Engineering Traffic Uncertainty in the OpenFlow Data Plane

Fei Chen*, Chunming Wu*, Xiaoyan Hong†, Zhouhao Lu*, Zhouhao Wang*, Changting Lin*
* College of Computer Science and Technology, Zhejiang University
† Department of Computer Science, University of Alabama
{fei_chen, wuchunming, bctlzh, wzh1994, linchangting}@zju.edu.cn hxy@cs.ua.edu

***Abstract*—This paper is driven by a simple question of whether traffic engineering in Software Defined Networking (SDN) can react quickly to bursty and unpredictable changes in traffic demand. The key challenge is to strike a careful balance between the overhead (frequently involving the SDN controller) and performance (the degree of congestion measured as the maximum load and the balance between the minimum and the maximum loads). Exploiting OpenFlow (OF) features, quick shift of routing paths for unpredictable traffic bursty is the focal point of this work. It is achieved by using a dual routing scheme and letting the data plane to select the appropriate path in reacting to uncertainty in traffic load. The proposed work is called DUCE (Demand Uncertainty Configuration sElection). Further, we describe a traffic distribution model, an optimization solution that calculates congestion-free traffic distribution plan which guarantees that each switch can select one of the paths in a distributed way, and moreover, OF details about detaching the functionality of responding to the demand uncertainty from the control plane and delegating it to the data plane. Simulations are performed validating the efficiency of DUCE under various network scenarios.**

I. INTRODUCTION

A. *Traffic Engineering in SDN and Demand Uncertainty*

The simplicity of SDN can alleviate the complexities of traditional Traffic Engineering (TE) mechanisms [1]. The latter often relays on proactive methods, which first collect a set of traffic demands, and then compute optimal routes for these traffic demands. A straightforward use of SDN has been a centralized traffic engineering paradigm. Existing work has shown that centralized TE reduces network congestion and increases efficiency [2]–[6]. In such systems, a TE controller typically takes the form of a control loop that measures the current traffic demand and network topology, solves an optimization problem, and then reconfigures the network to match current traffic demand.

While centralized TE can be highly effective, it is still worthy to ask the question of whether centralized TE can work effectively in the presence of bursty and unpredictable changes in traffic demands (referred to as *demand uncertainty*). Understanding the characteristics of the traffic demand is the key to effective and efficient utilization of link capacities [4], [7]. Theoretically, if the traffic demand is predictable, optimal routes can be obtained to organize the traffic flows in the network to minimize network utilization [8]. However, this approach can only be used if the traffic demand is stable. Unfortunately, traffic can be highly dynamic and may contain unpredictable shifts. Prior studies [1], [9], [10] have shown that the shifts in traffic may leave no time for a proactive TE algorithm to re-compute or adjust. Consequently, packet losses occur due to congestion. One way to deal with the unpredictable traffic shifts is oblivious routing [7]. In oblivious routing, a robust routing for a class of traffic demand is computed, and thus has the potential to handle traffic spikes well. A potential drawback of oblivious routing is that optimizing for the worst-case performance may incur a high cost when traffic is predictable and stable, which may account for a majority of time. These early work show that it is desirable for centralized TE to have strategies able to capture the uncertainty in traffic demands and to balance the traffic load accounting for the uncertainty with adequate link utilization.

B. *Distributed TE in SDN and Implications*

Given a TE strategy dealing with demand uncertainty, we can plot its cost (*e.g.,* the frequency of involving the SDN controller) and performance (*e.g.,* the efficiency of responding to congestion) on a 2-D plane. The desirable strategy, *i.e.,* real-time strategy, will require constantly monitoring network conditions and rapidly reacting to problems [2], [3], [11]. As an example, load balancing via a controller involves timely collecting statistics about flows in the data plane [2], then adaptively rerouting traffic from over-utilized to under-utilized paths. However, frequent flow installation and network-wide statistics collection, may yield significant overhead on both the control plane and the data plane, and consequently limit the scalability of SDN [12]–[14]. *Implication: we should adjust network configurations when demands shift while reducing the overhead on the control plane.*

A natural strategy to reduce overhead would be a distributed one, which moves control functionalities toward distributed control planes [12], [15], where each controller has a partial view of the network. However, applications, such as traffic engineering that requires an up-to-date view of the network to optimize their objective functions, would have to cope with a network of distributed controllers [16] . *Implication: we should deal with demand uncertainty in traffic demand in a distributed and simple way that does not require collaborating and exchanging information with each other.*

The main problem is how to deal with unpredictable traffic spikes in a timely way with less control traffic overhead. Inspired by oblivious routing, we can proactively compute a routing for normal traffic demand and an alternative routing for unpredictable traffic demand, in a way that the latter can handle demand uncertainty. The basic idea, thus, is to reduce the time needed in responding to the sudden changes of traffic load with minimum control overhead.

C. Proposed Approach

We attempt to find a balance between the frequency of involving the SDN controller (*i.e.,* control overhead) and the efficiency of responding to congestion (*i.e.,* network performance), synthesizing existing routing strategies for SDN. Our approach is to consider both proactive and reactive methods as complementary with each other. The proactive methods, *e.g.,* prediction-based TE and oblivious TE, focus on minimizing the maximum link load observed over a period of time. However, they rely on precise knowledge of the traffic demand, and thus may not prepare for handling demand uncertainty. The reactive methods, *e.g.,* real-time routing and distributed control plane, attempt to adjust routing configurations as demands shift. However, their effectiveness is tied to how quickly they adapt to changing demands. The key method we use is to delegate the control function of responding to demand uncertainty to the data plane, with the guideline presented by SDN controller. Thus, while traffic flows are passing through the switches, reactions to changes in traffic demands will occur directly without passing traffic statistics to the controller and waiting for a flow table update.

Based on this insight, we propose a new scheme called DUCE (Demand Uncertainty Configuration sElection). DUCE will solve the following two problems: Can responding to bursty and unpredictable changes in traffic demand be done in the data-path without adding new data-path mechanisms to the switches? How such a distributed data-path-based approach achieve acceptable performance globally?

For the first problem, the underlying issue is that OpenFlow (OF) switch under SDN is dumb. They cannot dynamically change the proportion of traffic that is routed along each path according to the network states. They only explicitly react to commands from the controller. When bursty traffic occurs, they may fail to adjust network configurations in time, and thus cause serious network congestion. To solve this problem, DUCE uses a simple yet effective way: performing only a selection function at each OF switch based on traffic marks. Specifically, DUCE fully exploits the features of OpenFlow, including the meter table and group table. The former measures the rate of packets and remarks the DSCP (Differentiated Services Code Point) field. The latter provides the ability to define multiple forwarding behaviors according to the DSCP mark. The issues relating to measure the bandwidth consumed by each flow and to make decisions based only on these local states will be studied in this paper.

