Path Provisioning for Fibbing Controlled Load Balanced IP Networks

Saeed Barkabi Zanjani, Kuang-Yi Li, Steven S. W. Lee, and Yuan-Sun Chu
Department of Communications Engineering and Advanced Institute of Manufacturing with High-tech Innovations
National Chung Cheng University
Chiayi, Taiwan
barkabi@gmail.com; Nickykyli@gmail.com; ieeswl@ccu.edu.tw; chu@ee.ccu.edu.tw

***Abstract*—With the help of distributed link state routing protocols, IP networks are highly reliable and robust. However, routing in IP networks are relatively inflexible. Flows can follow only shortest paths toward their destinations. In contrast, Software Defined Networking (SDN) offers fine-grained control over routing, at the expense of controller overhead, failover latency, and deployment challenges. The best choice is to combine both approaches to achieve "central control over the distributed route computation" paradigm. Recently a new idea called Fibbing is proposed to realize flexible fine-grained control in conventional IP networks. Fibbing as an alternative loop-free destination based solution brings the flexibility for traditional networks without changing router configurations and/or protocols. The Fibbing controller can steer a traffic flow to follow any specific route. However, how to achieve traffic engineering through dynamic traffic monitoring and route computation is still an open question. In this paper, we propose a framework to achieve an optimized load balancing solution for a Fibbing controlled network. In our system, there are a light weight traffic monitoring module, a demand estimation module, and an optimization engine for routing path computation. The system periodically determines a set of routing paths to the Fibbing controller for path provisioning. In order to evaluate the performance, we have implemented the whole system in a testbed. The experimental results show that the average time for file transfer is significantly reduced with the support of the proposed system.**

Keywords—IP Network Routing, Fibbing, Load Balancing, Optimization

I. INTRODUCTION

In order to enhance utilization and avoid congestion, a network needs the capability to effectively control traffic routing. OSPF and IS-IS are the most common intra-domain routing protocols used in an autonomous system. They are famous for their high scalability and robustness. IP networks have to follow the shortest path constraint that only least cost paths can be taken for traffic delivery. To meet this rule, the only way to change routing paths in an IP network is through link weight modification [1][2][3]. However, traffic rerouting through changing link weight is a coarse grained approach. For example, two flows leaving from a router towards different destination hosts, they have to follow the same route if both destination hosts belonging to a same destination router.

The demand matrix considered in methods [4][5] are static. In [1][2][3][6], the traffic distribution is collected using online traffic monitoring e.g. SNMP. Although MPLS and RSVP-TE [7][8] can provide more flexible on traffic engineering, they have their own limitations. The details for the overhead and limitations are discussed in [9].

SDN uses centralized control that facilitates the controller to make routing decisions effectively. SDN has proved its success in many use cases especially in local area networks and data centers. However, scalability issue is the major concerns in SDN. As the network becomes large, the huge number of network states hinders central control by difficulty. In addition, SDN devices like OpenFlow switches have to compare a long range of packet headers. This complexity imposes challenging on designing cost effective high speed switches.

Generally speaking, by sharing the control loads on all of the nodes, conventional distributed link state routing protocols hold their benefits on high network scalability and robustness. Since only destination IP address is taken into account on determining next hop forwarding, it simplifies the processing efforts on routers. On the other hand, a central controlled SDN network has its benefits on realizing fine-grained control based on global network state information. Therefore, combining the best of both worlds to achieve "central control over the distributed route computation" is ideal the design of future network.

Recently a novel method called Fibbing is proposed to realize the above goal [10][11]. Fibbing combines the advantages of SDN with traditional legacy networks and is related to destination-based forwarding using loop-free paths. In additional to IP routers, there is a controller in a Fibbing network. Routers apply typical link state routing protocols like OSPF without making any change. By clever using OSPF Type 5 LSA [12], the controller generates fake nodes and fake links in the network. Those fake nodes and fake links do not exist in the network. The purpose of introducing them into the network is to generate a virtual network topology to cheat the routers. By manipulating the virtual topology, Fibbing is able to perform highly flexible routing control on the network.

