Traffic Engineering in SDN/OSPF Hybrid Network

Yingya Guo*‡, Zhiliang Wang†‡, Xia Yin*‡, Xingang Shi†‡,and Jianping Wu*‡
*Department of Computer Science and Technology, Tsinghua University
†Institute for Network Sciences and Cyberspace, Tsinghua University
‡Tsinghua National Laboratory for Information Science and Technology (TNLIST)
Beijing, P.R. China
Email:{guoyingya, wzl, yxia}@csnet1.cs.tsinghua.edu.cn, {shixg, jianping}@cernet.edu.cn

Abstract—**Traffic engineering under OSPF routes along the shortest paths, which may cause network congestion. Software Defined Networking (SDN) is an emerging network architecture which exerts a separation between the control plane and the data plane. The SDN controller can centrally control the network state through modifying the flow tables maintained by routers. Network operators can flexibly split arbitrary flows to outgoing links through the deployment of the SDN. However, SDN has its own challenges of full deployment, which makes the full deployment of SDN difficult in the short term.**

In this paper, we explore the traffic engineering in a SDN/OSPF hybrid network. In our scenario, the OSPF weights and flow-splitting ratio of the SDN nodes can both be changed. The controller can arbitrarily split the flows coming into the SDN nodes. The regular nodes still run OSPF. Our contribution is that we propose a novel algorithm called SOTE that can obtain a lower maximum link utilization. We reap a greater benefit compared with the results of the OSPF network and the SDN/OSPF hybrid network with fixed weight setting. We also find that when only 30% of the SDN nodes are deployed, we can obtain a near optimal performance.

Index Terms—**traffic engineering; OSPF; SDN; partial deployment; hybrid network**

I. INTRODUCTION

Open Shortest Path First (OSPF) protocol is one of the most widely used Interior Gateway Protocols (IGPs) in recent years [1]. In OSPF protocol, each link is assigned a weight (or a cost) by the operators and the shortest paths between the sources and the destinations are computed in terms of these weighted links within an Autonomous System(AS). The traffic directed to a same destination arrives at a router node and is evenly split across the next hops on the equal cost shortest paths. The multiple flows for a given destination are split across the multiple next hops to balance the load. In the traditional OSPF network, all router nodes support OSPF protocol. Each router maintains a forwarding table and always chooses the nearest next hop to forward packets. In order to get better network performance in traffic engineering, we should optimize the OSPF weights to balance the flows. Much work has been done previously on searching for the optimized weight setting under OSPF protocol. The problem is proven to be NP-complete in [2]. Therefore, many heuristic algorithms are proposed to solve the problem. The refined local search heuristic algorithms [3] [4] are thus proposed to search for the optimized weight setting. The algorithm in [3] allows us to cope with 50%-110% more demand than other weight setting algorithms, i.e, InvCapOSPF, UnitOSPF, L2OSPF, RandomOSPF. Other heuristic search algorithms, such as genetic algorithm [5], simulated annealing algorithm [6], have also been used in searching for the optimal weight setting. These heuristic algorithms can improve the network performance but can only obtain the local optimal solutions, which may be far from the global optimal solutions [7].

Software Defined Networking (SDN) is an emerging network architecture that attracts people's attention in recent years. In this new architecture, its control plane and data plane are separated. The SDN controller is a logically-centralized control plane and takes charge of the flows in the SDN network. It can flexibly control the flows in the network and assign arbitrary flows to the outgoing links of the SDN nodes. Some techniques can be exploited to direct the different flows to the same destination to different next hops and the traffic is split on the network layer. The deployment of SDN in the network provides a convenient and effective way to do the traffic engineering and can improve the network performance on a large scale. Microsoft designs a Software-driven WAN which interconnects the data centers and achieves a high utilization of the network throughput [8], which is near optimal. Google also exploits SDN and runs many links at the utilization of near 100% [9]. The advantages of SDN provide an incentive to the transition of SDN network. However, fully deploying the SDN in the network is by no means an easy job. We may encounter economical, organizational and technical challenges [10]. As a result, a full deployment of SDN, i.e, all the router nodes support SDN and can be centrally controlled by the controllers, in the network will not work out in the short term. Deploying SDN in the network incrementally can be a better choice. In this paper, we consider the scenario of partial SDN deployment, i.e, SDN/OSPF hybrid network.