For the second problem, the underlying issue is that there is a potential risk of adjusting the traffic distribution leading to network congestion, because the adjustment of traffic distribution is performed at each switch in an online and distributed fashion. The distributed adjustment has the advantage of responding quickly to changes in traffic, but is limited by lacking of global knowledge of the network state. The oblivious method may have the over-provisioning problem if they attempt to fully cover real-time traffic spikes with fixed bandwidth reservation in their offline approach. To this problem, instead of providing a single routing for all possible uncertain scenarios, DUCE precomputes and uses two routes for each flow in the network: one route is for normal traffic conditions and the other is for unpredictable conditions. The second route is a congestion-free traffic distribution plan, which guarantees that each switch can adjust the distribution of its flows in a distributed way and irrespective of the order in which the distributions are adjusted. The route calculation directs the unpredictable flow demands to links that can accommodate their bandwidth requirements. The solutions to aforementioned first problem allow the installations of these routing configurations to the switches, so the switches handle traffic uncertainty in the data plane. In this paper, we study issues relating to estimating traffic load and compute the congestion-free routes.

DUCE is evaluated through simulations. First, we compare DUCE to standard offline TE approaches, *i.e.,* prediction-based TE and COPE, based on Abilene topology and traffic traces, and show that it can efficiently utilize network bandwidth. Second, we compare DUCE to an online TE approach which periodically shifts flows to least loaded paths, and show that with bursty traffic demand, the traffic distribution under DUCE is stable and load balancing is achieved.

This paper makes the following key contributions.

- Propose the idea of detaching the functionality of responding to demand uncertainty from the control plane and delegating it to the data plane. Moreover, exploit the features of OF to make it happen, so to achieve the goal of enabling fast reaction to uncertainty in traffic load.
- Use a dual routing strategy at the data plane which handles normal case and bursty case separately by two separate policies. Moreover, present an optimization solution that calculates congestion-free traffic distribution plan which guarantees that each switch can select one of the policies in a distributed way and irrespective of the order in which the distributions are adjusted at different switches.
- Evaluate DUCE with comparisons with both offline optimal TE schemes and an online greedy TE scheme. Comparisons to the former show that DUCE can reduce maximum load on bottleneck links, while keep the degree of imbalance over all the links low. Results also show that DUCE follows the traffic demand changes quicker by showing low frequency of traffic fluctuations.

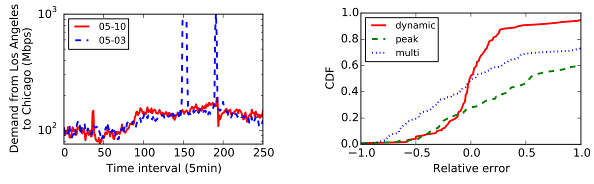
The rest of the paper is organized as followings. Section II presents case studies about traffic uncertainty using Abilene data. Section III introduces the two key ideas of DUCE, namely, selection function and traffic distribution plan. Section IV describes the model and optimization problem for the alternative path that handles traffic uncertainty. Section V presents simulation based evaluations. Sections VI and VII are related work and conclusions respectively.

II. MOTIVATION

Our work is motivated by the following observations, for examples, the demand uncertainty occurs frequently in the network, and TE solutions have to consider how to minimize the impact of demand uncertainty on a specific performance goal, such as minimizing bandwidth consumption and balancing the load distribution.

A. Demand Uncertainty

Understanding the characteristics of the traffic demand is the key to efficient utilization of link capacities. We briefly



(a) Traffic dynamics (b) CDF of relative error

Fig. 1: Traffic dynamics in Abilene data (a) and relative error between the predicted demand and the real demand (b).

present the properties of Abilene traces [17] and MapReduce bandwidth requirement, highlighting specific features that influence DUCE design.

First, we examine the Abilene (Internet2) data to show the traffic demand uncertainty. Figure 1(a) shows the existence of variations in traffic demand over time (the red solid line) and huge traffic spikes while the rest remains relatively stable most of the time (the blue dashed line). Second, we examine the time-varying traffic of MapReduce in Figure 8(a) according to the traffic pattern depicted in [18]. The figure shows that using a fixed-bandwidth reservation can potentially waste the resources of the datacenter.

These observations validate the need to quickly react to demand uncertainty, and the need to avoid over-provisioning. In this work, we strive to achieve a balance between the two while also to offer optimization for bursty traffic.

B. *Analysis of Existing Methods*

We classify exiting TE approaches in terms of the availability of traffic demand or the timescale of operations in two classes: offline TE and online TE. The offline TE class configures the set of link weights that minimize the maximum network utilization with an available and fixed traffic demand matrix. In contrast, the online TE class sets link weights adaptively based on current network conditions, reacting quickly to demand uncertainty.

Most of the proposed offline TE methods are *prediction-based*. The prediction-based TE methods can be further categorized into the following classes [1]: (1) *dynamic*, methods based on the traffic demand in the previous interval; (2) *peak*, methods based on the traffic demand in the peak interval (in terms of the total volume of traffic) of the previous day and the same day of the previous week; and (3) *multi*, methods based on a set of traffic matrices in the previous day and the same day of the previous week. We would like to understand the gap between predicted traffic demand and real traffic demand.

Figure 1(b) shows a comparison in terms of relative error (in the -1 to 1 range) between predicted traffic demand (*e.g., dynamic*, *peak*, and *multi*) and real traffic demand for the Abilene traffic trace. For the *multi* method, we plot the minimum relative error for each IE-pair. The figure shows that both *peak* and *multi* methods have high relative error. For example, the relative error greater than 0.5 for the *peak* and *multi* approaches are 52.5% and 31.4% respectively. As a result, *peak* and *multi* methods typically require bandwidth over-provisioning. On the other hand, the *dynamic* method has a lower fraction above the 0.5 threshold (nearly 10%) than *peak* and *multi* methods. In fact, the fraction of relative error in the range of [-0.2, 0.2] constitutes 63%. Thus, the *dynamic* method has a tight relative error, and may fail to cope with bursty and unpredictable changes in traffic demands.

To account for the gap between predicted traffic demand and real traffic demand, another type of offline TE methods, which referred to as oblivious routing, computes routes that are optimized for the worst-case performance. A potential drawback of oblivious routing is that optimizing for the worst-case performance may incur a high waste when traffic is predictable and stable, which may account for a majority of time. COPE (*cope*) is proposed to combine the best of both prediction-based TE and oblivious routing. It optimizes routing based on the convex hull constructed from the set of traffic matrices collected from the previous day and the same day last week, subjected to a performance-ratio penalty envelope on all nonnegative traffic demands. Similar to *multi*, *cope* also requires bandwidth over-provisioning.

