Intra-Data-Center Traffic Engineering with Ensemble Routing

Ziyu Shao*, Xin Jin+, Wenjie Jiang◇, Minghua Chen*, and Mung Chiang+
* The Chinese University of Hong Kong, + Princeton University, ◇Google
Email: {zyshao, minghua}@ie.cuhk.edu.hk, xinjin@cs.princeton.edu, chiangm@princeton.edu, joe.wenjie.jiang@gmail.com

***Abstract*—Today's data centers are shared among multiple tenants running a wide range of applications. These applications require a network with a scalable and robust layer-2 network management solution that enables load-balancing and QoS provisioning. Ensemble routing was proposed to achieve management scalability and robustness by using Virtual Local Area Networks (VLANs) and operating on the granularity of flow ensembles, *i.e.* group of flows. The key challenge of intra-data-center traffic engineering with ensemble routing is the combinatorial optimization of VLAN assignment, *i.e.*, optimally assigning flow ensembles to VLANs to achieve load balancing and low network costs. Based on the Markov approximation framework, we solve the VLAN assignment problem with a general objective function and arbitrary network topologies by designing approximation algorithms with close-to-optimal performance guarantees. We study several properties of our algorithms, including performance optimality, perturbation bound, convergence of algorithms and impacts of algorithmic parameter choices. Then we extend these results to variants of VLAN assignment problem, including interaction with TCP congestion and QoS considerations. We validate our analytical results by conducting extensive numerical experiments. The results show that our algorithms can be tuned to meet different temporal constraints, incorporate fine-grained traffic management, overcome traffic measurement limitations, and tolerate imprecise and incomplete traffic matrices.**

I. Introduction

In recently years, various ways to leverage multi-path routing in Ethernet-based data center networks [1]–[4] have been proposed for scalable traffic management. The use of multi-path routing raises the need of intelligent traffic engineering and flow control. *Ensemble routing* [4] provides an efficient way to dynamically manage a huge amount of traffic flows in large-scale data center networks with arbitrary topology. The core idea of ensemble routing is to operate on the granularity of flow ensembles, rather than individual flows, by adopting Hash-Based Routing (HBR). A flow ensemble is a collection of flows, each of which follows the same routing path. Each flow, identified by the tenant ID and packet header tuple, is classified into a *traffic class*, which determines the QoS treatment for the flow, and a *hash class*, which is simply calculated by a hash function. The traffic class and the hash class determine the *routing class* of the flow, *i.e.*, the flow ensemble the flow belongs to. Ensemble routing greatly reduces management overhead and is scalable to manage a huge amount of traffic flows. Compared to other data center traffic management schemes, ensemble routing is also more promising for OpenFlow based Software Defined Networks (SDN) due to its ability to perform hash-based routing [4].

Ensemble routing makes use of Virtual Local Area Networks (VLANs) [5], [6] to provide multiple routing networks, similar to [3]. VLANs are extensively used in enterprise networks to improve Ethernet scalability, where each VLAN constitutes a subnet that behaves logically like a conventional LAN but is independent of the physical locations of the hosts. Each VLAN constitutes a separate broadcast domain, and a separate spanning tree rooted at a switch is constructed per VLAN. Figure 1 shows an example of VLAN configuration on a fully connected (clique) topology with four switches. We list four VLANs (four one-hop spanning trees, with each switch as its root, respectively). For a network with n switches over a clique topology, there are n^{n-2} spanning trees by the well-known Cayley's formula [7]. However, in practice, the total number of VLANs is limited due to protocol and implementation limitations. Typically, a network is limited to have *4096* VLANs and each switch is restricted to support *300-500* VLANs [5].

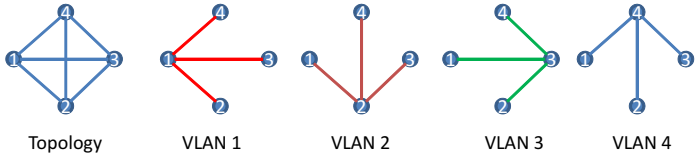


Fig. 1. A fully-connected network topology with four switches. Four VLANs are constructed with each switch as the root of a one-hop spanning tree. Each traffic flow has many choices of paths. For example, one flow from switch 1 to switch 3 can adopt either a one-hop path at VLAN 1 ($1 \rightarrow 3$) or a two-hop path at VLAN 2 ($1 \rightarrow 2 \rightarrow 3$).

The key problem for intra-data-center traffic engineering with ensemble routing is VLAN assignment, *i.e.*, assigning flow ensembles, or routing classes, to VLANs to achieve globally optimized network performance for multiple tenants through load balancing. This problem is challenging not only because the number of possible VLAN assignments can be huge, but also because of the following three major practical concerns:

- *Re-optimization intervals*: Data center applications usually have dynamic traffic patterns. Enforcing right temporal constraints on how fast we re-optimize VLAN assignments and shift existing traffic improves network stability and management efficiency. However, enforcing inappropriate re-optimization intervals diminishes network performance.

- *The number of flows per routing class*: Ensemble routing can be generalized to have different numbers of flows in one routing class. It raises a question of how this generalization can be done in terms of in both effectiveness and scalability. Increasing the number of flows per routing class leads to fine-grained traffic management on one hand, while increasing the computational complexity on the other hand.
- *Inaccurate traffic measurements*: In practice, traffic measurement may be imprecise and may only be taken on a set of limited locations. This will undermine the effectiveness of traffic estimation accuracy, which further leading to potentially suboptimal VLAN assignments.

Several recent papers have tried to solve the VLAN assignment problem. In [4], a greedy heuristic algorithm was proposed to assign VLAN for each routing class. In [8], a linear programming based heuristic was proposed. In [9], a water-filling based local search heuristic was proposed and its effectiveness on the Open Cirrus cloud computing platform was also demonstrated. All these existing efforts [4], [8], [9] enrich the fundamental understandings of ensemble routing and data center traffic engineering. However, algorithms with performance guarantee and analytical studies of the impacts of system parameters are yet to be found. These features are indispensable for the building of service level agreement (SLA) and QoS guarantee in data center networks.