Our work mainly focus on how to adjust the weight setting of the entire network to balance the flows coming out of the regular nodes and how to split the flows that aggregate at the SDN nodes so that the maximum link utilization of the whole network can be minimized. In this paper, we are original in doing the traffic engineering in the hybrid network scenario that the weight setting of the network and the splitting ratio of the SDN nodes can both be changed at the same time. The algorithm SOTE (SDN/OSPF Traffic Engineering) we propose in this paper is novel in minimizing the maximum link utilization in this scenario. We obtain a better network performance compared to the traffic engineering in traditional OSPF network [3] and the hybrid network with fixed weight setting (WFSOTE) [11]. Moreover, we discuss the number of SDN nodes to deploy that can best meet our demands. We deploy the network incrementally and find that when 30% of the nodes are deployed, we can achieve a near optimal performance.

The rest of the paper is organized as follows. We introduce the related work in Section II. Section III depicts the scenario of our hybrid network and formulates the problem as a mathematical problem. We also analyze the complexity of the problem in this section. Section IV is the introduction of our algorithm, which solves the problem of optimizing the weight setting and the splitting ratio of the flows at the SDN nodes. Section V presents several experiments, which compare the maximum link utilization and CPU time of our algorithm with the algorithms in [3] and [11]. We also talk about the locations and number of SDN nodes to deploy in this section. Section VI is the conclusion and summary of this paper. It also outlines the direction for the future work.

II. RELATED WORK

In this section, we present some related work on traffic engineering in recent years.

[3] firstly proposes an IP-based intra-domain traffic engineering solution by setting link weights of intra-domain routing protocol (OSPF or IS-IS) to achieve traffic engineering objectives. This solution makes a constraint that the traffic is evenly split among all the next hop routers on shortest paths from source node to destination node as OSPF and IS-IS have done. They prove that optimizing the setting of link weights is NP-hard and propose a local search heuristic solution. [7] breaks down the constraint of evenly splitting traffic. In this solution, a subset of the next hops for each routing prefix is selected to evenly forward the traffic, which emulates the uneven splitting of traffic to achieve a near optimal solution. This solution needs complex configuration for each routing prefix. Recently, Xu et al. present new link-state routing protocols DEFT, PEFT to achieve optimal traffic engineering in [12] and [13]. In this solution, traffic is not only carried by multiple shortest paths but also longer paths, by splitting traffic over multiple paths with an exponential penalty on longer paths. [14] models optimal intra-domain traffic engineering as the utility maximization of multi-commodity flows with the generic objective function. This solution extends OSPF routing protocol by adding a second weight for each link to support arbitrary splitting of traffic over all shortest paths. Generally speaking, in the most existing IP-based intra-domain traffic engineering solutions, link state routing protocol needs configuration or modifications. In this paper, we utilize the flexibility of SDN and control the flows coming out the SDN nodes.

Traffic engineering in hybrid network scenarios is definitely a hot topic. [15] introduces a MPLS/OSPF hybrid network. It formulates the problem as a linear programming multi-commodity flow problem. It avoids the drawbacks of MPLS and OSPF and provides a good solution to traffic engineering. We can flexibly split the traffic among the multiple tunnels using MPLS. However, MPLS traffic engineering lacks of global view and it distributes the flows regardless of other flows. Moreover, they need to establish the MPLS tunnels, i.e, Label Switching Paths (LSP) to forward the packets. In our SDN/OSPF traffic engineering, SDN nodes have a global view and can flexibly control the flow in the network. [11] is the first paper to address network performance issues in an incrementally deployed SDN network. It formulates the traffic engineering as a linear programming problem and refines a polynomial algorithm to solve it. In [11], the weight setting is fixed and all link weights are set to 1. The network performance is thus limited. In our research, the weight setting is undetermined and we should search for the optimized weight setting exploiting heuristic algorithm.

III. PROBLEM FORMULATION

In this section, we firstly depict the SDN/OSPF hybrid network scenario. Then we formulate the problem as mathematical problem and analyze the complexity of the problem.

