Software Defined Networking-based Traffic Engineering for Data Center Networks

Yoonseon Han*, Sin-seok Seo†, Jian Li*, Jonghwan Hyun†, Jae-Hyoung Yoo†, James Won-Ki Hong†

*Division of IT Convergence Engineering, POSTECH, Korea
{seon054, gunine}@postech.ac.kr

†Department of Computer Science and Engineering, POSTECH, Korea
{sesise, noraki, styoo, jwkhong}@postech.ac.kr

***Abstract*—Today's Data Center Networks (DCNs) contain tens of thousands of hosts with significant bandwidth requirements as the needs for cloud computing, multimedia contents, and big data analysis are increasing. However, the existing DCN technologies accompany the following two problems. First, power consumptions of a DCN is constant regardless of the utilization of network resources. Second, due to a static routing scheme, a few links in DCNs are experiencing congestions while other majority links are being underutilized. To overcome these limitations of the current DCNs, we propose a Software Defined Networking (SDN)-based Traffic Engineering (TE), which consists of *optimal topology composition* and *traffic load balancing*. We can reduce the power consumptions of the DCN by turning off links and switches that are not included in the optimal subset topology. To diminish network congestions, the traffic load balancing distributes ever-changing traffic demands over the found optimal subset topology. Simulation results revealed that the proposed SDN-based TE approach can reduce power consumptions of a DCN about 41% and Maximum Link Utilization (MLU) about 60% on average in comparison with a static routing scheme.**

***Index Terms*—Data Center Network, Traffic Engineering, Software Defined Networking**

I. Introduction

A Data Center (DC) is a facility used to house computer servers or hosts, and a Data Center Network (DCN) interconnects these hosts using dedicated links and switches. Today's DCs may contain tens of thousands of hosts with significant bandwidth requirements as the needs for cloud computing, multimedia contents, and big data analysis are increasing [1].

Current DCNs are suffering from high operational expenditures and frequent link congestions. First of all, the amount of power consumed by a current DCN is constant regardless of the usage ratio of network resources, while the network utilization fluctuates depending on the time of day. The varying traffic demands in the DCN can be satisfied by a subset of the network resources most of the time [2]. As a result, operating costs of the DCN are higher than what are actually required. Second, due to a static flow path selection scheme, a few specific links are frequently experiencing congestions while other majority links are being under-utilized [3]. The static ECMP [4], most widely used for core routers, does not consider dynamic nature of DCN traffic characteristics and this leads to a congestion of a specific link even in a situation where almost every other links are underutilized.

Considering these problems of current DCN technologies, this paper proposes a dynamic Traffic Engineering (TE) system for a DCN using Software Defined Networking (SDN) technologies. With SDN, a network administrator can vendor independently control the entire network from a single logical point (i.e. SDN controller) and this simplifies the network design and operation.

TE is defined as "network engineering dealing with the issue of performance evaluation and performance optimization of operational IP networks" [5]. Typical objectives of TE include balancing network load and minimizing network utilization. A lot of studies have been proposed in the area of TE and they can be classified into three categories according to routing enforcement mechanisms: MultiProtocol Label Switching (MPLS)- [6], [7], Internet Protocol (IP)- [8], [9], or SDN-based [2], [10]–[15]. Our proposed dynamic TE proposal for a DCN takes the SDN-based approach. The benefits of SDN-based TE approaches for a DCN are summarized as follows.

- It is relatively easier to obtain traffic and failure information via a (logically) centralized SDN controller.
- Any flow format with arbitrary granularity can be exploited for TE
- It is easy to apply TE results to switches in a DCN by modifying flow tables within the switches.