In this study of the VLAN assignment problem, we adopt and extend the Markov approximation framework [10]. This framework not only enables us to design approximation algorithms with close-to-optimal performance guarantee, but also gives us degrees of freedom to design algorithms allowing parallel and distributed implementations. Our main results and contributions are listed as follows:

- For any objective function of network performance and arbitrary network topology, we design an approximation algorithm to solve VLAN assignment problem with provable near-optimal performance guarantee. This algorithm implicitly finds the best VLAN assignment by implementing a Markov chain over all VLAN assignments and performing state jumping. (Section II.A)
- We characterize the key properties of the designed Markov chain: approximation gap, perturbation error bound, mixing time, and trade-off between approximation gap and mixing time. These studies enable us to analyze performance optimality, convergence, and impacts of design parameters. (Section II.C)
- Our algorithm allows parallel and distributed implementation if we replace the objective function of network performance with a local estimation value. The induced optimality gap can be analyzed via perturbation analysis. (Section II.B)
- We outline how to extend the Markov approximation framework in two aspects: one is the characterization of perturbation bound due to estimation errors of traffic matrix and the other is the characterization of the relationship between mixing time of our designed Markov chain and the fastest mixing Markov chain. (Section II.D)
- We study variants of VLAN assignment problem, including interaction with TCP congestion control and QoS considerations. (Section IV)
- We conduct numerical results to show performance of our algorithm under many practical concerns. It is found that our algorithm can be tuned to meet different temporal constraints, incorporate fine-grained traffic management, and tolerate imprecise and incomplete traffic matrices. (Section III)

Due to the page limitation, all proofs can be found in our technical report [11].

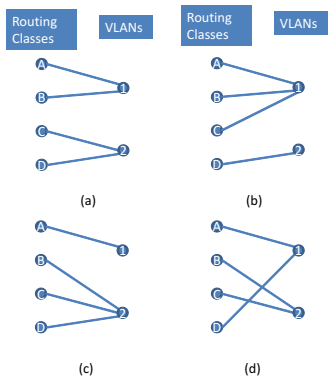


Fig. 2. Example of VLAN assignments. There are four routing classes (A,B,C and D) and two pre-configured VLANs (1 and 2). Four of all sixteen possible assignments are shown here as (a),(b),(c) and (d).

II. VLAN Assignment Problem

First we introduce some notation. The underlying physical network is denoted by $G = (N, L)$, where N denotes the set of network nodes and L denotes the set of physical links. The set of routing classes is denoted by $\mathcal{H}$ and the set of pre-configured VLANs by $\mathcal{V}$. The set of all feasible VLAN assignments is denoted by $\mathcal{X}$, where each assignment $x \in \mathcal{X}$ maps every routing class $h \in \mathcal{H}$ to one VLAN $v \in \mathcal{V}$. This mapping relationship is denoted by $v = x(h)$. Multiple routing classes can be assigned to the same VLAN, while one routing class *cannot* be assigned to multiple VLANs. Examples of VLAN assignment are shown in Figure 2. In general, given $|\mathcal{H}|$ routing classes and $|\mathcal{V}|$ pre-configured VLANs, there are totally $|\mathcal{V}|^{|\mathcal{H}|}$ VLAN assignments, *i.e.*, $|\mathcal{X}| = |\mathcal{V}|^{|\mathcal{H}|}$.

For each routing class $h \in \mathcal{H}$, $\boldsymbol{T}^{(h)}$ represents the estimated flow(rate) vector for routing class h. Given a VLAN $v \in \mathcal{V}$, $\boldsymbol{P}^{(v)}$ represents the corresponding link-flow routing matrix, and $\boldsymbol{r}^{(v)}$ represents a vector of traffic rates contributed by VLAN v. Let $\boldsymbol{r}$ represent the vector of aggregate traffic rates on physical links. Then we have the following relationships between $\boldsymbol{T}$ and $\boldsymbol{r}$:

$$\sum_{h:v=x(h)} \boldsymbol{P}^{(v)}\boldsymbol{T}^{(h)} = \boldsymbol{r}^{(v)}, \forall v \in \mathcal{V} \quad (1)$$

$$\boldsymbol{r} = \textstyle\sum_{v\in\mathcal{V}} \boldsymbol{r}^{(v)}. \quad (2)$$

Let $\boldsymbol{C}$ represent the capacity vector for the physical links in the underlying network. The corresponding utilization for any link $l \in L$ under VLAN assignment x is defined as

$u_l(x) = \frac{\sum_{v\in\mathcal{V}} r_l^{(v)}}{C_l}$. Let $\boldsymbol{u}(x)$ denote the vector of link utilizations. We also let $\Phi_x = \Phi(\boldsymbol{u}(x))$ denote a function of link utilization under VLAN assignment x that reflect the network performance under x. In the literatures of traffic engineering [12], [13], Φ_x represents network costs under assignment x. By minimizing some network cost function, we can avoid solutions that operate near the capacity of the links and shift flows to less utilized links. A common example is to minimize the maximum link utilization, where $\Phi_x = \max_{l\in L} u_l(x)$. The other common objective is to minimize a network-wide objective, where $\Phi_x = \sum_{l\in L} f(u_l(x))$ and $f(\cdot)$ is a convex and increasing function that penalizes solutions with heavily-loaded links. In practice, function f is usually set to be a piecewise-linear form for faster computation time.

Given a general network cost function Φ, the problem of minimizing network cost by choosing the best VLAN assignment can be formulated as the following combinatorial optimization problem:

$$\text{MP}: \ \min \ \Phi_x \tag{3}$$

$$\text{s.t.} \ \ x \in \mathcal{X}. \tag{4}$$

where Φ_x is the network cost under given VLAN assignment x. Φ_x is a function of link utilization $\boldsymbol{u}(x)$, which depends on $\boldsymbol{T}$ and $\boldsymbol{r}$ satisfying (1) and (2).

Problem $\boldsymbol{MP}$ is challenging to solve. Indeed, we have the following result:

Proposition 1. *Problem* MP *is NP-complete and APX-hard (no effective polynomial-time approximate solution).*

A. Markov Approximation

We apply Markov approximation framework [10] to solve the above combinatorial optimization problems. Instead of directly solving problem $\boldsymbol{MP}$, we solve the following approximated problem [10]:

$$\text{MP}-\beta: \ \min \ \sum_{x\in\mathcal{X}} p_x\Phi_x + \frac{1}{\beta}\sum_{x\in\mathcal{X}} p_x \log p_x \tag{5}$$

$$\text{s.t.} \ \ \sum_{x\in\mathcal{X}} p_x = 1, \tag{6}$$

$$p_x \geq 0, \forall x \in \mathcal{X} \tag{7}$$

where p_x is the probability that the system is operated under VLAN assignment $x \in \mathcal{X}$, and β is a positive constant that controls the approximation accuracy.

In fact, the optimal solution to problem $\boldsymbol{MP}-\beta$ is given by

$$p_x^* = \frac{\exp(-\beta\Phi_x)}{\sum_{x'\in\mathcal{X}}\exp(-\beta\Phi_{x'})}, \forall x \in \mathcal{X}. \tag{8}$$

and the corresponding optimal object value is

$$\hat{\Phi} = -\frac{1}{\beta}\log\left(\sum_{x\in\mathcal{X}}\exp(-\beta\Phi_x)\right). \tag{9}$$

The approximation accuracy is known as follows [10]:

$$\min_{x\in\mathcal{X}}\Phi_x - \frac{1}{\beta}\log|\mathcal{X}| \leq \hat{\Phi} \leq \min_{x\in\mathcal{X}}\Phi_x \tag{10}$$

Clearly, as β approaches infinity, the approximation gap approaches zero. The tradeoff is that larger β slows down the convergence. We will elaborate on this tradeoff soon.