A. Hybrid Network Scenario

In our SDN/OSPF hybrid network, we assume that the SDN nodes are a subset of the nodes in the network. SDN nodes are the hybrid routers that run traditional network protocol stack and at the same time support centrally control from the SDN controller. The remaining are the regular nodes that only run traditional network protocol stack. Therefore, all the nodes support OSPF protocol. The regular nodes can compute the shortest paths and route the flows in the entire network. The weight setting of the network have an impact on the shortest path selection between the regular nodes and the destinations, but they do not have an impact on the splitting ratio of the flows out the SDN nodes.

Fig.1 shows an example of the SDN/OSPF hybrid network scenario. It is a research and education network of America with 11 nodes, 28 links. In this topology, nodes 3,4,9 are SDN nodes and are controlled by the controller. The others are regular nodes and routes the flows along the shortest paths.

B. Optimization Problem Definition

We are given an undirected network graph $G = (V, A)$ (V is the vertexes set, A is the arcs set), capacity matrix C (C_{ij} denotes the capacity of link $(i, j) \in A$) and traffic matrix TM (TM_{ij} denotes the estimated traffic demands from i to j). ω denotes the weight setting matrix and ω_{ij} is an integer chosen from [1,20]. CN is the SDN nodes set. $In(v)$, $v \in V$ denotes the edge sets that there are traffic flowing into the node V. $Out(c)$, $c \in CN$ represents the edge sets that there are traffic flowing out of the node c. $x_e(\omega)$ denotes that the splitting flow on the edge $e \in Out(c)$ with the weight setting ω. $f_e(\omega)$ denotes the flow on the edge e with the weight setting ω. $g_e^{vt}(\omega)$ denotes the flow on the edge e from v to t with the

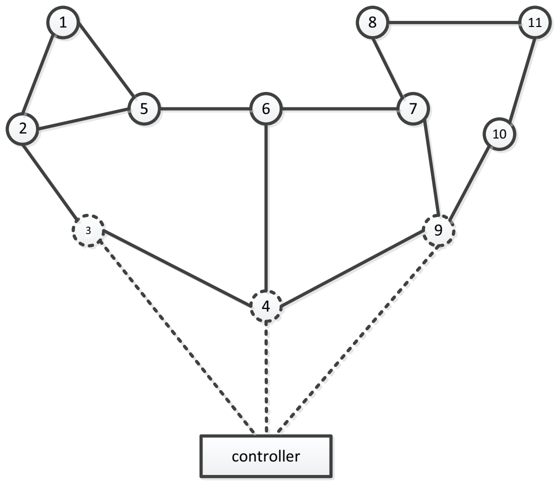

Fig. 1. A hybrid network scenario

weight setting ω. U is the maximum link utilization of the network. We assume that the traffic matrix and the topology of the network are steady for a short period. The locations and number of the SDN nodes are fixed and are determined by the experiments in section V. By adjusting the weight setting matrix ω and the splitting ratio of the flows x_i out the SDN nodes, we intend to minimize the maximum link utilization of the network, i.e., minimize U. We can formulate the problem as following.

$$\textbf{minimize} \quad U$$

$$\sum_{e \in In(t)} f_e(\omega) = \sum_{v \in V} TM_{vt} \quad t \in V \tag{1}$$

$$\sum_{e \in In(c)} f_e(\omega) + TM_{ct} = \sum_{e \in Out(c)} x_e(\omega) \quad c \in CN, t \in V \tag{2}$$

$$\sum_{v,t \in V} g_e^{vt}(\omega) \leq UC(e) \quad e \in A \tag{3}$$

$$\omega \in \mathbf{Z} \tag{4}$$

$$x_e(\omega) \geq 0 \quad e \in A \tag{5}$$

(1) denotes that the traffic demands are meet in the flow routing. (2) denotes that the flows flow into the node equals the flows that flow out of this node. (3) denotes that the total flows on a edge cannot exceed its capacity. (4)-(5) denote that the weights are integers and the flow on a link must be non-negative.

C. Problem Complexity

When the weight setting is fixed, the problem boils down to a multi-commodity linear programming problem, which can be solved in polynomial time [16]. However, the weights are integers and undetermined, which adds up the complexity of the problem. It has been proved that optimizing of weight setting is a NP-complete problem [2]. Our proposed model can be reduced to the weight optimization problem in [3] if all the traffic is split in a pre-defined manner, i.e, all traffic is split to the shortest path and none to the longer paths. So the complexity of the entire problem is easily proved to be NP-complete. Due to the space limitations, the detailed proof is omitted.

