Dynamic Switch-Controller Association and Control Devolution for SDN Systems

Xi Huang[1], Simeng Bian[1], Ziyu Shao[1], Hong Xu[2]
[1]School of Information Science and Technology, ShanghaiTech University
[2] NetX Lab @ City University of Hong Kong
Email: {huangxi,biansm,shaozy}@shanghaitech.edu.cn, henry.xu@cityu.edu.hk

Abstract—In software-defined networking (SDN), as data plane scale expands, scalability and reliability of the control plane have become major concerns. To mitigate such concerns, two kinds of solutions have been proposed separately. One is multi-controller architecture, i.e., a logically centralized control plane with physically distributed controllers. The other is control devolution, i.e., delegating control of some flows back to switches. Most of existing solutions adopt either static switch-controller association or static devolution, which may not adapt well to the traffic variation, leading to high communication costs between switches and controller, and high computation costs of switches. In this paper, we propose a novel scheme to jointly consider both solutions, i.e., we dynamically associate switches with controllers and dynamically devolve control of flows to switches. Our scheme is an efficient online algorithm that does not need the statistics of traffic flows. By adjusting some parameter V, we can make a trade-off between costs and queue backlogs. Theoretical analysis and extensive simulations show that our scheme yields much lower costs and latency compared to static schemes, and balanced loads among controllers.

Index Terms—Cloud computing, .

1 Introduction

Software-defined networking (SDN) holds great promises to improve network performance and management. The key idea of SDN is to decouple the control plane from the data plane [1]. Data plane can focus on performing basic functionalities such as packet forwarding at high speed, while the logically centralized control plane manages the network. Usually, switches at data plane send request to control plane for processing some flow events, e.g. flow-install events.

The control plane is a potential bottleneck of SDN in terms of scalability and reliability. As the data plane expands, control plane may not be able to process the increasing number of requests if implemented with a single controller, resulting unacceptable latency to flow setup. Reliability is also an issue since a single controller is a single point of failure , resulting in disastrous break-down of the control plane and the network.

Existing proposals to address such problems fall broadly into two categories. One is to implement the control plane as a distributed system with multiple controllers [2] [3]. Each switch then associate with certain controller for fault-tolerance and load balancing [4] [5] [6] [7]. The other is to devolve partial loads of request processing from controllers to switches [8] [9] [10], reducing the work load of controllers.

For switch-controller association, the first category of solution, initial design choice is to make a static switch-controller association [2] [3]. However, such static association may result in overloading of controllers and increasing flow setup latency due to its inflexibility to handle traffic variations. An elastic distributed controller architecture is proposed in [5], with an efficient protocol to migrate switches across controllers. However, it remains open to determine the switch-controller association. Then [6] took a step further by formulating the switch-association problem as a deterministic optimization problem, i.e., an integer linear problem with prohibitively high computational complexity. A local search algorithm was proposed to find suboptimal associations within a given time limit (e.g., 30 seconds). In [7], the controller is assumed to be modeled as M/M/1 queue (Poisson arrival and exponential service). Under such assumption, the switch-association problem with steady-state objective function was formulated as a many-to-one stable matching problem with transfers. Then a novel two-phase algorithm was proposed to connect stable matching with utility-based game theoretic solutions, i.e., coalition formation game with Nash stable solutions.

For control devolution, the second category of solution, initial design choice is to make a static devolution for certain functions and certain flows [8] [9] [10]. It remains open for dynamic devolution with respect to traffic variations.

Then several interesting questions are raised and answers to such questions will definitely shape our design for SDN networks:

i. Instead of deterministic switch-controller association with infrequent re-association [6] [7], can we directly perform dynamic switch-controller association with respect to the traffic variation? What is the benefit that we can obtain from a fine-grained control at the request level?
ii. How to perform dynamic devolution?
iii. How to make a trade-off between dynamic switch-controller association and dynamic control devolution?

In this paper, we consider a general SDN network with traffic variations, resulting variations of requests to handle with flow events. We assume each request can be either processed at switch (incurs computation costs) or be uploaded to certain controllers (incurs communication costs) [1]. We aim at reducing the computational cost by control devolution at data plane, the communication cost by switch-user association between data plane and control plane, and the response time experienced by switches, which is mainly caused by queueing delay on controllers. Under such settings, we provide a new perspective and a novel scheme to answer those questions. Our key results and contributions are summarized as follows:

i. *Problem Formulation*: According to the best of our knowledge, this paper is the first to study the joint optimization problem of dynamic switch-controller association and dynamic control devolution.

ii. *Finer Granularity Control*: According to the best of our knowledge, this paper is the first to perform the control decisions at the granularity of request-level. Note that request-level information such as time-varying queue backlog sizes and number of request arrivals presents the actual time-varying state of data plane. Hence it'd help for more accurate decision making of dynamic association and dynamic devolution when compared to coarse-grained control.

iii. *Online Algorithm Design* We formulate a stochastic network optimization problem, aiming at minimizing the long-term average sum of communication cost and computational cost, while keeping time-average queue backlogs of both switches and controllers small [2]. By employing Lyapunov drift technique [11] and exploiting sub-problems structure, we develop an efficient greedy algorithm to achieve optimality asymptotically. Our algorithm is online, which means it does not need the statistics of traffic workloads and does not need the prior assumption of traffic distribution.

iv. *Algorithm Analysis*: We show that our algorithm yields a tunable trade-off between $O(1/V)$ deviation from minimum long-term average sum of communication cost and computational cost and $O(V)$ bound for average queue backlogs. We also find that the positive parameter V determines the switches' willingness of uploading requests to controllers, i.e., performing switch-controller association.

v. *Simulation*: We conduct large-scale trace-driven simulations to evaluate the performance of our algorithm within two widely adopted data center networking topologies, i.e., Canonical 3-Tiered topology and Fat-tree topology. Simulation results verify the effectiveness and the trade-off of our algorithm. In addition, in the extreme case that without control devolution, we compare our dynamic association scheme with other association schemes including Static, Random, and JSQ (join the shortest queue). Simulation results verify the advantages of our schemes.

1. The scenario that some requests can only be processed by controller is a straightforward extension of our model.

2. By applying *Little's law*, small queue backlog implies small queueing delay or short response time.

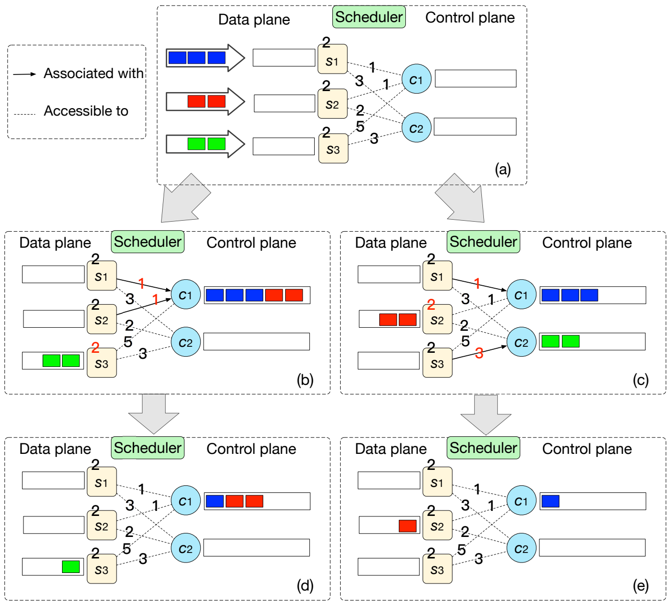


Fig. 1. An example that shows the request-level scheduling process. There are 3 switches(s_1, s_2, s_3), 2 controllers(c_1, c_2), and 1 global scheduler. Each switch or controller maintains a queue that buffers requests. During each time slot, each switch can decide to store and process its requests locally or to upload requests to some controller at each time slot. Each controller can serve 2 requests while each switch can serve only 1 request. There is a communication cost per request if switches upload requests to controllers, and a computational cost (2 per request on each switch) from local processing by switches themselves. At the beginning of time slot t, s_1, s_2, and s_3 generates 3, 2, and 2 requests, respectively. The scheduler then collects system dynamics and decides a switch-controller association (could be (b) or (c)), aiming at minimizing the sum of communication cost (could be the number of hops, RTTs, etc.) and computational cost, as well as maintaining small queue backlog size. Each switch chooses to either locally process its requests or send them to controllers according to the scheduling decision.

We organize the rest of paper as follows. We present the basic idea and formulation in Section 2. Then we show our algorithm design and corresponding performance analysis in Section 3. In Section 4, we present simulation results. We conclude this paper in Section 5.

2 Problem Formulation

In this section, we first provide a motivating example for the dynamic switch-controller association and dynamic control devolution. Then we introduce the system model and formulate the problem.

2.1 Motivating Example

The example of dynamic association and devolution is shown in Fig. 1.

First, we focus on the behavior of s_3. In Fig. 1 (b), s_3 chooses to process its requests locally, and that incurs a computational cost of 2 per request. In Fig. 1 (c), s_3 decides to upload requests to c_2 and that incurs a communication cost of 3 per request. Although the computational cost is less than communication cost, the decision of locally processing leaves one request not processed yet at the end of the time slot. Hence, it is not necessarily a smart decision for a switch to perform control devolution when its computational cost is lower than its communication cost. Instead, the scheduler should jointly decide control devolution and switch-controller association at the same time.