Next, we will design a time-reversible VLAN-hopping Markov chain with a state space being the set of all feasible VLAN assignments $\mathcal{X}$ and a stationary distribution being the product-form distribution p_x^* in (8). Then our solution is to hop among different states (different sets of VLAN assignments) according to this Markov chain. When this Markov chain converges, we solve the problem $\boldsymbol{MP}$ (3) approximately.

There are two degrees of freedom in designing a time-reversible VLAN-hopping Markov chain:

- **The state space structure:** we choose such structure of state space that direct transitions between two states corresponds to one and only one routing class switching its assigned VLAN. For example, in Figure 2, suppose these four VLAN assignments are states of some Markov chain, then we allow the direct state transitions between (a) and (b), which correspond to routing class *C* switching between VLAN 1 and VLAN 2. Similarly, we allow the direct state transitions between (c) and (d), which correspond to routing class *D* switching between VLAN 1 and VLAN 2. However, we do *not* allow the direct state transitions between (b) and (c), which correspond to routing classes *B* and *C* switching between VLAN 1 and VLAN 2. It is not hard to show the state space is connected and any two states are reachable from each other.
- **Direct transition rates**: for any two states x, x' with direct transitions, let $q_{x,x'}$ be the transition rates from state x to another state x', then we set

$$q_{x,x'} = \alpha\exp(\beta\Phi_x), \tag{11}$$

where α is a positive constant. It is not hard to see these transition rates satisfy detailed balance equation for the time reversible VLAN-hopping Markov chain.

B. VLAN Assignment Algorithm

We can implement the designed VLAN-hopping Markov chain as follows: Initially, each routing class $h \in \mathcal{H}$ is assigned a VLAN v randomly picked from $\mathcal{V}$. Under the current VLAN assignment x, each routing class is associated with an exponentially distributed random number with a mean equal to

$$\frac{\exp(-\beta\Phi_x)}{\alpha(|\mathcal{V}|-1)} \tag{12}$$

and counts down according to this number. When the count down of a routing class h expires, this routing class is assigned a new VLAN randomly picked from $\mathcal{V}-x(h)$ (its $|\mathcal{V}|-1$ not-in-use VLANs). System will informs other routing classes to terminate their current count down processes and start fresh ones using new measurements under the new VLAN assignments x'. We will refer to this the "Wait-and- Hop" algorithm. The corresponding pseudocode is shown in Algorithm 1.

The correctness of the "Wait-and-Hop" algorithm is shown in the following proposition.

Proposition 2. *The Wait-and-Hop algorithm realizes a continuous-time VLAN-hopping Markov chain with stationary distribution shown in* (8)*.*

Algorithm 1 "Wait-and-Hop" algorithm

The following procedure runs on each individual routing class independently. We focus on a particular routing class h.

procedure INITIALIZATION
  $x(h) \leftarrow v$ randomly picked from $\mathcal{V}$
  invoke Procedure Wait(h)
end procedure

procedure WAIT(h)
  Obtains the value of Φ_x
  generates a timer $\tau_h \sim \exp(\lambda)$ with rate $\lambda = \alpha(|\mathcal{V}| - 1)\exp(\beta\Phi_x)$
  begins counting down
  while the timer τ_h does not expire **do**
    if receives a message of RESET **then**
      index $\leftarrow 1$
      break
    end if
  end while
  if index = 1 **then**
    terminates current countdown process and invoke Procedure WAIT(h)
    index $\leftarrow 0$
  else
    Invoke Procedure HOP(h)
  end if
end procedure

procedure HOP(h)
  $x'(h) \leftarrow v'$ randomly picked from $\mathcal{V} - v$
  broadcasts a RESET message to other routing classes
end procedure

Our "Wait-and-Hop" algorithm is parallel and distributed if we can obtain local estimations of Φ_x. The optimality gap due to estimation errors can be analyzed via the following perturbation analysis.

C. Perturbation Analysis

For each state $x \in \mathcal{X}$, if we obtain the accurate value of Φ_x for any state x, then the state distribution of the designed Markov chain will converge to the desired stationary distribution shown in (8). Hence guided by Markov chain, we obtain a close-to-minimal network cost.

However, in practice, we usually obtain *perturbed* (inaccurate) values of Φ_x for any $x \in \mathcal{X}$. This inaccuracy is caused by two kinds of perturbations:

- **Passive Perturbation**: this means the imprecise and incomplete measurements of traffic flow rates, leading to the imprecise and incomplete traffic matrices.
- **Active Perturbation**: this means for any state x, we replace Φ_x (which may need global network information and hard to compute) with problem-specific local estimates to enable parallel and distributed implementations.

Consequently, with perturbed errors of $\Phi_x, x \in \mathcal{X}$, the perturbed VLAN hopping Markov chain may converge to a sub-optimal steady-state distribution, resulting in an optimality gap. To characterize the optimality gap due to the perturbation errors, we adopt the quantization error model proposed in [14].

For each state $x \in \mathcal{X}$ with Φ_x, we assume its corresponding perturbation error belongs to the bounded region $[-\Delta_x, \Delta_x]$, where Δ_x is the error bound and can be different for different x. We also assume the perturbed Φ_x takes only one of the following $2n_x + 1$ discrete values:

$$\left[\Phi_x - \Delta_x, \ldots, \Phi_x - \frac{1}{n_x}\Delta_x, \Phi_x, \Phi_x + \frac{1}{n_x}\Delta_x, \ldots, \Phi_x + \Delta_x\right],$$

where n_x is a positive constant. Further, with probability $\eta_{j,x}$, the perturbed Φ_x takes the value $\Phi_x + \frac{j}{n_x}\Delta_x$, $\forall j \in \{-n_x, \ldots, n_x\}$ and $\sum_{j=-n_x}^{n_x} \eta_{j,x} = 1$.