The major weakness of offline TE is the lack of adaptive traffic manipulation according to traffic and network dynamics. In order to rapidly respond to demand uncertainty, online TE class typically performs on a timescale of minutes or even milliseconds in order to evenly distribute traffic within the network promptly. Online TE methods aim to dynamically split traffic among multiple paths, *i.e.,* decides on when and how to shift traffic among the paths, based on the view of the load situation in the network. Specifically, an online TE solution takes the form of a control loop that periodically polls the switches for collecting flow statistics, and then tires to moves traffic from the over-utilized paths to the under-utilized paths until the utilization is balanced. However, since online TE mechanism is implemented at (logically) centralized controller, this would require continually distribute the rules across the affected switches to dynamically adaptive to time-varying network states. This forwarding rule installation process may take time and yield significant overhead at both the control plane and data plane. Indeed it is challenging to build an online TE scheme for SDN that respond quickly to changes in traffic, yet requires no additional interactions between the controller and switches during demand uncertainty. Therefore, there is a need for an online TE method that combines practical implementation with clear performance advantage.

Again, the goal of our work is to rapidly respond to demand uncertainty by, performing traffic distribution on a timescale of minutes or even milliseconds.

III. DUCE CORE IDEAS

As analyzed above, the two fundamental limitations (*i.e.,* high latency and reduced scalability) of the existing SDN-based TE approaches stem from the fact that they process the demand uncertainty in the control plane. In this paper, we propose a new design called DUCE which deals with the demand uncertainty in the data plane. DUCE is based on two key ideas. The first idea is to perform a *selection function* at each OF switch to adjust traffic distribution in an online and distributed fashion. However, adjusting the traffic distribution locally without incurring network congestion is challenging. Thus, our second idea is to combine the selection function with a *traffic distribution plan*, which guarantees that adjusting traffic distribution locally will not incur network congestion.

Algorithm 1 Path selection

```
   At each interval of $t$:
1: Calculate the moving average rate $H$ for every flow $X$.
2: If rate $H$ is less than a pre-defined threshold value $T$,
   select route $r = 0$. Otherwise, select route $r = 1$.
3: If route is find, using the corresponding weight to configure path selection function $Q$.
   At each packet of flow $X$ arrival:
4: Forward packet to path $p$ using traffic splitting algorithm.
5: Record the number of packets that have passed through
   for each time interval of $t$.
```

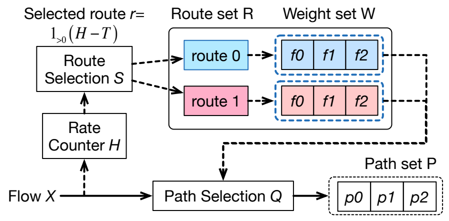


Fig. 2: Path selection.

A. Selection Function

For the selection function, three components are added to the switches as shown in Figure 2. First, a *rate counter*, which monitoring the rate of incoming flows and determining whether there is a spike or not. Then, a *route selection*, when there is a spike in traffic demand, it will switch to alternative route. Finally, a globally optimized *route table*, provides two routes for each source-destination pair. One for normal traffic conditions and the other for unpredictable conditions. As a result, load assigned on each path can be adapted to dynamic changes of the traffic condition.

Selection function is supported using path selection as shown in Figure 2. The input to the path selection function Q is each packet of flow X. The output is a path to which the packet should be assigned according to the load. Path selection for each flow is decided independently. The path selection function can be implemented by using a weight-based traffic splitting algorithm (*e.g.,* WRR, THR, etc [19]). Each path is associated with a weight that determines how much traffic load should be sent via that path as compared with other paths, *i.e.,* traffic splitting ratio. The weights among the available paths are adjusted according to real-time flow rates by the route selection function S. The assignment of weight is based on the arrival rate of flow X: whenever the rate of flow X exceeding a pre-defined rate threshold T, the weight will be assigned with route 0; otherwise, it will be assigned with route 1. When a flow arrival rate changes, the weight is adjusted. Therefore, load assigned on each path can be adapted to dynamic changes of the traffic condition. Congestion induced by bursty and sudden spikes in traffic rate can be mitigated. The value of flow rate H can be found by using an exponentially weighted moving average function: $H = (1 - \alpha)\, H_{old} + \alpha H_{new}$. The value of T determines the frequency of updating weight assignment.

The procedure for this basic idea is summarized in Algorithm 1. First, the weight assignment will be updated at each time interval t: calculates the moving average rate for flow X (Step 1), determines which weight to be chosen (Step 2) and updates the weight assignment (Step 3). Second, when a packet of flow X arrives, it will be sent via a path selected according to its weight (Step 4). The corresponding counter will be updated (Step 5).

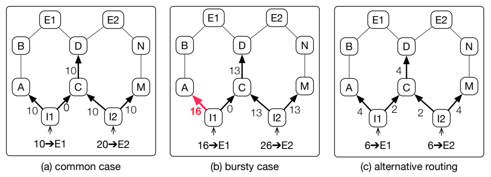


Fig. 3: This example shows how to perform a dynamic traffic distribution through incremental router reconfiguration.

B. Traffic Distribution Plan

We illustrate by example that how changes in traffic lead to congestion and how to prevent the congestion through a carefully designed traffic distribution plan.

Figure 3(a) shows a simple network where the capacity of each link is 15. The numbers besides the ingress switches are traffic demands and the arrow indicates the traffic direction. The number on each link is the traffic load. This figure shows a normal traffic demand and the optimal traffic distribution where an offline algorithm uses a multi-commodity flow linear programming to optimally split the traffic among the links. For example, the load on link I1 $\rightarrow$ A is 10, all the demand of I1 $\rightarrow$ E1, while the load on link I2 $\rightarrow$ C and I2 $\rightarrow$ M are 10, half of demand of I2 $\rightarrow$ E2. No link is congested.

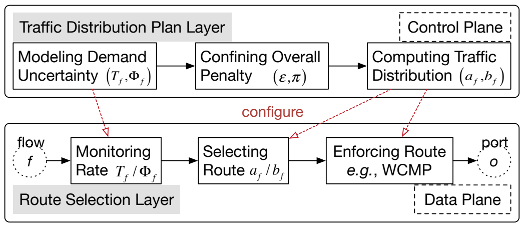
Suppose there is a sudden spike in traffic demand, as shown in Figure 3(b). The demand of I1 $\rightarrow$ E1 and I2 $\rightarrow$ E2 increases to 16 and 26, respectively. However, applying the routing plan of common case to the bursty traffic demand, *i.e.,* the splitting ratio used in Figure 3(a) to the case in Figure 3(b), will break the load balance of the network. As a result, link I1 $\rightarrow$ A will have a load of 16, exceeding its capacity.

