing, Congestion Contribution, Traffic Engineering, Software Defined Networking

I. Introduction

A lot of hardware such as network devices and many software including protocols form a huge computer network. Recently, the network traffic volume has been increasing dramatically due to the development of multimedia and technologies such as Internet of Things (IoT), cloud services, and so on [1]. However, the existing computer network environment was not designed for enduring the rapid increase of network traffic volume. Therefore, the rapidly increasing traffic volume is very likely to cause congestion that is made by the imbalanced traffic flow in traditional computer network environments.

Moreover, an Interior Gateway Protocol (IGP) used for an Autonomous System (AS), such as (RIP) [2], and (OSPF) [3], can also cause congestion in an AS [4] because it finds the shortest paths or best paths for routing using the snapshot of information for network topology. For RIP, the routing paths are decided by counting minimum hop from source to destination. On the other hand, for OSPF, the routing paths are determined by calculating the minimum cost based on the bandwidth of the link in the network topology. It means that those protocols can not estimate near future that traffic jam can occur at particular nodes.

Consequently, the researches on Traffic Engineering (TE) [5] have been conducting to enhance the performance regarding network traffic and resource level by Internet Engineering Task Force (IETF) [6] and other institutions. In other words, it needs to control the network traffic proactively toward an optimization. As an example of TE, there is a scheme for so-called Multi-Protocol Label Switching (MPLS) [7] using Resource Reservation Protocol (RSVP) [8]. The MPLS is a data forwarding method to support high-speed data transfer. And the RSVP is a protocol to support Quality of Service (QoS) by reserving resource. It is similar to control protocol such as Internet Control Message Protocol (ICMP) or Internet Group Management Protocol (IGMP). A suitable combination of the two techniques makes TE support. However, it may not only be difficult to find a provider that can support MPLS, but also the costs for constructing and operating the infrastructure for MPLS network are expensive compared to the public internet. And there are many difficulties in modifying the determined labeled traffic path because the MPLS is not flexible to change the labeled path. Finally, the resource overhead of each network device can not be ignored.

To resolve these problems, we propose a TE scheme for routing in an AS using Software-Defined Networking (SDN) [9] and OpenFlow [10] with congestion contribution proposed by vehicular network research area [11], which is a virtual metric to measure future congestion by the current and approaching vehicles in a road segment. SDN is a networking technology that user can control the network using the network devices separated into two planes: 1) the data plane and 2) control plane. The OpenFlow is a core component of SDN and a standard interface to communicate between two planes. The reason why we decide to use SDN and OpenFlow is that it allows flexible traffic path configuration and reduces the cost. Also, the congestion contribution is a concept that accumulates congestion weight to each link for future congestion level in the network topology. We will address congestion contribution in detail at following section. The proposed scheme consists of network devices which support OpenFlow, SDN controller, and the application for TE. The application manages congestion contribution matrix and determines the near-optimized routing path in terms of average bandwidth utilization and congestion level using the matrix. And then, the application makes the flows that can steer the traffic and configures them to the network devices that located at routing path. Thus, the application makes the traffic balanced.

Note that this paper is an enhanced version of our previous paper [12]. The rest of this paper is organized as follows Section II describes the problem formulation for our proposed scheme. Section III explains the design of our scheme. Section IV presents the implementation for our scheme in detail and the result of it. We finally conclude this paper along with future work in Section V.