IV. ALGORITHM FOR TRAFFIC ENGINEERING IN HYBRID NETWORK

Given the NP-completeness of the problem, we resort to heuristic algorithms. In this section, we firstly introduce our novel algorithm called SOTE, which is a local search heuristic algorithm. Then we give a simple example to illustrate our algorithm in detail.

A. A novel algorithm SOTE

Algorithm 1: SOTE

```
Input: $G = (V, A)$ , $CN$ , $C$ , $TM$
Output: $U$
1  Initial weight setting matrix $W$ , $U$ ;
2  $currutil$ = floydwarshall $(G, W, TM)$ ;
3  $bestutil = currutil$ ;
4  $currweights = W$ ;
5  $bestweights = W$ ;
6  foreach iteration times increase by one do
7      $currweights$ =neighbour_search $(G, W)$;
8      $currutil$ =Splitting_ratio
       $(currweights, CN, TM, C, V, A)$;
9      if $currutil < bestutil$ then
10         $bestutil = currutil$ ;
11         $bestweights = currweights$ ;
12 $U = bestutil$ ;
```

Algorithm 1 is SOTE. At the core of the algorithm is the neighbour_search function (line 7) and the Splitting_ratio function (line 8). We first use Floyd algorithm to find the shortest paths between the two nodes and route the traffic demands to obtain the current link utilization (line 2). Then we set the current weight setting and link utilization to be the best weight setting and utilization (line 3-5). In each iteration of local search (line 6-11), we change the weights of the links (line 7). When we obtain a weight setting, we use the splitting_ratio function to optimize the splitting ratio of the flows coming out the SDN nodes (line 8). We evaluate the maximum link utilization with the current weight setting and the optimized splitting ratio (line 9-11). We iterate to get the solution finally.

Algorithm 2 shows when the weight setting is fixed, how to optimize the splitting ratio of the SDN nodes. We firstly construct Directed Acyclic Graphs (DAGs) for treating every node d as a destination node (line 2). DAG in the algorithm denotes the matrix that stores all possible paths between the

Algorithm 2: Splitting_ratio

```
Input: local optimal weight setting $currweights$, $CN$, $TM$, $C$, $V$, $A$
Output: $U$
1 foreach vertex $d \in V$ do
2     $DAG$ =Dijkstra_Shortest_Path($currweights, d$) ;
3     foreach arc $(i, j)$ in outgoing_links($SDN_i$) do
4         $DAG(i, j) = 1$;
5         if check_loop($DAG$) == 1  $DAG(i, j) = 0$;
6     foreach vertex $v$ in Topological_Sort($V$) do
7         Route_Flow ($v$);
8 $expr$ = Multi_Commodity ($A, V$);
9 cplex_solve ($expr$);
```

nodes to a specific node d. $DAG(i, j) == 0$ if there is no link between i and j in the DAG and $DAG(i, j) == 1$ if there is a link between i and j in the DAG. When constructing a DAG, we first choose a node and find all the shortest paths to this specific node with Dijkstra_Shortest_Path function. The shortest paths tree ensures there are no loops. Then we add all the edges that are outgoing links of the SDN nodes to the DAG (line 4). If there is a loop after adding a link, we remove the link and ensure there is no loops (line 5). When the DAG is constructed, we can route the traffic demands in Route_Flow function. We choose the node according to the topologically sort order so that all the flows passing through the current node are calculated before assigning the flows to the next hops. Then we route the flows to all outgoing links in DAG (line 6-7). If the node chosen is a SDN node, the splitting flows are denoted as x_i and are added to the next hops. The flows to the same destination are injected to the next hops in an arbitrary proportion. We can exploit the hash function to map flows to different next hops to realize the splitting of the flow; if the node chosen is a regular node, the flows are equally split among the shortest paths and added to next hops. In this way, we have all the flows from the chosen node to node d routed (line 7). We repeat this process and route the flow of every node in topologically sort order. As the weight setting is fixed, our problem boils down to a multi-commodity problem (line 8). We can list the expressions and employ the CPLEX to solve the optimized problem (line 9). Cplex solves LP problems using dual simplex algorithm and the complexity of the algorithm is in polynomial time.