In this paper, we focus on how to achieve network load balancing in a Fibbing controlled network. In [13], the authors proposed a load balancing method based on ECMP by adding some equal cost paths using Fibbing as a central controller. They try to introduce a better tool for reacting to flash crowds without changing the configuration and link weights. Although network load balancing has been studied in [13], a user has to determine a Directed Acyclic Graph (DAG) for each destination. Based on the given DAG, their algorithms can determine how to generate fake nodes and the corresponding link weight. In other words, the approaches can be considered to be tools for path configuration. However, how to determine the load balanced routes are not in the scope of their design.

This work tries to fill the gap for network load balancing in a Fibbing controlled network. We design and implement a

control system that constantly monitors network traffic distribution based on a light weight traffic sampling technique. Based on the sampled packets, the controller can estimate the end-to-end demand matrix. We formulate the load balancing routing and path provisioning problem as an integer linear programming problem. The objective function of this problem is to minimize traffic loads on the bottleneck links. By periodically re-configuring the routing paths, the network can avoid traffic congestion so as to reduce data transmission time.

The remainder of this paper is organized as follows. In Section II, we introduce the architecture and the major functional blocks in the system. The optimization model to determine the routing for a Fibbing controlled network is presented in Section III. We have implemented a testbed to evaluate the performance of the proposed control system. The experimental results and performance comparisons for a system with and without our using our control are presented in Sec. IV. Finally, concluding remarks and future works are listed in Sec V.

II. System Architecture

The architecture of the system controller is shown in Fig.1. It includes two major layers:

- Traffic Monitoring and Path Provision (TMPP) Layer
- Virtual Topology Management (VTM) Layer

In the TMPP layer, the general orchestration of the system is being handled. It monitors the traffic distribution to determine new routing paths. TMPP leverages Fibbing controller in the VTM layer to configure the whole network.

The TMPP layer consists of three main subsystems: Monitoring Subsystem (MS), Bandwidth Estimation Module (BEM), and Optimization Engine (OE). The MS is based on sFlow Technology [14]. The sFlow manager in this subsystem configures the sFlow agents to perform packet sampling with specified frequency. To reduce the overhead, only the headers of the sampled packets are sent to the BEM.

BEM analyzes the incoming packets by extracting the source address, destination addresses, and the packet length. Based on the collected data, it makes the estimation of the end to end bandwidth usage for each subnet pair. BEM periodically sends the estimated demand matrix to the OE for route optimization. Please note that the subnet considered here is logical. A logical subnet does not need to be the same as they appear in the physical network. Since Fibbing can aggregate any continuous destination IP addresses into its routing path, we can take any continuous IP addresses to form a logical subnet. The traffic volume in an entry of the demand matrix is the demand between two logical subnets.

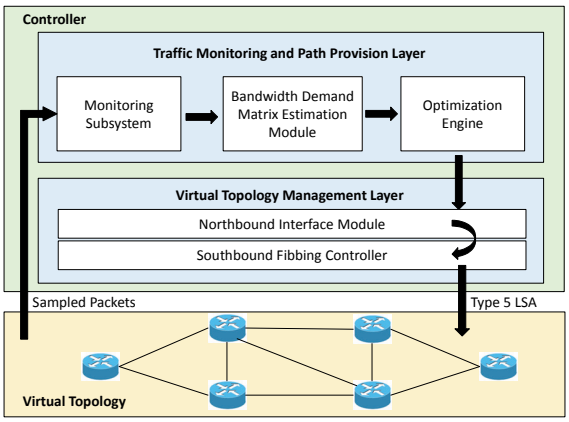


Fig. 1. System Architecture

The load balancing problem is modelled as an ILP problem in this work. OE performs an ILP solver to obtain the solution for load balancing routing. The details of this ILP problem are described in Sec. III.

The output of the OE will be the input of the VTM Layer. In VTM, we have two main modules: Northbound Interface (NI) Module and Southbound Fibbing Controller (SFC). NI is the interface between TMPP and VTM Layers. It navigates the process of preparing well-defined input entries to the SFC.

