# Hierarchical Edge-Cloud SDN Controller System With Optimal Adaptive Resource Allocation for Load-Balancing

Frank Po-Chen Lin, *Student Member, IEEE*, and Zsehong Tsai, *Member, IEEE*

***Abstract*—Although the openflow-based software defined network (SDN) architecture can ease the workload of network control and management and separate it from the switch/routing operations, a computation-resource limited controller can still be congested by heavy flows and then experiences serious delay. To enhance network scalability and reduce computation delay on SDN networks under Quality of Service (QoS) requirements, a hierarchical edge-cloud SDN (HECSDN) controller system design is proposed with three features. First, by sharing computational resources in the edge and the cloud, the system architecture provides a flexible mechanism for devices to allocate their computational tasks according to traffic loads. The second feature is to design a queueing model of the proposed architecture. The model description of the networking architecture enables the network designers to quickly estimate the performance of design without considerable time and cost in experimental setups. Third, derived from the queueing model, an efficient load-balancing algorithm satisfying QoS requirements or fairness allocation for different applications in the HECSDN architecture is proposed. This newly-developed multi-tier controller system has been proved to be effective even when working on a large-scale SDN, without sacrificing the overall performance. Moreover, this system stays highly stable in transient periods even under highly fluctuating flow arrivals.**

***Index Terms*—Adaptive optimization, cloud computing, hierarchical software defined network (SDN), load-balancing, network modeling, queuing theory.**

## I. Introduction

THANKS to revolutionary wired and wireless communications, new network applications, such as smart/digital programs, intelligent sensors, and AI-based Internet-of-Things applications have brought much more convenience to the society than ever before. However, many applications based on the conventional network architecture could not afford enough flexibility in their network management and operations. In many cases, the architecture cannot efficiently deal with huge network sizes or overload traffic conditions, especially if they need to provide sufficient Quality of Service (QoS) for different delay-sensitive applications, such as voice over Internet protocol (VoIP), video streaming, and online games.

Manuscript received September 6, 2018; revised July 11, 2018 and December 31, 2018; accepted January 13, 2019. Date of publication February 12, 2019; date of current version March 2, 2020. *(Corresponding author: Frank Po-Chen Lin.)*

The authors are with the Graduate Institute of Communication Engineering, National Taiwan University, Taipei 10617, Taiwan (e-mail: frank555076@gmail.com; ztsai@ntu.edu.tw).

Digital Object Identifier 10.1109/JSYST.2019.2894689

Openflow-based software defined network (SDN) [1], [2] has seen a very promising approach in the future landscape of the Internet. It can help facilitate the deployment and operation of new services [3]. The SDN can also improve network efficiency through high level novel abstractions so as to provide dynamic programmability for real-time centralized intelligence [4].

The SDN mainly consists of controllers and switches. When a new network flow arrives, the switch will match the header of the packet with flow entries and send the unmatched packet to the controller with a packet-in message [5]. The controller then computes the routing path for each data flow and sets the flow table to all corresponding switches to meet the need of the global network. This enables an SDN to simplify its network configurations and resource management.

However, when the amount of computation requests grows large along with the size of the network, insufficient processing capacity of a single controller may make itself a bottleneck of incoming traffic flows [6]. The limited processing capacity of the controllers may result in unbearable delay under high traffic demands. The performance and scalability of the controller architecture thus become a critical problem [7], [8].

Various methodology and performance models have aimed to enhance the scalability of SDN networks. Here, we examine two categories of these previous works related to this paper: Delay-reduction in SDN and scalable cloud computing.

### A. Delay Reduction in SDN

Many existing SDN studies have tried different approaches in improving end-to-end QoS, e.g., Joksch [9] formulated the problem as a constrained shortest path problem and solved it by using dynamic programming. Jia and Varaiya [10] discovered the best routing path through the Bellman–Ford algorithm.

In turn, other research works focused on the load-balancing design of multi-controllers as a way to deal with the scalability of SDN controllers. Load-balancing of multiple controllers can be mainly divided into two categories: centralized and distributed. For typical centralized load-balancing, Zhao *et al*. [11] proposed an implementation on communication protocol in the control plane to realize their hierarchical design of multiple controllers. Ma *et al*. [12] presented a centralized load-balancing mechanism to eliminate the bottleneck of the centralized control in a network. For distributed load-balancing approaches, Zhou *et al*. [13] proposed a method that allow every controller to make load-balancing decisions locally.

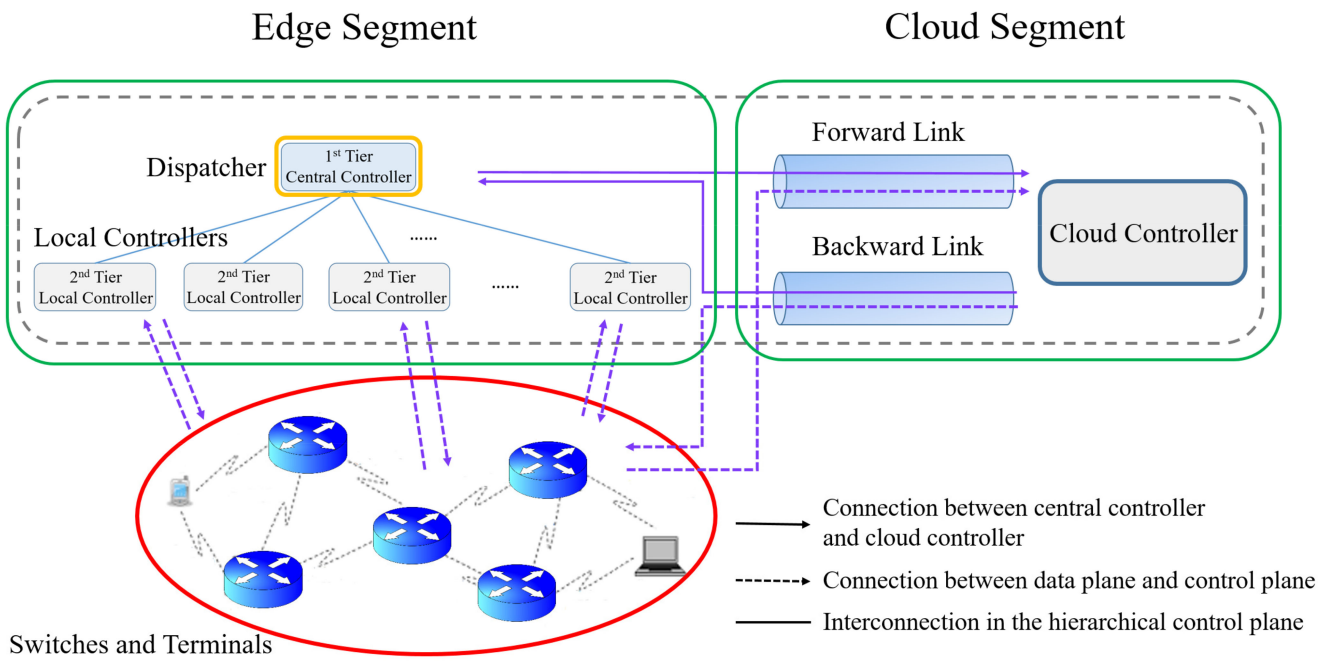


Fig. 1. New hierarchical edge-cloud SDN controller system.

### B. Scalable Cloud Computing

Cloud computing can serve as a computing architecture as well as network design that can provide easy access to computing resources [14]. Several developments of applications have been made possible through the aid of cloud computing. A good case in point is mobile cloud computing [15].

Many works were initiated in this regard and extensive surveys on cloud computing are available in the literature. For instance, a scalable algorithm is implemented to choose a set of near clouds that minimizes offload duration and mobile power consumption in a two-tiered cloud architecture [16]. Ramezani *et al.* [17] designed a multiobjective cloud load-balancing technique and proposed a particle swarm optimization-genetic algorithm that can provide good QoS for different mobile applications. Moreover, analysis and techniques on parallel programming and distributed computing have also been developed to increase processing efficiency [18], [19].

