An Effective Approach to Controller Placement in Software Defined Wide Area Networks

Guodong Wang*, Yanxiao Zhao*, Jun Huang†, Yulei Wu‡
*Department of Electrical and Computer Engineering,
South Dakota School of Mines and Technology, Rapid City, SD 57701, USA
†School of Computer Science and Technology,
Chongqing University of Posts and Telecom, Chongqing, 400065, China
‡College of Engineering, Mathematics and Physical Sciences,
University of Exeter, Exeter, EX4 4QF, UK

***Abstract*—One grand challenge in Software Defined Networking (SDN) is to select appropriate locations for controllers to shorten the latency between controllers and switches in wide area networks. In the literature, the majority of approaches are focused on the reduction of packet propagation latency, but propagation latency is only one of the contributors of the overall latency between controllers and their associated switches. In this paper, we explore and investigate more possible contributors of the latency, including the end-to-end latency and the queuing latency of controllers. In order to decrease the end-to-end latency, the concept of network partition is introduced and a Clustering-based Network Partition Algorithm (CNPA) is then proposed to partition the network. The CNPA can guarantee that each partition is able to shorten the maximum end-to-end latency between controllers and switches. To further decrease the queuing latency of controllers, appropriate multiple controllers are then placed in the subnetworks. Extensive simulations are conducted under two real network topologies from the Internet Topology Zoo. The results verify that the proposed algorithm can remarkably reduce the maximum latency between controllers and their associated switches.**

***Index Terms*—Controller Placement Problem, SDN, Latency, WAN**

I. Introduction

The proliferation of new computing technologies such as big data, Internet of Things (IoT) and cloud services fundamentally change the way we store, acquire and transfer information and data. Accompanied with these new computing trends, challenges including network upgrade and management, resource utilization, and high-speed transmission, are raised. To tackle these challenges, Software Defined Networking (SDN) is envisioned as a promising paradigm for the next-generation networking. Compared with conventional networking, SDN decouples the control plane from data plane. That is, the control plane is formed by a set of dedicated controllers and each of them manages one or more simplified packet-forwarding switches. As a result, the control and management functions are implemented at SDN controllers so that applications and network services are abstracted from the underlying infrastructure.

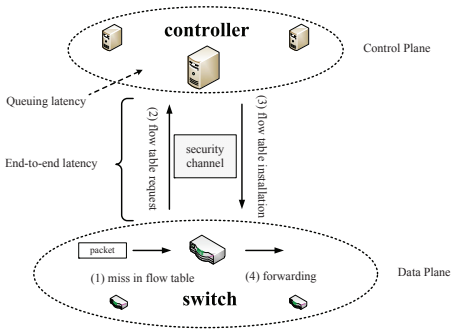


Fig. 1. Demonstration of the packet process in OpenFlow

In SDN, the software based controller serves as the network intelligence, and network switches become simple packet forwarding devices, which can be programmed via an open interface, e.g., OpenFlow [2]. A simplified model is illustrated in Fig. 1 to demonstrate the packet delivery process in the OpenFlow: (1) if a packet arrives at a switch but the flow table in this switch does not match this packet, (2) the switch will generate a request packet and send it to its associated controller. After receiving this request, (3) the controller will respond this switch with a new forwarding policy and the switch will update its flow table accordingly. Afterwards, (4) the packet is delivered based on the

This paper is an extended version of our paper published in the 2016 IEEE International Conference on Communications (ICC) [1] Please refer to Dr. Guodong Wang via wgdaaa@gmail.com for more information. Dr. Jun Huang's work was supported in part by NSFC (Grant number 61309031 and 61671093)

updated flow table.

As illustrated in this model, all functions of SDN are carried out through frequent message exchanges between controllers and switches. Therefore, the location of controller(s) dramatically influences message exchanges, and hence affects the performance of SDN. The problem of exploring appropriate locations for controllers is termed as the controller placement problem. This problem refers to how to place controllers in an SDN-enabled network and allocate associated switches to those controllers so as to achieve an objective. The controller placement is more significant for wide area networks, because of their irregular topology and large packet propagation latency.