SFC makes adjacency with the network. It creates fake nodes by injecting the Type-5 LSAs. Accordingly, the target routers change their Forwarding Information Base (FIB) in accordance with the values of the LSAs. In other words, the Fibbing controller in the SFC generates a virtual topology for the network. By updating the state of the network in each predefined period, the network is triggered to reconfigure the routing paths.

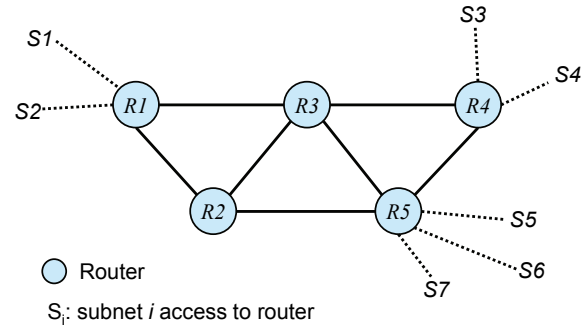
III. Problem Formulation

We use graph transformation as an example shown in Fig. 2 to facilitate our problem formulation. Figure 2(a) is the original topology denoted by graph G. Each node is a router in the network. In this example, routers $R1$, $R4$, and $R5$ connect to subnets $\{S1, S2\}$, $\{S3, S4\}$, and $\{S5, S6, S7\}$, respectively.

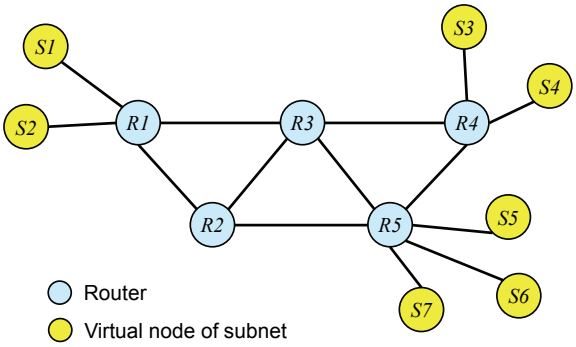
We transform G into G', shown in Fig. 2(b). In the transformation, we take logical subnets into consideration. A logical subnet here is not necessary mapping to a physical subnet. Any continuous IP address space under a router can be considered to be a logical subnet. For example, assuming there is a physical subnet connecting to a router with address space V where its IP address starts from IP_a and ends at IP_b. We could further divide the physical subnet V into multiple logical subnets if needed. For example, V can be divided into two subnets V_1 and V_2, where the address space for V_1 is $[IP_a, IP_i]$ and the address space for V_2 is $[IP_i, IP_b]$.

In the transformed graph G', each logical subnet is represented by a virtual node that is connecting to the network through a virtual link. In the example shown in Fig. 2, since $R1$ consists of subnet $S1$ and subnet $S2$, those two virtual nodes are connecting to $R1$ through virtual links ($S1$, $R1$) and ($S2$, $R1$).

On the basis of the graph transformation, we formulate the load balancing routing and path provisioning problem as an integer linear programming problem. The input of the problem includes the transformed graph $G'(N, L)$ and the demand matrix for subnet pairs in the network. The problem determines the routing configurations so as to minimize the utilization on the most congested link. Based on the results, Fibbing controller generates fake nodes to achieve balanced routing paths provisioning.



(a) An input graph G with subnet to access router



(b) Each subnet transform to a node to construct G'

Fig. 2. Illustration of graph transformation

The problem formulation and notations are listed below.