The proposed TE system consists of two procedures: *optimal topology composition* and *traffic load balancing*. To reduce power consumptions of a DCN, the optimal topology composition finds a subset DCN topology that can accommodate expected traffic demands at the moment. We can reduce the power consumptions of the DCN by turning off links and switches that are not included in the optimal subset topology. To diminish network congestions, the traffic load balancing distributes ever-changing traffic demands over the found optimal subset topology. This traffic distribution makes it possible to accommodate more traffic without causing network congestions. To validate the proposed dynamic TE system, we have implemented a prototype of the proposed TE system for a DCN. The APIs provided by SDN controller were used for applying outputs of the proposed TE system to the

DCN. We also have evaluated performance of the proposed TE system using simulations in terms of power saving ratio and Maximum Link Utilization (MLU).

II. Related Work

Many TE techniques for the traditional Internet have been proposed including [4], [6]–[9], but TE for DCNs is still in a preliminary stage [16]. In the meantime, the needs for large-scale DCNs are increasing sharply as well as the needs for efficient utilization of DCN resources. Recently, for that reason, several studies about TE for DCNs have been published including [2], [10]–[12], [15], [16]. We briefly introduce major TE studies that are highly related with our proposed approach.

Hedera [11], a dynamic flow routing system for a multi-stage switching fabric, utilizes a centralized approach to route *elephant* flows exceeding 10 percent of the host Network Interface Card (NIC) bandwidth while it utilizes static ECMP for the rest short lived *mice* flows. The purpose of Hedera is maximizing bisection bandwidth of a DCN by appropriately placing the elephant flows among multiple alternative paths; estimated flow demands of the elephant flows are used for the placement. The main limitation of Hedera is that it let the static ECMP routes the mice flows, which comprise more than 80% of the actual DCN traffic [3], [17].

Benson *et al.* proposed microTE [10], a centralized system that adapts to traffic variations by leveraging the short term predictability of the DCN traffic, to achieve fine grained TE. It constantly monitors traffic variations, determines which Top-of-Rack (ToR) pairs have predictable traffic, and assigns the predicted traffic to the optimal path. Similar to Hedera, the remaining unpredictable traffic is then routed using weighted ECMP, where the weights reflect the available capacity after the predictable traffic has been assigned. The major shortcomings of microTE is twofold: a) it requires host modifications to collect fine-grained traffic data and b) it lacks of scalability due to its extremely short execution cycle

Penalizing Exponential Flow-spliTting (PEFT) was proposed by Xu *et al.* [18] to achieve optimal TE for wide-area ISP networks. Switches running PEFT make forwarding and traffic splitting decisions locally and independently with each other. In addition, packets, even in a same flow, can be forwarded through a set of unequal cost paths by exponentially penalized higher cost paths. Later, Tso *et al.* [16] modified the PEFT for DCN TE as a reactive online version to cope with dynamic and unpredictable nature of DCN traffic. However, PEFT imposes heavy load on switches because each switch in PEFT has to measure traffic volume incoming to and outgoing from its ports and it has to calculate optimal routing paths using the measured traffic data. Another limitation of PEFT is that routing decisions of switches are not global optimal solutions. Finally, PEFT delivers packets in the same flow through a set of unequal cost paths by splitting them and this causes a packet reordering problem.

ElasticTree [2] is a centralized system for dynamically adapting the energy consumption of a DCN by turning off the links and switches that do not essentially necessary to meet the varying traffic demands at the time. The fact that traffic can be satisfied by a subset of the entire network links and switches most of the time makes ElasticTree feasible. To find the minimum subset topology, i.e. essential links and switches, ElasticTree proposed three algorithms with different optimality and scalability: Linear Programming (LP)-based formal model, greedy bin packing heuristic, and topology-aware heuristic. The main problem of ElasticTree is that its flow allocation algorithms probably cause severe link congestions because they allocate flows to links while maximizing the utilization of link capacity.

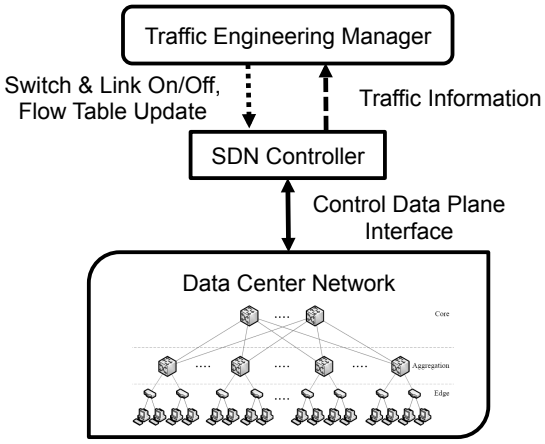


Fig. 1. Traffic engineering system architecture.