Second, we focus on the behavior of associations. Fig. 1 (b) and (c) show two different associations. Fig. 1 (b) shows the switch-controller association with (s_1, c_1) and (s_2, c_1) (s_3 processes requests locally), denoted by X_1. X_1 incurs the total cost of communication and computation by 9 but results in uneven queue backlogs, leaving four requests unfinished after time slot t. Fig. 1 (c) shows another association with (s_1, c_1) and (s_3, c_2) (s_2 processes requests locally), denoted by X_2. X_2 incurs the total cost by 13 but does better in balancing queue backlogs. Thus there is a non-trivial trade-off between minimizing the total cost of communication and computation and maintaining small queue backlogs on each controller.

2.2 Problem Formulation

We consider a time slotted network system, indexed by $\{0, 1, 2, \dots\}$. Its control plane comprises a set $\mathcal{C}$ of physically distributed controllers, while its data plane consists of a set of switches $\mathcal{S}$. Each switch $i \in \mathcal{S}$ keeps a queue backlog of size $Q_i^s(t)$ for locally processing requests, while each controller $j \in \mathcal{C}$ maintains a queue backlog $Q_j^c(t)$ that buffers requests from data plane. We denote $[Q_1^c(t), \dots, Q_{|\mathcal{C}|}^c(t)]$ as $\mathbf{Q}^c(t)$ and $[Q_1^s(t), \dots, Q_{|\mathcal{S}|}^s(t)]$ as $\mathbf{Q}^s(t)$. We use $\mathbf{Q}(t)$ to denote $[\mathbf{Q}^s(t), \mathbf{Q}^c(t)]$.

At the beginning of time slot t, each switch $i \in \mathcal{S}$ generates some amounts $0 \le A_i(t) \le a_{max}$ of requests. Then each switch could choose to process its requests either locally or by sending to its associated controller. We assume that each switch $i \in \mathcal{S}$ has a service rate $0 \le U_i(t) \le u_{max}$ to process the devoluted requests, while each controller $j \in \mathcal{C}$ has an available service rate $0 \le B_j(t) \le b_{max}$. We denote $[A_1(t), \dots, A_{|\mathcal{S}|}(t)]$ as $\mathbf{A}(t)$, $[B_1(t), \dots, B_{|\mathcal{C}|}(t)]$ as $\mathbf{B}(t)$, and $[U_1(t), \dots, U_{|\mathcal{S}|}(t)]$ as $\mathbf{U}(t)$. For $i \in \mathcal{S}$ and $j \in \mathcal{C}$, we assume that all $A_i(t)$, $B_j(t)$, and $U_i(t)$ are i.i.d.; besides, their first and second raw moments are all finite.

Then the scheduler collects system dynamics information $(\mathbf{A}(t), \mathbf{B}(t), \mathbf{Q}(t))$ during current time slot and makes a scheduling decision, denoted by an association matrix $\mathbf{X}(t) \in \{0, 1\}^{|\mathcal{S}| \times |\mathcal{C}|}$. Here $\mathbf{X}(t)_{i,j} = 1$ if switch i will be associated with controller j during current time slot and 0 otherwise. An association is feasible if it guarantees that each switch is associated with at most one controller during each time slot. We denote the set of feasible associations as $\mathcal{A}$,

$$\mathcal{A} \triangleq \left\{ \mathbf{X} \in \{0,1\}^{|\mathcal{S}| \times |\mathcal{C}|} \mid \sum_{j \in \mathcal{C}} \mathbf{X}_{i,j} \le 1 \text{ for } i \in \mathcal{S} \right\} \tag{1}$$

According to the scheduling decision, each switch i sends its request to controller j if $\mathbf{X}_{i,j} = 1$. However, if $\sum_{j \in \mathcal{C}} \mathbf{X}_{i,j} = 0$, switch i appends its requests to local queue backlog. Then both switches and controllers serve as many requests in their queues as they could[3]. As a result, the update equation for $Q_i^s(t)$ at switch i is

$$Q_i^s(t+1) = \left[Q_i^s(t) + \left(1 - \sum_{j \in \mathcal{C}} \mathbf{X}_{i,j}(t) \right) A_i(t) - U_i(t) \right]^+ \tag{2}$$

and the update equation for $Q_j^c(t)$ at controller j is given by

$$Q_j^c(t+1) = \left[Q_j^c(t) + \sum_{i \in \mathcal{S}} \mathbf{X}_{i,j}(t) \cdot A_i(t) - B_j(t) \right]^+ \tag{3}$$

where $[\,x\,]^+ = \max(x, 0)$.

Having covered the necessary notations and queueing dynamics, we switch to the objective and constraints of our problem.

2.2.1 *Time-Average Communication Cost*

We define the communication cost between switch i and controller j as $W_{i,j}$ [4]. Accordingly, we have a communication cost matrix $\mathbf{W} = \{W_{i,j}\}$. Fixing some association $\mathbf{X} \in \mathcal{A}$, the communication cost within one time slot is

$$f_{\mathbf{X}}(t) = \hat{f}(\mathbf{X}, \mathbf{A}(t)) \triangleq \sum_{j \in \mathcal{C}} \sum_{i \in \mathcal{S}} W_{i,j} \cdot \mathbf{X}_{i,j} \cdot A_i(t) \tag{4}$$

where we can view $W_{i,j}$ as the price of transmitting one request from switch i to controller j. Then, given a series of associations $\{\mathbf{X}_0, \mathbf{X}_1, \dots, \mathbf{X}_{t-1}\}$, the time-average expectation of communication cost is shown as follows

$$\bar{f}(t) \triangleq \frac{1}{t} \sum_{\tau=0}^{t-1} E\left\{ f_{\mathbf{X}_\tau}(\tau) \right\} \tag{5}$$

2.2.2 *Time-average Computational Cost*

There is a computational cost α_i for each devoluted request to i when switch i appends its requests to its local queue backlog for processing. Given some association $\mathbf{X} \in \mathcal{A}$, we define the one-time-slot computational cost as

$$g_{\mathbf{X}}(t) = \hat{g}(\mathbf{X}, \mathbf{A}(t)) \triangleq \sum_{i \in \mathcal{S}} \alpha_i \cdot \left(1 - \sum_{j \in \mathcal{C}} \mathbf{X}_{i,j} \right) \cdot A_i(t) \tag{6}$$

Given a series of associations $\{\mathbf{X}_0, \mathbf{X}_1, \dots, \mathbf{X}_{t-1}\}$, the time-average expectation of computational cost is

$$\bar{g}(t) \triangleq \frac{1}{t} \sum_{\tau=0}^{t-1} E\left\{ g_{\mathbf{X}_\tau}(\tau) \right\} \tag{7}$$

2.2.3 *Queueing Stability*

In this paper, we say that a queueing process $\{Q(t)\}$ is stable, if the following condition holds:

$$\lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} E\left\{ Q(\tau) \right\} < \infty \tag{8}$$

3. Note that we do not fix the serving principle, which can be *FIFO*, *LIFO*, etc. But in our simulation, we use *FIFO*.

4. The communication cost can be the number of hops or round trip times (RTT).

Accordingly, for each switch $s \in \mathcal{S}$, the queueing process $\{\mathbf{Q}^s(t)\}$ is stable if

$$\lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{i\in\mathcal{S}} E\left\{Q_i^s(\tau)\right\} < \infty \tag{9}$$

Likewise, for each controller $c \in \mathcal{C}$, the queueing process $\{\mathbf{Q}^c(t)\}$ is stable if

$$\lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{j\in\mathcal{C}} E\left\{Q_j^c(\tau)\right\} < \infty \tag{10}$$

Queueing stability implies that both switches and controllers would process buffered requests timely, so that queueing delay is controlled within a small range.

Consequently, our problem formulation is given as follows

$$\begin{array}{cc} \underset{\mathbf{X}(t)\in\mathcal{A} \text{ for } t\in\{0,1,2,\dots\}}{\text{Minimize}} & \underset{t\to\infty}{\lim\sup} \left(\bar{f}(t) + \bar{g}(t)\right) \\ \text{subject to} & (2), (3), (10), (9). \end{array} \tag{11}$$

3 Algorithm Design and Analysis of its Performance

In this section, we solve our stochastic optimization problem (11) by first transforming it into a series of one-time-slot problems, and designing optimal algorithm that solves the problem in each time slot. Our algorithm design is then followed by a theoretical analysis of its performance.

3.1 Algorithm Design

To design a scheduling algorithm that solves problem (11), we adopt the Lyapunov optimization technique in [11].

Define the quadratic Lyapunov function as

$$L(\mathbf{Q}(t)) \triangleq \frac{1}{2}\left(\sum_{j\in\mathcal{C}} \left(Q_j^c(t)\right)^2 + \sum_{i\in\mathcal{S}} \left(Q_i^s(t)\right)^2\right) \tag{12}$$

Next, we define the conditional Lyapunov drift for two consecutive time slots as

$$\Delta\left(\mathbf{Q}(t)\right) \triangleq E\left\{L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t)) \,|\, \mathbf{Q}(t)\right\} \tag{13}$$

This conditional difference measures the general change in queues' congestion state. We want to push such difference as low as possible, so as to prevent queues $\mathbf{Q}^s(t)$ and $\mathbf{Q}^c(t)$ from being overloaded. However, to maintain small queue backlogs, the action we take, e.g. $\mathbf{X}$, might incur considerable communication cost $f_{\mathbf{X}}(t)$ or computational cost $g_{\mathbf{X}}(t)$, or both. Hence, we should jointly consider both queueing stability and the total cost $f_{\mathbf{X}}(t) + g_{\mathbf{X}}(t)$.