Given input constant values:

Input values:

- N: set of nodes (including routers and subnet) in the given network;
- L: set of directional links in the given network;
- $L_n^{in(out)}$: set of links entering (leaving) node n;
- D: set of subnet nodes;
- C_l: physical capacity of link l;
- H_d: total demand volume to destination subnet node d;
- h_n^d: demand volume from source node n to destination node d;

Decision variables:

- α: utilization of the most congested link;
- x_l^d: total traffic amount of flows with destination d carried on link l;
- y_l^d: =1, if link l is used to transmit packets with destination subnet d; =0, otherwise. y_l^d is used to determine routing table;

Problem (IP):

$$\min \alpha$$

subject to:

$$\sum_{l \in L_d^{in}} x_l^d = H_d \qquad \forall d \in D \qquad (1)$$

$$\sum_{l \in L_n^{out}} x_l^d - \sum_{l \in L_n^{in}} x_l^d = h_n^d \qquad \forall n \in N \setminus \{d\}, d \in D \qquad (2)$$

$$x_l^d \le y_l^d C_l \qquad \forall l \in L, d \in D \qquad (3)$$

$$\sum_{l \in L_n^{out}} y_l^d \le 1 \qquad \forall n \in N \setminus \{d\}, d \in D \qquad (4)$$

$$\sum_{d \in D} x_l^d \le \alpha C_l \qquad \forall l \in L \qquad (5)$$

$$y_l^d = 0 \text{ or } 1 \qquad \forall l \in L, d \in D \qquad (6)$$

The objective function is to minimize the link utilization on the most congested link. The flow conservation law for each OD-pair is realized by Constraints (1) and (2). Constraint (3) states that a link l can be used to route traffic toward destination d only if the corresponding route is configured. Constraint (4) requires that each router can select at most one outgoing link to deliver packets for destination d. Constraint (5) requires that the total flow amount on each link is no larger than its upper bound capacity. Finally, Constraint (6) requires the decision variables for routing table configurations are binary values.

IV. Experimental Results

We have implemented a network testbed to evaluate the performance of the system. We made performance comparisons for the network with and without our controller. In the experiments, we apply different inter arrival times for connection setup requests. Various file sizes are also applied to generate different network loads. In addition, we also evaluate the impact of reconfiguration frequency on the performance.

A. Environment

Figure 3 depicts an experimental network. It consists of 6 Cisco 3750 routers and 20 hosts. The capacity of each link between each adjacent router is 100 Mbps and the link rate between a host to its access router is 100 Mbps. Since our routers do not have sFlow inside, an additional HP 5900 sFlow agent (not shown in Fig. 3) is inserted for each link to

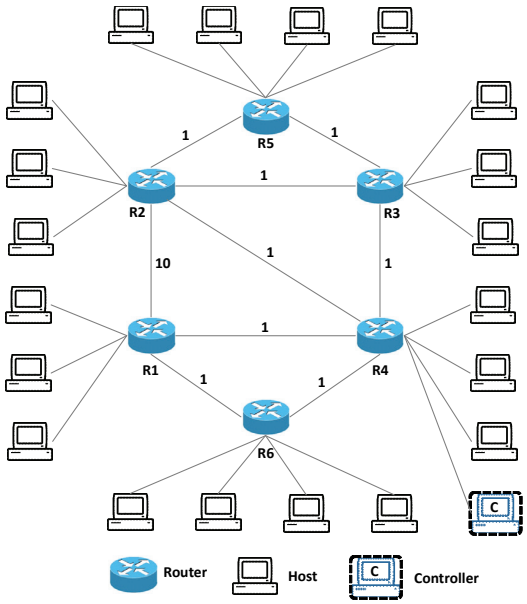


Fig. 3. Experimental Network Topology

collect sample packets. The sampling rate is set to 1/10000 in our experiments.

The OSPF protocol is running in our network with the hello interval of 5 seconds and with the dead interval of 40 seconds. In the experiments, TCP is used to support file transfer between a pair of hosts. The incoming requests arrive to the network follow the Poisson process. The Poisson arrival rates λ considered in the experiments are 15, 20 and 25 seconds. For each connection, the source node is randomly selected from one the 20 hosts. The destination node is also randomly selected but the selected source node and the destination node for each connection must connect to different access routers.

We apply three file sizes 1, 2, and 3 GBytes for file transfer. The total duration for injecting the requests to the network is 90 minutes. After 90 minutes, there is no further connection generated, the system continues to work until all of the TCP connections inside the network completing their file transmission. We evaluate three reconfiguration periods-10, 15, and 20 minutes for Fibbing. We make performance comparisons on average duration for file transfer for network with and without our load balancing controller.