III. Traffic Engineering for DCNs

In this section, we present the proposed dynamic TE system for DCNs in detail. First, we explain the overall system architecture of the proposed TE system. Thereafter, detailed algorithms for optimal topology composition and traffic load balancing are described.

A. Traffic Engineering System Architecture

Fig. 1 shows an overall architecture of the proposed dynamic TE system that has three components: a Data Center Network (DCN), an SDN Controller, and a TE Manager. The DCN, which contains many servers and SDN switches, is a target network of our TE system. The SDN switches in the DCN report their traffic and failure status to the SDN controller through the control data plane interface (e.g., OpenFlow). The SDN controller aggregates and summarizes the collected information. The TE manager takes the summarized traffic and failure information to dynamically make an appropriate TE decision at the time. The SDN controller also changes switching behavior of SDN switches by updating their flow tables, and turns on/off switches and links in the DCN to apply the TE decision that minimizes power consumptions and link congestions.

The TE manager, a core component of our dynamic TE system, periodically takes traffic and failure information from the SDN controller, makes a TE decision using the information, and notifies the decision results to the SDN controller.

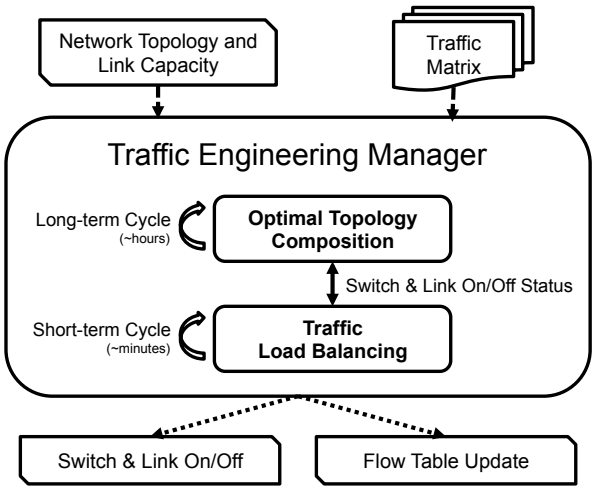


Fig. 2. Traffic engineering manager.

The TE manager consists of two major procedures: optimal topology composition and traffic load balancing (see Figure 2). Optimal topology composition periodically finds a minimum subset of links and switches that can accommodate hourly estimated traffic demands at the moment within the whole DCN topology. Traffic load balancing periodically distributes ever-changing traffic demands over the found optimal topology to minimize Maximum Link Utilization (MLU). Note that traffic load balancing is repeatedly executed in the time scale of minutes whereas the optimal topology composition is executed in the time scale of hours.

B. Optimal Topology Composition

The traffic demands in the DCN varies depending on the time. However, power consumption of the DCN are constant at higher lever than what are actually required. To reduce power consumption of a DCN, Heller *et al.* [2] proposed *ElasticTree*. Their proposal contains both an MCF-based formal model using linear programming, and a greedy bin packing heuristic which quickly produces a *near-optimal* solution. MCF (Multi Commodity Flow) problem is a network flow problem with multiple flow demands between different source and target nodes. In this paper, we propose a quicker algorithm to find the minimum subset topology using path-based MCF, which improves the link-based MCF *ElasticTree*. We also propose a refined greedy bin packing heuristic for minimum subset topology composition corresponding to the path-based MCF.

1) Subset Topology Composition using Path-based MCF: To reduce the computation time, we propose a minimum subset topology composition model based on path-based MCF, which requires significantly less decision variables and constraints than the MCF-based model [2]. The minimum subset topology composition model using path-based MCF is defined as follows.