In addition to the two categories of the abovementioned works, many research works related to network performances also focused on optimization modeling techniques for seeking efficient solutions [20], [21]. Nevertheless, despite the amount of abovementioned works, the integration of cloud computing and SDN are seldom considered. Most of the solutions for enhancing the overall performance of SDN systems are not good enough to satisfy the needs for different QoS. Only a few works focused on the requirements of various applications [22]. However, in nation-wide networks, controllers are required to process millions of flows per second without compromising the QoS of network flows [23]. Therefore, the objective of this paper is to help enhance network scalability and reduce computation delay in SDN while providing guarantee for QoS and fair resource allocation. For these reasons, a hierarchical edge-cloud SDN (HECSDN) controller system is proposed in this paper.

The HECSDN controller system has the following three features. First, by sharing computational resources in the edge and the cloud, the system architecture provides a flexible mechanism for devices to allocate their computational tasks according to traffic loads. Second, the queueing model of the investigated SDN networking architecture enables network designers to quickly estimate the performance of their designs without spending considerable time for expensive experimental setups [24]. Third, an efficient load-balancing algorithm is designed along with the proposed queuing model to provide QoS guarantee and fairness allocation for different applications.

The proposed HECSDN model can work as a basis that provides optimized formulation and adaptively minimizes the system delay or maintains fair resource allocation according to the status of traffic flows. Through the optimization of delay waiting time in the queuing model, the best load-balancing solution will overcome the delay bottleneck and maintain high-level QoS for various network applications in different traffic patterns in large-scale networks.

Later in this paper, with simulation results of several example networks, we also validate the advantages of the proposed HECSDN system.

## II. Hierarchical Edge-Cloud SDN Controller System

### A. System Architecture

The HECSDN controller system consists of two segments, the edge and the cloud (see Fig. 1). The *edge segment* consists of two types of controllers, forming a hierarchical control plane, which includes the first-tier central controller and the second-tier local

controllers. The *cloud segment* consists of a cloud controller and the communication links. A cloud controller, which is a supporting computation resource for the purpose of load-balancing in signal-packet processing, borrows the idea of the cloud computing infrastructure and offers strong computation power in a data center. As for the communication links, they can be further categorized into forward links and backward links. The forward links are in charge of flow signals and control messages sending from the switches and the central controller to the cloud controller, while the backward links take care of flow signals and controller messages sending back to the switches and the central controller from the cloud controller.

In operations, as signal-packets from new flows are sent from the SDN switches to the control plane, the central controller shall first make load-balancing decisions for signal-packets between the local controllers and the cloud controller, acting as a dispatcher. The local controllers and the cloud controller then deal with the processing requests of signal-packets according to the dispatching decisions.

*1) Switches and Terminals:* The switches and the terminals are shown in the lower part of Fig. 1. Each switch can support various data flows with total of $k$ types in the network. Flow data-packets originating from the same source and ending in the same destination are considered as the same data flow. In this paper, transmission control protocol (TCP) flows are considered to play the major role among all data flows, and the extracted header information in first TCP packet in a data flow is the flow signal-packet.

*2) Central Controller:* With the global view of traffic status in the control plane, the central controller, which includes a dispatcher mechanism, is responsible for managing the load distribution of flow signals among the local controllers and the cloud controller. The modes of the dispatcher in the central controller can be divided into two: the delay-objective mode and the fairness-objective mode. The delay-objective mode is activated when the system is capable of handling all input traffic loads of flow signal-packets. The dispatcher in this mode aims to minimize total mean delay while satisfying QoS for all network applications. The fairness-objective mode, in addition, is activated when the total signal load is too heavy for all the local controllers and the cloud controller to satisfy QoS requirements. In this mode, instead of focusing on the performance of the overall mean delay, the objective of the dispatcher is to provide minimax fairness allocation on delay for all network applications to assure resource allocation fairness.

*3) Local Controller:* The local controllers are responsible for computing the routing path for these incoming flow signal-packets and setting flow tables for all the corresponding switches, from which the signal-packets arrived. Meanwhile, to ensure that the central controller maintains the real-time traffic status of all local controllers in the control plane, the local controllers must update their own traffic conditions for the central controller after path computation. These traffic-updates can be classified into two fashions, reactive and proactive. In reactive traffic updates, a controller informs its traffic status to the central controller through a control message when being overloaded by flow signal-packets. In proactive traffic-updates, all local controllers send their own traffic conditions to the central controller periodically so as to ensure that every alteration in the data flow traffic is well-considered all times.

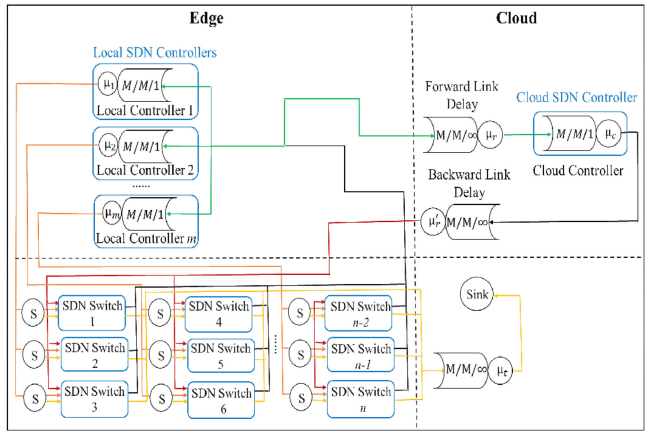


Fig. 2. System architecture of the edge-cloud queueing network.

*4) Cloud Controller:* As a supporting computation resource for signal-packet processing, the cloud controller provides strong computation capacity on the basis of cloud computing. With the aid of the cloud controller, the system is capable of handling larger traffic amount and significantly decreasing the flow level of waiting time in control plane processing. For small scale systems, the cloud controller can be just a single virtual server operating on one virtual machine (VM). When the system scalability is a concern, it can be extended to operate with a large number of VMs. However, for simplicity, in the model of this paper the cloud controller is assumed to operate on a single VM in one remote data center.

### B. System Operation

Upon the arrival of a new data flow, the SDN switch should extract its matching fields from all protocol headers to compute the flow key [3]. The switch then looks up its flow tables to match an entry with the key. If the match fails, the switch fires off a flow signal-packet and this signal-packet is forwarded to the central controller for its dispatching decision. The signal packets are then processed by its connected controller, which can be either one of the local controllers or the cloud controller. After receiving new processing requests for a period of time, the controllers must update their own traffic status to the central controller. If any alteration in data flow traffic patterns occurs, or when any local and the cloud controller is overloaded, or when the QoS requirements for current signal-packet arrivals is no longer satisfied, the central controller then activates the load-balancing mechanism and reassigns the workload distribution for the signal-packets.

### C. System Queueing Model

In order to design and analyze the load-balancing system, this research models the SDN network as a queuing network (see Fig. 2). The network model of the system is divided into two

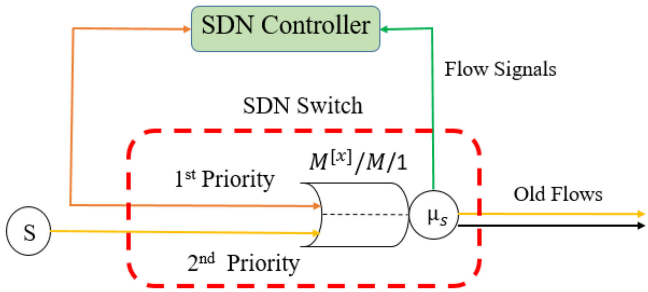


Fig. 3. Queueing model of an SDN switch with service rate $\mu_s$.

TABLE I
NOTATIONS OF THE SDN QUEUEING MODEL

| Notations | Definitions |
|---|---|
| $n$ | *total amount of switches* |
| $m$ | *total amount of controllers* |
| $\lambda_j$ | *batch arrival rate of data-packets in switch j (packets/sec)* |
| $S_j$ | *average number of data-packets in a batch arrival of switch j* |
| $p_j$ | *the probability for switch j that a new data-packet will form a new flow-signal packet to be sent to the local or cloud controller* |
| $\phi_{ij}$ | *the probability of a signal-packet sent from switch j to controller i* |
| $\phi_{cj}$ | *the probability of a signal-packet sent from switch j to cloud.* |
| $\mu_i$ | *service rate of controller i (packets/sec)* |
| $\mu_r$ | *service rate for the edge-to-cloud link queue(packets/sec)* |
| $\mu'_r$ | *service rate for the cloud-to-edge link queue (packets/sec)* |
| $\mu_c$ | *service rate of the cloud controller (packets/sec)* |
| $\mu_s$ | *service rate of the SDN switch for table-lookup proc. (packets/sec)* |

major segments, edge, and cloud. The network model in the edge consists of the SDN switches and local controllers. Since the phenomenon of packet groups is a natural artifact of the TCP so that the SDN switch is modeled as a component composed of a $M^{[x]}/M/1$ non-preemptive priority queue with service rate $\mu_s$ as shown in Fig. 3.