Let $\Phi_{\min} = \min_{x\in\mathcal{X}} \Phi_x$ denote the minimal network cost, $\Delta_{\max} = \max_{x\in\mathcal{X}} \Delta_x$ the maximum perturbation error, $\Phi^*_{ave} = \sum_{x\in\mathcal{X}} p^*_x \cdot \Phi_x$ the expected network cost with VLAN-hopping Markov chain, and $\bar{\Phi}_{ave} = \sum_{x\in\mathcal{X}} \bar{p}_x \cdot \Phi_x$ the expected network cost with perturbed VLAN-hopping Markov chain. By perturbation analysis developed in [14], we have the following result:

Theorem 1. *(a) The stationary distribution of the perturbed VLAN-hopping Markov chain is*

$$\bar{p}_x(\mathbf{\Phi}) = \frac{\sigma_x \exp\left(-\beta\Phi_x\right)}{\sum_{x'\in\mathcal{X}} \sigma_{x'} \exp\left(-\beta\Phi_{x'}\right)}, \forall x \in \mathcal{X} \qquad (13)$$

where $\sigma_x = \sum_{j=-n_x}^{n_x} \eta_{j,x} \exp\left(\beta \frac{j\Delta_x}{n_x}\right)$.
(b) The optimality gap are shown as follows:

$$0 \le \Phi^*_{ave} - \Phi_{\min} \le \frac{|\mathcal{H}|\log|\mathcal{V}|}{\beta}, \qquad (14)$$

$$0 \le \bar{\Phi}_{ave} - \Phi_{\min} \le \frac{|\mathcal{H}|\log|\mathcal{V}|}{\beta} + \Delta_{\max}, \qquad (15)$$

Remarks:

- The upper bound on optimality gap of perturbed VLAN-hopping Markov chain shown in (15) is quite general, as it is independent of the values of $n_x, x \in \mathcal{X}$ and the values of $\eta_{j,x}$ $(-n_x \le j \le n_x, x \in \mathcal{X})$.
- When β increases, the optimality gap for both the perturbed VLAN-hopping Markov chain and the VLAN-hopping Markov chain decreases. However, increasing β may also increase the mixing time of the Markov chain. We will present bounds on the mixing time in the next subsection.
- The upper bound on optimality gap of perturbed VLAN-hopping Markov chain is more loose than the counterpart of VLAN-hopping Markov chain because of perturbation errors. The difference is $\Delta_{\max}$ and we call it "the price of perturbation errors".

Given the exact form of objective function and suppose there are only flow estimation errors, we can characterize the relationship between the flow estimation errors and perturbation errors, and then apply Theorem 1. However, this bound may be loose. Therefore, we extend the Markov approximation framework and show that we can replace perturbation bound

with maximum perturbation error $\Delta_{\max}$ by a more tight bound with the expected value of perturbation errors. For example, when $\Phi_x = \max_{l \in L} u_l(x), \forall x \in \mathcal{X}$, we have following result:

Corollary 1. *The optimality gap of corresponding perturbed Markov chain is*

$$\bar{\Phi}_{ave} - \Phi_{\min} \leq \frac{|\mathcal{H}| \log |\mathcal{V}|}{\beta} + \max_{l \in L} \frac{d_l}{C_l} \epsilon_l, \tag{16}$$

where d_l denotes the number of flows traversing link l, ϵ_f denotes the estimated error of flow rates for flow f, $\epsilon_l = \max_{f \in l} E|\epsilon_f|$ denotes the maximum expected absolute value of the estimation error of flow rates for any flow traversing link l.

D. Mixing Time Analysis

We study the mixing time (convergence time) of the VLAN-hopping Markov chain. The perturbed version is a straightforward extension. First, we introduce the definition of total variation distance between any two probability distributions $\boldsymbol{p}$ and $\boldsymbol{p}'$ over state space $\mathcal{X}$ as follows:

$$||\boldsymbol{p} - \boldsymbol{p}'||_{TV} \triangleq \frac{1}{2} \sum_{x \in \mathcal{X}} |p_x - p'_x|. \tag{17}$$

Now let $\boldsymbol{P}_t(x)$ denote the probability distribution of all states in $\mathcal{X}$ at time t given that the initial state is x. Then the mixing time of the VLAN-hopping Markov chain is defined as follows:

$$t_{mix}(\epsilon) \triangleq \inf \left\{ t \geq 0 : \max_{x \in \mathcal{X}} ||\boldsymbol{P}_t(x) - \boldsymbol{p}^*||_{TV} \leq \epsilon \right\}, \tag{18}$$

where $\boldsymbol{p}^*$ is the stationary distribution shown in (8).

We adopt the spectral analysis method [15] to obtain both a lower bound and an upper bound of $t_{mix}(\epsilon)$ for general values of β. We also adopt path coupling method [16] to obtain a tight upper bound of $t_{mix}(\epsilon)$ for some values of β. Denote $\Phi_{\max} \triangleq \max_{x \in \mathcal{X}} \Phi_x$ and $\Phi_{\min} \triangleq \min_{x \in \mathcal{X}} \Phi_x$, we have the following results:

Theorem 2. *The mixing time (convergence time) of the VLAN-hopping Markov chain is bounded as follows:*
(a) for general $\beta \in (0, \infty)$

$$t_{mix}(\epsilon) \geq \frac{\exp(-\beta \Phi_{\max})}{2\alpha(|\mathcal{V}| - 1)|\mathcal{H}|} \cdot \ln \frac{1}{2\epsilon}, \tag{19}$$

and

$$t_{mix}(\epsilon) \leq \frac{2}{\alpha} \cdot (|\mathcal{V}| - 1) \cdot |\mathcal{H}| \cdot |\mathcal{V}|^{2|\mathcal{H}|} \cdot \exp(\beta(4\Phi_{\max} - 3\Phi_{\min})) \cdot \left[\ln \frac{1}{2\epsilon} + \frac{1}{2}|\mathcal{H}| \cdot \ln |\mathcal{V}| + \frac{1}{2}\beta \cdot (\Phi_{\max} - \Phi_{\min}) \right]. \tag{20}$$

(b) When $0 < \beta < \beta_{th} = \frac{1}{2(\Phi_{\max} - \Phi_{\min})} ln(\frac{|\mathcal{H}| + \frac{1}{|\mathcal{V}|-1}}{|\mathcal{H}|-1})$, we have a tighter upper bound

$$t_{mix}(\epsilon) \leq \frac{\exp(\beta(2\Phi_{\max} - \Phi_{\min})) \cdot \ln \frac{|\mathcal{H}|}{\epsilon}}{\alpha(|\mathcal{V}| - 1) \left[|\mathcal{H}| + \frac{1}{|\mathcal{V}|-1} - (|\mathcal{H}| - 1) \exp(2\beta(\Phi_{\max} - \Phi_{\min}) \right]} \tag{21}$$

Recall that in practice, $|\mathcal{V}|$, *i.e.*, the total number of VLANs is limited by some constants [5]. Now given constant $|\mathcal{V}|$, $\Phi_{\max}$ and $\Phi_{\min}$, we have following observations when β scales with $|\mathcal{H}|$:

- As $\beta \sim \log(|\mathcal{H}|)$, optimality gap $\Phi^*_{ave} - \Phi_{\min} \sim O(\frac{|\mathcal{H}|}{\log(|\mathcal{H}|)})$ in (14), $t_{mix}(\epsilon) \sim \exp(\Omega(|\mathcal{H}|))$ in (20), and $t_{mix}(\epsilon) \sim O(|\mathcal{H}| \log(|\mathcal{H}|))$ in (21).
- As $\beta \sim |\mathcal{H}|$, optimality gap $\Phi^*_{ave} - \Phi_{\min} \sim O(1)$ in (14), $t_{mix}(\epsilon) \sim \exp(\Omega(|\mathcal{H}|))$ in both (20) and (21).