- **Input**
  - Network topology: $G(V, E)$
  - Traffic matrix: T
  - Link capacity: $\forall (u, v) \in E, c(u, v)$
  - Set of considered paths of flows: $\forall i, P_{T_i} = \{p_{i,0}, \cdots, p_{i,j}, \cdots, p_{i,l}\}$
  - Subset of considered paths that contain a link (u, v): $\forall i, P_{T_i}^{(u,v)} \subseteq P_{T_i}$
  - Set of all switches: $S \subset V$
  - Set of nodes connected to a switch: $\forall u \in S, V_u$
  - Power cost of links: $\forall (u, v) \in E, a(u, v)$
  - Power cost of switches: $\forall u \in S, b(u)$
- **Decision Variables**
  - Flows along each path: $\forall i, \forall p \in P_{T_i}, f_i(p)$
  - Binary decision variable indicating whether a link (u, v) is powered on: $\forall (u, v) \in E, X_{u,v}$
  - Binary decision variable indicating whether a switch u is powered on: $\forall u \in S, Y_u$
- **Objective**
  Minimize $\sum_{(u,v)\in E} X_{u,v} \times a(u, v) + \sum_{u \in S} Y_u \times b(u)$
- **Constraints**
  - Capacity limitation: s$\forall (u, v) \in E, \sum_{i=1}^{k} \sum_{p \in P_{T_i}^{(u,v)}} f_i(p) \leq X_{u,v} \times c(u, v)$
  - Demand satisfaction: $\forall i, \sum_{p \in P_{T_i}} f_i(p) = d_i$
  - Bidirectional link power: $\forall (u, v) \in E, X_{u,v} = X_{v,u}$
  - Switch-to-link correlation: $\forall u \in S, \forall w \in V_u, X_{u,w} = X_{w,u} \leq Y_u$
  - Link-to-switch correlation: $\forall u \in S, Y_u \leq \sum_{w \in V_u} X_{w,u} = \sum_{w \in V_u} X_{u,w}$

The decision variables $f_i(p)$ represent allocation of flow T_i to each path p. In order to allocate flows in T to the network topology $G(V, E)$, path-based MCF model finds values of $f_i(p)$ that satisfy both capacity limitation and demand satisfaction constraints. The binary decision variables $X_{u,v}$ or Y_u indicate whether a link (u, v) or a switch u is powered on respectively. With these binary decision variables, we can define an *objective function* that represents minimization of power costs of a DCN.

The capacity limitation constraints force the sum of flows along each link (u, v) does not exceed the link capacity $c(u, v)$ and ensure flows are allocated to only those links that are powered on. The demand satisfaction constraints ensure each source s_i or target t_i in a flow T_i sends or receives an equal amount of traffic to its demand d_i. The bidirectional link power constraints make the power statuses of both a link (u, v) and (v, u) consistent. The switch-to-link correlation constraints ensure that when a switch u is powered off, all links connected to this switch are also powered off. Similarly, the link-to-switch correlation constraints ensure that when all links connected to a switch u are powered off, the switch is also powered off.

2) Subset Topology Composition using Heuristic: We can find a subset topology using a heuristic within a short period of time even for a large scale DCN. While this heuristic does not guarantee a solution within a bound of optimal, it produces a high-quality subset topology in practice. In brief, this heuristic, which is called greedy bin packing [2], evaluates

Algorithm 1: Heuristic for Subset Topology Composition

Input : T (Traffic Matrix), DCN Topology, Link Capacity
Output: On/Off Status of $switches$ and $links$