B. Discussion and Results

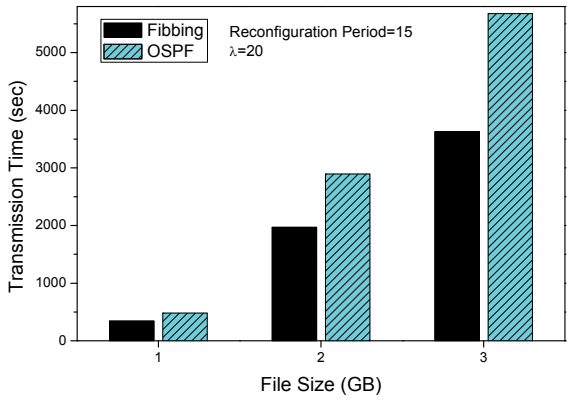
In the first experiment, the inter arrival time of connections is λ=20 seconds. The period for Fibbing to reconfigure the routing paths is 15 minutes. The experimental results are shown in Fig. 4(a). It is clear that our controller can reduce average file transfer times. The gaps between the network running pure OSPF and that controlled by our controller are 28.52%, 31.94%, and 36.1% for files with 1 GB, 2 GB, and 3GB, respectively.

We further examine the performance with different inter arrival times. The results are shown in Fig. 4(b). The gaps between pure OSPF and our system are 37.97%, 31.94%, and 36.38% for lambda= 15 sec, 20 sec, and 25 sec, respectively. The results indicate that our controller can help a network to reduce time for file transfer for a wide range of inter arrival times.

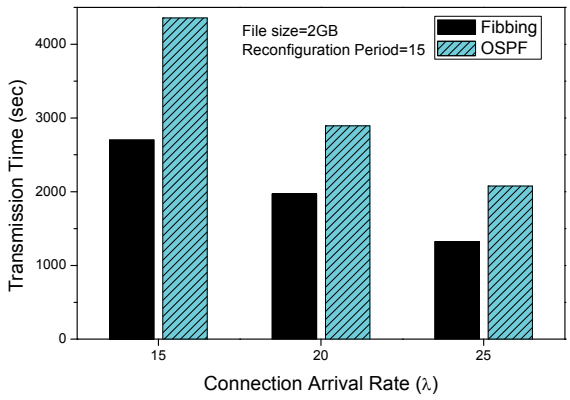
In the third set of experiments, we take different reconfiguration periods into consideration. The inter arrival time is fixed to λ=20. Each TCP connection is used to send a 2 GB file. We use Fibbing to reconfigure paths every 10, 15 and 20 minutes. The results are plotted in Fig. 4(c). As the reconfiguration period is small, the network can use better paths for file transfer so as to reduce the average time for TCP connection. However, the frequent reconfiguration becomes a large burden on control plane as the network size increases.

To clearly identify the distribution of transmission, we compare the ranking of connection times between OSPF and Fibbing. The transmission times are sorted in ascending order. In the experiments, the file size for a connection is fixed to 2 GByte and the reconfiguration period for Fibbing is fixed to 15 minutes. Figure 5(a), 5(b), and 5(c) show the results for inter arrival times 15 sec, 20 sec, and 25 sec, respectively. Observing the results, we find that our system outperforms pure OSPF in all of those network loads. In addition, the time variance of file transfer in our system is much smaller than that in pure OSPF.