Given any feasible association $\mathbf{X}$, we define one-time-slot conditional drift-plus-penalty function as

$$\Delta_V(\mathbf{Q}(t)) \triangleq \Delta(\mathbf{Q}(t)) + V \cdot E\left\{f_{\mathbf{X}}(t) + g_{\mathbf{X}}(t) | \mathbf{Q}(t)\right\} \tag{14}$$

where $f_{\mathbf{X}}(t)$ is defined by (4), $g_{\mathbf{X}}(t)$ is defined by (6), and $V > 0$ is a constant that weights the penalty brought by $f_{\mathbf{X}}(t)$ and $g_{\mathbf{X}}(t)$.

By minimizing the upper bound of the drift-plus-penalty expression (14), the time-average communication cost can be minimized while stabilizing the network of request queues.

We employ the concept of *opportunistically minimizing an expectation* in [11], and we transform the long-term stochastic optimization problem (11) into the following drift-plus-penalty minimization problem at every time slot t.

$$\begin{aligned} \underset{\mathbf{X}\in\mathcal{A}}{\text{Minimize}} \quad & V \cdot \left(\hat{f}(\mathbf{X}, \mathbf{A}(t)) + \hat{g}(\mathbf{X}, \mathbf{A}(t))\right) + \\ & \sum_{j\in\mathcal{C}} Q_j^c(t) \cdot \left[\sum_{i\in\mathcal{S}} \mathbf{X}_{i,j} \cdot A_i(t)\right] + \\ & \sum_{i\in\mathcal{S}} Q_i^s(t) \cdot \left[(1 - \sum_{j\in\mathcal{C}} \mathbf{X}_{i,j}) \cdot A_i(t)\right] \end{aligned} \tag{15}$$

After rearranging the terms in (15), our optimization problem turns out to be

$$\begin{aligned} \underset{\mathbf{X}\in\mathcal{A}}{\text{Minimize}} \quad & \sum_{i\in\mathcal{S}} \left[V\alpha_i + Q_i^s(t)\right] A_i(t) + \sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{C}} [VW_{i,j} + \\ & Q_j^c(t) - V\alpha_i - Q_i^s(t)] \, \mathbf{X}_{i,j} A_i(t) \end{aligned} \tag{16}$$

Since the first summing term $\sum_{i\in\mathcal{S}}[V\alpha_i + Q_i^s(t)]$ in (16) has nothing to do with $\mathbf{X}$, then we fix it as constant and put our focus on minimizing the second term of (16) only.

For each $i \in \mathcal{S}$, we split $\mathcal{C}$ into two disjoint sets $\mathcal{J}_1^i$ and $\mathcal{J}_2^i$, i.e. $\mathcal{J}_1^i \dot{\bigcup} \mathcal{J}_2^i = \mathcal{C}$, and

$$\mathcal{J}_1^i \triangleq \{j \in \mathcal{C} \,|\, VW_{i,j} + Q_j^c(t) > V\alpha_i + Q_i^s(t)\} \tag{17}$$

$$\mathcal{J}_2^i \triangleq \{j \in \mathcal{C} \,|\, VW_{i,j} + Q_j^c(t) \le V\alpha_i + Q_i^s(t)\} \tag{18}$$

Then, for each switch $i \in \mathcal{S}$,

$$\begin{aligned} & \sum_{j\in\mathcal{C}} \left[VW_{i,j} + Q_j^c(t) - V\alpha_i - Q_i^s(t)\right] \mathbf{X}_{i,j} A_i(t) \\ = & \left\{ \sum_{j\in\mathcal{J}_1^i} \left[VW_{i,j} + Q_j^c(t) - V\alpha_i - Q_i^s(t)\right] \mathbf{X}_{i,j} + \right. \\ & \left. \sum_{j\in\mathcal{J}_2^i} \left[VW_{i,j} + Q_j^c(t) - V\alpha_i - Q_i^s(t)\right] \mathbf{X}_{i,j} \right\} A_i(t) \end{aligned} \tag{19}$$

Next, we show how to minimize (19) with $\mathbf{X} \in \mathcal{A}$. Given any $(i, j) \in \mathcal{S} \times \mathcal{C}$, we define

$$\omega(i, j) \triangleq VW_{i,j} + Q_j^c(t) - V\alpha_i - Q_i^s(t) \tag{20}$$

Here, we define $\mathbf{X}^*$ as the optimal solution to minimize (19). For each switch $i \in \mathcal{S}$, we should consider two different cases.

i. If $\mathcal{J}_2^i = \emptyset$, i.e., $\omega(i, j) > 0$ for all $j \in \mathcal{C}$, then the only way to minimize (19) is setting $\mathbf{X}_{i,j}^* = 0$ for all $j \in \mathcal{C}$.
ii. If $\mathcal{J}_2^i \neq \emptyset$, then we handle with $\mathbf{X}_{i,j}$ for $j \in \mathcal{J}_1^i$ and $j \in \mathcal{J}_2^i$ separately.
    - For $j \in \mathcal{J}_1^i$, to minimize (19), it is not hard to see we should set $\mathbf{X}_{i,j}^* = 0$ for all $j \in \mathcal{J}_1^i$.
    - For $j \in \mathcal{J}_2^i$, $\omega(i, j) \le 0$. Then we should make $\mathbf{X}_{i,j^*}^* = 1$ for such j^* that

    $$j^* = \underset{j\in\mathcal{J}_2^i}{\arg\min} \quad \omega(i, j) \tag{21}$$

    and $\mathbf{X}_{i,j}^* = 0$ for $j \in \mathcal{J}_2^i - \{j^*\}$.

In such a way, given any $\mathbf{X}' \in \mathcal{A}$, for switch i the following always holds

$$\begin{aligned}
&\sum_{j\in\mathcal{J}_1^i} \omega(i,j)\cdot\mathbf{X}'_{i,j} + \sum_{j\in\mathcal{J}_2^i} \omega(i,j)\cdot\mathbf{X}'_{i,j} \\
\geq\ & \left[\sum_{j\in\mathcal{J}_1^i} \omega(i,j)\right]\cdot 0 + \min_{j\in\mathcal{J}_2^i}\omega(i,j) \qquad (22)\\
=\ & \sum_{j\in\mathcal{J}_1^i} \omega(i,j)\cdot\mathbf{X}^*_{i,j} + \sum_{j\in\mathcal{J}_2^i} \omega(i,j)\cdot\mathbf{X}^*_{i,j}
\end{aligned}$$

Therefore, the association $\mathbf{X}^*$ produced by the above process is the optimal solution that minimizes (16).

As a result, we have the algorithm shown as follows:

Algorithm 1 Greedy Scheduling Algorithm

```
Input: During time slot $t$, the scheduler collects queue
    lengths information from individual controllers and
    switches, i.e. $\mathbf{Q}^c(t)$, $\mathbf{Q}^s(t)$, and $\mathbf{A}(t)$
Output: A scheduling association $\mathcal{X} \subset \mathcal{S}\times\mathcal{C}$
1: Start with an empty set $\mathcal{X} \leftarrow \emptyset$
2: for each switch $i \in \mathcal{S}$ do
3:     Split all controllers $\mathcal{C}$ into two sets $\mathcal{J}_1^i$ and $\mathcal{J}_2^i$, where
       $\mathcal{J}_1^i = \{j\in\mathcal{C} \,|\, \omega(i,j) > 0\}$ and
       $\mathcal{J}_2^i = \{j\in\mathcal{C} \,|\, \omega(i,j) \leq 0\}$
4:     If $\mathcal{J}_2^i = \emptyset$, then skip current iteration.
5:     If $\mathcal{J}_2^i \neq \emptyset$, then choose controller $j^* \in \mathcal{J}_2^i$ such that
           $j^* \in \arg\min_{j\in\mathcal{C}} \omega(i,j)$
6:     $\mathcal{X} \leftarrow \mathcal{X} \bigcup \{(i,j^*)\}$
7: end for
8: return $\mathcal{X}$
   According to $\mathcal{X}$, switches upload requests to controllers or
   append requests to their local queues. Then controllers and
   switches update their queue backlogs as in (2) and (3) after
   serving requests.
```

Remarks:

i. Our algorithm is greedy since it greedily associates each switch with controllers that either with small queue backlog size or close to the switch, and otherwise it leaves all requests locally processed.

ii. For switch i, given any controller j far enough from i, i.e., $W_{i,j} > \alpha_i$, switch i decides to upload requests to j only if $\omega(i,j)$ is non-positive and smaller than any other. This requires switch i itself holds enough requests locally, i.e., $Q_i^s(t) \geq V\cdot(W_{i,j}-\alpha_i)+Q_j^c(t)$. Then it will upload requests. Thus smaller V will invoke more effectively the willingness of switch i to upload requests to control plane.

iii. On the other hand, given any controller k close to switch i, i.e., $W_{i,j} < \alpha_i$, switch i will process requests locally if control plane holds large amounts of requests, i.e., $Q_i^s(t) < V\cdot(W_{i,j}-\alpha_i) + Q_j^c(t)$. Thus given any large V, controllers will have to hold great loads of requests before switches become willing to process requests locally.

iv. Therefore, the parameter V actually controls switches' willingness of uploading requests to controllers, i.e., performing switch-controller association. In other words, it controls the trade-off between communication cost and the computational cost, which are incurred by uploading requests to control plane and locally processing, respectively.