1: **for all** flow f in T **do**
2: $list_p \leftarrow$ possible paths from f_{src} to f_{dst}
3: sort $list_p$ by position from left to right
4: **for all** path p in $list_p$ **do**
5: **if** All links in p satisfy $cap_{avail} > f_{dmd}$ **then**
6: Assign f to p
7: **for all** link l in p **do**
8: $cap_{avail}[l] = cap_{avail}[l] - f_{dmd}$
9: **end for**
10: break
11: **end if**
12: **end for**
13: **end for**
14: **for all** link l in DCN **do**
15: **if** $cap_{avail}[l] < cap_{max}[l]$ **then**
16: Set l and connected $switches$ to turn on
17: **end if**
18: **end for**

possible paths between a source and a target of each flow, and allocates the flow to the leftmost or the rightmost path with sufficient capacity. Similar to the linear programming-based formal models, this approach requires knowledge of the traffic matrix in advance, but it can compute the solution incrementally; we can process the traffic matrix on-line with this heuristic.

Algorithm 1 describes the heuristic for subset topology composition in detail. It takes a traffic matrix T, a DCN topology, and a capacity of each link as input. The traffic matrix T is a set of flows specified by (source, target, demand) tuples. The output of this algorithm is on/off status of switches and links that constitute the entire physical DCN topology. By using only turned on switches and links specified by this algorithm, we can construct the near-minimum subset topology that satisfies all the traffic demands in T. This subset topology composition algorithm consists of two parts: 1) allocation of each flows in T to the left most path with sufficient capacity (line 1–13) and 2) determination of switches and links to be turned on (line 14–18).

3) Extra Switch and Link Addition: The subset topology composition algorithms try to find the subset topology with the minimum number of links and switches. Thus, the probability that these subset links would experience traffic congestions is very high. By adding extra switches and links on the found minimum topology, the link congestion probability could be diminished. A procedure for re-distributing the ever-changing traffic demands over the augmented optimal topology is necessary to fully utilize the reinforced extra links and to reduce possible link congestions.

C. Traffic Load Balancing

A few specific links in DCN are experiencing congestions, especially the links connected to core switches, while other majority links are being underutilized due to a static routing path selection scheme [3]. We propose a dynamic traffic load balancing algorithms to minimize possible link congestions by distributing traffic demands over the entire DCN topology or over the subset topology found by the optimal topology composition algorithms. We propose two different traffic load balancing algorithms for predicted traffic demands using path-based MCF and a heuristic. These algorithms periodically distribute the ever-changing estimated traffic demands, which are predicted in the form of traffic matrix and in the time scale of a few minutes or seconds, over the found optimal topology. This makes it possible to accommodate more traffic demands without installing extra network resources.

1) Traffic Load Balancing using Path-based MCF: We can allocate flows in a traffic matrix T to network resources using Path-based MCF while distributing traffic load to minimize MLU. It is required to define additional decision variable m, which represents MLU, and to modify capacity limitation constraints with this variable m. The predicted traffic load balancing model using path-based MCF is defined as follows.

The modified capacity limitation constraints ensure the sum of flows along each link (u, v) does not exceed the adjusted link capacity $m \times c(u, v)$. This predicted traffic load balancing model exploits the modified capacity limitation constraints to minimize the MLU of all the links in the network by setting the objective function to minimize m.

- **Input**
  - Network topology: $G(V, E)$
  - Traffic matrix: T
  - Link capacity: $\forall (u, v) \in E, c(u, v)$
  - Set of considered paths of flows: $\forall i, P_{T_i} = \{p_{i,0}, \cdots, p_{i,j}, \cdots, p_{i,l}\}$
  - Subset of considered paths that contain a link (u, v): $\forall i, P_{T_i}^{(u,v)} \subseteq P_{T_i}$
- **Decision Variables**
  - Maximum Link Utilization (MLU): m
  - Flows along each path: $\forall i, \forall p \in P_{T_i}, f_i(p)$
- **Objective**: Minimize m
- **Constraints**
  - Capacity limitation: $\forall (u, v) \in E, \sum_{i=1}^{k} \sum_{p \in P_{T_i}^{(u,v)}} f_i(p) \leq m \times c(u, v)$
  - Demand satisfaction: $\forall i, \sum_{p \in P_{T_i}} f_i(p) = d_i$

2) Traffic Load Balancing using Heuristic: We can allocate flows to minimize MLU using a heuristic within a short period of time even for a large scale DCN. While this heuristic does not guarantee a solution within a bound of optimal, it produces a high-quality traffic allocation that minimizes MLU in practice. Algorithm 2 describes the heuristic for predicted traffic load balancing in detail. It takes a traffic matrix T, a DCN topology, and a capacity of each link as input.