The complexity of the floydwarshall, neighbour_search are both $O(n^3)$ and the complexity of Dijkstra_Shortest_Path is $O(n^2)$. The complexity of other functions are no more than $O(n^3)$. Therefore, the overall complexity of the algorithm is $O(n^3)$.

B. A simple example of the algorithm

We use Fig.2 as a simple example to illustrate our algorithm. The graph on the top is a simple network topology. It has three nodes and node 0 is a SDN node. We first search the neighbour of our present weight setting. We assume that the numbers on the links are the weights of the links for the moment and in this example, they are 2,3,4, respectively. When the weight setting is determined, we construct the DAGs to node 0,1,2, respectively, as shown in the bottom three graphs in Fig.2. The solid lines are the shortest path we find exploiting the Dijkstra algorithm. Because the flows pass through the SDN nodes can flexibly choose the outgoing links regardless of their weights, we use dotted lines to represent the extra possible links that the SDN node can split the flows to. As long as these extra links cause no loops in the network, we can add them into the DAG. Then in each DAG, we can calculate the flows on each link corresponding to the nodes in topologically sort order. We take the DAG in the middle for an example. The result of the topologically sort is 0,2,1. We choose 0 first and it is a SDN node. x_1, x_2 denote the splitting ratio of node 0 on link $(0, 1)$ and link $(0, 2)$. We can get the equation $x_1 + x_2 = TM_{01}$. At the same time, the flows are routed to node 1,2 and we add the flows x_2 to the traffic demands TM_{21}. Therefore, all the flows originated from node 0 have been assigned and the flows that has to be transferred from node 2 to node 1 are $TM_{21} + x_2$. Then we choose the second node 2 in the order and it is a regular node. We route the flow to the next hop on the shortest paths to node 1. The flows on the link $(2, 1)$ are $TM_{21} + x_2$ and all the demands to node 1 are meet. The procedures are the same when we calculate the flows on the links in the other two DAGs. We add up the flow load on each link and restrict the flow load as (1)-(5) shows. The specific equations and inequations are listed in Fig.2. Finally, we take advantage of CPLEX to solve the linear programming problem.

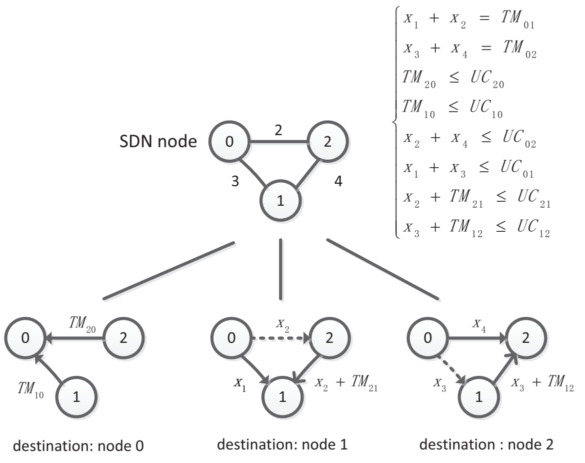

Fig. 2. A simple example of the algorithm

V. EXPERIMENTS AND EVALUATION

In this section, we conduct the simulation experiments. The topologies are shown in TABLE I. They are the same with the topologies in [3], which are inferred by Rocketfuel [17]. The traffic matrices are randomly generated. Due to space limitations, we choose the first three topologies to conduct the experiments. The results are similar for the remaining two topologies.

TABLE I
THE TOPOLOGIES FROM ROCKETFUEL

Name	Nodes	Links
Abovenet	17	74
Ebone	18	66
Exodus	21	72
Sprint	27	126
Tiscali	28	132

A. *Number of the SDN nodes*

Firstly, We should determine the number of SDN nodes in the hybrid network before the experiments of comparison. We decide to deploy the SDN in the network with greedy algorithm. We choose to deploy the nodes whose outgoing link has the maximum link utilization every time. We plot the curves that the maximum link utilization varies with the increasing of the deployment rate and the results are shown in Fig.3.