To prevent the preceding congestion, we need to change the routing, *i.e.,* the traffic split ratio on both $I1$ and $I2$. While a traditional solution is to change the splitting ratio of the common case in order to handle the bursty case when it occurs, our approach is to introduce a separate traffic distribution plan for increased demand. Figure 3(c) shows such an alternative routing, where $I1$ splits traffic by 4:2 and $I2$ splits traffic by 2:4. It is easy to verify that no link is congested.

IV. DUCE Design

DUCE separates the functionality of responding to demand uncertainty from the control plane and delegates to the data plane. Thus, DUCE consists of two layers as shown Figure 4: the first layer, traffic distribution plan (TDP), computes routing for the common case demand and the unpredictable case demand; while the second layer, route selection (RS), achieves dynamic transition between two different routes (forwarding behaviors).

TDP uses a simple model to capture the traffic demand of each flow f: the traffic demand of f is T_f at common case, while it is upper bounded by Φ_f. Instead of optimizing the routing for all possible traffic demand, TDP seeks to provide different routings for different traffic demand. Given a predicted traffic demand T_f and a possible peak traffic demand



Traffic Distribution Plan Layer
Control Plane
Modeling Demand Uncertainty (T_f, Φ_f)
Confining Overall Penalty (ε, π)
Computing Traffic Distribution (a_f, b_f)
configure
flow f
Monitoring Rate T_f / Φ_f
Selecting Route a_f / b_f
Enforcing Route e.g., WCMP
port o
Route Selection Layer
Data Plane

Fig. 4: The high-level working process of DUCE.

Φ_f, TDP will attempt to compute a common case traffic distribution a_f for T_f and a worst case traffic distribution b_f for $\Phi_f - T_f$. Considering that optimizing performance for worst case may come at the expense of poor performance for common case, TDP ensures the performance of routing a_f to be slightly higher than optimal under common case, specified by parameter ε; and uses the parameter π to control how far the worst case traffic demand is away from the common case.

The TDP layer feeds the computed routing a_f and b_f to the RS layer. For each flow f, the RS layer on the OF switch uses a rate-limiter to limit the traffic. The rate-limit is assigned by the TDP layer, *i.e.,* T_f and Φ_f. RS aims to transition from a_f to b_f when traffic rate changes from T_f to Φ_f. Specifically, when the rate of flow f is greater than T_f, RS will change the corresponding rules from a_f to b_f on involved switches. To enforce traffic distribution (*i.e.,* a_f or b_f), Weighted-Cost-Multi-Path (WCMP) is used to direct a matching packet to an output port. DUCE leverages the hash function (*i.e.*, WCMP) on the OF switches. Doing so, DUCE can adjust the bandwidth among different flows to achieve high network utilization, while avoiding the congestion caused by uneven flow assignment.

A. Network Model

We first describe a network model under which we formally define the traffic demand as the input to DUCE TDP layer.

1) Traffic Demand: Let $G = (V, E)$ be a network under consideration, where V is the set of switches, and E is the set of links connecting the switches. Let c_e or c_{ij} denote the capacity of a directed link e_{ij} from switch i to switch j. Let $F = \{f\}$ be the set of flows aggregated by ingress-egress switches. A flow f enters the network from an ingress to an egress switch through one or multiple paths (H_f). A traffic demand T defines the size of each flow T_f.

2) Traffic Distribution: Let $a_{f,h}$ be the bandwidth allocated for flow f on path h. A traffic distribution, denoted by $\vec{A}$, is specified by a set of values $\{a_{f,h} | f \in F, h \in H_f\}$. Given a traffic demand T, the traffic distribution $\vec{A}$ can be computed by using a multi-commodity flow linear program, as follows:

$$\min \quad MLU_T \tag{1a}$$
$$subject\ to:$$
$$\forall e \in E: \frac{\sum_{f \in F, h \in H_f} a_{f,h} \delta[h,e]}{c_e} \le MLU_T \tag{1b}$$
$$\forall f \in F: \sum\nolimits_{h \in H_f} a_{f,h} = T_f \tag{1c}$$
$$\forall f \in F, h \in H_f: 0 \le a_{f,h} \le T_f \tag{1d}$$

Where the binary indicator $\delta[h,e]$ denotes if path h traverse link e. The objective in the formulation is to minimize the maximum link utilization (MLU). The second condition specifies that all demands of a flow should be routed.

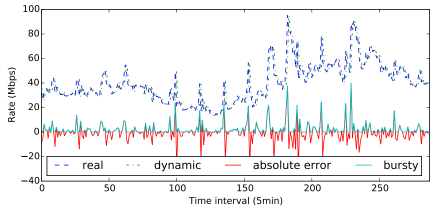


Fig. 5: The bursty line captures the burst traffic demand when using predicted traffic demand (*dynamic*) to approximate the real traffic demand (*real*).

3) Flow Rule: Given a traffic distribution $\vec{A}$, we can compute the rule set R for $\vec{A}$. Let r_{ij}^f be the rule for flow f on link e_{ij}, where $r_{ij}^f = \sum_{h \in H_f} a_{f,h} \delta[h, e_{ij}] \Big/ T_f$. A rule set R is specified by a set of values $\left\{ r_{ij}^f | f \in F, e_{ij} \in E \right\}$.

B. Modeling Demand Uncertainty

We assume that the demand of each flow f is unknown and varies with time, but that always lies inside an explicitly defined region, *i.e.,* $[T_f, \Phi_f]$. Note that Φ_f is the bound for the unpredictable traffic, and independent of time; if we have a better understanding of the traffic, we can have a tight bound. Let θ_f be the set of time when there exists changes in flow f. Given θ_f, we can model demand uncertainty of flow F as follow:

$$S_f(t) = \begin{cases} X_f & : t \in \theta_f,\ X_f \in (T_f, \Phi_f] \quad (2a) \\ T_f & : otherwise \quad (2b) \end{cases}$$

Function 2 attempts to model demand uncertainty into pulse functions with varying height and width peaks. To see how this model capture traffic demand uncertainty, we examine the Abilene data. Figure 5 plots the real traffic demand and predicted traffic demand (by using *dynamic* method) from Los Angeles to Indianapolis. To capture the traffic demand, we can use the predicted traffic demand as the lower bound, *i.e.,* T_f; and use the absolute error of *dynamic* method as the unpredictable demand, *i.e.,* $X_f - T_f$, where absolute error is defined as the predicted demand minus the real demand. Since the value of $X_f - T_f$ in our model is always greater than 0, we then set the value of $X_f - T_f$ as $\max\{absolute\ error, 0\}$ (the bursty line in Figure 5).