3.2 Performance Analysis

Next we characterize the performance of our algorithm. We suppose g^* and f^* are the infimum of time-average computational cost and communication cost that we want to achieve, respectively. We also suppose $d_{\max} = \max_{i,j}(E(B_j^2(t)), E(U_i^2(t)), E(A_i^2(t)))$. The we have the following theorem[5] on the $O(1/V), O(V)$ trade-off between costs and queue backlogs:

Theorem 1. Given the parameters $V > 0$, $\epsilon > 0$, and constant $K \geq \dfrac{d_{\max}\cdot(|\mathcal{C}|+|\mathcal{S}|+|\mathcal{S}|^2)}{2}$, then the queue vector process $\mathbf{Q}(t)$ is stable; besides, the time-average expectation of communication cost and computational cost, as well as queue backlogs on switches and controllers satisfy:

$$\begin{aligned}
&i. \quad \limsup_{t\to\infty}\left(\bar{f}(t)+\bar{g}(t)\right) \leq f^* + g^* + \frac{K}{V}\\
&ii. \quad \limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\left[\sum_{j\in\mathcal{C}} E\left\{Q_j^c(\tau)\right\} + \sum_{i\in\mathcal{S}} E\left\{Q_i^s(\tau)\right\}\right]\\
&\qquad \leq \frac{K + V\cdot(f^*+g^*)}{\epsilon}
\end{aligned} \qquad (23)$$

4 Simulation Results

4.1 Basic Settings

Topology: We evaluate our **Greedy** scheduling algorithm under two widely adopted topologies: Canonical 3-Tiered topology and Fat-tree [13] topology. We show two of their instances in Fig. 2 and Fig. 3, respectively.

To make our performance analysis comparable in both topologies, we construct a Fat-tree network and a Canonical 3-Tiered network with almost the same number of switches. Regarding the Canonical 3-Tiered topology, we set the number of core switches as 26. Accordingly, the total number of switches is 702. Regarding the Fat-tree topology, we set the port number as 24 and thusly there are 720 switches in total. Note that the two resulting topologies are also comparable to the size of commercial data centers [14].

In both topologies, we deploy controllers on the hosts (one controller for every two pods[6]), which are denoted by the blue circles in Fig. 2 and Fig. 3.

Traffic Workloads: We conduct trace-driven simulations, where the flow arrival process on each switch follows the distribution of flow inter-arrival time in [14], which is drawn from measurements within real-world data centers. In [14], the average flow inter-arrival time is about $1700\mu s$. We then set the length of each time slot as $10ms$. Accordingly, the average flow arrival rate on each switch is about 5.88 flows per time slot.

5. See the proof in appendix B.
6. In Canonical 3-Tiered topology, we regard the group of switches that affiliate the same aggregation switch as one pod (including the aggregation switch itself).

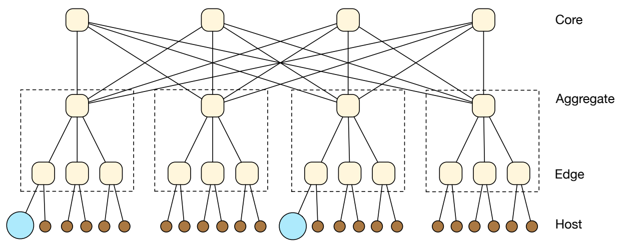


Fig. 2. An instance of Canonical 3-Tiered topology with $k = 4$. k denotes the number of core switches. In this paper, the number of aggregate switches is also set to k, and each connects to $k - 1$ edge switches. The total number of switches is $k^2 + k$. Each edge switch is directly connected to $\frac{k}{2}$ hosts. Therefore, there are $\frac{k^3-k^2}{2}$ hosts in total.

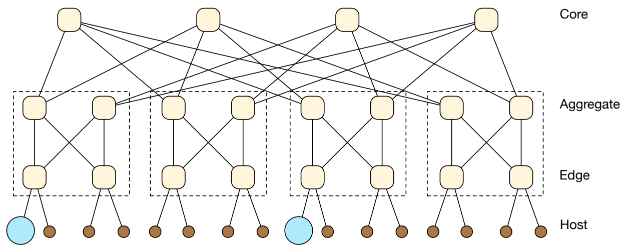


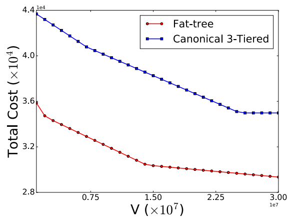
Fig. 3. An instance of Fat-tree topology with $k = 4$. k denotes the number of pods. The number of core, aggregate, edge switches are $\frac{k^2}{4}$, $\frac{k^2}{2}$, $\frac{k^2}{2}$, respectively. And the total number of switches is $\frac{5}{4}k^2$. Each edge switch is directly connected to $\frac{k}{2}$ hosts. Accordingly, there are $\frac{k^3}{4}$ hosts in total.

In fact, there do exist hot spots within pods in real-world data center networks, where the switches have significantly high flow arrival rates. In our simulation, we pick the first pod as a hot spot and all switches there have significantly high flow arrival rate, i.e., 200 flows per time slot. As for controllers, we set their individual capacity as 600 flows per time slot. That is consistent with the capacity of a typical NOX controller [15].

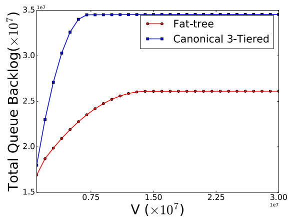
Costs: Given any network topology, we define the communication cost $W_{i,j}$ between switch i and controller j as the length (number of hops) of shortest path from i to j. Then we set a common computation cost α for all switches, which equals to the average hop number between switches and controllers of its underlying topology. In Fat-tree topology, $\alpha = 4.13$; while in 3-Tiered topology, $\alpha = 4.81$.

4.2 Evaluation of Greedy Algorithm

Fig. 4(a) presents how the summation of long-term average communication cost and computational cost changes with different V in Fat-tree and 3-Tiered topologies. As V varies from 0 to 3.0×10^7, we can observe that the total cost goes down gradually. This is consistent with our previous theoretic analysis. The intuition behind such decline is as follows. Remind that V controls the switches' willingness of uploading requests. For switches that are close to controllers (their communication cost is less than the average), large V makes them unwilling to process requests locally unless the controllers get too heavy load. As V increases, those switches will choose to upload requests to further reduce the costs since for those switches, communication costs are less than the computation cost. Another observation we make

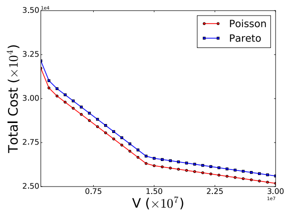


(a) Total cost vs. V

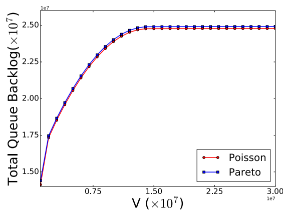


(b) Total queue backlog vs. V

Fig. 4. Performance of **Greedy** under Fat-tree topology and Canonical 3-Tiered topology in terms of (a) the sum of total communication cost and computational cost, and (b) total queue backlog.



(a) Total cost vs. V



(b) Total queue backlog vs. V

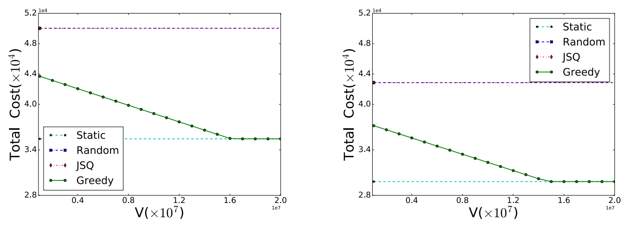
Fig. 5. Performance of **Greedy** under Fat-tree topology with request arrivals that follow Poisson and Pareto process in terms of (a) the sum of total communication cost and computational cost, and (b) total queue backlog.

is that the total cost of 3-Tiered topology is more than Fat-tree's. The reason is that 3-Tiered has a higher computational cost ($\alpha = 4.81$ compared to 4.13) and it cost more when switches process requests locally. In Fig. 5(a), we also show the total cost of **Greedy** in Fat-tree topology with other two request arrival processes. The curves of Poisson and Pareto exhibit qualitatively similar decline in total cost, although Pareto incurs slightly ($\sim 1.5\%$) larger queue backlog size than Poisson.

Fig. 4(b) shows the curve of total queue backlog size with different values of V. From the figure, we notice that total queue backlog size increases until V reaches about 0.75×10^7 and 1.5×10^7 in Fat-tree and 3-Tiered topologies, respectively. This is also consistent with the $O(V)$ queue backlog size bound in (23). Recall our analysis in *Total Cost*: larger V invokes most switches to spend more time uploading requests to control plane. However, control plane's service capacity is fixed and requests will keep accumulating. Thus when V becomes sufficiently large, control plane will eventually hold most of requests in the system. This explains the increasing queue backlog size in Fig. 4(b). Fig. 5(b) present the total queue backlog size in Fat-tree topology when we apply **Greedy** with request arrivals that follow Poisson and Pareto processes. Note that we do not show that of 3-Tiered topology because curves there almost overlap with those in Fig. 5(b).