When β is a given constant, we consider the trade-off between the optimality gap of VLAN-hopping Markov chain (Theorem 1) and its mixing time (Theorem 2), e.g., the two ends of this spectrum.

- As $\beta \to \infty$, the optimality gap of VLAN-hoping Markov chain approaches zero while the upper bound of its mixing time scales with $\exp(\Omega(|\mathcal{H}|))$ and approaches infinity (slow-mixing).
- As $\beta \to 0$, the optimality gap of VLAN-hoping Markov chain approaches infinity while the upper bound of its mixing time scales with $O(\log(|\mathcal{H}|))$ and remain limited (fast-mixing).

This resembles the phase transition phenomenon in statistics physics: when $\beta \leq \beta_{th}$, the whole system is fast mixing, while when $\beta > \beta_{th}$, the whole system is slow mixing. Here β_{th} is the threshold value for phase transition. In our case, $\beta_{th} = \frac{1}{2(\Phi_{\max} - \Phi_{\min})} ln(\frac{|\mathcal{H}| + \frac{1}{|\mathcal{V}|-1}}{|\mathcal{H}|-1})$, a small value. In practice, β_{th} can be larger. In extreme case with $|\mathcal{H}| = 1$, $\beta_{th} \to \infty$ and the whole system is always fast mixing.

Now suppose the topology of state space of VLAN-hopping Markov chain and the stationary distribution (13) are kept the same, let $t^*_{mix}(\epsilon)$ denote mixing time of corresponding fastest mixing Markov chain, which may be obtained with prohibitive computational complexities. By the variational characterization of eigenvalues [15], [17], we have the following result:

Proposition 3. *The mixing time of VLAN-hopping Markov chain $t_{mix}(\epsilon)$ is bounded as follows:*

$$t^*_{mix}(\epsilon) \leq t_{mix}(\epsilon) \leq \frac{|\mathcal{H}| \cdot (|\mathcal{V}| - 1)}{\exp(\beta(\Phi_{\min} - \Phi_{\max}))} \cdot t^*_{mix}(\epsilon) \tag{22}$$

Note that in practice, we usually observe more tight bounds.

III. Performance Evaluation and Practical Issues

In this section, we evaluate performance of our algorithm with practical issues including re-optimization intervals, non-uniform traffic matrices, the number of flows per routing class, imprecise and incomplete traffic matrices. In practice, the switch is essentially a rack under which many (e.g., hundreds of) hosts are connected to. The capacity of link connecting the host to the switch (e.g., the rack) is usually lower than inter-switch (rack) capacities. What is more, the clique topology is a good starting point for benchmarking. Therefore, in experiments, we use a clique topology with four switches as shown in Figure 1. Four VLANs are constructed with each switch as the root of a one-hop spanning tree. One single host is attached to each switch. The bandwidth of each link that connects switches is 1Gbps, and the bandwidth of each link that connects a host and a switch is 100Gbps. We choose 8 hash classes and use all-pair shuffle traffic for each

hash class. There are 96 flows in total. We also use both uniform and non-uniform traffic matrices in experiments.

We use realistic dynamic traffic patterns, where VLAN assignments should be re-optimized over time. VLAN assignment re-optimization can be conducted at different time scales, ranging from a single-flow duration to once every several seconds. The need for responsive traffic engineering may require the timescale to be sub-second, which is practically feasible in openflow-enabled networks. Different optimization granularity can result in different performance. A short re-optimization time can lead to lower congestion cost, higher throughput and shorter completion time. On the other hand, it can also increase computation cost. Tuning to re-optimize VLAN assignments at different time-scales allows the operator to strike a complexity-optimality tradeoff.

A. Under Uniform Traffic Matrix

We evaluate how the optimization interval influences the algorithm performance, first under uniform traffic matrix. A uniform traffic matrix means all entries in the traffic matrix have the same value. In our setting, we set each flow with 1Gb. We varied the optimization interval from small time interval to large time interval, *i.e.*, upon every flow completion, every 1 second, every 5 seconds, and every 10 seconds.

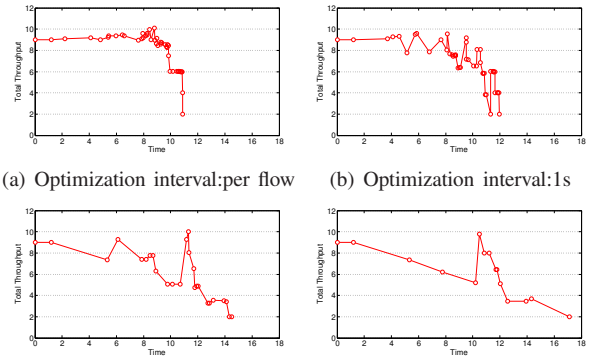
Figure 3 shows how the total throughput evolves over time under different optimization intervals. *The completion time of all traffic flows increases as the optimization interval increases. The system uses the least time to complete all traffic flows when optimization is performed upon completion of each flow.* The system achieves high throughput most of the time when optimization is performed upon completion of each flow, except in the end when there are only a few unfinished traffic flows, as shown in Figure 3(a). But when the optimization interval increases, the system can no longer achieve high throughput most of the time. For example, as shown in Figure 3(c), when the optimization interval is 5 seconds, it only achieves high throughput after each re-optimization, i.e. at 0s, 6s, and 11s. Between two re-optimizations, the total throughput decreases over time since some flows finish in the interval and no re-optimization is done to utilize the idle or underutilized links. For instance, between 6s and 11s, the total throughput decreases.

What is more, for the four optimization intervals, the total completion time is 10.9s, 12.0s, 14.5s and 17.1s respectively. This means when we perform optimization every several seconds, the system can still finish all work in a reasonable time. This allows a good complexity-optimality tradeoff.

B. Under Non-uniform Traffic Matrix

We now evaluate how the optimization interval influences the algorithm performance under non-uniform traffic matrix. Each flow is generated from normal distribution $\mathcal{N}(1, 0.2)$ and 100 samples of traffic matrices are used.