However, Function 2 needs prior knowledge on when bursty will happen. To reduce this assumption, we introduce a vector $\varphi(t) = [\varphi_{t,f} | f \in F]$ that indicates the status of each flow f at time t, where $\varphi_{t,f} = S_f(t)/\Phi_f$; and a slave variable π that bounds the total number of changes in flows. Then we covert function 2 into the following constraint:

$$S_f(t) = \begin{cases} X_f & : \|\varphi(t)\|_1 \le \pi,\ X_f \in (T_f, \Phi_f] \quad (3a) \\ T_f & : otherwise \quad (3b) \end{cases}$$

where $\|\varphi(t)\|_1 = \sum_{f \in F} \varphi_{t,f}$.

C. Computing Traffic Distribution

The basic idea is that we do not change the distribution of a flow f across its available paths as long as the bandwidth consumed by this flow below a certain threshold, *i.e.*, T_f. Only when the bandwidth consumption of this flow exceeds T_f, will the distribution for it be adjusted. The main reason for limiting the number of routings to 2 is that we want to limit the amount of rules installed in the memory used in OpenFlow switch.

A key challenge is to implement these adjustments without causing congestion. The underlying problem is that the adjustment of traffic distribution is performed at each switch in an online and distributed fashion. This has the advantage of responding quickly to changes in traffic, but is limited by lacking of global knowledge of the network state. Hence, there is a risk of collision, moving traffic to over-utilized paths, and thus leading to congestion. To avoid congestion distribution adjustments, DUCE computes a congestion-free traffic distribution plan, which guarantees that each switch can adjust the distribution on its flows in a distributed way and irrespective of the order in which the distributions are adjusted.

To compute such a distribution plan, we first remove the time variable in Function 3. Since both Φ_f and T_f are independent of time, we can convert demand uncertainty model into time independent demand set as follow:

Lemma 1: Function 3 can be convert into the sum of T and $\Delta_\pi =$

$$\left\{\Delta_f = X_f{}' - T_f | X_f{}' \in [T_f, \Phi_f], \sum\nolimits_{f\in F} \frac{X_f{}' - T_f}{\Phi_f - T_f} \le \pi\right\}$$

where T is normal traffic demand, and Δ_π is bursty demand.

Proof: From the definition of $\varphi_{t,f}$ and $S_f(t)$, we have:

$$\frac{X_f{}' - T_f}{\Phi_f - T_f} \le \frac{X_f{}'}{\Phi_f} = \frac{S_f(t)}{\Phi_f} = \varphi_{t,f} \tag{4}$$

Thus, Δ_f captures the increased demand of flow f. Then we can convert Function 3 into: $S_f(t) = T_f + \Delta_f$. ■

Given Δ_π, we denote $\vec{B}$ the traffic distribution for traffic demand Δ_π, where $\vec{B} = \left\{b_{f,h} | f \in F, h \in H_f, \sum_{h\in H_f} b_{f,h} = \Delta_f, b_{f,h} \ge 0\right\}$. Then, we can replace constraints 1a and 1b with:

$$\min \quad MLU_S \tag{5a}$$

$$\forall e \in E : \frac{\sum_{f\in F, h\in H_f} (a_{f,h} + b_{f,h})\, \delta[h,e]}{c_e} \le MLU_S \tag{5b}$$

To solve (5), we then bound the load on each link offered by increased demand with the following lemma.

Lemma 2: Let $\psi_e = \max\limits_{\Delta_\pi} \sum_{f\in F, h\in H_f} b_{f,h}\delta[h,e]$, if the distribution $\{a_{f,h}, b_{f,h}\}_{\forall f\in F, h\in H_f}$ can guarantee that: $\forall e \in E : \sum_{f\in F,h\in H_f} a_{f,h}\delta[h,e] + \psi_e \le c_e$, then each switch can independently adjust its distribution on its flows without causing congestion.

Proof: Denote $\omega_{f,h} = b_{f,h} \Big/ \sum_{h'\in H_f} b_{f,h'}$. Let d_f and d_h be the instantaneous rate of flow f and path h respectively. When d_f exceeds T_f, *i.e.,* $T_f \le d_f \le \Phi_f$, the switch adjusts the distribution of flow f from $a_{f,h}$ to $b_{f,h}$, the traffic load of flow f on a path h is:

$$d_h = a_{f,h} + (d_f - T_f)\,\omega_{f,h} = a_{f,h} + \Delta_f \omega_{f,h} \tag{6}$$

which is directly derived from the definition of Δ_π, *i.e.,* $\exists \Delta_\pi : (d_f - T_f) \in \Delta_\pi$. Therefore, the total traffic on a link e is:

$$\sum\nolimits_{f\in F,h\in H_f} d_h \delta[h,e] \le \sum\nolimits_{f\in F,h\in H_f} a_{f,h}\delta[h,e] + \psi_e \le c_e$$

which finishes the proof. ■

Using linear programming duality, we can show that the $\psi_e \le \vartheta$ if and only if the following set of constraints can be satisfied:

$$\sum\nolimits_{f\in F} \mu_{e,f} + \lambda_f \pi \le \vartheta \tag{7a}$$

$$\forall f \in F : \frac{\mu_{e,f} + \lambda_e}{\Phi_f - T_f} \ge \sum\nolimits_{h\in H_f} b_{f,h}\delta[h,e] \tag{7b}$$

$$\forall f \in F : \mu_{e,f} \ge 0 \tag{7c}$$

$$\lambda_e \ge 0 \tag{7d}$$

The variables $\mu_{e,f}$ and λ_e are dual multipliers on the constraint $\Delta_f \le \Phi_f - T_f$ and $\sum_{f\in F} \Delta_f/(\Phi_f - T_f) \le \pi$, respectively. We can then covert (1) into a set of linear constraints as following:

$$\min \quad MLU_S \tag{8a}$$

$$subject\ to:$$

$$\forall e \in E:$$

$$\frac{\sum_{f\in F,h\in H_f} a_{f,h}\delta[h,e] + \sum_{f\in F}\mu_{e,f} + \lambda_f \pi}{c_e} \le MLU_S \tag{8b}$$

$$\forall e \in E : \frac{\sum_{f\in F,h\in H_f} a_{f,h}\delta[h,e]}{c_e} \le \varepsilon \cdot MLU_T \tag{8c}$$

$$\forall e \in E, f \in F : \frac{\mu_{e,f} + \lambda_e}{\Phi_f - T_f} \ge \sum\nolimits_{h\in H_f} b_{f,h}\delta[h,e] \tag{8d}$$

$$\forall f \in F : \sum\nolimits_{h\in H_f} a_{f,h} = T_f ; \sum\nolimits_{h\in H_f} b_{f,h} \le \Phi_f - T_f \tag{8e}$$

$$\forall f \in F, h \in H_f : 0 \le a_{f,h} \le T_f ; 0 \le b_{f,h} \le \Phi_f - T_f \tag{8f}$$

$$\forall f \in F, e \in E : \mu_{e,f} \ge 0 \tag{8g}$$

$$\forall e \in E : \lambda_e \ge 0 \tag{8h}$$

Constraint (8c) bounds the penalty on normal traffic demand by $\varepsilon \cdot MLU_T$, and (8e) guarantees that all the traffic of T is fully delivered and total traffic delivered should not exceed Φ. Constraint (8b) guarantees the worst case performance.