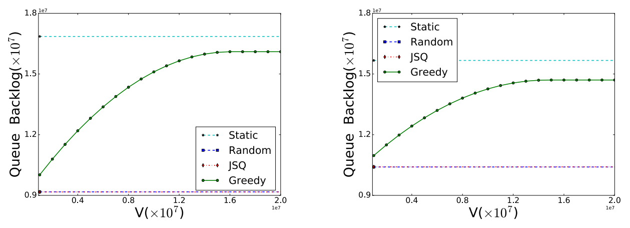
4.3 Comparison with Other Association Schemes

In this subsection, we consider the extreme case by setting common computational cost $\alpha = 2.0 \times 10^{28}$ for all switches. This means the cost of local processing requests are prohibitively high and each time slot switches choose to upload



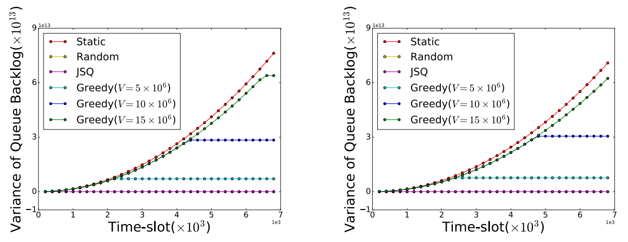
(a) Canonical 3-Tiered topology (b) Fat-tree topology

Fig. 6. Communication cost comparison among four scheduling schemes under Canonical 3-Tiered topology and Fat-tree topology, respectively.



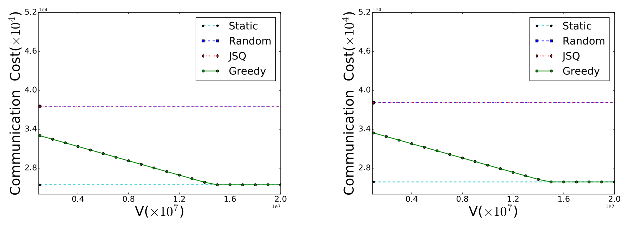
(a) Canonical 3-Tiered topology (b) Fat-tree topology

Fig. 8. Average queue backlog size comparison among four scheduling schemes under Canonical 3-Tiered topology and Fat-tree topology, respectively.



(a) Canonical 3-Tiered topology (b) Fat-tree topology

Fig. 7. Variance of queue backlog size comparison among four scheduling schemes under Canonical 3-Tiered topology and Fat-tree topology, respectively.



(a) Poisson (b) Pareto

Fig. 9. Communication cost comparison among four scheduling schemes under Fat-tree topology, when the flow arrival follows Poisson and Pareto, respectively.

requests to controllers. Thus our greedy algorithm degenerates into a dynamic switch-controller association algorithm. We compare its performance with three other schemes: **Static**, **Random** and **JSQ(Join-the-Shorest-Queue)**. In static scheme, each switch i chooses the controller j with minimum communication cost $W_{i,j}$ and then fixes this association for all time slots. In random scheme, the scheduler randomly picks up a controller for each switch at each time slot. In JSQ scheme, it randomly picks up one switch i without replacement round by round until all switches have chosen the target controllers. At each round, the selected switch i chooses the controller j with the smallest queue backlog size at present, and appends all its requests to j's queue.

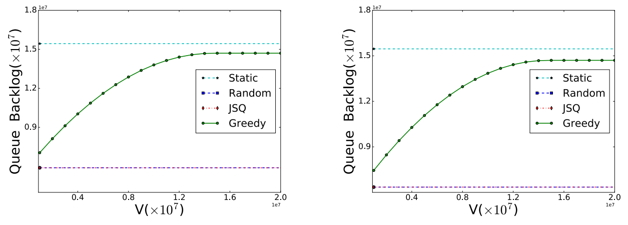
Fig. 6 presents a comparison among **Static**, **Random**, **JSQ**, and **Greedy** in terms of communication cost under Canonical 3-Tiered topology and Fat-tree, respectively. First, the communication cost under **Static** is the minimum among all schemes, which is consistent with its goal of minimizing the overall communication cost. **Greedy** cuts down the communication cost with increasing V. Eventually, when V is sufficiently large, communication cost stops decreasing and remains unchanged. Both **Random** and **JSQ** exhibit much higher communication costs, compared to **Greedy** and **Static**.

Fig. 7 presents a comparison among the four schemes in terms of the variance of queue backlog size under Canonical 3-Tiered topology and Fat-tree topology, respectively. In fact, smaller queue backlog size variance indicates better capability of load balancing. The variance of **Static** grows exponentially with time, showing that **Static** is incompetent in load balancing. The reason is that **Static** greedily associates switches with their nearest controllers, ignoring different controllers' loads. When it comes to **Random** and **JSQ**, the variance is almost 0, which shows the two schemes' advantage in load balancing. While the variance of **Greedy** is in between the other three: it increases at the beginning and then remains stable after about thousands of time slots. Furthermore, **Greedy** exhibits higher variance of queue backlog size with larger V, i.e., the load of controllers is more imbalanced.

Fig. 8 presents a comparison among the four schemes in terms of the average queue backlog size under Canonical 3-Tiered topology and Fat-tree topology, respectively. Actually, the observations in Fig. 8 are very consistent with that of Fig. 7. Intuitively, the more balanced the load of controller is, the smaller of the average queue backlog size. In Fig. 7, the variance of **Static** is high while that of **Random** and **JSQ** are almost 0, so the average queue backlog size of **Static** is large while that of **Random** and **JSQ** is small in Fig. 8. As for **Greedy**, the variance increases with V in Fig. 7, so the average queue backlog size of **Greedy** rises as V increases as shown in Fig. 8.

In addition to trace-driven simulation, we also conduct the comparison under other two assumptions of flow arrival processes, i.e., *Poisson* and *Pareto* processes. They two are widely adopted in traffic analysis. We only show the simulation results under Fat-tree topology, because the simulation results in 3-Tiered topology is qualitatively similar to that in Fat-tree topology. Fig. 9 shows the communication cost comparison when the flow arrival process follows Poisson and Pareto, respectively. Fig. 10 shows the average queue backlog size comparison when the flow arrival process follows Poisson and Pareto processes, respectively. We can see from these figures that the scheduling policies perform qualitatively consistent under different arrival processes.

In summary, among four schemes, **Static** is on the one end of performance spectrum: it minimizes communication



(a) Poisson (b) Pareto

Fig. 10. Average queue backlog size comparison among four scheduling schemes under Fat-tree topology, when the flow arrival follows Poisson and Pareto, respectively.

cost while incurring extremely large queue backlogs; both **Random** and **JSQ** are on the other end of performance spectrum: they minimize the average queue backlog while incurring much large communication costs. In contrast, our **Greedy** scheme achieves a trade-off between minimization of communication costs and minimization of queue backlogs. Through a tunable parameter V, we can achieve different degrees of balance between cost minimization and latency (queue backlog) minimization.

5 Conclusion

In this paper, we studied the joint optimization problem of dynamic switch-controller association and dynamic control devolution for SDN networks. We formulated the problem as a stochastic network optimization problem, aiming at minimizing the long-term average summation of total communication cost and computational cost while maintaining low time-average queue backlogs. We proposed an efficient online greedy algorithm, which yields a long-term average sum of communication cost and computational cost within $O(1/V)$ of optimality, with a trade-off in an $O(V)$ queue backlog size for any positive control parameter V. Extensive simulation results show the effectiveness and optimality of our online algorithm, and the ability to maintain a tunable trade-off compared to other dynamic association schemes.

Appendix A
Problem Transformation by Opportunistically Minimizing an Expectation

By minimizing the upper bound of the drift-plus-penalty expression (14), the time average of communication cost can be minimized while stabilizing the network of request queues. We denote the objective function of (16) at time slot t by $J_t(\mathbf{X})$ and its optimal solution $\mathbf{X}^* \in \mathcal{A}$.

Therefore, for any other scheduling decision $\mathbf{X} \in \mathcal{A}$ made during time slot t, we have

$$J_t(\mathbf{X}) \geq J_t(\mathbf{X}^*) \tag{24}$$

Taking the conditional expectation on both sides conditional on $\mathbf{Q}^c(t)$, we have

$$E\left[J_t(\mathbf{X}) \mid \mathbf{Q}^c(t)\right] \geq E\left[J_t(\mathbf{X}^*) \mid \mathbf{Q}^c(t)\right] \tag{25}$$

for any $\mathbf{X} \in \mathcal{A}$. In such way, instead of directly solving the long-term stochastic optimization problem (11), we can opportunistically choose a feasible association to solve problem (16) during each time slot.