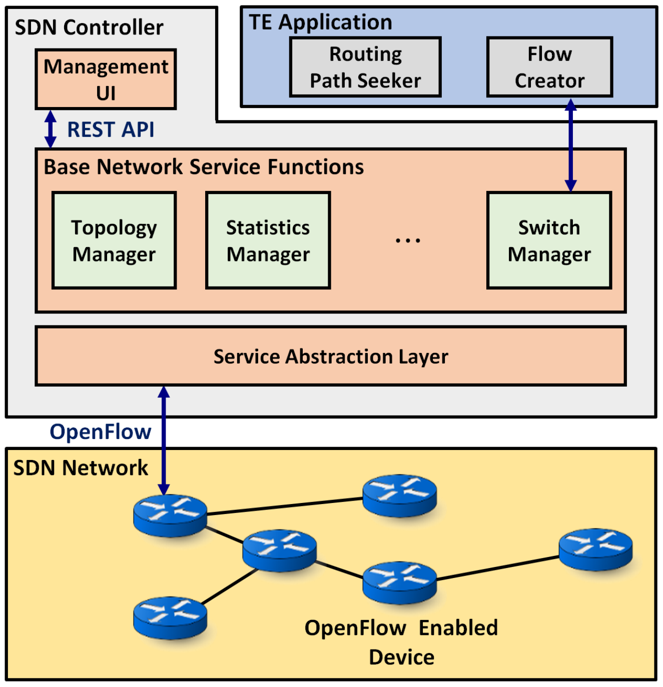
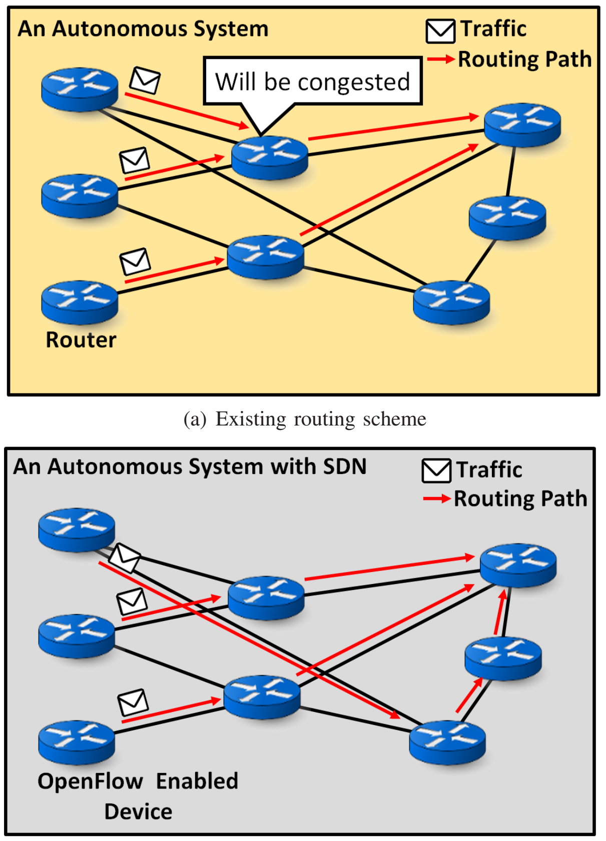


Fig. 1. SDN architecture with traffic engineering application



(a) Existing routing scheme

(b) Congestion contribution-based routing scheme

Fig. 2. Two routing scheme for handling the traffic

II. Problem Formulation

This section describes the goal, SDN architecture with TE application, and the concept of our TE scheme. We assume the target network topology is an AS in this paper. Given the target network topology, our goal is to determine a near-optimal routing path for each traffic. The proposed TE scheme aims to maximize the average bandwidth utilization per each link using congestion contribution formulation.

A. SDN Architecture with TE Application

This section presents the SDN architecture with TE application and its components. Fig. 1 shows the SDN architecture with TE application. The items explain each component in the SDN architecture with TE application as follows:

- **SDN Controller**: The SDN controller is a management node in the SDN network. Originally, if a packet arrives from the OpenFlow supported network device, the SDN controller creates a flow automatically based on the global view and configures it to the devices located at the routing path. In our scheme, however, the SDN controller just notifies it to the TE application to conduct TE and configures the flow forwarded by TE application. The SDN controller can also collect the statistics for the traffic via OpenFlow. The OpenDaylight [13] can be used for SDN controller.
- **Network Devices**: The network devices we used support OpenFlow. The OpenFlow supported network devices can forward the packets to other devices or a controller by processing the packets using the flow entry that contains a set of matching field. Thus, if a packet arrives that does not match any flow, they can notify it of the controller. They are the components constructing the target network topology. A tool called Mininet [14] can be used for creating network devices to construct the target network topology.
- **TE Application**: The application consists of the routing path seeker and the flow creator. The routing path seeker finds the TE routing path based on the congestion contribution matrix and Yen's algorithm [15]. And then, the application creates a flow using the routing path and forwards it to the SDN controller to configure the flow to the OpenFlow supported network devices located at the routing path.