D. Route Selection and Implementation

The output of DUCE TDP layer is a sequence of traffic distributions, each of which can be implemented by installing its corresponding flow table entries into the switches. Given a traffic distribution $\vec{A}$, we then compute the rule set R_1 for $\vec{A}$. As defined before, the rule set R_1 is specified by a set of values $\left\{r_{ij}^f | f \in F, e_{ij} \in E\right\}$, where $r_{ij}^f = \sum_{h\in H_f} a_{f,h}\delta[h, e_{ij}] \Big/ T_f$. Then the rule set should be installed into a switch i is given by $\left\{r_{ij}^f | \forall f \in F, \forall j \in V, e_{ij} \in E\right\}$. Similarly, we compute a rule set R_2 for distribution $\vec{B}$, where $R_2 = \left\{r_{ij}^f = \sum_{h\in H_f} b_{f,h}\delta[h, e_{ij}] \Big/ \sum_{h'\in H_f} b_{f,h'} | e_{ij} \in E\right\}$.

We assume that each switch has a group table and a meter table. Rules in the group table include an ordered list of action buckets. Each action bucket contains a set of actions to execute, and provides the ability to define multiple forwarding

Algorithm 2 Route Selection on Switch i

```
When switch $i$ receives a packet of flow $f$:
1: A meter measures the rate of flow $f$.
2: If current rate is greater than $T_f$, remark the DSCP field.
3: If DSCP is remarked, select route $R_1$; otherwise, $R_2$.
4: If route is $R_1$, select outport 1; otherwise 2.
```

Table 0

Match	Instructions
SrcPrx = I1, DstPrx = E1	Meter 1, Table 1

Table 1

Match	Instructions
SrcPrx = I1, DstPrx = E1, DSCP = 0	Group 1
SrcPrx = I1, DstPrx = E1, DSCP = 1	Group 2

Meter Table

Meter ID	Band Type	Rate	Parameter
1	DSCP remark	10	1

Group Table

Group ID	Group Type	Bucket1 Weight	Bucket2 Weight	Bucket1 Action	Bucket2 Action
1	WCMP	1.0	0.0	Fwd A, Vlan = 1	Fwd C, Vlan = 1
2	WCMP	2/3	1/3	Fwd A, Vlan = 2	Fwd C, Vlan = 2

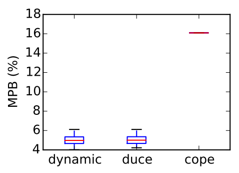
Fig. 6: Implementing DUCE on OpenFlow switch.

behaviors. r_{ij}^{f} can be implemented on switch i by using group table with type WCMP. The meter table measures the rate of packets and remarks the DSCP filed. Algorithm 2 describes how a packet of flow f will be processed within a switch i. When a packet of flow f arrives, it will be processed as follows: First, the meter table decides whether DSCP field of this packet should be remarked (Step 1 and Step 2). Second, the group table select an outport, which depends on whether DSCP field has been remarked.

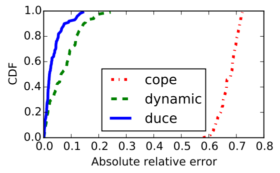
Figure 6 shows an example of how to implement DUCE on an OF switch. For simplicity, we have replaced the names of ports with the switches they are connected to, *e.g.,* in place of the name of the port connecting I1 to A, we simply write A. The OF pipeline processing always starts at the first flow table (*i.e.,* Table 0): the packet must be first matched against flow entries of flow table 0. It consists of a match that specifies packet attributes (*e.g.,* packet aggregated by ingress-egress switch I1 $\rightarrow$ E1) and a list of instructions that specify how to process matching packets. In this case, the rule states that all packets coming from ingress switch I1 to egress switch E1 should be processed by the entry with identifier 1 in meter table and then processed by Table 1. Thus, when a packet arrives, it is first processed by meter band with type DSCP remark specified by the 1-the meter entry, which will increase the value of DSCP to 1 if the current rate is higher than 10. Then, the packet header is matched against flow entries of flow table 1. Table 1 determines which forwarding behavior (defined by group table) to adopt by using DSCP field. Finally, group table uses the WCMP group type to forward the packet to the next hop. As a result, the forwarding behavior of the packet is adjusted when the flow arrival rate changes. For ease of explanation, we assume that the default value of DSCP filed in the header of each packet is 0. It can be easily extend to general cases.

V. EVALUATION

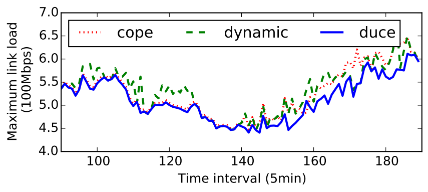
In this section, we evaluate DUCE by simulations. We first compare DUCE with offline TE approaches and show that it can efficiently utilize network bandwidth. Then we compare DUCE with an online TE approach and show that within a network that has bursty traffic demand, the traffic distribution under DUCE is stable and load balancing is achieved.



(a) MPB



(b) CDF of absolute relative error



(c) Time series of maximum link load

Fig. 7: Comparison with offline TE.

A. Comparison with Offline TE

1) Traffic demands and topologies: Similarly to [1], we use the real topology and traffic demand of Abilene [17] for comparison with offline TE methods.

2) Performance metrics: We use the following two performance metrics to compare different offline algorithms: maximum provided bandwidth (*MPB*), defined as

$$MPB\left(\vec{A}\right) = \max_{e\in E} \sum\nolimits_{f\in F, h\in H_f} a_{f,h}\delta\left[h,e\right] \qquad (9)$$

and maximum link load [10], defined as

$$MLL\left(\vec{A}, D\right) = \max_{e\in E} \sum_{f\in F, h\in H_f} \frac{D_f a_{f,h}}{\sum_{h'\in H_f} a_{f,h'}} \delta\left[h,e\right] \qquad (10)$$

where D is the set of real traffic demand of each flow D_f. Given the maximum provided bandwidth MPB and the maximum link load *MLL*, the maximum link utilization is given by $MLU = MLL/MPB$ (for simplicity we assume here that the capacity of each link is given by *MPB*). Instead of using *MLU* as performance metric, we use *MLL* and *MPB* separately because each reveals unique performance aspects that matter to DUCE design. Specifically, *MPB* captures the ability of a TE scheme for capacity planning (*i.e.,* bandwidth reservation), while *MLL* captures the extent of congestion given the *MPB*. Minimizing the *MPB* leads more available bandwidth, which can be used by other flows. Confining the *MLL* helps reducing the occurrence of packet losses and retransmissions.