As data-packet arrivals enter from the external traffic sources into the switch, which performs table look-up operation, the switch classifies the data-packet arrivals into two types, old and new. Old arrivals correspond to data-packet flows coming from the same source to the same destination as historical arriving flows. For new arrivals, the signal-packets will be send to the controllers for path computation over switch $j$ ( $j \in \mathrm{S}_1$ where $\mathrm{S}_1 = \{1,2,\ldots,\mathrm{m}\}$) with rate $p_j\lambda_j S_j$.

A summary of mathematical notations is available in Table I. After the path computation of their packets, the signal-packets from the same flow are transferred back to the $M^{[x]}/M/1$ queue with first priority (first Priority). Flows from the data source are considered as the arrivals with the second priority (second Priority).

Local controller $i$ ($i \in S_2$ where $S_2 = \{1,2,\ldots,\mathrm{n}\}$) is modeled as an $M/M/1$ queue with service rate $\mu_i$, which is determined by the computational ability of controller $i$. Arrivals of every controller are considered the combination of flow signals from every switch in the network with different proportions.

Every switch sends the different proportion of the new network flows signals into every controller for path computation. The arrival rate of local controller $i$ can be formulated as $\sum_j \phi_{ij} p_j \lambda_j S_j$. In addition, the cloud consists of the cloud controller and the communication links. The cloud controller is modeled as an $M/M/1$ queue with service rate $\mu_c$. The arrival rate of the cloud controller can be written as $\sum_j \phi_{cj} p_j \lambda_j S_j$.

For arrivals of signal-packets sent to the cloud controller, the total waiting time of a packet is the sum of the round trip communication delay in the links and the queuing processing delay in the cloud controller. For arrivals sent to the local controllers, the communication delay inside the edge is negligible for that the distance between the control plane and the data plane is much shorter than the edge-to-cloud distance. The edge-to-cloud distance represents the physical length of optic fiber between $d$ the edge and cloud controller. For the communication links in the queueing system, two queues model the delay of link communication. The delay between edge and cloud is modeled as an $M/M/\infty$ queue with service rate $\mu_d$. The value of $\mu_d$ is determined by $d$ with $\mu_d = c/d$, where the packet traveling speed $c$ is considered as the speed of light without loss of generality. The mathematical notations of the queueing model are given in Table I.

## III. EDGE-CLOUD SDN LOAD-BALANCING ALGORITHM

### A. Edge-Cloud SDN Load-Balancing Algorithm

To enhance the sustainability and efficiency on the operation of the network for QoS requirements and resource allocation fairness, the ECSDN load-balancing algorithm is proposed. For simplicity, the term *original delay bound* refers to the delay limitations of different applications; *adjusted delay bound* to delay bounds added with an additional guard interval; *enlarged delay bound* to the newly assigned delay bound under the fairness objective mode.

The central controller monitors the status of all the local controllers and the cloud controller. If one of the three conditions occurs in any of the controllers, i.e., input arrival alteration, controller overloading, or QoS of certain application becoming unsatisfied, the controller send a control message to the central controller requesting for load assignment modifications. Simultaneously, the other controllers are requested to send their current traffic status to the central controller.

After receiving the traffic status from all controllers (local and cloud), the central controller, acting as a dispatcher, determines whether the QoS requirements of all signal-packet arrivals can be satisfied. If QoS can be satisfied, the dispatcher operates in the delay-objective mode. Optimization in this mode is to minimize a combined objective consisting of the total mean delay and the allowed additional guard interval for QoS constraints. However, if the dispatcher finds out that the QoS cannot be satisfied under the current traffic condition, the dispatcher moves on to operate in the fairness-objective mode. Optimization in this mode is performed both to satisfy the original delay bound $D_k$ of all resource-affordable applications and to minimize the enlarged delay bound $\delta$ for all other resource-unaffordable applications. Mathematical formulation of the optimization problems is derived in the following sections. Additional notations employed in the optimization formulations are given in Table II.

TABLE II
ADDITIONAL NOTATIONS FOR THE OPTIMIZATION FORMULATIONS

| Notations | Definitions |
|---|---|
| $\gamma$ | *the amount of guard interval for protecting the delay bound* |
| $\beta$ | *weighting factor* |
| $D_k$ | *delay bound* |
| $D_k'$ | *adjusted delay bound $D_k$ - $\gamma$* |
| $D_\Omega$ | *original delay bound for type $\Omega$ application* |
| $\alpha_{ik}$ | *the probability that the waiting delay $w_i$ satisfies the adjusted delay bound $D_k'$ for type k application in local controller i* |
| $\alpha_{ck}$ | *the probability that the waiting delay $w_c$ satisfies the adjusted delay bound $D_k'$ for type k application in cloud controller* |
| $\eta_k$ | *required QoS on packet-overdue ratio for type k application* |
| $\alpha_{i\Omega}$ | *the probability that the waiting delay $w_i$ satisfies the adjusted delay bound $D_k'$ for type $\Omega$ application in local controller i* |
| $\alpha_{i\omega}$ | *the probability that the waiting delay $w_i$ satisfies the adjusted delay bound $D_k'$ for type $\omega$ application in local controller i* |
| $\alpha_{c\Omega}$ | *the probability that the waiting delay $w_c$ satisfies the adjusted delay bound $D_k'$ for type $\Omega$ application in cloud controller* |
| $\alpha_{c\omega}$ | *the probability that the waiting delay $w_c$ satisfies the adjusted delay bound $D_k'$ for type $\omega$ application in cloud controller* |

### B. Delay-Objective Optimization

When computational resource suffices, the dispatcher operates in a delay-objective mode. In this mode, the dispatcher minimizes the average delay and protects the delay bounds by adding guard intervals for all types of applications. This is achieved by allocating appropriate portion of flow signal load message to the local controllers and the cloud controller. The guard intervals adjust the delay bounds of all applications by decreasing the amount of time from the original delay bound.

As the guard interval increases, all applications receive a better performance with reduced packet delays and overdue ratio. To fulfill the minimax fairness criteria on the guard interval, the delay bounds of all applications are protected with the same amount $\gamma$. The objective function be formulated as

$$\text{minimize } f = \beta\left[\sum_i \left(\mu_i - \sum_j \phi_{ij} p_j \lambda_j S_j\right)^{-1} + \left(\mu_c - \sum_j \phi_{cj} p_j \lambda_j S_j\right)^{-1}\right] - (1-\beta)\gamma \tag{1}$$

where $\sum_i (\mu_i - \sum_j \phi_{ij} p_j \lambda_j S_j)^{-1} + (\mu_c - \sum_j \phi_{cj} p_j \lambda_j S_j)^{-1}$ corresponds to the total mean signal-packet waiting time in all controllers, $\gamma$ to the amount of guard interval for protecting the delay bound, $\beta$ to the weighting factor when considering the different importance between a total of the mean waiting time and the size of the guard interval. The objective function is to minimize the sum of total mean delay and negative value of guard interval $-\gamma$ to achieve a better delay to QoS for all applications. The constraints for the optimization problem are derived in the following sections.

*1) Optimization Constraints:* For every local controller *i*, the QoS formulation acts to assure the probability that the waiting delay $w_i$ satisfies the adjusted delay bound $D_k$ for type $k$ ($\forall k \in S_3$ where $S_3 = \{1,2,\ldots,K\}$) application is larger or equal to $\alpha_{ik}$, which can be described as

$$\Pr\{w_i \le D_k' D\} \ge \alpha_{ik} \ \forall i \in \mathrm{S}_1, k \in \mathrm{S}_3 \tag{2}$$

where $D_k{}' = D_k - g$.