Appendix B
Proof of Theorem 1

Given $L(\mathbf{Q}(t))$ defined in (12), we have

$$\begin{aligned}
& L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t)) \\
= \; & \frac{1}{2}\left(\sum_{j\in\mathcal{C}}\left[\left(Q_j^c(t+1)\right)^2 - \left(Q_j^c(t)\right)^2\right] + \sum_{i\in\mathcal{S}}\left[\left(Q_i^s(t+1)\right)^2 - \right.\right. \\
& \left.\left.\left(Q_i^s(t)\right)^2\right]\right) \\
\leq \; & \frac{1}{2}\sum_{j\in\mathcal{C}}\left\{\left(Q_j^c(t) - B_j(t) + \sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}\cdot A_i(t)\right)^2 - \left(Q_j^c(t)\right)^2\right\} + \\
& \frac{1}{2}\sum_{i\in\mathcal{S}}\left\{\left(Q_i^s(t) - U_i(t) + \mathbf{Y}_i\cdot A_i(t)\right)^2 - \left(Q_i^s(t)\right)^2\right\} \\
= \; & \frac{1}{2}\sum_{j\in\mathcal{C}}\left\{2Q_j^c(t)\cdot\left(\sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}\cdot A_i(t) - B_j(t)\right) + \right. \\
& \left.\left(\sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}\cdot A_i(t) - B_j(t)\right)^2\right\} + \frac{1}{2}\sum_{i\in\mathcal{S}}\left\{\left(\mathbf{Y}_i\cdot A_i(t) - U_i(t)\right)^2\right. \\
& \left.+2Q_i^s(t)\cdot\left(\mathbf{Y}_i\cdot A_i(t) - U_i(t)\right)\right\} \\
\leq \; & \sum_{j\in\mathcal{C}}\left\{Q_j^c(t)\cdot\left(\sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}\cdot A_i(t) - B_j(t)\right) + \right. \\
& \left.\frac{\left(\sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}\cdot A_i(t)\right)^2 + \left(B_j(t)\right)^2}{2}\right\} + \\
& \sum_{i\in\mathcal{S}}\left\{Q_i^s(t)\cdot\left(\mathbf{Y}_i\cdot A_i(t) - U_i(t)\right) + \frac{\left(Y_i\cdot A_i(t)\right)^2 + \left(U_i(t)\right)^2}{2}\right\}
\end{aligned} \tag{26}$$

Then with the definition of $\Delta(\mathbf{Q}(t))$ in (13), we have

$$\begin{aligned}
& \Delta(\mathbf{Q}(t)) \\
= \; & E\left\{L\left(\mathbf{Q}(t+1)\right) - L\left(\mathbf{Q}(t)\right) \mid \mathbf{Q}(t)\right\} \\
\leq \; & E\left\{\sum_{j\in\mathcal{C}}Q_j^c(t)\cdot\left(\sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}(t)A_i(t) - B_j(t)\right) \mid \mathbf{Q}(t)\right\} + \\
& E\left\{\sum_{i\in\mathcal{S}}Q_i^s(t)\cdot\left(\mathbf{Y}_i(t)A_i(t) - U_i(t)\right) \mid \mathbf{Q}(t)\right\} + \\
& \frac{1}{2}E\left\{\sum_{j\in\mathcal{C}}\left[\left(\sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}A_i(t)\right)^2 + \left(B_j(t)\right)^2\right] \mid \mathbf{Q}(t)\right\} + \\
& \frac{1}{2}E\left\{\sum_{i\in\mathcal{S}}\left[\left(\mathbf{Y}_iA_i(t)\right)^2 + \left(U_i(t)\right)^2\right] \mid \mathbf{Q}(t)\right\} \\
= \; & \sum_{j\in\mathcal{C}}Q_j^c(t)\cdot E\left\{\left(\sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}(t)A_i(t) - B_j(t)\right) \mid \mathbf{Q}(t)\right\} + \\
& \sum_{i\in\mathcal{S}}Q_i^s(t)\cdot E\left\{\left(\mathbf{Y}_i(t)A_i(t) - U_i(t)\right) \mid \mathbf{Q}(t)\right\} + \\
& \frac{1}{2}\sum_{j\in\mathcal{C}}\left[\left(\sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}A_i(t)\right)^2 + \left(B_j(t)\right)^2\right] + \\
& \frac{1}{2}\sum_{i\in\mathcal{S}}\left[\left(\mathbf{Y}_iA_i(t)\right)^2 + \left(U_i(t)\right)^2\right]
\end{aligned} \tag{27}$$

The last equality in (27) holds because of conditional expectation on $\mathbf{Q}(t)$, then both $Q_i^s(t)$ and $Q_j^c(t)$ can be regarded as a constant. Besides, the queueing process $\{\mathbf{Q}(t)\}$ is independent of the arrival process $\{\mathbf{A}(t)\}$ and service process $\{\mathbf{B}(t)\}$, $\{\mathbf{U}(t)\}$. Hence, the last two terms have nothing to do with $\mathbf{Q}(t)$. Now consider the last two terms in (27). We have

$$\begin{aligned}
&\frac{1}{2}\sum_{j\in\mathcal{C}}\left[\left(\sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}A_i(t)\right)^2+(B_j(t))^2\right]+\\
&\frac{1}{2}\sum_{i\in\mathcal{S}}\left[(\mathbf{Y}_iA_i(t))^2+(U_i(t))^2\right]\\
=\;&\frac{1}{2}\left[\sum_{j\in\mathcal{C}}(B_j(t))^2+\sum_{i\in\mathcal{S}}(U_i(t))^2\right]+\\
&\frac{1}{2}\sum_{j\in\mathcal{C}}\left[\left(\sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}A_i(t)\right)^2\right]+\\
&\frac{1}{2}\sum_{i\in\mathcal{S}}\left[\left((1-\sum_{j\in\mathcal{C}}\mathbf{X}_{i,j})\cdot A_i(t)\right)^2\right]
\end{aligned}\tag{28}$$

Then by taking expectation on (28), the following holds

$$\begin{aligned}
&E\left\{\frac{1}{2}\sum_{j\in\mathcal{C}}\left[\left(\sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}A_i(t)\right)^2+(B_j(t))^2\right]+\right.\\
&\left.\frac{1}{2}\sum_{i\in\mathcal{S}}\left[(\mathbf{Y}_iA_i(t))^2+(U_i(t))^2\right]\right\}\\
=\;&E\left\{\frac{1}{2}\left[\sum_{j\in\mathcal{C}}(B_j(t))^2+\sum_{i\in\mathcal{S}}(U_i(t))^2\right]+\right.\\
&\frac{1}{2}\sum_{j\in\mathcal{C}}\left[\left(\sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}A_i(t)\right)^2\right]+\\
&\left.\frac{1}{2}\sum_{i\in\mathcal{S}}\left[\left((1-\sum_{j\in\mathcal{C}}\mathbf{X}_{i,j})\cdot A_i(t)\right)^2\right]\right\}\\
=\;&\frac{1}{2}\left[\sum_{j\in\mathcal{C}}E\left\{(B_j(t))^2\right\}+\sum_{i\in\mathcal{S}}E\left\{(U_i(t))^2\right\}\right]+\\
&\frac{1}{2}\sum_{j\in\mathcal{C}}E\left\{\left(\sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}A_i(t)\right)^2\right\}+\\
&\frac{1}{2}\sum_{i\in\mathcal{S}}E\left\{(1-\sum_{j\in\mathcal{C}}\mathbf{X}_{i,j})^2\cdot(A_i(t))^2\right\}\Bigg\}\\
=\;&\frac{1}{2}\left[\sum_{j\in\mathcal{C}}E\left\{(B_j(t))^2\right\}+\sum_{i\in\mathcal{S}}E\left\{(U_i(t))^2\right\}\right]+\\
&\frac{1}{2}\sum_{j\in\mathcal{C}}E\left\{\sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}^2\,(A_i(t))^2+2\sum_{i<i'}\mathbf{X}_{i,j}\mathbf{X}_{i',j}A_i(t)A_{i'}(t)\right\}+\\
&\frac{1}{2}\sum_{i\in\mathcal{S}}E\left\{(1-\sum_{j\in\mathcal{C}}\mathbf{X}_{i,j})^2\cdot(A_i(t))^2\right\}
\end{aligned}\tag{29}$$

Remind that the request arrival processes $\{\mathbf{A}(t)\}$ are independent and they are also independent of $\mathbf{X}_{i,j}$ for $(i,j)\in\mathcal{S}\times\mathcal{C}$. Then we have

$$\begin{aligned}
&E\left\{\frac{1}{2}\sum_{j\in\mathcal{C}}\left[\left(\sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}A_i(t)\right)^2+(B_j(t))^2\right]+\right.\\
&\left.\frac{1}{2}\sum_{i\in\mathcal{S}}\left[(\mathbf{Y}_iA_i(t))^2+(U_i(t))^2\right]\right\}\\
=\;&\frac{1}{2}\left[\sum_{j\in\mathcal{C}}E\left\{(B_j(t))^2\right\}+\sum_{i\in\mathcal{S}}E\left\{(U_i(t))^2\right\}\right]+\\
&\frac{1}{2}\left[\sum_{j\in\mathcal{C}}\sum_{i\in\mathcal{S}}E\left\{\mathbf{X}_{i,j}^2\right\}E\left\{(A_i(t))^2\right\}+\right.\\
&\left.2\sum_{i<i'}E\left\{\mathbf{X}_{i,j}\right\}E\left\{\mathbf{X}_{i',j}\right\}E\left\{A_i(t)\right\}E\left\{A_{i'}(t)\right\}\right]+\\
&\frac{1}{2}\sum_{i\in\mathcal{S}}E\left\{(1-\sum_{j\in\mathcal{C}}\mathbf{X}_{i,j})^2\right\}\cdot E\left\{(A_i(t))^2\right\}\\
\le\;&\frac{1}{2}\left(C\cdot\max_{j\in\mathcal{C}}\{E(B_j^2(t))\}+S\cdot\max_{i\in\mathcal{S}}\{E(U_i^2(t))\}+\right.\\
&\max_{i\in\mathcal{S}}\{E(A_i^2(t))\}\left[\sum_{j\in\mathcal{C}}E\left\{\left(\sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}\right)^2\right\}+\right.\\
&\left.\left.\sum_{i\in\mathcal{S}}E\left\{\left(1-\sum_{j\in\mathcal{C}}\mathbf{X}_{i,j}\right)^2\right\}\right]\right)\\
\le\;&\frac{1}{2}\max_{i,j}(E(B_j^2(t)),E(U_i^2(t)),E(A_i^2(t)))\cdot\\
&\left(C+S+\max_{\mathbf{X}\in\mathcal{A}}\left\{\sum_{j\in\mathcal{C}}\left(\sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}\right)^2+\sum_{i\in\mathcal{S}}(\mathbf{Y}_i(t))^2\right\}\right)
\end{aligned}\tag{30}$$

where the first inequality holds because of the following reasoning. We suppose $i^*\in\arg\max_{i\in\mathcal{S}}A_i(t)$. Then for any $i,i'\in\mathcal{S}$

$$\begin{aligned}
&E\left\{A_i(t)\right\}\cdot E\left\{A_{i'}(t)\right\}\\
\le\;&\left(E\left\{A_{i^*}(t)\right\}\right)^2
\end{aligned}\tag{31}$$