In the efforts to address the controller placement problem in SDN, numerical metrics have been proposed in the literature. Among those metrics, network latency between controllers and switches plays the most critical role, since it heavily affects the overall performance of SDN [3] [4]. For example, if a packet does not match the flow table in a switch, the switch needs to wait for a period of time (processes (2) and (3) in Fig. 1) to get new flow tables installed before processing the packet. Apparently, a long period of waiting time will degrade the overall performance of SDN, e.g., time-insensitive applications may slow down, while real-time tasks may become infeasible. Generally speaking, links with limited bandwidth and heavy traffic are prone to experience congestion, and thus lead to extra latency. Due to the unique feature of SDN, the latency resulted from congestion between controllers and switches is quite limited in an SDN-enabled network. First, the control plane is separated from the data plane in SDN, so the control message between controllers and switches is transmitted in a dedicated channel (out-of-band mode) [5] [6]. In addition, the control messages are relatively light flows in comparison to the loads in the data plane [7].

Considering the significance of latency between controllers and switches, methods have been proposed in the literature to decrease the propagation latency. However, the propagation latency is only one component of the overall latency. Other components include packet transmission latency, switch processing latency and the queuing latency of controllers. An in-depth and quantitative investigation of the overall latency, especially the queuing latency in controllers, has received limited investigation in the literature.

In this paper, we investigate the overall latency between controllers and switches, and seek solutions to shorten the latency in wide area networks. Since the overall latency includes the end-to-end latency (packet transmission latency, packet propagation latency and switch processing latency) and the controller's queuing latency, we address them separately. First, we propose to partition a network into subnetworks to shorten the end-to-end latency between controllers and switches. A Clustering-based Network Partition Algorithm (CNPA) is developed to conduct the network partition. The CNPA can guarantee that each partition is able to shorten the maximum end-to-end latency between controllers and their associated switches. Second, we propose to place multiple controllers in subnetworks that have a large number of switches to further reduce the queuing latency. In order to quantitatively measure the queuing latency, queuing theory is adopted in this paper to formulate the relationship between the number of controllers and the queuing latency. Appropriate number of controllers is calculated for each subnetwork and multiple controllers are placed in subnetworks to shorten the queuing latency.

The main contributions of this paper are briefly highlighted as follows.

- The overall latency between controllers and switches in an SDN-enabled network is well investigated in this paper. Different from the existing work, we explore more possible contributors to the overall latency. Specifically, the overall latency consists of the end-to-end latency and the controllers queuing latency. The end-to-end latency further includes the packet transmission latency, packet propagation latency and switch processing latency.
- The concept of network partition is introduced in this paper to tackle the controller placement problem. In order to shorten the end-to-end latency, we propose to partition a network into subnetworks and place appropriate controllers for each subnetwork. A clustering-based network partition algorithm is then proposed to address the network partition problem. This algorithm guarantees that each partition is able to shorten the maximum end-to-end latency between controllers and their associated switches.
- Multi-controller placement is proposed to shorten the queuing latency of controllers. Leveraged by queuing theory, appropriate number of controllers is calculated for each subnetwork, and multiple controllers are placed in subnetworks that have dense switches to decrease the queuing latency of controllers, and thus reduce the total latency between controllers and switches.
- Extensive simulations are conducted under two real network topologies from the Internet Topology Zoo. The simulation results verify that the proposed algorithm has greatly decreased the latency

between controllers and their associated switches in comparison with the K-means and K-center, which are adopted widely in the literature to address the controller placement problem.

The rest of the paper is organized as follows. In Section II, the related work is introduced. Section III formulates the problem mathematically. In Section IV, the proposed new solution is described in details. Section V presents the performance analysis of the solution. Concluding remarks are drawn in Section VI.

II. Related Work

Controller placement in SDN is one of critical problems and has attracted extensive attention in the literature. The controller placement aims to find out the best k locations in an SDN-enabled network to place controllers in order to enhance the efficiency of network management [8]. Due to the logic separation in SDN, network management is achieved through efficient communications between controllers and switches (north/south), as well as the communications among controllers (east/west). Neither of these two topics is a trivial problem in the literature. In this paper, we mainly focus on the controller placement aiming to optimize the communication between controllers and switches, because: (1) the majority of controller placement methods are proposed to enhance the communication between controllers and switches, and (2) the communication among controllers can be referred from multi-controller approaches, e.g., the inter-domain component in DISCO [9] and the Network Information Base in Onix [10]. In this section, we briefly review the research on controller placement in the literature. The existing research can be classified into four categories with regard to their objectives [8]: reducing network latency between controllers and switches, increasing reliability and resilience, reducing deployment cost and energy consumption, and the multi-objective approach.