3) Alternative algorithms: We consider *dynamic, peak, multi,* and *cope*, as discussed earlier. We solve the corresponding linear programming problems with MOSEK [20]. Our simulations show that both *peak* and *multi* algorithms perform poorly, while *dynamic* achieves close-to-optimal performance most of the time. Thus in the following presentations, we omit the results of both algorithms, and report only the results of *dynamic* and *cope* algorithms.

4) Bandwidth provisioning: Figure 7(a) compares the algorithms in terms of *MPB* (*i.e.,* reserved bandwidth). Since *cope* attempts to reduce the worst-case penalty, and thus requiring a high bandwidth provisioning ($\sim$16% of link capacity); while *duce* is only slightly higher than *dynamic*.

5) Maximum link load: The traffic of 10 May is used as the input of the maximum link load for the comparisons of the algorithms. Figure 7(c) plots the maximum link load versus the time interval. For clarity, we zoom in to a few intervals in the day. We make the following observations. First, *duce* show lower *MLL* than *dynamic* at most of the time intervals. As we saw before (Figure 1(b)), the predicted error greater than 0.2 of *dynamic* can be as high as 36.5% by combining 16.1% (relative error less than -0.2) and 20.4% (relative error greater than 0.2), thus resulting a higher performance penalty than *duce*. Second, the maximum link load of *duce* and *cope* are close before the time interval of 140. However, after the time interval of 140, *duce* achieves better performance than *cope*.

For clarity, we plot the CDF of absolute relative error between *MPB* and *MLL* for these algorithms, as shown in Figure 7(b). The absolute relative error of a given *MLL* with respect to a certain MPB measures how far is the *MLL* from being optimal, *i.e.,* minimizing the impact of demand uncertainty given the constraint of bandwidth consumption. It is defined as the relative error between *MPB* and *MLL* divided by the *MPB*. Formally, the absolute relative error is $|(MLL - MPB)/MPB|$. Correspondingly, the closer to 0 the value of absolute relative error means the better of the performance. We can find that *duce* has better performance in terms of absolute relative error than both *dynamic* and *cope*.

In summary: (1) For the same traffic demands, *duce* can deal with traffic uncertainty compared to *cope*, but requires only one-fourth bandwidth provisioning. (2) For the same bandwidth provisioning, *duce* can adapt to traffic uncertainty when *dynamic* cannot.

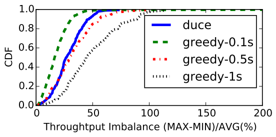
B. Comparison with Online TE

1) Traffic demands and topologies: We use a simple topology as shown in Figure 3, where link (C,D) is shared between paths from pairs (I1,E1) and (I2,E2). Thus, there is an interaction between the pairs, which captures the main feature of online algorithms. We write a packet level discrete-event simulator in Python to conduct this experiment. Figure 8(a) shows the packet arrival rates in our experiments, which is generated according to the traffic pattern depicted in [18], which captures the bandwidth requirement of MapReduce appellations with a coarse-grained pulse functions with different widths and heights. For clarity, we only show a few intervals from 20s to 80s of our experiments.

2) Alternative algorithms: We consider a greedy online algorithm (*greedy*): At each time interval t, for each flow, *greedy* searches all possible paths linearly to find one which is least loaded. If such a path is found, then that flow is placed on the path. This algorithm models online load balancing schemes used in MATE [10], TeXCP [9], Hedera [2], and CONGA [21].

3) Performance metrics: We use the following two metrics to compare *duce* with greedy: *throughput imbalance*, defined as the maximum throughput minus the minimum divided by the average, *i.e.,* $(MAX - MIN)/AVG$; and *stability*, defined by the frequency of load changes.

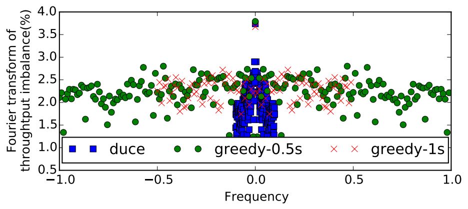
4) Overall performance: Figure 8 compares the algorithms in terms of link load. Figure 8(b) shows that as traffic demand



(a) Extent of imbalance

	(A,B)	(C,D)	(M,N)
duce	46.4	73	69
greedy 1s	89.5	166	100
greedy 0.5s	65.1	103.1	83.5

(b) Standard deviation



(c) Fourier transform of throughput imbalance

Fig. 9: Throughput imbalance and stability.

changing, *greedy* with the frequency of 1s cannot react to changing demand. It is expected that *greedy* method increases the frequency of path adjustment. As shown in Figure 8(c), when *greedy* increases its frequency to 0.5s, the performance of *greedy* improves significantly. In contrast, *duce* (Figure 8(d)) reacts in realtime to the actual demand and performs comparably with *greedy* of 0.5s. It means that *greedy* need to run every 500ms to approach the performance of *duce*. However, rerouting traffic to the least loaded link too frequently may cause oscillations. Figure 9(b) shows the standard deviation of load on each link. The result indicates that *duce* can achieve comparable and better performance with *greedy* for 0.5s and slower update frequencies while prevent oscillations.

5) Throughput imbalance: Figure 9(a) shows the CDF of the *throughput imbalance* across the 3 links (*e.g.,* (A,B), (M,B), (C,D)). This is calculated during the time interval (10s, 100s). The results show that *duce* is slightly better than *greedy* with 0.5 seconds adjustment interval.

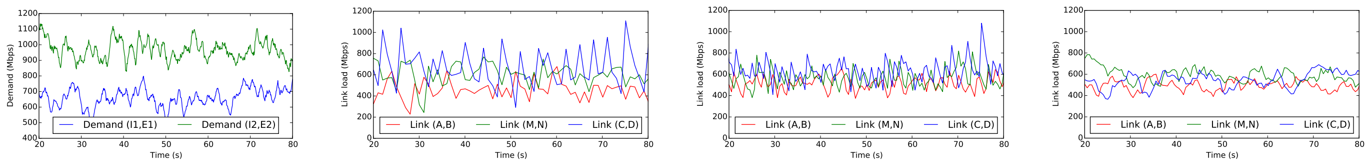
6) Stability: To quantify the stability, we use the Fourier transform to convert the results of throughput imbalance into the frequency domain, see Figure 9(c). The figure shows that the oscillation frequencies of *duce* are closely centered together while those of the two cases of *greedy* are scatted widely. The observation suggests that *duce* produce more stable traffic flows than the *greedy* algorithm.