Figure 4 and Figure 5 show the total completion time under different optimization intervals. Figure 4 shows the CDF(Cumulative Distribution Function) of total completion time. It is shown that the total completion time increases when



(a) Optimization interval:per flow (b) Optimization interval:1s

(c) Optimization interval:5s (d) Optimization interval:10s

Fig. 3. Total throughput under uniform traffic matrix and different optimization intervals.

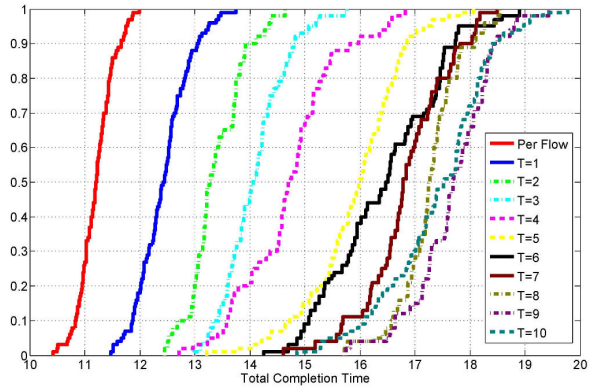


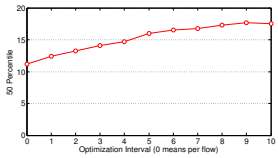
Fig. 4. CDF of total flow completion time under non-uniform traffic matrix and different optimization intervals.

the optimization interval increases. When optimization is performed upon every flow completion, the system uses the least time to complete. We also find that *the speed of performance degrades decreases with the increasing optimization interval*. For example, the difference between "Per Flow" and "$T = 1$" is larger than the difference between "$T = 9$" and "$T = 10$". Intuitively, this follows some diminishing marginal utility rule.

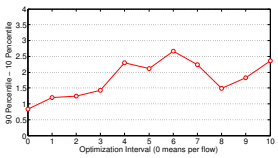
Figure 5(a) shows the 50 percentile of total completion time under different optimization intervals. It is clear that the 50 percentile value increases when the optimization interval increases. This is because *when optimization is performed more frequently, the system can make better use of idle network bandwidth*. Figure 5(b) shows the difference between the 90 percentile and 10 percentile of total completion time, and Figure 5(c) shows the variance of total completion time. They follow the similar trend of going up as the optimization interval increases. This is because when optimization is not performed frequently, the system cannot make good use of network bandwidth. Sometimes some links might be idle, while sometimes all links make full use of bandwidth. The difference between different cases is not compensated by re-optimization. When the optimization interval is large, the system shows a large variance of performance.

C. Impact of Number of Flows Per Class

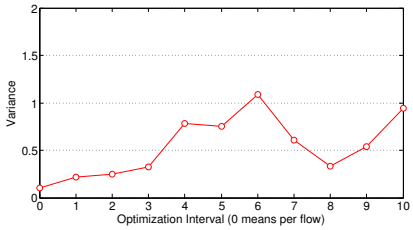
Ensemble routing reduces traffic management overhead by operating on the granularity of flow ensembles, rather



(a) 50 percentile of total completion time

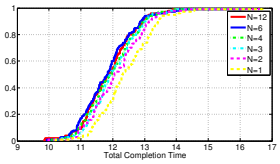


(b) Difference of 90 percentile and 10 percentile of total completion time

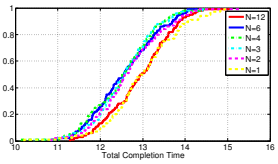


(c) Variance of total completion time

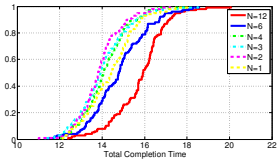
Fig. 5. Total flow completion time under non-uniform traffic matrix and different optimization intervals.



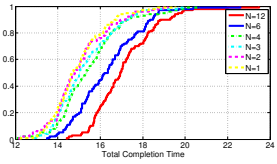
(a) Optimization interval:per flow



(b) Optimization interval:1s



(c) Optimization interval:5s



(d) Optimization interval:10s

Fig. 6. CDF of total completion time when each traffic class contains 12, 6, 4, 3, 2, 1 flows under different optimization intervals.

than individual flows. However, controlling each individual flow can make a better use of link capacities. There is a performance gap between ensemble routing and individual flow management. In fact, if we let each ensemble class contain only one flow, ensemble routing can be generalized to incorporate the individual flow management.

In the generalized form, an ensemble class can be further split into several sub-classes where each sub-class contains a fraction of flows in the original ensemble class. In the extreme case, each sub-class contains only one flow. The more sub-classes each ensemble class is split into, the better link capacity utilization can be achieved. But more sub-classes also introduces more computational cost. There is a tradeoff between complexity and optimality. Through experiments we want to explore whether splitting ensemble classes can make a difference and if it can how big the difference is. What is more, how well splitting ensemble classes can perform also depends on the temporal constraint(the optimization interval). So in our experiments we also incorporate this variable.

Each flow is generated from normal distribution $\mathcal{N}(1, 0.2)$ and 100 samples of traffic matrices are used. Since each ensemble class has 12 flows, we split each ensemble class into 1, 2, 3, 4, 6, 12 sub classes respectively. Thus each sub-class contains 12, 6, 4, 3, 2, 1 flows respectively. Different optimization intervals are also used including per flow, 1s, 5s and 10s.

Figure 6 shows CDF of total completion time when each traffic class contains 12, 6, 4, 3, 2, 1 flows under different optimization intervals. Intuitively, the more flows each sub-class contains, the better VLAN assignments can be made, and the shorter total completion time the system can achieve. However, from the figure we can see that optimization interval also plays an important role. In Figure 6(a) and Figure 6(b), we find that *the gap between different flow numbers per class is not big when the optimization interval is small*. This is because when the optimization interval is small, the system can frequently reassign VLANs to traffic classes so as to make effective use of link capacity. The drawback of coarse-grained traffic management can be compensated by frequently reassigning of VLANs. However, from Figure 6(c) and Figure 6(d) we find that when *the optimization interval is large, the advantage of letting each traffic class contains fewer flows becomes more obvious*. For example, in Figure 6(d), the optimization interval is 10s, we find that the total completion time when each traffic class only contains one flow is shorter than the total completion time when each traffic class contains 12 flows. This gap increases along with the optimization interval.

D. Impact of Imprecise and Incomplete Traffic Matrices

In practice, traffic measurement may be imprecise and may only be taken on a set of limited locations, such as core switches and bottleneck links.

We now investigate the sensitivity of our algorithm with respect to imprecise and incomplete traffic matrices. Given an imprecise traffic matrix, our algorithm may come up with a VLAN assignment far from optimal. Similar, given an incomplete or partial traffic matrix, our algorithm has to estimate missing entries first. If the estimation is far from the reality, the VLAN assignment given by our algorithm may also be far from the optimal one.