As we know that $\text{Var}\left\{A_{i^*}(t)\right\}\ge 0$, then

$$\begin{aligned}
&E\left\{A_i(t)\right\}\cdot E\left\{A_{i'}(t)\right\}\\
\le\;&E\left\{A_{i^*}^2(t)\right\}
\end{aligned}\tag{32}$$

Thus the first inequality in (30) holds.

Next, we focus on the upper bound of $\sum_{j\in\mathcal{C}}\left(\sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}\right)^2+\sum_{i\in\mathcal{S}}\left(1-\sum_{j\in\mathcal{C}}\mathbf{X}_{i,j}\right)^2$ for $\mathbf{X}\in\mathcal{A}$. At each time slot, $\mathbf{X}_{i,j}\in\{0,1\}$ and for each switch $i\in\mathcal{S}$, it must decide either to upload requests to one of controllers or process them locally. Then among all $\mathbf{X}_{i,j}$ (for all $(i,j)\in\mathcal{S}\times\mathcal{C}$) and $(1-\sum_{j\in\mathcal{C}}\mathbf{X}_{i,j})$ (for $i\in\mathcal{S}$), there are exactly $|\mathcal{S}|$ of them that's equal to one. Let $a\in[0,|\mathcal{S}|]$ denote the number of switches that decide to upload requests to control plane, i.e., there are a terms among all $\mathbf{X}_{i,j}$ (for $(i,j)\in\mathcal{S}\times\mathcal{C}$) that's equal to one. Likewise, let $b\in[0,|\mathcal{S}|]$ denote the number of switches that process requests locally. Accordingly, we know that $a+b=|\mathcal{S}|$. Besides, $\sum_{i\in\mathcal{S}}\left(1-\sum_{j\in\mathcal{C}}\mathbf{X}_{i,j}\right)^2=b$ since there are exactly b switches such that for any switch i among them $\sum_{j\in\mathcal{C}}\mathbf{X}_{i,j}=0$.

Now we prove that the upper bound of $\sum_{j\in\mathcal{C}}\left(\sum_{i\in\mathcal{S}}\mathbf{X}_{i,j}\right)^2$ is a^2 and the bound is reached when all a switches is associated with the same controller. We use $\mathcal{R}$ to denote the set of those a switches. We introduce indicator $\mathcal{I}_{k,l}$ such that $\mathcal{I}_{k,l} = 1$ if switch k and switch l are associated with the same controller and 0 otherwise. Therefore, for any switch-controller association $\mathcal{N} \subseteq \mathcal{R} \times \mathcal{C}$ such that $|\mathcal{N}| = a$, we have

$$\begin{aligned} & \sum_{j\in\mathcal{C}} \left(\sum_{i\in\mathcal{S}} \mathbf{X}_{i,j} \right)^2 \\ = & \sum_{j\in\mathcal{C}} \left\{ \sum_{i:(i,j)\in\mathcal{N}} \mathbf{X}_{i,j}^2 + 2 \cdot \sum_{\substack{i,i'\in\mathcal{R}:i<i' \\ \text{and } (i,j),(i',j)\in\mathcal{N}}} \mathbf{X}_{i,j}\mathbf{X}_{i',j} \right\} \\ = & \sum_{(i,j)\in\mathcal{N}} \mathbf{X}_{i,j}^2 + 2 \cdot \sum_{j\in\mathcal{C}} \sum_{i,i':i<i'} \mathcal{I}_{i,i'} \mathbf{X}_{i,j}\mathbf{X}_{i',j} \\ = & \sum_{(i,j)\in\mathcal{N}} \mathbf{X}_{i,j}^2 + 2 \cdot \sum_{i,i'\in\mathcal{R}:i<i'} \mathcal{I}_{i,i'} \\ = & \ a + 2 \cdot \sum_{i,i'\in\mathcal{R}:i<i'} \mathcal{I}_{i,i'} \end{aligned} \tag{33}$$

where the last equality holds because for any pair of switches (i, i'), $\mathcal{I}_{i,i'} = 1$ only when i and i' upload requests to the same controller. From (33), we know that the upper bound is reached when $\mathcal{I}_{i,i'} = 1$ for all $i, i' \in \mathcal{R}$, i.e., when all switches in $\mathcal{R}$ connected to the same switches. In such case, since there are $\frac{1}{2}a(a-1)$ pairs of different switches, then the upper bound of $\sum_{j\in\mathcal{C}} \left(\sum_{i\in\mathcal{S}} \mathbf{X}_{i,j}\right)^2$ is $a + a(a-1) = a^2$. Hence,

$$\begin{aligned} & \sum_{j\in\mathcal{C}} \left(\sum_{i\in\mathcal{S}} \mathbf{X}_{i,j} \right)^2 + \sum_{i\in\mathcal{S}} (1 - \sum_{j\in\mathcal{C}} \mathbf{X}_{i,j})^2 \\ \leq & \ a^2 + b \\ = & \ a^2 + |\mathcal{S}| - a \\ = & \ (a - \tfrac{1}{2})^2 + |\mathcal{S}| - \tfrac{1}{4} \end{aligned} \tag{34}$$

Now that a is a non-negative integer and $0 \leq a \leq |\mathcal{S}|$, then the upper bound in (34) reaches its maximum value $|\mathcal{S}|^2$ when $a = |\mathcal{S}|$. In other words, the upper bound reaches maximum when all switches in $\mathcal{S}$ upload requests to the same controller. As a result,

$$\begin{aligned} & E\left\{ \frac{1}{2} \sum_{j\in\mathcal{C}} \left[\left(\sum_{i\in\mathcal{S}} \mathbf{X}_{i,j} A_i(t) \right)^2 + (B_j(t))^2 \right] + \right. \\ & \left. \frac{1}{2} \sum_{i\in\mathcal{S}} \left[(\mathbf{Y}_i A_i(t))^2 + (U_i(t))^2 \right] \right\} \\ \leq & \ \frac{1}{2} \max_{i,j} (E(B_j^2(t)), E(U_i^2(t)), E(A_i^2(t))) \cdot \left(|\mathcal{C}| + |\mathcal{S}| + |\mathcal{S}|^2 \right) \\ = & \ \frac{d_{max}}{2} \left(|\mathcal{C}| + |\mathcal{S}| + |\mathcal{S}|^2 \right) = K \end{aligned} \tag{35}$$

We assume the whole control plane is capable of handling all requests from data plane in the mean sense. Therefore, for $j \in \mathcal{C}$, there exists $\epsilon_j^c > 0$ such that $E[B_j(t) - \sum_{i\in\mathcal{S}} \mathbf{X}_{i,j} A_i(t) \,|\, \mathbf{Q}^c(t)] = \epsilon_j^c$. Likewise, for $i \in \mathcal{S}$, there exists $\epsilon_i^s > 0$ such that $E[U_i(t) - \mathbf{Y}_i A_i(t) \,|\, \mathbf{Q}^c(t)] = \epsilon_i^s$. Following (35) and the definition in (14), after taking expectation on $\Delta_V(\mathbf{Q}(t))$, we have

$$\begin{aligned} & E\left\{ \Delta_V(\mathbf{Q}(t)) \right\} \\ \leq & \ K + \sum_{j\in\mathcal{C}} E\left\{ Q_j^c(t) \right\} \cdot E\left\{ E\left\{ \sum_{i\in\mathcal{S}} \mathbf{X}_{i,j}(t) A_i(t) - B_j(t) \,|\, \mathbf{Q}(t) \right\} \right\} \\ & + \sum_{i\in\mathcal{S}} E\left\{ Q_i^s(t) \right\} \cdot E\left\{ E\left\{ [1 - \sum_{j\in\mathcal{C}} \mathbf{X}_{i,j}(t)] A_i(t) - U_i(t) \,|\, \mathbf{Q}(t) \right\} \right\} \\ & + V \cdot E\left\{ E\{ f(t) + g(t) | \mathbf{Q}(t) \} \right\} \\ = & \ K + \sum_{j\in\mathcal{C}} E\left\{ Q_j^c(t) \right\} \cdot E\left\{ \sum_{i\in\mathcal{S}} \mathbf{X}_{i,j}(t) A_i(t) - B_j(t) \right\} \\ & + \sum_{i\in\mathcal{S}} E\left\{ Q_i^s(t) \right\} \cdot E\left\{ [1 - \sum_{j\in\mathcal{C}} \mathbf{X}_{i,j}(t)] A_i(t) - U_i(t) \right\} \\ & + V \cdot E\left\{ f(t) + g(t) \right\} \\ \leq & \ K - \epsilon^c \sum_{j\in\mathcal{C}} Q_j^c(t) - \epsilon^s \sum_{i\in\mathcal{S}} Q_i^s(t) + V \cdot (f^* + g^*) \end{aligned} \tag{36}$$