B. Concept of Congestion Contribution-based TE

This section describes the concept of congestion contribution-based TE. Fig. 2(a) shows the existing routing schemes [2], [3] in an AS. They use the snapshot of information for a target network topology to make the shortest paths or best paths. Since the routing paths are updated when there is any change for the links, the congestion can be occurred when there are lots of traffic that have the duplicated links on their routing path. This method does not consider the global view of the network and it also does not consider the traffic for in near future.

On the other hand, Fig.2(b) shows the congestion contribution-based routing scheme in an AS with SDN. The scheme uses predicted information for the target network topology to make the routing path, considering the all of

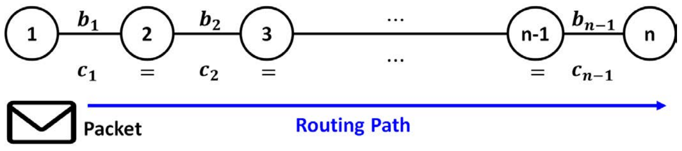


Fig. 3. Link congestion contribution model in network topology

traffic in the network. It can spread out network traffic by steering the routing path using SDN. For instance, the OpenFlow enabled device, which is located at upper left in Fig.2(b), detours the traffic to avoid the congestion. Thus, the congestion in the target network topology will be minimized and it leads to support QoS indirectly.

III. Design of Our Traffic Engineer Scheme

This section explains the link congestion contribution metric, network delay, and overview of our traffic engineering scheme with an algorithm.

A. Link Congestion Contribution Metric

This section introduces link congestion contribution metric that we modify and reuse the proposed concept by Jeong *et al.* [11]. Jeong *et al.* proposed a system to optimize vehicular traffic. A main idea of the system is congestion contribution formulation that is a concept of conditional delay reservation for each road segment in a vehicular network.

In this paper, however, we redefine this concept because the network traffic is quite different from the vehicular network. In the network topology, there is a link between two nodes and each link have a bandwidth. If a host generates traffic, it occupies a certain amount of the bandwidth from the host to destination. In other words, it is a congestion that affects the network topology by the host. Therefore, we add the occupied amount of the bandwidth on each link or subtract it from each link to predict congestion level for the network topology in near future.

We have a following setting for our TE scheme in the target network topology. Fig 3 shows the redefined link congestion contribution formulation in the network topology for a given routing path (denoted $P_{1,n}$). There is a maximum bandwidth for each link (denoted b_i) between two nodes that has a shape of a circle. The link congestion contribution value (denoted c_i) is defined as the reserved bandwidth of a link caused by the traffic to be generated. The packet (labeled in Fig. 3) moves by following the given $P_{1,n}$. In this paper, the design of link congestion contribution can be defined as the connection is opened, the link congestion contribution increases by reserving the bandwidth for the traffic. According to the congestion contribution metric, the equation in link congestion contributions of the links along $P_{1,n}$ is that $c_1 = c_2 = \cdots = c_{n-1}$. This equation means the reserved bandwidth is the same, corresponding to the bottleneck link that is the smallest b_i as the width of $P_{1,n}$. One of the congestion contribution value can be the same or less compared to b_i ($c_i \leq \min b_i$ in $P_{1,n}$). Let the adjacency matrix M store the congestion contribution values of each link for the target network topology.

B. Network Delay Prediction

This section presents the latency for an end-to-end (E2E) routing path based on the network delay and a routing path selection algorithm with the bounded detour latency for our scheme. The four main causes of network delay between E2E are as follows [16]:

1) Transmission Delay (denoted D_{tr}): The time for pushing the bit information in a packet to the link between two nodes.
2) Propagation Delay (denoted D_{pg}): The time for moving the bit information from a node to another node.
3) Processing Delay (denoted D_{pr}): The time for investigating a packet by a node.
4) Queuing Delay (denoted D_{qu}): The time for a packet wait in input and output queues.