In all, considering achieving the same throughput imbalance, *duce* can produce stable flows than *greedy*. Moreover, *duce* can achieve acceptable performance under bursty traffic.

VI. RELATED WORK

For brevity, we focus below on centralized flow management techniques supported by SDN. Existing traffic engineering schemes can be mainly classified into two types: the proactive methods and the reactive methods. Each type of method achieves a different tradeoff between data-plane performance and control-plane overhead.

Reactive. The reactive methods will constantly monitoring the network conditions and reacting to traffic changes accordingly. Hedera [2] detects elephant flow at the edge switches and finds an alternative path with sufficient capacity for each elephant flow. The performance of Hedera is constrained by



(a) demand (b) greedy 1s (c) greedy 0.5s (d) duce

Fig. 8: Comparison with online TE. Synthesized traffic demand (a) and overall performance in terms of link load (b, c, and d).

the rates and durations of flows. DevoFlow [3] delegates the handling of mice flows to the data plane, and therefore reduces the number of interactions between the controller and switches. Both Hedera and DevoFlow involve controller in maintaining an accurate and global view of the traffic demands, and rerouting the traffic. However, it is differs from the existing work by performing these all in the data plane, without involving the controller.

Proactive. The proactive methods typically rely on prior knowledge of the traffic demand observed over a period time. Prediction based methods are used to optimize the overall performance over most of traffic demands [1]. The oblivious routing [7] optimizes the worst-case performance over all traffic demands. MicroTE [4] performs multipath routing by leveraging the short term and partial predictability of the traffic demand. SWAN [5] focuses on control when and how much traffic each service sends. Both MicroTE and SWAN perform traffic distribution based on a global view of traffic demands and use OF to coordinate scheduling. As a hybrid approach, DUCE proactively computes two routing plan: one for normal conditions and the other for unpredictable situations. However, its path selection is reactive, triggered by and performed at the data plane.

VII. CONCLUSION

We have presented the design and evaluation of DUCE. DUCE delegates the functionality of responding to the bursty and unpredictable changes in traffic demands to the data plane. It does not require any changes to the OF switch by fully utilizing the meter table and group table. In comparison to the existing approaches, DUCE achieves a better utilization than standard offline TE approaches; and it also produces a more stable and load balanced network when the bursty traffic demands occur. In our future work, we plan to realize it in a deployed network testbed to further validate its benefits.

VIII. ACKNOWLEDGEMENT

This research is supported by the National Basic Research Program of China (2012CB315903), the Program for Key Science and Technology Innovation Team of Zhejiang Province (2011R50010-21, 2013TD20), 863 Program of China (2015AA015602, 2015AA016103), the National Natural Science Foundation of China (61379118), and the Research Fund of ZTE Corporation.

REFERENCES

[1] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, “COPE: Traffic Engineering in Dynamic Networks,” in *SIGCOMM*. ACM, Aug. 2006, pp. 99–110.

[2] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, “Hedera: Dynamic Flow Scheduling for Data Center Networks,” in *NSDI*. USENIX Association, Apr. 2010.

[3] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, “DevoFlow: Scaling Flow Management for High-performance Networks,” in *SIGCOMM*. ACM, 2011, pp. 254–265.

[4] T. Benson, A. Anand, A. Akella, and M. Zhang, “MicroTE: Fine Grained Taffic Engineering for Data Centers,” in *CoNEXT*, Dec. 2011, pp. 8:1–8:12.

[5] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, “Achieving High Utilization with Software-Driven WAN,” in *SIGCOMM*, New York, Aug. 2013, p. 15.

[6] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, “B4: Experience with a Globally-Deployed Software Defined WAN,” in *SIGCOMM*, New York, Aug. 2013, p. 3.

[7] D. Applegate and E. Cohen, “Making Routing Robust to Changing Traffic Demands: Algorithms and Evaluation,” *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1193–1206, Dec. 2006.

[8] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, “Deriving Traffic Demands for Operational IP Networks: Methodology and Experience,” *IEEE/ACM Transactions on Networking (ToN)*, vol. 9, no. 3, pp. 265–280, 2001.

[9] S. Kandula, D. Katabi, B. Davie, and A. Charny, “Walking the Tightrope: Responsive Yet Stable Traffic Engineering,” in *SIGCOMM*, New York, Aug. 2005, pp. 253–264.

[10] A. Elwalid, C. Jin, S. H. Low, and I. Widjaja, “MATE: MPLS Adaptive Traffic Engineering,” *INFOCOM*, pp. 1300–1309, 2001.

[11] J. Rasley, B. Stephens, C. Dixon, E. Rozner, W. Felter, K. Agarwal, J. Carter, and R. Fonseca, “Planck: Millisecond-scale Monitoring and Control for Commodity Networks,” in *SIGCOMM*, Aug. 2014.

[12] S. Hassas Yeganeh and Y. Ganjali, “Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications,” in *HotSDN*. ACM, 2012, pp. 19–24.

[13] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, “On Scalability of Software-Defined Networking,” *Communications Magazine, IEEE*, vol. 51, no. 2, pp. 136–141, 2013.

[14] S. Schmid and J. Suomela, “Exploiting Locality in Distributed SDN Control,” in *HotSDN*. ACM, 2013, pp. 121–126.

[15] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama *et al.*, “Onix: A Distributed Control Platform for Large-scale Production Networks,” in *OSDI*, vol. 10, 2010, pp. 1–6.

[16] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, “Logically Centralized? State Distribution Trade-offs in Software Defined Networks,” in *HotSDN*. ACM, 2012, pp. 1–6.

[17] Abilene, “http://www.cs.utexas.edu/~yzhang/research/AbileneTM/.”

[18] D. Xie, N. Ding, Y. C. Hu, and R. Kompella, “The Only Constant is Change: Incorporating Time-Varying Network Reservations in Data Centers,” in *SIGCOMM*, New York, Aug. 2012, p. 199.

[19] S. Prabhavat, H. Nishiyama, N. Ansari, and N. Kato, “On Load Distribution over Multipath Networks,” *IEEE Communications Surveys and Tutorials*, vol. 14, no. 3, pp. 662–680, 2012.

[20] MOSEK, “http://mosek.com.”

[21] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese, “CONGA: Distributed Congestion-aware Load Balancing for Datacenters,” in *SIGCOMM*. ACM, Aug. 2014, pp. 503–514.