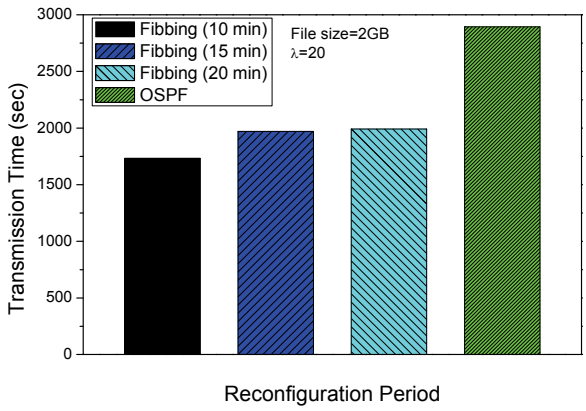
In the final set of experiments, we examine the impact of sampling error by sFlow. We conducted 30 experiments for



(a) Average Transmission Time with different File Sizes



(b) Average Transmission Time with different connection arrival rate



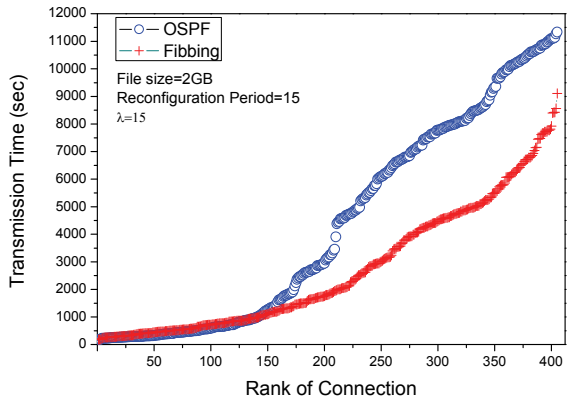
(c) Average Transmission Time with different Reconfiguration Period

Fig. 4. Comparing OSPF with and without the presence of Fibbing

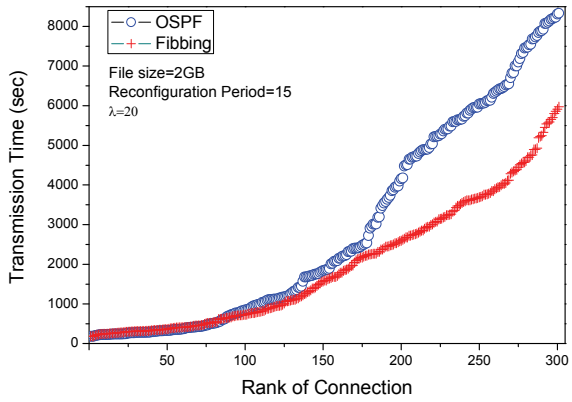
the case of transmission 2 GBytes file with call inter arrival time λ=20 second. We examine the average file transmission time of those 30 experiments and plot the results in Fig. 6. The worst and best of them are 2018 sec and 1704 sec, and their average is 1827 sec. In other words, the gap of each individual to the average falls within 10.5%. This indicates the effectiveness of using light weight sFlow in the application.

V. CONCLUSION AND FUTURE WORKS

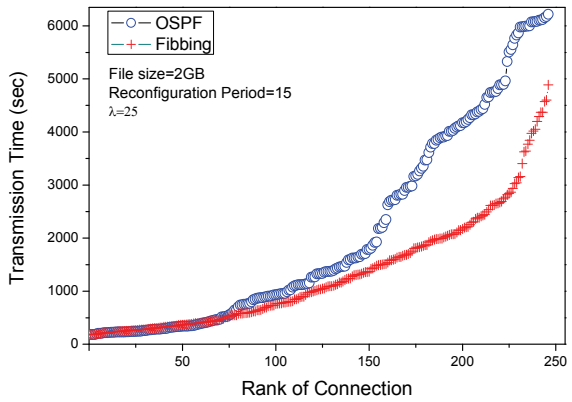
In this work, we propose a control system for dynamic path provisioning in a Fibbing controlled IP networks. Our



(a) Average Transmission Time with Rank of Connection in λ=15



(b) Average Transmission Time with Rank of Connection in λ=20



(c) Average Transmission Time with Rank of Connection in λ=25

Fig. 5. Comparing OSPF with and without the presence of Fibbing

system monitors the traffic distribution in the underneath IP network through light weight sFlow packet sampling. Based on the sampled data, our controller can derive the demand matrix of the network. We have modelled the load balancing problem as an ILP to obtain the routing paths. By leveraging the Fibbing, the average time to finish a TCP connection can be significantly improved through periodically reconfiguring the routing paths of the networks.