The traffic matrix T is a set of flows specified by (source, target, demand) tuples. The output of this algorithm is allocation, which minimizes MLU, of flows in T to paths. This heuristic sorts T in descending order of traffic demands to allocate large flows first (line 1). Then, for each flow in T, it evaluates possible paths from a source f_{src} to a target f_{dst}

Algorithm 2: Heuristic for Traffic Load Balancing

Input : T (Traffic Matrix), DCN Topology, Link Capacity
Output: Minimizing MLU path allocation of flows in T

```
1: sort T in descending order of demands
2: for all flow f in T do
3:     list_p ← possible paths from f_src to f_dst
4:     list_MLU ← null
5:     for all path p in list_p do
6:         list_MLU[p] = MLU of p
7:     end for
8:     p_sel = list_p[index of minimum list_MLU]
9:     Assign f to p_sel
10:    for all link l in p_sel do
11:        cap_avail[l] = cap_avail[l] − f_dmd
12:    end for
13: end for
```

TABLE I
TRAFFIC MATRIX DATA SETS.

Category	Set 1	Set 2	Set 3
k of Fat tree	**4–36**	32	32
# of Hosts	**16–11,664**	8,192	8,192
# of Flows per Host	2	**1–5**	2
Intra-Rack Traffic %	50	50	**10–90**
Traffic Demands	10–20% of a maximum link capacity		

(line 3). Among these considered paths, this heuristic allocates the flow f to a path p_{sel} with the minimum MLU (line 4–9). Finally, it decreases the available capacities cap_{avail} of links consisting the selected path p_{sel} as many as the amount of traffic demand f_{dmd} of the flow f (line 10–12).

IV. Implementation and Evaluation

We employed *Mininet* [19] network emulation tool for constructing a virtual DCN topology. Each switch in the virtual topology emulated by Mininet is a virtual instance of *Open vSwitch* [20]. To generate traffic among hosts, we used *iperf* [21], which is a network testing tool that can create TCP or UDP data streams. It also provides measurement logs of the data streams. we have chosen to use *Floodlight* [22], a Java-based OpenFlow controller, as a centralized manager of the DCN. It updates flow tables of switches using OpenFlow protocol to apply TE results. The traffic engineering manager, a core component of our TE system, executes the proposed TE mechanism, and notifies the results to the SDN controller using Floodlight APIs. We implemented prototype of this TE manager using a Python 2.7.4 programming language.

We used three different data sets of traffic matrix summarized in Table I for experiments. In Table I, k means the Fat-Tree topology is organized using only k-port commodity switches. Each flow in these three traffic matrices has a demand that was randomly selected in between 10–20% of a maximum link capacity in common. The data set 1 were generated by increasing the number of hosts to examine influences of the size of a DCN. The data set 2 was generated by increasing the number of flows per host to identify influences of a traffic load in a DCN. Lastly, the data set 3 was generated by changing the intra-rack traffic ratio. Moreover, the data sets contain unpredicted traffic in common; the unpredicted traffic was assumed to contain 5 flows per host with traffic demands that were randomly selected in between 1 to 5% of a maximum link capacity.