Similarly, for the cloud controller, the QoS formulation works to assure the probability that the waiting delay $w_c$ satisfies the adjusted delay bound $D_k'$ for type $k$ application is larger or equal to $\alpha_{ck}$, which can be described as

$$\Pr\{w_c \le D_k'\} \ge \alpha_{ck} \ \forall k \in S_3. \tag{3}$$

To ensure that the round-trip delays of a type $k$ flow signal arrival across all controllers is below $D_k'$ with probability larger or equal to $\eta_k$, where $\eta_k$ is a predetermined value for type $k$ flow signal, the optimization process will then search and find appropriate values of $\alpha_{ik}$ and $\alpha_{ck}$ so that their combined QoS satisfy the criteria in

$$\left(\sum_i \sum_{j \in \text{type k}} \alpha_{jk}\phi_{ij} p_j \lambda_j S_j + \sum_{j \in \text{type k}} \alpha_{ck}\phi_{cj} p_j \lambda_j S_j\right) \times \left(\sum_{j \in \text{type } k} p_j \lambda_j S_j\right)^{-1} \ge \eta_k \cdot 100\% \ \forall i \in \mathrm{S}_1, j \in \mathrm{S}_2, k \in \mathrm{S}_3. \tag{4}$$

In other words, the restriction in (4) means that the weighted average of delay constraint conforming probability, $\alpha_{ik}$ and $\alpha_{ck}$, across all controllers in the SDN network for type *k* flow signal is assured to be larger or equal to $\eta_k$. To obtain individual nodal parameters $\alpha_{ik}$ and $\alpha_{ck}$ is one part of the optimization problem.

*2) Constraint Formulations for Edge Local Controllers:* To define the constraints for local controllers that corresponds to (2), let $X_{i,\text{local}}$ be the random variable of packet waiting time in local controller $i$. Since we use a single server queue with exponential service time to model every SDN switch, the waiting time of a packet follows exponential distribution, i.e.,

$$X_{i,\text{local}} \sim \text{exponential}\left(\mu_i - \sum_j \phi_{ij} p_j \lambda_j S_j\right) \ \forall i \in \mathrm{S}_1, j \in \mathrm{S}_2 \tag{5}$$

and its cumulative density function (CDF) is denoted as $F_{\text{local}}()$. The constraints for local controllers can then be written as

$$F_{\text{local}}(D_k') \ge \alpha_{ik} \ \forall i \in \mathrm{S}_1, k \in \mathrm{S}_3 \tag{6}$$

where

$$F_{\text{local}}(D_k') = \Pr\{w_i \le D_k'\}. \tag{7}$$

By Shortle *et al*. [25], it can be derived that

$$1 - \exp\left[-\left(\mu_i - \sum_j \phi_{ij} p_j \lambda_j S_j\right) D_k'\right] \ge \alpha_{ik} \ \forall i \in S_1, j \in S_2, k \in S_3 \tag{8}$$

and it is equivalent to

$$\sum_j \phi_{ij} p_j \lambda_j S_j - \mu_i - \ln(1-\alpha_{ik}) D_k'^{-1} \leq 0 \ \forall i \in S_1, j \in S_2, k \in S_3 \quad (9)$$

which implies that the delay constraint conforming probability in controller *i* is assured to be larger or equal to the probability $\alpha_{ik}$.

Meanwhile, to stabilize the network system in the edge, the utilization of a local controller (assuming controller $i$) needs to satisfy

$$U_{i,\text{local}} = \sum_j \phi_{ij} p_j \lambda_j S_j \mu_i^{-1} \leq 1 \ \forall i \in S_1, j \in S_2. \quad (10)$$

*3) Constraint Formulations for Cloud Controller:* Similarly, to derive the required constraints for the cloud controller that satisfies (3), one should set $X_c$ to be the random variable of delay waiting time for packets sent to the cloud controller. As shown in Fig. 2, $X_c$ is the sum of the edge-cloud round trip delay components, i.e., $X_{c,\text{forward}}$ , $X_{c,\text{backward}}$ the link delay over the forward and the backward links, and the packet waiting delay $X_{c,\text{cloud}}$ in the cloud controller. In this paper, the forward link from a switch to the cloud and the backward link for the cloud to a switch are both modeled as an infinite server queue with service rate $\mu_d$.

Similar to the edge local controller, the distribution of the packet waiting time $(X_{c,\text{cloud}})$ in the cloud controller follows an exponential distribution with rate $\mu_c - \sum_j \phi_{cj} p_j \lambda_j S_j$ as shown in (5). Now, the total waiting time $X_c$ can be then be derived via $X_c = X_{2-\text{hops}} + X_{c,\text{cloud}}$, where $X_{2-\text{hop}}$ is the sum of $X_{c,\text{forward}}$ and $X_{c,\text{backward}}$.

Finally, using the CDF of $X_c$, the constraint that corresponds to (3) can now be derived as

$$F_c\left(D_k'; \mu_d, \mu_c - \sum_j \phi_{cj} p_j \lambda_j S_j\right) \geq \alpha_{ck} \ \forall j \in \mathrm{S}_2, k \in \mathrm{S}_3 \quad (11)$$

where

$$F_c\left(D_k'; \mu_d, \mu_c - \sum_j \phi_{cj} p_j \lambda_j S_j\right) = \Pr\{w_c \leq D_k'\} \quad (12)$$

which represents that the delay constraint conforming probability in the cloud controller is guaranteed to be larger or equal to the probability $\alpha_{ck}$.

Similarly, to stabilize the network system in the cloud, using the cloud controller is also controlled within 1 with constraint

$$U_{\text{cloud}} = \sum_j \phi_{cj} p_j \lambda_j S_j \mu_c^{-1} \leq 1 \ \forall i \in S_1, j \in S_2. \quad (13)$$

*4) General Constraint Formulations:* For every designated probabilities $\phi_{ij}$, $\alpha_{ik}$, and $\alpha_{ck}$, the value should be restricted between 0 and 1. That is

$$0 \leq \phi_{ij} \leq 1 \ \forall i \in S_1, j \in S_2 \quad (14)$$

$$0 \leq \alpha_{ik} \leq 1 \ \forall i \in S_1, k \in S_3 \quad (15)$$

and

$$0 \leq \alpha_{ck} \leq 1 \ \forall k \in \mathrm{S}_3. \quad (16)$$

Since the sum of the assigned probabilities from switch $j$ to controller $i$ is one, we have

$$\phi_{cj} + \sum_i \phi_{ij} = 1 \ \forall i \in \mathrm{S}_1, j \in \mathrm{S}_2. \quad (17)$$

The guard interval $\gamma$ should be limited to the range of 0 to the minimum delay bound of all applications to ensure that newly assigned delay bounds are all positive values, i.e.,

$$0 \leq \gamma \leq \min(D_k) \ \forall k \in \mathrm{S}_3. \quad (18)$$

*5) Optimization Formulation:* In general, when there are $n$ switches and $m$ controllers, the formulation of the convex optimization problem under adequate resources is depicted as

$$\underset{\phi_{ij}, \alpha_{ik}, \alpha_{ck}, \gamma}{\text{minimize}} \ f$$

$$f = \beta\left[\sum_i \left(\mu_i - \sum_j \phi_{ij} p_j \lambda_j S_j\right)^{-1} + \left(\mu_c - \sum_j \phi_{cj} p_j \lambda_j S_j\right)^{-1}\right] - (1-\beta)\gamma$$

s.t.