In the current implementation, the ILP is solved using a commercial optimization solver. As the network size increases, the computation time should increase dramatically. To deal with this issue, in the future, we will develop a heuristic algorithm like [15] to obtain a near optimal solution in short computation time.

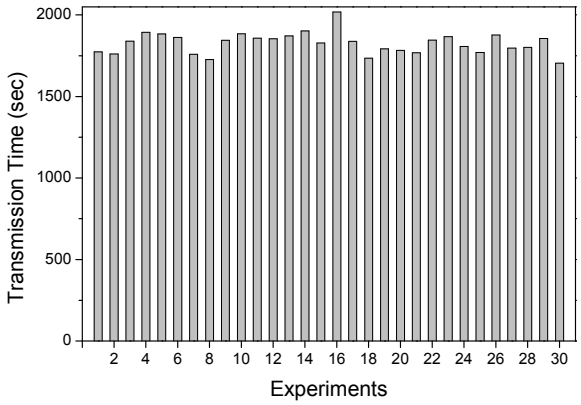


Fig. 6. Variations on transmission time caused by sampling error

ACKNOWLEDGEMENT

This work was partially supported by the Advanced Institute of Manufacturing with High-tech Innovations (AIM-HI) from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan. This work is also supported by the Ministry of Science and Technology, Taiwan, under Grant no. 106-2221-E-194 -019.

REFERENCES

[1] Bernard Fortz and Mikkel Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights," in *Proc. IEEE INFOCOM*, 2014.

[2] O. Brun and J. M. Garcia, "Dynamic igp weight optimization in ip networks," *in First International Symposium on Network Cloud Computing and Applications*, 2011.

[3] J.Vallet, O.Brun, "Adaptive routing in IP networks using SNMP link counts", in *Teletraffic Congress,* 2013.

[4] E. Mulyana and U. Killat, "Optimizing ip networks for uncertain demands using outbound traffic constraints", in *INOC'2005* , pages 695–701, 2005.

[5] A. Altin, P. Belotti, and M. Pinar, "Ospf routing with optimal oblivious performance ratio under polyhedral demand uncertainty", in *Optimization and Engineering* , 2009.

[6] J. Chu and C. T. Lea, "Optimal link weights for ip-based networks supporting hose-model vpns", in *IEEE/ACM Transactions on Networking* , 17(3):778–786, June 2009.

[7] Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," *RFC 3209*, 2001.

[8] A. Farrel, J.-P. Vasseur, and J. Ash, "A Path Computation Element (PCE)-Based Architecture," *RFC 4655*, 2006.

[9] A. Pathak, M. Zhang, Y. C. Hu, R. Mahajan, and D. A. Maltz, "Latency inflation with MPLS-based traffic engineering," in *Internet Measurement Conference*, 2011, pp. 463–472.

[10] S. Vissicchio, L. Vanbever, J. Rexford, "Sweet little lies: fake topologies for flexible routing," in *Proc. ACM Workshop on Hot Topics in Networks*, 2014.

[11] S. Vissicchio, O. Tilmans, L. Vanbever, J. Rexford, "Central control over distributed routing," *ACM SIGCOMM Computer Communication Review*, vol. 45, pp. 43-56, Oct. 2015.

[12] J. Moy, "OSPF Version 2," *RFC 2328*, 1998

[13] Olivier Tilmans , Stefano Vissicchio , Laurent Vanbever , Jennifer Rexford, " Fibbing in action: On-demand load-balancing for better video delivery," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016.

[14] P. Phaal, sFlow Version 5, Jul. 2004. [Online]. Available: http://sflow.org/sflow_version_5.txt

[15] Steven S. W. Lee, Po-Kai Tseng, and Alice Chen, "Link Weight Assignment and Loop Free Routing Table Update for Link State Routing Protocols in Energy-Aware Internet," *Future Generation Computer Systems*, pp. 437-445, vol. 28, no. 2, February 2012.