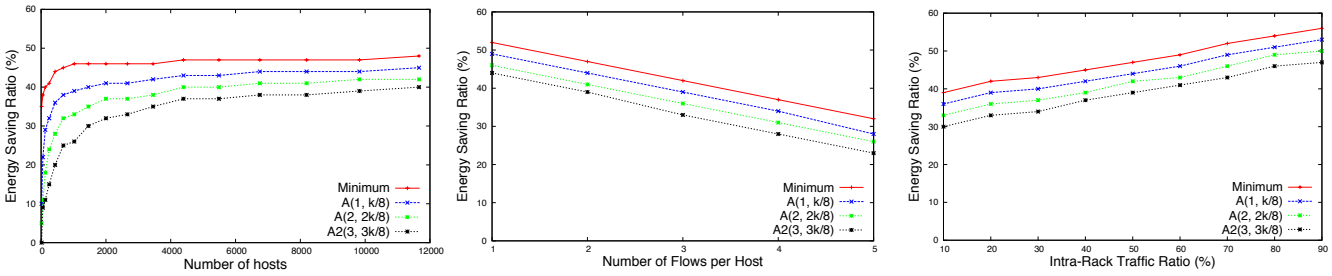
Fig. 3 shows power consumption ratio when the proposed optimal topology composition approach is applied. We assumed that the power costs of each switch and link were 150 and 1, respectively. We calculated power saving ratios after adding extra i aggregate switches per pod and j core switches to the found optimal subset topology (represented as A(i, j)). The result shows that our proposed mechanism reduces around 41% energy consumption with minimum topology composition (35% for A(1, k/8), 31% for A(2, 2k/8), and 27% for A(3, 3k/8)). We can identify four trends of the power saving ratio. 1) the power saving ratio was increased as the size of the DCN grows. 2) the saving ratio was decreased as the number of flows per host increases. 3) the ratio was increased as the intra-rack traffic ratio increases. 4) the gap of the power saving ratio between an optimal and an augmented topology was decreased as the size of the DCN grows.

Fig. 4 shows Maximum Link Utilization (MLU). Note that the static routing scheme of Fat-Tree cannot be applied on an optimal subset topology because it was designed to make use of an entire Fat-Tree topology to distribute traffic load. We also plotted an average Link Utilization (LU) of aggregate and core links in the Figure as a base line. We can identify that the MLU of the static routing scheme was increased 1) as the size of the DCN grows, 2) as the number of flows per host increases, and 3) as the intra-rack traffic ratio decreases. In other words, our traffic load balancing schemes significantly reduced the MLU in comparison with the static scheme (maximum 60%). Moreover, we can identify that our traffic load balancing schemes stably maintained the MLUs as low as possible against the average LUs, whereas the static scheme failed to address.

V. Conclusion and Future work

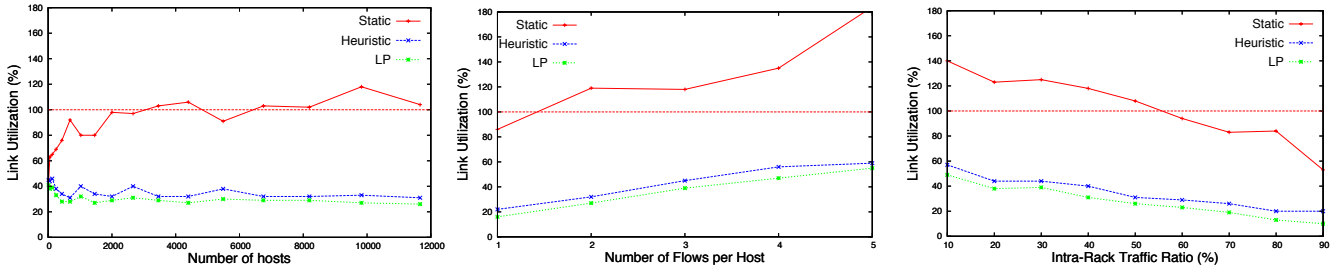
In this paper, we have presented a dynamic TE system for a DCN in detail. We have explained the overall system architecture of the TE system, which has three components: a DCN, an SDN controller, and a TE manager. Thereafter, optimal topology composition and traffic load balancing for the TE manager are described using both linear programming and a heuristic approaches. We have implemented a prototype of the proposed TE system for a DCN using various tools. The evaluation results in simulations shows that the proposed method reduces 41% power consumption with minimal topology composition, and 60% lower Maximum Link Utilization (MLU) compared to a static routing scheme.