$$\left(\sum_i \sum_{j \in \text{type } k} \alpha_{jk} \phi_{ij} p_j \lambda_j S_j + \sum_{j \in \text{type } k} \alpha_{ck} \phi_{cj} p_j \lambda_j S_j\right) \times \left(\sum_{j \in type k} p_j \lambda_j S_j\right)^{-1} \geq \eta_k \cdot 100\% \ \forall i \in S_1, j \in S_2, k \in S_3$$

$$\sum_j \phi_{ij} p_j \lambda_j S_j - \mu_i - \ln(1-\alpha_{ik}) D_k'^{-1} \leq 0 \ \forall i \in S_1, j \in S_2, k \in S_3$$

$$F_c\left(D_k'; \mu_d, \mu_c - \sum_j \phi_{cj} p_j \lambda_j S_j\right) \geq \alpha_{ck} \ \forall j \in S_2, k \in S_3$$

$$\sum_j \phi_{ij} p_j \lambda_j S_j \mu_i^{-1} \leq 1 \ \forall i \in S_1, j \in S_2$$

$$\sum_j \phi_{cj} p_j \lambda_j S_j \mu_c^{-1} \leq 1 \ \ \forall j \in S_2$$

$$\phi_{cj} + \sum_i \phi_{ij} = 1 \forall i \in \mathrm{S}_1, j \in \mathrm{S}_2$$

$$0 \leq \phi_{ij} \leq 1 \forall i \in \mathrm{S}_1, j \in \mathrm{S}_2$$

$$0 \leq \alpha_{ik} \leq 1 \ \forall i \in \mathrm{S}_1, k \in \mathrm{S}_3$$

$$0 \leq \alpha_{ck} \leq 1 \ \forall k \in \mathrm{S}_3$$

$$0 \leq \gamma \leq \min(D_k) \ \forall k \in \mathrm{S}_3 \quad (19)$$

where $\phi_{ij}$, $\alpha_{ik}$, $\alpha_{ck}$, and $\gamma$ are the decision variables of the load-balancing controller system. In the delay-objective mode,

given a pre-determined value $\eta_k$ for type $k$ flow signal, feasible solutions may not exist when the adjusted delay bound $D'_k$ for any type $k$ surpasses its original delay bound $D_k$ due to the growth of the total input flow signal $\sum_j \lambda_j$. When no feasible solution in the delay-objective mode can be found, the system changes to fairness-objective mode to release the original delay bound into the enlarged delay bound $\delta$ described in Section III-C. When this occurs, the optimization tool (such as MATLAB) used can easily provide model designer necessary feedback.

### C) Fairness-Objective Optimization

Under insufficient resource conditions, the optimization mode changes to fairness-objective, which implements fair resource allocation to guarantee that every application is provided with a fair amount of resources. The dispatcher minimizes the enlarged delay bound $\delta$ for the delay-unsatisfied applications and assigns the original delay bound for the other delay-satisfied applications. Compared with the delay-objective mode, the fairness-objective mode does not see the delay performance as a main concern. All resources in this mode are distributed based on the minimax delay criteria to make sure that every network application is under a relatively fair delay bound.

In this mode, the dispatcher allows the delay bound of all resource-affordable applications to be assigned as the initially requested (the original delay bound), and minimizes the enlarged delay bound of other resource-unaffordable applications to $\delta$, where $\delta$ is larger than the initially requested of the resource unaffordable. Applications (*K*-types in total) are classified into two sets, *S* and *U*. Resources affordable types belongs to set *S* and resources affordable types to set *U*. Compared with the delay-objective optimization, the constraints related to delay bound now need to be modified.

For flow signals sent from resources affordable types of applications in set $S$ to the local controllers, the QoS formulation is to assure the probability that the waiting delay $w_i$ satisfies the original delay bound $D_\Omega$ for type $\Omega \in S$ application in this set is larger or equal to $\alpha_{i\Omega}$, which can be described as

$$\Pr\{w_i \le D_\Omega\} \ge \alpha_{i\Omega} \ \forall i \in S_1, \Omega \in S. \tag{20}$$

For flow signals sent from resources unaffordable types of applications in set $U$ to the local controllers, the QoS formulation is to ensure that the waiting delay $w_i$ satisfies the enlarged delay bound $\delta$ for type $\omega \in U$ application in this set is larger or equal to$\alpha_{i\omega}$, which can be described as

$$\Pr\{w_i \le \delta\} \ge \alpha_{i\omega} \ \forall i \in \mathrm{S}_1, \omega \in U. \tag{21}$$

Similarly, for flow signals sent from resources affordable types of applications to the cloud controller, the QoS formulation also works to ensure that the waiting delay $w_c$ satisfies the original delay bound $D_\Omega$ for type $\Omega$ application in set $S$ is larger or equal to $\alpha_{c\Omega}$, which is described as

$$\Pr\{w_c \le D_\Omega\} \ge \alpha_{c\Omega} \ \forall \Omega \in S. \tag{22}$$

For flow signals sent from resources unaffordable types of applications in set $U$ to the cloud controller, the QoS formulation serves to assure the probability that the waiting delay $w_c$ satisfies the enlarged delay bound $\delta$ for type $\omega$ application in this set is larger or equal to $\alpha_{c\omega}$, which can be described as

$$\Pr\{w_c \le \delta\} \ge \alpha_{c\omega} \ \forall \omega \in U \tag{23}$$

From (20)–(23), the constraints formulation can be derived from the results obtained in the delay-objective mode as in (24) and (25). For resource affordable applications in set $S$, we then can show

$$\sum_j \phi_{ij} p_j \lambda_j S_j - \mu_i - \ln(1-\alpha_{i\Omega}) D_\Omega^{-1} \le 0 \ \forall i \in S_1, j \in S_2, \Omega \in S \tag{24}$$

and

$$F_c\left(D_\Omega; \mu_d, \mu_c - \sum_j \phi_{cj} p_j \lambda_j S_j\right) \ge \alpha_{c\Omega} \ \forall j \in S_2, \Omega \in S. \tag{25}$$

In other words, (24) and (25) means that the delay constraint conforming probability in controller *i* and the cloud controller is assured to be larger or equal to $\alpha_{i\Omega}$ and $\alpha_{c\Omega}$, respectively, for resource affordable applications.

For the resource unaffordable applications in set *U*, we can write

$$\sum\nolimits_j \phi_{ij} p_j \lambda_j S_j - \mu_i - \ln(1-\alpha_{i\omega})\, \delta^{-1} \le 0 \ \forall i \in S_1, j \in S_2, \omega \in U \tag{26}$$

and

$$F_c\left(\delta; \mu_d, \mu_c - \sum_j \phi_{cj} p_j \lambda_j S_j\right) \ge \alpha_{c\omega} \ \forall j \in \mathrm{S}_2, \omega \in U \tag{27}$$

where (26) and (27) are used to assure that the delay constraint conforming probability in controller *i* and the cloud controller is guaranteed to be larger or equal to $\alpha_{i\omega}$ and $\alpha_{c\omega}$, respectively, for resource unaffordable applications.

Finally, the load-balancing problem in fairness-objective mode can be modeled as a convex optimization problem and described as

$$\underset{\phi_{ij}, \alpha_{ik}, \alpha_{ck}, \delta}{\text{minimize}} \ \delta$$

s.t.

$$\left(\sum_i \sum_{j \in \text{type } k} \alpha_{jk} \phi_{ij} p_j \lambda_j S_j + \sum_{j \in \text{type } k} \alpha_{ck} \phi_{cj} p_j \lambda_j S_j\right) \times \left(\sum_{j \in \text{type } k} p_j \lambda_j S_j\right)^{-1} \ge \eta_k \cdot 100\% \ \forall i \in S_1, j \in S_2, k \in S_3$$

$$\sum_j \phi_{ij} p_j \lambda_j S_j - \mu_i - \ln(1-\alpha_{i\Omega}) D_\Omega^{-1} \le 0 \ \forall i \in \mathrm{S}_1, j \in \mathrm{S}_2, \Omega \in S$$

$$\sum_j \phi_{ij} p_j \lambda_j S_j - \mu_i - \ln(1-\alpha_{i\omega})\delta^{-1} \leq 0 \ \forall i \in \mathrm{S}_1, j \in \mathrm{S}_2, \omega \in U$$

$$F_c\left(D_\Omega; \mu_d, \mu_c - \sum_j \phi_{cj} p_j \lambda_j S_j\right) \geq \alpha_{c\Omega} \ \forall j \in \mathrm{S}_2, \Omega \in S$$

$$F_c\left(\delta; \mu_d, \mu_c - \sum_j \phi_{cj} p_j \lambda_j S_j\right) \geq \alpha_{c\omega} \ \forall j \in \mathrm{S}_2, \omega \in U$$

$$\sum_j \phi_{ij} p_j \lambda_j S_j \mu_i^{-1} \leq 1 \ \forall i \in \mathrm{S}_1, j \in \mathrm{S}_2$$

$$\sum_j \phi_{cj} p_j \lambda_j S_j \mu_c^{-1} \leq 1 \ \forall j \in \mathrm{S}_2$$

$$0 \leq \phi_{ij} \leq 1 \ \forall i \in \mathrm{S}_1, j \in \mathrm{S}_2$$

$$0 \leq \alpha_{ik} \leq 1 \ \forall i \in S_1, k \in S_3$$

$$0 \leq \alpha_{ck} \leq 1 \ \forall k \in S_3$$

$$\phi_{cj} + \sum_i \phi_{ij} = 1 \ \forall i \in S_1, j \in S_2. \quad (28)$$

In the fairness-objective mode, given a pre-determined value $\eta_{\,\mathrm{k}}$ for type $k$ flow signal, feasible solutions may not exist when any application in set $S$ surpasses its original delay bound $D_\Omega$ due to the growth of the total input flow signal $\Sigma_j \lambda_j$.