where $\epsilon^c = \min_{j\in\mathcal{C}}\{\epsilon_j^c\}$, $\epsilon^s = \min_{i\in\mathcal{S}}\{\epsilon_i^s\}$. Expanding the term $E\left\{\Delta_V(\mathbf{Q}(t))\right\}$, then for any time slot τ,

$$\begin{aligned} & E\left\{ L(\mathbf{Q}(\tau+1)) - L(\mathbf{Q}(\tau)) \right\} + V \cdot E\left\{ f(\tau) + g(\tau) \right\} \\ \leq & \ K - \epsilon^c \sum_{j\in\mathcal{C}} E\left\{ Q_j^c(\tau) \right\} - \epsilon^s \sum_{i\in\mathcal{S}} E\left\{ Q_i^s(\tau) \right\} + V(f^* + g^*) \end{aligned} \tag{37}$$

Next, summing over $\tau \in \{0, 1, 2, \ldots, t-1\}$ for some $t > 0$, we have

$$\begin{aligned} & E\left\{ L(\mathbf{Q}(t)) - L(\mathbf{Q}(0)) \right\} + V \cdot \sum_{\tau=0}^{t-1} E\left[f(\tau) + g(\tau) \right] \\ \leq & \ t \cdot K - \epsilon^c \sum_{\tau=0}^{t-1} \sum_{j\in\mathcal{C}} E\left\{ Q_j^c(\tau) \right\} - \epsilon^s \sum_{\tau=0}^{t-1} \sum_{i\in\mathcal{S}} E\left\{ Q_i^s(\tau) \right\} + \\ & \ t \cdot V \cdot (f^* + g^*) \end{aligned} \tag{38}$$

By re-arrangement of terms at both sides and ignoring some non-negative term such as $E\left\{L(\mathbf{Q}(t))\right\}$ and $E\left\{Q_j^c(t)\right\}$, with $\epsilon^c, \epsilon^s > 0$ and $V > 0$, we have

$$\begin{aligned} & V \cdot \sum_{\tau=0}^{t-1} E\left[f(\tau) + g(\tau) \right] \\ \leq & \ t \cdot V \cdot (f^* + g^*) + t \cdot K + E\left\{ L(\mathbf{Q}(0)) \right\} \end{aligned} \tag{39}$$

$$\begin{aligned} & \epsilon^c \cdot \sum_{\tau=0}^{t-1} \sum_{j\in\mathcal{C}} E\left\{ Q_j^c(\tau) \right\} \\ \leq & \ t \cdot V \cdot (f^* + g^*) + t \cdot K + E\left\{ L(\mathbf{Q}(0)) \right\} \end{aligned} \tag{40}$$

$$\begin{aligned} & \epsilon^s \cdot \sum_{\tau=0}^{t-1} \sum_{i\in\mathcal{S}} E\left\{ Q_i^s(\tau) \right\} \\ \leq & \ t \cdot V \cdot (f^* + g^*) + t \cdot K + E\left\{ L(\mathbf{Q}(0)) \right\} \end{aligned} \tag{41}$$

Then by dividing both sides of (39) by $V \cdot t$, (40) by $\epsilon^c \cdot t$, and (41) by $\epsilon^s \cdot t$, we have

$$\begin{aligned} & \frac{1}{t} \cdot \sum_{\tau=0}^{t-1} E\left[f(\tau) \right] \\ \leq & \ (f^* + g^*) + \frac{K}{V} + \frac{E\left\{ L(\mathbf{Q}(0)) \right\}}{V \cdot t} \end{aligned} \tag{42}$$

$$\begin{aligned}&\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{j\in\mathcal{C}}E\left\{Q_j^c(\tau)\right\}\\ \leq\quad&\frac{V\cdot(f^*+g^*)+K}{\epsilon^c}+\frac{E\left\{L(\mathbf{Q}(0))\right\}}{\epsilon^c\cdot t}\end{aligned}\tag{43}$$

$$\begin{aligned}&\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{i\in\mathcal{S}}E\left\{Q_i^s(\tau)\right\}\\ \leq\quad&\frac{V\cdot(f^*+g^*)+K}{\epsilon^s}+\frac{E\left\{L(\mathbf{Q}(0))\right\}}{\epsilon^s\cdot t}\end{aligned}\tag{44}$$

At last, taking the limit as $t\to\infty$ for both equations, we have the desired results:

$$\lim_{t\to\infty}\frac{1}{t}\cdot\sum_{\tau=0}^{t-1}E\left[f(\tau)+g(\tau)\right]\leq f^*+g^*+\frac{K}{V}\tag{45}$$

$$\lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{j\in\mathcal{C}}E\left\{Q_j^c(\tau)\right\}\leq\frac{V\cdot(f^*+g^*)+K}{\epsilon^c}\tag{46}$$

$$\lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{i\in\mathcal{S}}E\left\{Q_i^s(\tau)\right\}\leq\frac{V\cdot(f^*+g^*)+K}{\epsilon^s}\tag{47}$$

By setting $\epsilon=\min\{\epsilon^c,\epsilon^s\}$, the following desired result holds

$$\begin{aligned}\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\left[\sum_{j\in\mathcal{C}}E\left\{Q_j^c(\tau)\right\}+\sum_{i\in\mathcal{S}}E\left\{Q_i^s(\tau)\right\}\right]\quad\leq\\ \frac{K+V\cdot(f^*+g^*)}{\epsilon}\end{aligned}\tag{48}$$

□

Acknowledgments

This work was supported in part by NSFC grant No.6130 2114, and Hong Kong RGC CRF-C7036-15G.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[2] 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 *Proceedings of OSDI*, 2010.

[3] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, 2010.

[4] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically centralized?: state distribution trade-offs in software defined networks," in *Proceedings of ACM HotSDN*, 2012.

[5] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, "Towards an elastic distributed sdn controller," in *Proceedings of ACM HotSDN*, 2013.

[6] A. Krishnamurthy, S. P. Chandrabose, and A. Gember-Jacobson, "Pratyaastha: An efficient elastic distributed sdn control plane," in *Proceedings of ACM HotSDN*, 2014.

[7] T. Wang, F. Liu, J. Guo, and H. Xu, "Dynamic sdn controller assignment in data center networks: Stable matching with transfers," in *Proceedings of IEEE INFOCOM*, 2016.

[8] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: scaling flow management for high-performance networks," in *Proceedings of ACM SIGCOMM*, 2011.

[9] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: a framework for efficient and scalable offloading of control applications," in *Proceedings of ACM HotSDN*, 2012.

[10] K. Zheng, L. Wang, B. Yang, Y. Sun, Y. Zhang, and S. Uhlig, "Lazyctrl: Scalable network control for cloud data centers," in *Proceedings of IEEE 35th International Conference on Distributed Computing Systems (ICDCS)*, 2015.

[11] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[12] X. Huang, S. Bian, Z. Shao, and H. Xu, "Dynamic switch-controller association and control devolution for sdn systems," 2016, https://arxiv.org/abs/1702.03065.

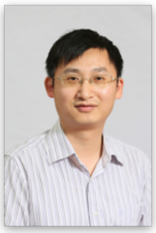
[13] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proceedings of ACM SIGCOMM*, 2008.

[14] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of ACM IMC*, 2010.

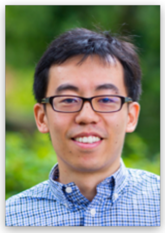
[15] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Proceedings of ACM Hot-ICE*, 2012.

Xi Huang received the B.Eng. degree from Software Institute, Nanjing University, China, in 2014. Since 2014, he has been with the School of Information Science and Technology at ShanghaiTech University, where he is currently a PhD candidate. He is now visiting the Department of Electrical Engineering and Computer Sciences at UC Berkeley, during February to July in 2017. His current research interests include: datacenter networking, software-defined networking, and coflow scheduling.

Simeng Bian Simeng Bian received the B.Eng. degree from the School of Software and Microelectronics, Northwestern Polytechnical University, China, in 2016. Since 2016, she has been with the School of Information Science and Technology at ShanghaiTech University, where she is currently a Master Degree candidate. Her current research interests include: datacenter networking, resource allocation, and job scheduling.



Ziyu Shao Biography text here.



Hong Xu Hong Xu, Henry, is an assistant professor in Department of Computer Science, City University of Hong Kong. His research area is computer networking, particularly NFV and data center networks. He received the B.Eng. degree from The Chinese University of Hong Kong in 2007, and the M.A.Sc. and Ph.D. degrees from University of Toronto in 2013. He was the recipient of an Early Career Scheme Grant from the Research Grants Council of the Hong Kong SAR, 2014. He also received the best paper awards from ACM TURC 2017 (Sigcomm China), IEEE ICNP 2015, and ACM CoNEXT Student Workshop 2014. He is a member of ACM and IEEE.