Using the above delays and the number of nodes in an E2E (denoted n), the following equation for the total delay for the E2E can be derived.

$$D_{total} = (n-1)(D_{tr} + D_{pg} + D_{pr}) + (n)(D_{qu}) \quad (1)$$

C. Overview of Our Traffic Engineering Scheme

This section explains a TE algorithm and a procedure of our proposed TE scheme. The Algorithm 1 returns P_{te} that is a routing path for TE using input such as, the adjacency matrix M for congestion contribution values, the target network topology graph $G = (V, E)$ for a topology map where V is the set of vertices (i.e., network node) and E is the set of directed edges e_{ij} (i.e., link between two network nodes) for $i, j \in V$, source node $u \in V$, and destination node $v \in V$. In

Algorithm 1 TE Algorithm

```
1:  function GET_TE_ROUTING_PATH($M, G, u, v$)
2:      $P_{te} \leftarrow \emptyset$      ▷ $P_{te}$ will contain the list of OpenFlow
        supported network devices for TE routing path.
3:      $K \leftarrow$ Compute-$k$-Smallest-Delay-Paths($G, u, v$)      ▷
        compute the next $k$ smallest delay paths arranged in
        nondecreasing order by Yen's $k$-shortest-path algorithm.
4:      $C_1 \leftarrow$ Compute-Path-Congestion($M, K, 1$) ▷ compute
        the maximum congestion value for the 1st path in $K$.
5:      $min \leftarrow 1$
6:      $n \leftarrow$ Count-Path-Numbers($K$) ▷ count the number of
        paths in $K$.
7:      for $i \leftarrow 2, n$ do
8:          $C_i \leftarrow$ Compute-Path-Congestion($M, K, i$)      ▷
        compute the maximum congestion value for the $i$th path
        in $K$.
9:          if $C_i < C_{min}$ then
10:             $min \leftarrow i$
11:         end if
12:     end for
13:     $P_{te} \leftarrow$ Get-Path($K, min$) ▷ get the $min$th path in $K$.
14:     return $P_{te}$
15: end function
```

line 2, P_{te} is allocated to store the list of OpenFlow supported network devices for TE routing path. In line 3, Compute-k-Smallest-Delay-Paths computes k shortest paths in terms of the

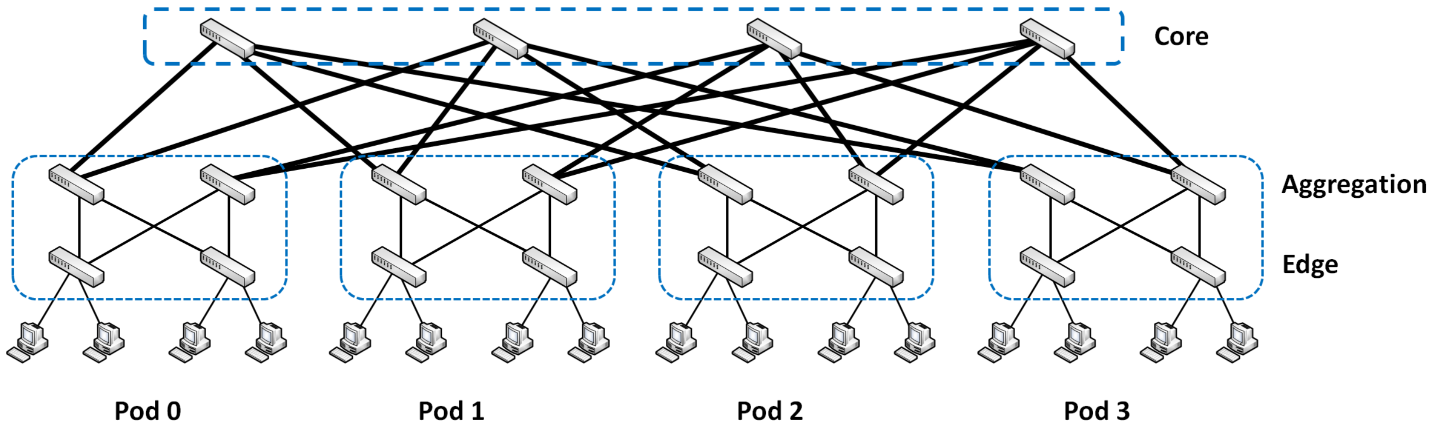


Fig. 4. The target topology called k-array fat tree (e.g., k=4)

network delay. The paths can be computed by *Yen's k-shortest-path algorithm* [15] with equation 1. In line 4, the maximum link bandwidth utilization ratio between the 1st path in K is stored in C_1 to compare the congestion level at line 9. Line 5 stores the index of 1st path. Line 6 counts the number of the available shortest paths in K to use for-loop in lines 7-12. In lines 7-12, it finds the index of minimum link bandwidth utilization ratio among the K-1 paths until escaping the for-loop. Line 13 updates P_{te} corresponding to the path that has the minimum ratio for the whole target network topology. If there is no such path among K-1 path, the 1st path in K is returned. And the TE procedure is organized as follows:

1) If there is a new connection detected at the edge layer node, the node notifies it of the SDN controller.
2) The SDN controller requests a routing path for TE to the TE application.
3) Routing path seeker located at TE application finds the routing path for TE using TE algorithm.
4) Routing path seeker returns the path discovered to flow creator.
5) According to the path, flow creator creates a flow.
6) Flow creator configures the created flow to the devices located at the discovered path.
7) The matrix M is updated according to the congestion contribution value.
8) The traffic moves along with the flow until the connection is maintained.
9) If the edge layer node detects the connection is closed, flow creator deletes the flow corresponding to the connection.
10) Lastly, the matrix M is updated according to the congestion contribution value.

By reserving the bandwidth with the above algorithm and procedure, the traffic can avoid the congested network device by detouring to the destination when a new connection detected.

IV. Implementation

This section describes how to implement the proposed scheme, including the target network topology and congestion contribution metric. We use an open source project called OpenDaylight [14] for SDN controller and a tool called Mininet [15] for our implementation.

A. Target Network Topology

The target network topology is addressed in this section. A. Mohammad *et al*. [17] proposed an architecture for data center network as shown in Fig. 4. The k-array fat tree consists of hosts and k-port switches. It can support $k^3/4$ hosts. It has three layers called 1) core, 2) aggregation, and 3) edge layer. It also has k pods that include aggregation and edge layers (each layer has $k/2$ switches). In the edge layer, the switch is connected to $k/2$ hosts. In the aggregation layer, the switch is connected to the edge layer switches in the same pod. In the core layer, there are $(k/2)^2$ switches and each i^{th} port of the core switch is connected to the switch located at the aggregation layer in the pod i. The source node and destination node are selected randomly among the hosts, and the traffic is generated sequentially.

The reasons why we chose the k-array fat tree are that it is possible to consider a fat tree as an AS and it is well-known. We construct the fat tree (k=4) using Mininet and configure the maximum bandwidth for each layer as follows. (core layer = 80Mbit/s, aggregation layer = 40Mbit/s, and edge layer = 20Mbit/s)

B. Congestion Contribution Metric

As we explained above, we add congestion contribution value on corresponding link when the connection is open and subtract it when closed. In our performance evaluation, we only consider reserving the maximum bandwidth of each routing path for the congestion contribution value in the matrix M explained above. This method cannot achieve perfect optimization because the utilized bandwidth of each routing path can be the same or less than the maximum bandwidth. However, not only is there no perfect way to calculate the available bandwidth but also it is impossible to predict the actual traffic volume when the traffic occurred. This limitation is left as our future work.

C. Flow Control

This section presents how to control the traffic in detail for OpenDaylight case. The flow control is managed by the flow

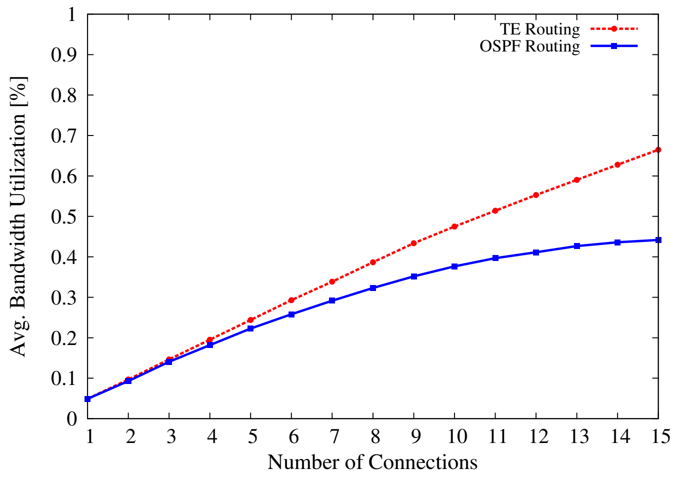


Fig. 5. Average Bandwidth Utilization by Connections

creator in performance evaluation. After the TE routing path is found by the routing path seeker, the flow creator makes Extensible Markup Language (XML) [18] format based flow using the found TE path. The made flow is stored at database and then it is pushed to the SDN controller via RESTCONF [19]. The flow creator can also push the removing flow to the SDN controller via RESTCONF for the TE path when the connection is closed.