When no feasible solution in fairness-objective mode is available, the system then removes the QoS unsatisfied applications from set $S$ into set $U$ and provide new feasible solution. There always exist a feasible solution when all applications are categorized into set $U$ since their delay bound $\delta$ can always be enlarged.

## IV. Simulation and Analysis

In this section, we validate the performance of the proposed HECSDN system. The ECSDN algorithm in the system is compared with the Greedy scheme facing different patterns of traffic arrivals. In the Greedy scheme, the arrivals are allowed to dynamically choose the local controllers and the cloud controller with the smallest number of waiting packets.

Here, the employed simulation program consists of two parts. The first part for the queueing model simulation is coded with Python. The second part for optimization is carried out using MATLAB Optimization Toolbox Version 8.1.

### A. Parameter Setup

A testbed with three local controllers, a cloud controller and one central controller is established in this simulation. The setting of model parameters on hardware (switches and controller) processing rate and network traffic are considered as the benchmark for numerical analysis according to real conditions in previous related works. Considering the possible processing rate variations in real conditions, the processing rate of signal-packets for local controllers are set with different values around the processing rate of a typical controller, which is 30 000 packets/s [12], which is $\mu_1 = 33\,000$ (packets/s), $\mu_2 = 35\,000$ (packets/s) and $\mu 3 = 32\,000$ (packets/s), respectively. The cloud controller is assumed to operate over a VM with combined processing capability of $\mu_c = 150\,000$ (packets/s). Each switch is set with the data-packet forwarding rate $\mu_3 = 50\,000$ (packets/s) [24].

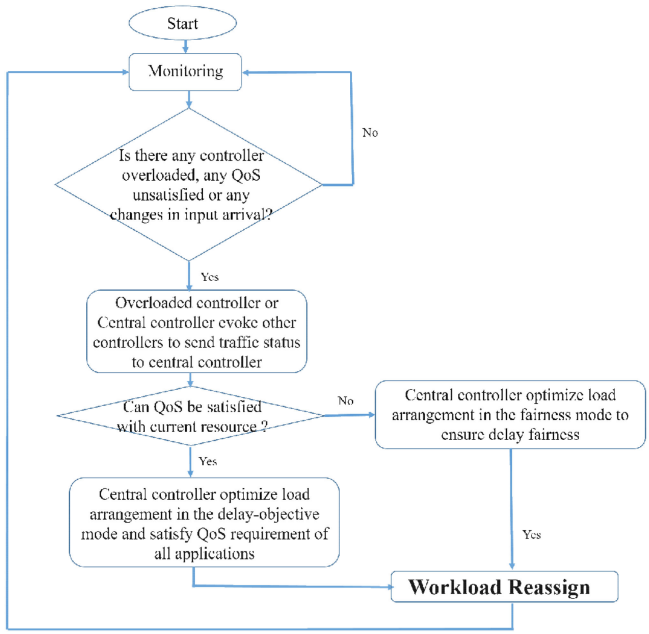


Fig. 4. Flowchart of the ECSDN algorithm.

According to real network traffic measurements [3], a packet in the switch belongs to a new data flow with average probability of $p_j = 0.03$. The average number of packets per batch are set to $S_j = 8$ for all switches $j$. The optimization parameter $\beta$ can be set to different values according to network requirements. In this paper, $\beta$ is set to 0.5 considering equivalent importance on signal-packet mean delay and delay guard interval.

Performances are evaluated considering two major types of delay-sensitive interactive applications: VoIP and video streaming. Based on a series of subject tests, The International Telecommunication Unit G.114 specification recommends less than a 150 millisecond (ms) one-way end-to-end delay for high-quality real-time traffic in audio and video streaming [26]. To achieve great transmission performance, this paper considers the control plane delay to be within 0.6 and 0.4 ms for VoIP and video streaming. Moreover, the QoS requires the packet-overdue ratios of the two types to be limited within 1%. In the experiment, every switch allows data flow traffic arrivals of a certain type.

### B. Design of Arrival Patterns

This section describes the arrival pattern designed for testing our algorithms under various traffic conditions and demonstrate the result using two performance metrics, the packet mean waiting time and packet-overdue ratio in the following section. In the simulation, the packet corresponds to the flow signal-packets in

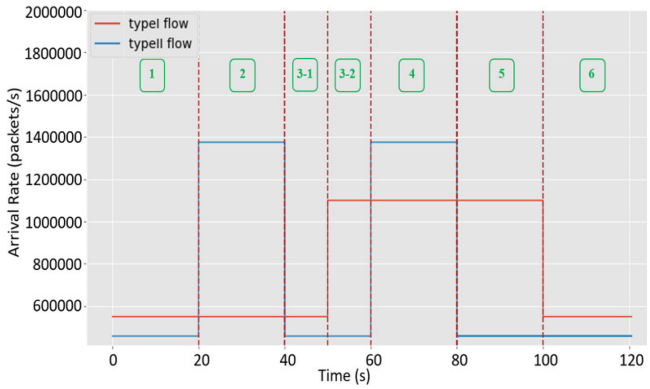


Fig. 5. Arrival pattern over time.

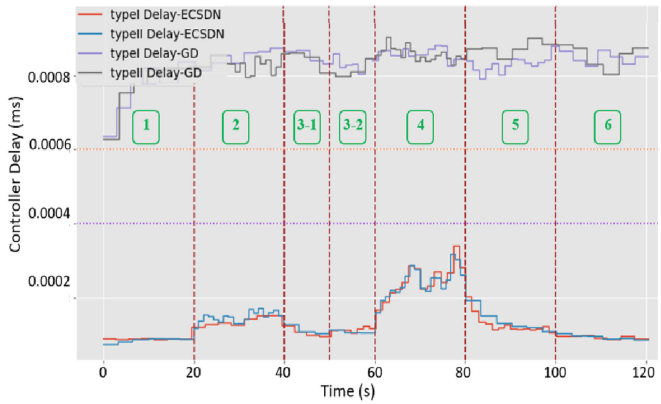


Fig. 6. Mean packet delay over time.

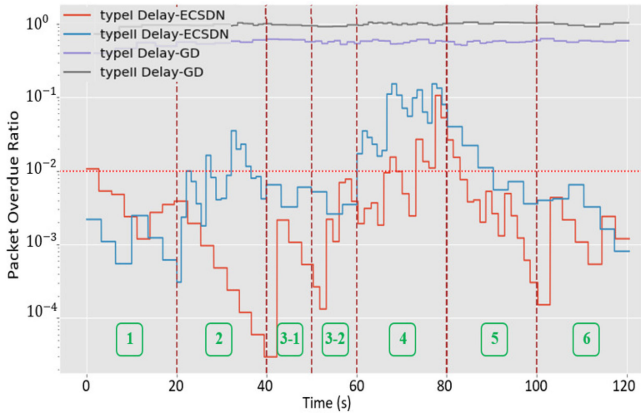


Fig. 7. Packet overdue ratio over time.

the control plane. Type I application refers to VoIP and type II to video streaming.