We conduct a series of experiments to understand how our algorithm performs under imprecise and incomplete traffic matrices. Intuitively, for an imprecise traffic matrix, the more biased the traffic matrix is, the poorer performance our algorithm achieves; for an incomplete traffic matrix, the further the estimation is from the true value, the poorer performance our algorithm achieves. Through experiment we want to explore how big the gap is between our algorithm with imprecise or incomplete information and our algorithm with precise and complete information.

Note that traffic matrix estimation and completion is a challenging topic in its own right and outside the scope of this paper. We take a simple method of traffic matrix completion to illustrate the robustness of our VLAN design methodology even under such cases. More refined traffic matrix completion algorithms will further enhance the end results.

1) Imprecise Traffic Matrices: We evaluate how our algorithm performs under imprecise traffic matrices. Each flow, or each entry in a traffic matrix is generated from normal

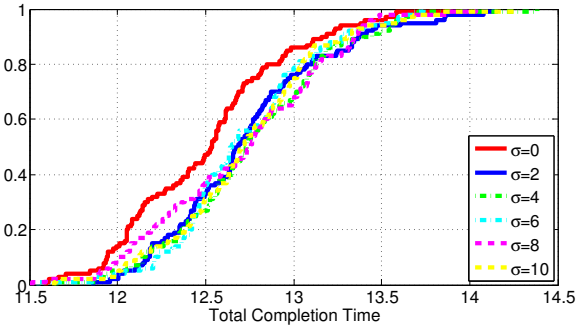


Fig. 7. CDF of total flow completion time under different bias level. The bias follows a normal distribution $N(0, \sigma)$, where σ denoted how imprecise a traffic matrix is.

distribution $\mathcal{N}(1, 0.2)$ and some bias is introduced to traffic matrices that the algorithm receives. The bias follows a normal distribution $\mathcal{N}(0, \sigma)$, where σ denote how imprecise a traffic matrix is. 100 samples of traffic matrices are used and σ is varied from 0 to 10.

Figure 7 shows the CDF of total completion time under different bias levels. From the figure, we find that the system takes the least time to complete all traffic flows when there is no bias ($\sigma = 0$). But *the difference between algorithm performance under no bias and algorithm performance under a large bias is not large*. The difference is no more than 0.5s. This means our algorithm can still perform well under imprecise traffic matrices.

2) Incomplete Traffic Matrices: We also evaluate how our algorithm performs under incomplete traffic matrices. Each flow, or each entry in a traffic matrix is generated from a normal distribution $\mathcal{N}(1, \sigma)$ and 100 samples of traffic matrices are used. For each row in the traffic matrix, we knock out *one* entry and use the average of other entries to estimate this entry. σ is varied from 0.2 to 8.0. Intuitively, if σ is big, each entry can be largely different from other entries, and the estimation using average values can be very imprecise, which consequently degrades the performance of the algorithm.

Figure 8 and Figure 9 show total completion time under complete and incomplete traffic matrices with different variances. Figure 8 shows CDF of total completion time. We find that *our algorithm under complete traffic matrices performs better than our algorithm under incomplete traffic matrices, but again the gap is small (less than 0.5s)*. Figure 9(a) shows that the 50 percentile of total completion time under complete traffic matrices is consistently lower than that under incomplete traffic matrices. The gap increases when σ increases. Figure 9(b) and Figure 9(c) show the difference of 90 percentile and 10 percentile and the variance of total completion time. *The gap increases when σ increases, but the gap is small.*

Next we knock out more than one entry for each row in the traffic matrix and investigate the impact as the number of unknown entries K increases. We set $\sigma = 0.2$ and the missing entries are estimated by the average of all non-missing entry values. Figure 10 shows the CDF of total completion time with the number of unknown entries. When more entries are missing (K increases), total completion time increases and

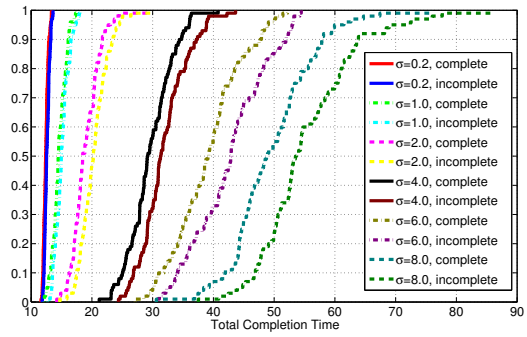
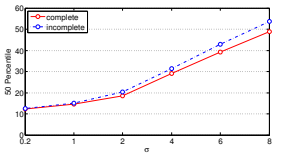
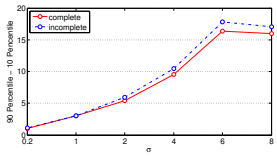


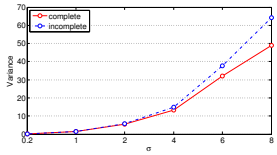
Fig. 8. CDF of total flow completion time under complete and incomplete traffic matrices.



(a) 50 percentile of total completion time



(b) Difference of 90 percentile and 10 percentile of total completion time



(c) Variance of total completion time

Fig. 9. Total flow completion time under complete and incomplete traffic matrices with different variances.

performance becomes worse. However, *when K is not larger than some threshold value, the performance gap is very small.*

These experiment results show that *our algorithm can still achieve good performance using incomplete traffic matrices.*

IV. Variants of VLAN Assignment Problem

In this section, we discuss two variants of VLAN assignment problem.

A. Interaction with TCP

In the original VLAN assignment problem (3), estimated traffic matrix $\boldsymbol{T}$ is given. In practice, flow rates are usually adjusted by TCP protocol. Thus we consider the interaction between VLAN assignment and TCP congestion control. We

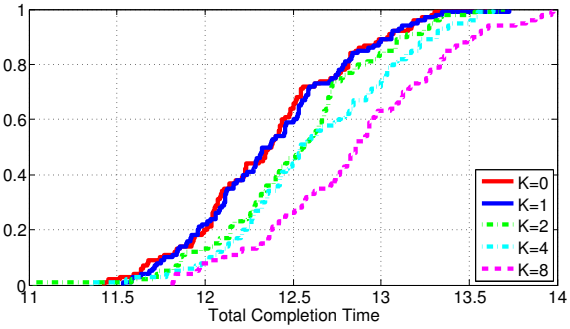


Fig. 10. CDF of total flow completion time with the number of unknown entries.

predict the traffic sending rate matrix $\boldsymbol{T}$ given a VLAN assignment x by solving the following Network Utility Maximization problem [18]:

$$\boldsymbol{JP_x}: \max \ U(\boldsymbol{r}) \tag{23}$$

$$\text{s.t.} \sum_{h:v=x(h)} \boldsymbol{P}^{(v)}\boldsymbol{T}^{(h)} = \boldsymbol{r}^{(v)}, \forall v \in \mathcal{V} \tag{24}$$

$$\boldsymbol{r} = \sum_{v\in\mathcal{V}} \boldsymbol{r}^{(v)} \leq \boldsymbol{C}, \tag{25}$$

$$\boldsymbol{T} \succcurlyeq \boldsymbol{0}, \tag{26}$$

where $U(\cdot)$ denotes the network utility function. Here the choices of the utility function capture different congestion control algorithms [18], including proportional fairness, max-min fairness and more general objectives with fairness considerations.