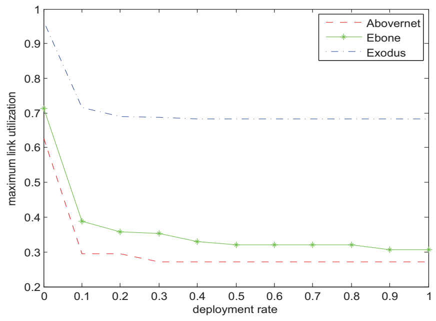

Fig. 3. The number of SDN nodes in the network

From Fig.3, we can obtain that the maximum link utilization decreases with the increasing of the deployment rate. When the deployment rate ranges from $0\%-30\%$, the maximum link utilization is decreasing rapidly. When the deployment rate is greater than 30%, the variation of maximum link utilization becomes relatively flat. We demonstrate that when deploying the SDN at a small rate, we can obtain the most of the benefit.

B. *simulation experiments*

From the aforementioned experiments, we determine that the deployment rate of SDN is 30% in SOTE and WFSOTE. We deploy the nodes in the network according to their outgoing links with maximum link utilization. To compare with the maximum link utilization in OSPF, we carry out the experiments in [3]. We implement the SOTE and all parameters in the heuristic algorithm are set the same with the parameters in [3]. The iteration times are set to 5000 (SOTE5000), 2000 (SOTE2000), 1000 (SOTE1000), respectively and the weights of links are integers in $[0, 20]$. We also conduct the comparison experiments in the scenario of [11] with fixed links weight setting (WFSOTE). We carry out twenty experiments with different traffic matrices and draw the Cumulative Distribution Function (CDF) graphs of the maximum link utilization. The results are shown in Fig.4, Fig.5 and Fig.6.

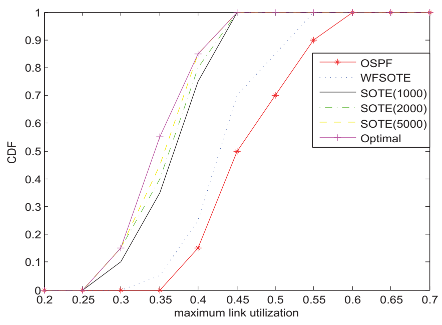

Fig. 4. The CDF curves of Abovenet

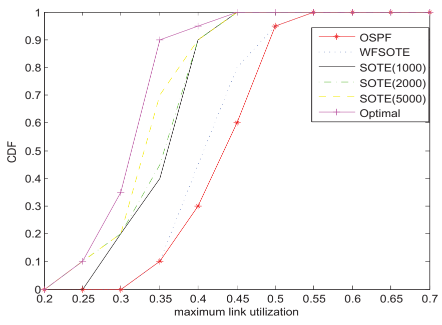

Fig. 5. The CDF curves of Ebone

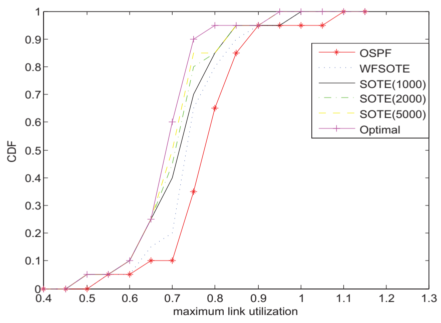

Fig. 6. The CDF curves of Exodus

As the above figures illustrate, our algorithm SOTE in the figures obtains a lower maximum link utilization compared with the other two algorithms. SOTE(5000) can improve the overall maximum link utilization by 5% and our algorithms with different iteration times do not have much effect on the maximum link utilization. The maximum link utilization of our proposed algorithm is near Optimal with a deployment rate of 30%.

C. CPU time

We also conduct the experiments to measure the CPU time of our algorithm. The experiments are done on a computer with 2.6GHz Intel Core 4 CPU and 4GB memory. The CPU time of the experiments on three different topologies are listed in table II. We record the CPU time of 20 experiments and calculate the average CPU time of three topologies.

TABLE II
THE CPU TIME OF DIFFERENT ALGORITHMS

Algorithms	Abovenet	Ebone	Exodu
OSPF	8.8s	13.3s	16.8s
SOTE(1000)	10.3s	15.2s	22.4s
SOTE(2000)	19.7s	27.9s	37.4s
SOTE(5000)	49.1s	57.5s	87.4s
WFSOTE	16.6ms	21.8ms	24.8ms

As shown in table II, WFSOTE without iterations is generally faster than other algorithms. SOTE(5000), SOTE(2000) can obtain a low maximum link utilization but occupy more CPU time. We should set the iteration times to be 1000 where we can get a low maximum link utilization with shorter CPU time. Moreover, in short terms, we can choose to timely adjust the splitting ratio of the SDN nodes according to the traffic demands without optimizing the link weights. In the long terms, we can optimize the OSPF weights together with adjusting the splitting ratio of the SDN nodes. In this way, we can fully utilize the potential of the SDN in the network.