According to different arrival rates of data flows, the pattern of type I arrival stream can be divided into three phases (0–50 s, 50 s-100 s, and $>$ 100 s) as shown in Fig. 5. The Type II arrival stream keeps oscillating every 20 s in the first 80 s and remains at constant rate in the last 40 s. The total network size of the switches is set to 56 with half of the switches serving type I arrival stream and the other serving type II arrival stream. In addition, the edge-to-cloud distance is set to be 100 km.

In the first region, type I arrival stream remains slightly larger than type II. Using the ECSDN algorithm, the mode of optimization remains in delay-objective. In the second region, the arrival rate of type II arrival stream is much higher than type I. In this situation, the computation resource remains sufficient and optimization stays in the delay-objective mode.

Focusing on rapid traffic fluctuations in shorter intervals, the third region can be further divided into two sub-regions 3-1 and 3-2, respectively. In region 3-1, the arrival rate declines and transforms back to the pattern in region 1. In region 3-2, type I arrival stream boosts up to the second phase while type II arrival stream drops back to the original value.

For the fourth region, type I arrival stream is with high arrival rate while type II arrival stream oscillates back to a value higher than type I arrival stream. In this region, the total arrival is boosted to high-level and the computational resources become unaffordable for the controllers to satisfy the QoS requirements of both types. The optimization procedure, therefore, changes to the fairness-objective mode.

In the fifth region, the arrival rate drops and transforms back to the pattern in region 3.2. In this region, computational resources are satisfied and the optimization returns to the delay-objective mode. In the final region of the scenario, the arrival rate of both type I and type II arrival stream is set to the original value in the first region. In this condition, computational resource is satisfied and the optimization in ECSDN returns back to delay-objective mode.

### C. Performance Evaluation Under Different Arrival Patterns

It can be clearly observed in Figs. 6 and 7 that the ECSDN algorithm outperforms the Greedy scheme in the two performance metrics, packet delay, and overdue ratio. The performance evaluations in Figs. 6 and 7 can be further divided into six regions according to the traffic arrival patterns. In order to observe both the transient and steady-state behavior of the flows, all regions are separated into equal time intervals with 20 s each in that the transient states last for around 18 s.

In region 1 (see Figs. 6 and 7), the mean packet delays of type I and type II arrival streams are both 0.08 ms and the packet overdue ratio of the two types are below 1% as required. However, using the Greedy approach, the mean packet delay approximates 0.82 ms for both and the packet overdue ratio for both types of applications are greatly violated against the requirement. It can be observed that ECSDN achieve greater performance than Greedy.

In region 2, although the mean packet delay of both arrivals increases under ECSDN, it only causes an average amount of 0.16 ms delay on the packets, compared with the delay of 0.83 ms in Greedy. This proves the stability of the ECSDN algorithm during high traffic load. For the performance on packet overdue ratio, Fig. 7 shows that the ratios of both types are significantly improved in ECSDN. However, the ratio of type II delay violation only becomes higher than the 1% requirement during a transient period and it soon becomes satisfied with the requirement after entering the steady state.

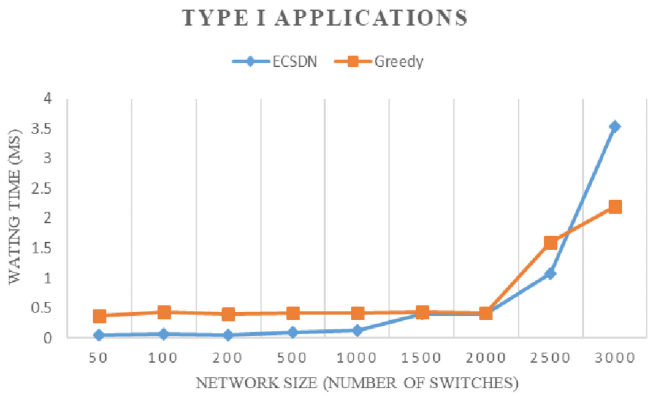


Fig. 8 Relationship of delay and network size for type I.

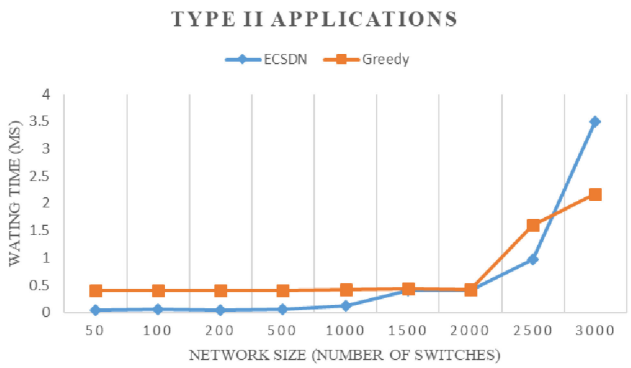


Fig. 9. Relationship of delay and network size for type II.

In region 3-1, computational resources are satisfied and the optimization returns to the delay-objective mode. The packet overdue ratio of both types in ECSDN drops back rapidly within the delay bound as expected while Greedy does not.

In region 3-2, computational resources remain sufficient and the optimization stays in the delay-objective mode. The amount of the mean packet delay in ECSDN is much smaller than Greedy by 0.6 ms. The overdue ratio in ECSDN for both types of applications are within the 1% requirement, while Greedy stays highly violated above.

In region 4, the mean packet delay of both types of applications raises up to approximately 0.33 ms in ECSDN, while in Greedy, the mean packet delay of both types of applications approximates 0.83 ms. The packet overdue ratio of either type of application is over 1% requirement in ECSDN. However, the ratio in ECSDN is still relatively small, compared with the Greedy scheme.

Other than the performances, the effectiveness of fair resource allocation in fairness-objective mode is demonstrated in region 4. From 60 s to 70 s, type II packet overdue radio is over 1% while type I packet overdue radio remains less than 1%. However, from 70 s to 80 s, the effect of fairness optimization now plays a role. The delay bounds for the two types are shifted to the same enlarged delay bound causing type I packet overdue ratio to exceed the 1% requirement and its performance is better than that of type II. Since the original delay bound for type II application is smaller, it can be expected that its packet overdue ratio stays higher than type I when the delay bounds of the applications are both shifted to the same enlarged delay bound.

In regions 5 and 6, the arrival pattern and optimization mode is similar to region 3-2 and region 1. Thus, the simulation result are as expected.

### D. Relationship of Network Size and Delay

In this section, the average arrival rate of flow signal-packets in each switch is approximately $\lambda = 100$ (packets/s). The network size is the number of switches in the network, and the simulation results are shown in Figs. 8 and 9. For both methods, delay increases as the network size grows large. Even in the conditions of fair resource allocation and strict QoS requirements, ECSDN still outperforms the Greedy scheme on delay performances in different network scales.

When the size of the network is small, the difference on delay time between Greedy and ECSDN is significant. For instance, when the size of the network is 150 (*switches*), the average delay in ECSDN is about $7 \times 10^{-2}(\text{ms})$, while in the Greedy scheme, the value of average delay approximates $0.42(\text{ms})$. Under conditions of serving a large network, i.e., when the size approaches 1000, the average delay in ECSDN is about $0.13(\text{ms})$, while in the Greedy scheme, the value of average delay is still close to $0.42(\text{ms})$.

From the result shown in Fig. 7, it can be observed that the optimization mode of the system changes into fairness objective with network size larger than 782. Figs. 8 and 9 show that the performance in controller delay outperforms Greedy when the scale of network approaches 2500. However, as the network size exceeds 2000, the control plane delay in both algorithms increase to millisecond scale. This delay is higher than that of greedy but is still within 4 ms. In conclusion, the ECSDN algorithm has more flexibility and effectiveness even in various network size.

### E. Relationship of Affordable Network Size and Edge-Cloud Distance

In this section, the average arrival rate of flow signal-packets in each switch is approximately $\lambda = 100$ (packets/s). An affordable network size refers to the size that the QoS of both applications in the network can be satisfied. The affordable network size varies along with the edge-to-cloud distance in a nonlinear relationship given in Table II. When the edge-to-cloud distance is 10 km, the controllers are capable of affording a large network with 1656 switches. When the distance increases to 1000 km, the affordable size decreases to only 644. The 958-switch difference in affordable size between 10 and 100 km is much larger than the 54-switch difference between 100 km and 1000 km. This nonlinear relationship between network size and edge-cloud distance indicates the importance in the allocation of the cloud.