D. Performance Result

This section presents the result of our scheme base on the performance evaluation. We compare our proposed TE routing scheme with an original routing scheme, such as OSPF that uses the weight based on the link bandwidth Dijkstra's shortest path algorithm, in the target network topology. For the test, a host generates the traffic from the host to another host under the target network topology. The two hosts are selected randomly among other hosts. The traffic is generated sequentially and a connection caused by the hosts is maintained until the end of the experiment. As shown in Fig. 5, it is clear that our TE routing scheme has better average bandwidth utilization by connections compared to the OSPF routing, because the maximum difference of average bandwidth utilization by connection between our scheme and OSPF scheme is 22%.

V. Conclusion

In this paper, we propose a Traffic Engineering (TE) scheme for routing in an autonomous system using Software Defined Networking and OpenFlow with congestion contribution metric. Finding the TE path can make the traffic balanced by distributing it to the network devices supporting OpenFlow in the target network topology. In order to show the feasibility of the proposed scheme, we tested the original routing scheme and our TE routing scheme in the target network topology. It is shown that our proposed scheme can utilize the average bandwidth of the whole target network topology up to 66%. This result is clear that the TE routing scheme has better performance than original one by showing the difference. But we realized our scheme has a limitation that we did not consider actual bandwidth utilization for each link. To resolve it, we will improve the congestion contribution metric by calculating the case of the utilized bandwidth of each routing path to support better traffic distribution for future work.

Acknowledgments

This research was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (Ministry of Science and ICT, MSIT) (No.2016-0-00078, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning). This research was also supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2018-2017-0-01633) supervised by the IITP(Institute for Information & communications Technology Promotion). Note that Jaehoon (Paul) Jeong is the corresponding author.

References

[1] CISCO, "The zettabyte era: trends and analysis," CISCO White Paper, 2017.
[2] C. Hedrick, "Routing information protocol," IETF RFC 1058, Jun. 1988.
[3] J. Moy, "OSPF version 2," IETF RFC 2328, Apr. 1998.
[4] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for Traffic Engineering Over MPLS," IETF RFC 2702, Sep. 1999.
[5] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "Overview and Principles of Internet Traffic Engineering," IETF RFC 3272, May. 2002.
[6] IETF, "The Internet Engineering Task Force," accessed 2018. [Online]. Available: https://ietf.org
[7] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," IETF RFC 3031, Jan. 2001.
[8] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin, " Resource ReSerVation Protocol (RSVP)," IETF RFC 2205, Sep. 1997.
[9] ONF, "Software-defined networking: the new norm for networks," ONF White Paper, 2012.
[10] OpenFlow Specification, accessed 2018. [Online]. Available: https://www.opennetworking.org/software-defined-standards/specifications/
[11] J. Jeong, H. Jeong, E. Lee, T. Oh, and D. H. C. Du, "SAINT: Self-Adaptive Interactive Navigation Tool for Cloud-Based Vehicular Traffic Optimization," in IEEE Transactions on Vehicular Technology, vol. 65, no. 6, pp. 4053-4067, Jun. 2016.
[12] D. Hong and J. Jeong, "A Traffic Load Balancing Scheme using K-Shortest Path Algorithm and Congestion Contribution based on SDN," in KICS-2018-Summer, vol. 66, pp. 411-412, Jun. 2018.
[13] OpenDaylight, "Open Source SDN Platform," accessed 2018. [Online]. Available: http://www.opendaylight.org/
[14] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, pp. 19:1–19:6, Oct. 2010.
[15] J. Y. Yen, "Finding the K shortest loopless paths in a network," Manage. Sci., vol. 17, no. 11, pp. 712-716, Jul. 1971.
[16] J. F. Kurose and K. W. Ross, Computer Networking: A Top–Down Approach, 6th ed., Pearson, 2012.
[17] A. Mohammad, L. Alexander, and V. Amin, "A Scalable, Commodity Data Center Network Architecture," in Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication, pp. 63–74, Oct. 2008.
[18] W3C, "Extensible Markup Language (XML) 1.0 (Fifth Edition)," W3C Recommendation, Nov. 2008.
[19] A. Bierman, M. Bjorklund, and K. Watsen, "RESTCONF Protocol," IETF RFC 8040, Jan, 2017.