VI. CONCLUSION

The SDN/OSPF hybrid network traffic engineering is a popular problem that raises people's attention worldwide. It deviates from the traditional traffic engineering scenario, where the flows are always routed along the shortest paths. The emerging of SDN provides a new method to solve the traffic engineering problem. It can centrally control the flows that directed to the outgoing links of the nodes, which is similar with the model of multi-commodity. In this paper, we concentrate on a new hybrid network scenario and propose a novel algorithm SOTE to solve the traffic engineering problems in this scenario. Compared with other traffic engineering algorithms, our algorithm performs better and obtains a lower maximum link utilization. We also conduct the experiments to show that a small portion deployment of SDN can reap a great benefit, which is near the benefit of the full deployment of SDN in the network.

In the future work, we will carry out the experiments on the testbed and consider more hybrid network scenarios.

VII. ACKNOWLEDGEMENTS

This work is partially supported by the National Natural Science Foundation of China (Grant No. 61202357, No. 61161140454), the Project for 2012 Next Generation Internet technology research and development, industrialization, and large scale commercial application of China (No. 2012 1763) and the National High Technology Research and Development Program of China (863 Program) (No. 20 11AA010704).

REFERENCES

[1] B. Fortz and M. Thorup, "Optimizing ospf/is-is weights in a changing world," *IEEE journal on selected areas in communications*, vol. 20, no. 4, 2002.

[2] Y. Wang, Z. Wang, and L. Zhang, "Internet traffic engineering without full mesh overlaying," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1. IEEE, 2001, pp. 565–571.

[3] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing ospf weights," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2. IEEE, 2000, pp. 519–528.

[4] ——, "Increasing internet capacity using local search," *Computational Optimization and Applications*, vol. 29, no. 1, pp. 13–48, 2004.

[5] M. Ericsson, M. G. C. Resende, and P. M. Pardalos, "A genetic algorithm for the weight setting problem in ospf routing," *Journal of Combinatorial Optimization*, vol. 6, no. 3, pp. 299–333, 2002.

[6] M. Pióro, A. Szentesi, J. Harmatos, A. Jüttner, P. Gajowniczek, and S. Kozdrowski, "On open shortest path first related network optimisation problems," *Performance evaluation*, vol. 48, no. 1, pp. 201–223, 2002.

[7] A. Sridharan, R. Guerin, and C. Diot, "Achieving near-optimal traffic engineering solutions for current ospf/is-is networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 13, no. 2, pp. 234–247, 2005.

[8] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*. ACM, 2013, pp. 15–26.

[9] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*. ACM, 2013, pp. 3–14.

[10] S. Vissicchio, L. Vanbever, and O. Bonaventure, "Opportunities and research challenges of hybrid software defined networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 70–75, 2014.

[11] S. Agarwal, M. Kodialam, and T. Lakshman, "Traffic engineering in software defined networks," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 2211–2219.

[12] D. Xu, M. Chiang, and J. Rexford, "Deft: Distributed exponentially-weighted flow splitting," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*. IEEE, 2007, pp. 71–79.

[13] ——, "Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering," *IEEE/ACM Transactions on Networking (TON)*, vol. 19, no. 6, pp. 1717–1730, 2011.

[14] K. Xu, H. Liu, J. Liu, and M. Shen, "One more weight is enough: Toward the optimal traffic engineering with ospf," in *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*. IEEE, 2011, pp. 836–846.

[15] M. Zhang, B. Liu, and B. Zhang, "Multi-commodity flow traffic engineering with hybrid mpls/ospf routing," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*. IEEE, 2009, pp. 1–6.

[16] N. Garg and J. Koenemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," *SIAM Journal on Computing*, vol. 37, no. 2, pp. 630–652, 2007.

[17] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring link weights using end-to-end measurements," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. ACM, 2002, pp. 231–236.