TABLE III
RELATIONSHIP OF AFFORDABLE NETWORK SIZE AND EDGE-CLOUD DISTANCE

| Distance (km) | 10 | 100 | 1000 |
|---|---|---|---|
| Accommodable network Size (switches) | 1656 | 698 | 644 |

TABLE IV
RELATIONSHIP BETWEEN UTILIZATION OF CLOUD VM AND EDGE-CLOUD DISTANCE

| Distance (km) | 1 | 10 | 100 | 1000 |
|---|---|---|---|---|
| Cloud VM Utilization | 74.6% | 60.9% | 41.5% | 34.4% |

### F. Relationship of Utilization and Edge-Cloud Distance

In this section, the network size of the system is set to 1500, representing a large-scale network. The data-packet arrival rate of each switch being $\lambda = 100$ (packets/s) operating in the ECSDN algorithm. When the edge-cloud distance increases, the utilization of the controller (cloud VM) will decrease as shown in Table IV. This indicates that the location of a data center for the cloud controller should be set within a certain range of distance to achieve better performance. As observed from the simulation result, when the distance between edge and cloud approaches 1000 km, the utilization of the cloud controller is only 34.4%, while when the distance of the cloud is only 10 km far away from the edge, the utilization of the cloud controller approaches 60.9%.

When the edge-to-cloud distance is between 100 and 1000 km, the drop in utilization due to the network size increase is only 7.1%. When the distance is 10 and 100 km, the difference is approximately 20% between conditions. The utilization of the cloud also becomes much smaller when the cloud is 100 km away from the edge, as compared to the situation when the location of the cloud is within 10 km of the local controllers. This coincides with the results concluded from Table III. The simulation results show a significance impact of edge-to-cloud distance on the system performance. It also shows a new and great significance in capacity improvement when a cloud controller appropriately installed. Meanwhile, we have shown in our example that this arrangement can support at least 1000 extra switches.

## V. CONCLUSION

In large-scale networks, controllers are required to process millions of flow signals per second without compromising the QoS of network applications. Well equipped to improve SDN's efficiency on large-scale networks and to make the best use of resources in the control plane, the proposed HECSDN controller system can dispatch certain computation workload to the cloud controller and handle various traffic conditions and fluctuations. As observed from the simulations, the system's performance remains highly stable even under highly fluctuating traffic loads.

Moreover, the system can strongly support large-scale networks under high standards of QoS. When the distance between the edge and the cloud is within 10 km, the HECSDN system allows 99% of the arrival data flows with at least 1656 switches to satisfy the QoS requirements and ensure fair resource allocation according to the Minimax criteria on the extra guard interval. In summary, the proposed controller system has proved effective on large-scale SDN without sacrificing the overall performance and the QoS of various network applications.

## REFERENCES


[1] R. Alvizu *et al.*, "Comprehensive survey on T-SDN: Software-defined networking for transport networks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2232–2283, Oct.–Dec. 2017.

[2] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: An intellectual history of programmable networks," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 87–98, 2014.

[3] B. Xiong *et al.*, "Performance evaluation of OpenFlow-based software-defined networks based on queueing model," *Comput. Netw.*, vol. 102, pp. 172–185, 2016.

[4] A. D. Gante, M. Aslan, and A. Matrawy, "Smart wireless sensor network management based on software-defined networking," in *Proc. 27th Biennial Symp. Commun.*, 2014, pp. 71–75.

[5] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[6] A. Tavakoli *et al.*, "Applying NOX to the datacenter," in *Proc. HotNets*, 2009.

[7] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 136–141, Feb. 2013.

[8] L. Yang, B. Ng, and W. K. G. Seah, "Heavy hitter detection and identification in software defined networking," in *Proc. 25th Int. Conf. Comput. Commun. Netw.*, 2016, pp. 1–10.

[9] H. C. Joksch, "The shortest route problem with constraints," *J. Math. Anal. Appl.*, vol. 14, no. 2, pp. 191–197, 1966.

[10] Z. Jia and P. Varaiya, "Heuristic methods for delay-constrained least cost routing using k-shortest-paths," *IEEE Trans. Automat. Control*, vol. 51, pp. 707–712, Apr. 2006.

[11] L. Zhao *et al.*, "Traffic engineering in hierarchical SDN control plane," in *Proc. IEEE 23rd Int. Symp. Quality Service*, 2015, pp. 189–194.

[12] Y.-W. Ma *et al.*, "Load-balancing multiple controllers mechanism for software-defined networking," *Wireless Pers. Commun.*, vol. 94, no. 4, pp. 3549–3574, 2017.

[13] Y. Zhou *et al.*, "A load balancing strategy of SDN controller based on distributed decision," in *Proc. IEEE TrustCom*, 2014, pp. 851–856.

[14] L. Guo *et al.*, "Dynamic performance optimization for cloud computing using m/m/m queueing system," *J. Appl. Math*, vol. 2014, 2014, Art. no. 756592.

[15] S. Rashidi and S. Sharifian, "A hybrid heuristic queue based algorithm for task assignment in mobile cloud," *Future Gener. Comput. Syst.*, vol. 68, pp. 331–345, 2017.

[16] M. R. Rahimi *et al.*, "MAPCloud: Mobile applications on an elastic and scalable 2-tier cloud architecture," in *Proc. IEEE 5th Int. Conf. Utility Cloud Comput.*, 2012, pp. 83–90.

[17] F. Ramezani *et al.*, "Evolutionary algorithm-based multi-objective task scheduling optimization model in cloud environments," *World Wide Web*, vol. 18, no. 6, pp. 1737–1757, 2015.

[18] F. P.-C. Lin and F. K. H. Phoa, "A performance study of parallel programming via CPU and GPU on swarm intelligence based evolutionary algorithm," in *Proc. Int. Conf. Intell. Syst., Metaheuristics Swarm Intell.*, 2017, pp. 1–5.

[19] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[20] J. Wu *et al.*, "Goodput-aware load distribution for real-time traffic over multipath networks," *IEEE Trans. Parallel Distrib. Syst., vo*l. 26, no. 8, pp. 2286–2299, Aug. 2015.

[21] X. Wang *et al.*, "Delay-cost tradeoff for virtual machine migration in cloud data center," *J. Netw. Comput. Appl.*, vol. 78, pp. 62–72, 2017.

[22] F. Li *et al.*, "A QoS guaranteed technique for cloud applications based on software defined networking," *IEEE Access*, vol. 5, pp. 21229–21241, 2017.

[23] D. Kreutz *et al.*, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.
[24] Y. Goto *et al.*, "Queueing analysis of software defined network with realistic openflow based switch model," in *Proc. IEEE 24th Int. Symp. Model., Anal. Simul. Comput. Telecommun. Syst.*, 2016, pp. 301–306.
[25] J. F. Shortle *et al.*, "Simple markovian queueing models" in *Fundamentals of Queueing Theory*, 4th ed. New York, NY, USA: Wiley, 2018, pp. 49–65.
[26] Y. Chen, T. Farley, and N. Ye, "QoS requirements of network applications on the Internet," *Inf. Knowl. Syst. Manage.*, vol. 4, no. 1, pp. 55–76, 2004.

**Frank Po-Chen Lin** received the B.S. degree in electrical engineering from National Sun Yat-sen University, Kaohsiung, Taiwan, in 2016 and the M.S. degree from National Taiwan University, Taipei, Taiwan in 2018.

He is currently a Research Assistant with the Institute of Statistical Science, Academia Sinica, Taipei, Taiwan. His current research interests include software defined networking, scheduling algorithm designs, parallel computing, and supercomputing.

Mr. Po-Chen Lin was a recipient of the College Student Research Creativity Award in the Ministry of Science and Technology, Taiwan, in 2016.

**Zsehong Tsai** (M'89) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1983 and the M.S. and Ph.D. degrees from the University of California, Los Angeles, CA, USA, in 1985 and 1988, respectively.

During 1988–1990, he was a member of the technical staff at AT&T Bell Laboratories, where he investigated the performance aspects of network management systems. Since 1990, he has been with the Department of Electrical Engineering and Graduate Institute of Communication Engineering, NTU, where he is currently a Professor.