As future work, we will further improve those TM estimation techniques considering characteristics of a DCN. Possible research directions to that include a) exploiting data obtained from end hosts as well as data from switches for estimating



(a) TM Set 1 (b) TM Set 2 (c) TM Set 3

Fig. 3. Power saving ratios. (k = a parameter to construct k-ary Fat-Tree topology)



(a) TM Set 1 (b) TM Set 2 (c) TM Set 3

Fig. 4. Maximum link utilization comparisons.

TM and b) extending TE algorithm by utilizing flow-level characteristics of DCN.

REFERENCES

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM '08*, Seattle, USA, Aug. 17–22, 2008, pp. 63–74.

[2] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "ElasticTree: Saving energy in data center networks," in *Proc. 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI '10)*, San Jose, USA, Apr. 28–30, 2010, pp. 1–16.

[3] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. ACM Internet Measurement Conference 2010 (IMC '10)*, Melbourne, Australia, Nov. 1–3, 2010, pp. 267–280.

[4] C. Hopps, "Analysis of an Equal-Cost Multi-Path algorithm," RFC 2992, Nov. 2000.

[5] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "Overview and principles of internet traffic engineering," RFC 3272, May 2002.

[6] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. MacManus, "Requirements for traffic engineering over MPLS," RFC 2702, Sep. 1999.

[7] D. Awduche, "MPLS and traffic engineering in ip networks," *IEEE Communications Magazine*, vol. 37, no. 12, pp. 42–47, 1999.

[8] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. 19th IEEE International Conference on Computer Communications (INFOCOM '00)*, Tel Aviv, Israel, Mar. 26–30, 2000, pp. 519–528.

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

[10] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers," in *Proc. 7th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '11)*, Tokyo, Japan, Dec. 6–9, 2011, pp. 1–12.

[11] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proc. 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI '10)*, San Jose, USA, Apr. 28–30, 2010, pp. 1–15.

[12] Y. Li and D. Pan, "OpenFlow based load balancing for Fat-Tree networks with multipath support," in *Proc. 12th IEEE International Conference on Communications (ICC '13)*, Budapest, Hungary, Jun. 9–13, 2013, pp. 1–5.

[13] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, and J. Zolla, "B4: Experience with a globally-deployed software defined wan," in *Proc. ACM SIGCOMM '13*, Hong Kong, China, Aug. 12–16, 2013, pp. 3–14.

[14] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *Proc. ACM SIGCOMM '13*, Hong Kong, China, Aug. 12–16, 2013, pp. 15–26.

[15] H. Long, Y. Shen, M. Guo, and F. Tang, "LABERIO: Dynamic load-balanced routing in OpenFlow-enabled networks," in *Proc. 27th IEEE International Conference on Advanced Information Networking and Applications (AINA '13)*, Barcelona, Spain, Mar. 25–28, 2013, pp. 291–297.

[16] F. P. Tso and D. P. Pezaros, "Improving data center network utilization using near-optimal traffic engineering," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1139–1148, Jun. 2013.

[17] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of datacenter traffic: Measurement & analysis," in *Proc. ACM Internet Measurement Conference 2009 (IMC '09)*, Chicago, USA, Nov. 4–6, 2009, pp. 202–208.

[18] D. Xu, M. Chiang, and J. Rexford, "Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering," *IEEE/ACM Transactions on Networking*, vol. 19, no. 6, pp. 1717–1730, Dec. 2011.

[19] "Mininet: An instant virtual network on your laptop (or other pc)," http://mininet.org/, [Online; accessed Nov. 27, 2013].

[20] "Open vSwitch: An open virtual switch," http://openvswitch.org/, [Online; accessed Dec. 2, 2013].

[21] "Iperf," http://en.wikipedia.org/wiki/Iperf, [Online; accessed Dec. 2, 2013].

[22] "Floodlight OpenFlow controller," http://www.projectfloodlight.org/floodlight/, [Online; accessed Oct. 23, 2013].