For any $x \in \mathcal{X}$, after solving problem $\boldsymbol{JP_x}$, we obtain corresponding optimal value of $\boldsymbol{r}^*$ and $\boldsymbol{T}^*$. Then we can compute the value Φ_x. The design of Markov chain, perturbation analysis of Markov chain and mixing time of Markov chain are very similar to those of original VLAN assignment problem. For example, the impacts of imperfect and incomplete traffic matrices can be modeled as estimation errors of traffic rates and can be studied by the perturbation analysis of Markov chain.

B. QoS Considerations

In the original VLAN assignment problem (3), each routing class $h \in \mathcal{H}$ can choose any one VLAN $v \in \mathcal{V}$. However, in practice, due to QoS considerations, the available VLAN set for each routing class h is limited. For example, a routing class h may want to choose only those VLANs providing it one-hop paths instead of paths with long hops due to some delay considerations. These QoS considerations can be modeled as follows: for each routing class $h \in \mathcal{H}$, it can only be assigned to a subset of VLANs, denoted by $S_h \subseteq \mathcal{V}$. VLANs belonging to S_h satisfy quality of service considerations for routing class h. Corresponding design of Markov chain, perturbation analysis of Markov chain and mixing time of Markov chain are very similar to those of original VLAN assignment problem.

V. Conclusions

In this paper, we study a key component of intra-data-center traffic engineering with ensemble routing: VLAN assignment schemes. By Markov approximation framework, we design a VLAN assignment scheme that achieve efficient load-balancing and close-to-minimal network cost for arbitrary network topology and arbitrary cost functions. Our scheme also allows parallel and distributed implementation. Key properties of this scheme are studied analytically, including performance optimality, perturbation bounds with estimation errors and mixing time of Markov chain. We characterize the optimality gap due to the estimation errors of flow rates, where the upper bounds of optimality gap increases linearly with the maximum flow estimation errors. We also obtain the bounds on mixing time, which exhibits an interesting phase transition phenomenon with some threshold value of β. We further compare our mixing time with the fastest mixing time and give out some bounds. We conduct performance evaluations on our scheme with many practical concerns and find that our scheme achieves efficient load balancing and shows desirable scalability and robustness. It can be tuned to meet different temporal constraints, incorporate fine-grained traffic management, overcome traffic measurement limitations, and tolerate imprecise and incomplete traffic matrices.

Acknowledgment

We thank Mike Schlansker and Yoshio Turner from HP Labs and Jennifer Rexford from Princeton University for many fruitful discussions on the topic, and a grant from the HP University Research Program. Minghua Chen' research was partially supported by a China 973 Program (Project No. 2012CB315904), the General Research Fund (Project No. 411209, 411010, and 411011) and an Area of Excellence Grant (Project No. AoE/E-02/08) both established under the University Grant Committee of the Hong Kong SAR, China, and two gift grants from Microsoft and Cisco.

References

[1] C. Kim, M. Caesar, and J. Rexford, "Floodless in SEATTLE: A scalable ethernet architecture for large enterprises," in *Proceedings of ACM SIGCOMM*, 2008.

[2] A. Greenberg, J. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, "VL2: A Scalable and Flexible Data Center Network," in *Proceedings of ACM SIGCOMM*, 2009.

[3] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. Mogul, "Spain: Cots data-center ethernet for multipathing over arbitrary topologies," in *Proceedings of the 7th USENIX*, 2010.

[4] M. Schlansker, Y. Turner, J. Tourrilhes, and A. Karp, "Ensemble routing for datacenter networks," in *Proceedings of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, 2010.

[5] M. Yu, J. Rexford, X. Sun, S. Rao, and N. Feamster, "A survey of virtual lan usage in campus networks," *IEEE Communications Magazine*, vol. 49, no. 7, pp. 98–103, 2011.

[6] X. Sun, Y. Sung, S. Krothapalli, and S. Rao, "A systematic approach for evolving vlan designs," in *Proceedings of IEEE INFOCOM*, 2010.

[7] R. Diestel, *Graph Theory*. Springer Verlag, 2006.

[8] W. Wu, Y. Turner, and M. Schlansker, "Routing optimization for ensemble routing," in *Proceedings of the 2011 ACM/IEEE Seventh Symposium on Architectures for Networking and Communications Systems*, 2011.

[9] W. Jiang, Y. Turner, J. Tourrilhes, and M. Schlansker, "Controlling traffic ensembles in open cirrus," *HP Technique Report*, 2011.

[10] M. Chen and S. Liew and Z. Shao and C. Kai, "Markov approximation for combinatorial network optimization," in *Proceedings of IEEE INFOCOM*, 2010.

[11] Z. Shao, X. Jin, W. Jiang, M. Chen, and M. Chiang, "Intra-Data-Center Traffic Engineering with Ensemble Routing," *Technical Report*, 2012, available at http://home.ie.cuhk.edu.hk/∼zyshao/ensemble.pdf.

[12] J. Rexford, "Route optimization in ip networks," *Handbook of Optimization in Telecommunications*, pp. 679–700, 2006.

[13] J. He, M. Bresler, M. Chiang, and J. Rexford, "Towards robust multi-layer traffic engineering: Optimization of congestion control and routing," *IEEE Journal on Selected Areas in Communications(JSAC)*, vol. 25, no. 5, pp. 868–880, 2007.

[14] S. Zhang, Z. Shao, and M. Chen, "Optimal Distributed P2P Streaming under Node Degree Bounds," in *Proceedings of IEEE ICNP*, 2010.

[15] P. Diaconis and D. Stroock, "Geometric bounds for eigenvalues of Markov chains," *The Annals of Applied Probability*, pp. 36–61, 1991.

[16] R. Bubley and M. Dyer, "Path coupling: A technique for proving rapid mixing in markov chains," in *Proceedings of IEEE FOCS*, 1997, pp. 223–231.

[17] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," *SIAM review*, vol. 46, no. 4, pp. 667–689, 2004.

[18] M. Chiang, S. Low, A. Calderbank, and J. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255–312, 2007.