A. Reduce network latency between controllers and switches

In an SDN-enabled network, the latency between controllers and switches is especially critical, because the control logic of the network is decoupled from simplified switches and all of the functions of the network are carried out through message exchanges between controllers and switches. Heller et al. [11] initiate the study on controller placement in SDN and propose that propagation latency (average propagation latency and worst-case propagation latency) is the main consideration in their study. This problem is formulated as a facility location problem and K-center is adopted to address this problem. Yao et al. [12] move forward by considering both the propagation latency and the controller capacity. Thus the placement is regarded as a variant of the capacitated K-center problem [13]. Their simulation results show that the proposed solution can reduce the number of controllers and the load of the busiest controllers. The tradeoff between propagation latency and traffic load is investigated in [14]. The simulation results show that the load in control plane can be balanced with a little increment of the delay. [15] investigates the controller placement problem from another point of view. Instead of directly minimizing the latency between controllers and switches, it aims to maximize the number of controlled switches within a latency bound.

B. Increase reliability and resilience

Reliability and resilience are drawing attention of researchers in the literature. In the efforts to increase the reliability of SDN, Zhang et al. [16] propose a min-cut based algorithm to minimize the likelihood of loss of connectivity between the controller and switches. Hu et al. further [17] study the controller placement problem for reliability and propose a greedy algorithm to achieve an efficient placement and reliability. The general goal of Survivor [18] is to maximize connectivity between forwarding devices and controllers instances. Their study verifies that the path diversity can increase the survivability during fail over states. The similar researches also include K-Critical [19]. Simulated annealing is adopted in [20] to increase the reliability of SDN by placing appropriate controllers in the network. Different from the research which improves the resilience of the southbound connections, an approach is proposed in [21] to endure controller failures in SDN-enabled wide area networks.

C. Reduce deployment cost and energy consumption

The network cost and energy consumption have also been investigated in the literature. The network cost is composed of the deployment cost and energy consumption. In order to formulate the deployment cost, authors of [22] develop an optimal model and the overall energy consumption is reduced according to their simulation. The main challenge of this model is its big complexity, e.g., some of the calculations cannot be achieved within 30 hours. In contrast, a dynamic solution is then proposed in [23] to dynamically add or delete controllers according to the change of loads. One of the methods to minimize the energy consumption is GreCo [24]. It proposes to shut down unnecessary

links while ensuring connectivity between switches and controllers. According to the simulation, GreCo is able to save up to 55% energy during off-peak hours. The other approach is developed in [25], which proposes to determine the number of controllers first and then activate required controllers to save energy. Another model [26] is developed to minimize the update cost when new switches are added to an existing network. Since the proposed model is not limited in SDN-enabled networks, it can also be used to plan a new network or update any existing networks.

D. Multi-objective approach

When multiple objectives are considered, it is necessary to address the controller placement problem through multiple-objective optimization. A representative work was conducted in [27], which discussed the controller placement problem by focusing on the resilience and failure of SDN-based core networks. A toolset named POCO was proposed to provide network operators with Pareto-optimal placements. Their findings further revealed that for most of the topologies more than 20% of all nodes should be controllers to guarantee a continuous connection of all nodes to one of the controllers in any arbitrary double link or node failure scenario. The POCO toolset is extended in [28] by adding a heuristic approach, which is less accurate, but yields faster computation times to cope with dynamics of large scale networks. Trade-off between time and accuracy is analyzed in detail via numerous real topologies. Authors of [29] propose an SDN-based management and control framework for fixed backbone networks. Algorithms were developed to allocate managers and controllers in the control layer to achieve adaptive load-balancing and energy management. In [30], a density based controller placement is developed to address the controller placement in terms of latency, fault tolerance, and number of controllers. Performance evaluations have verified that this approach achieves a desirable performance and reduces the executing speed at the same time. Authors of [31] formulate the controller placement problem to a multi-objective model, which aims to increase network reliability, maximize controller load balance and minimize latency between controllers and switches. The simulation results indicate that the proposed approach improves network performance and achieves a desirable tradeoff among these objectives.

According to our investigation, most of the above-mentioned approaches only consider the packet propagation latency between controllers and switches when proposing solutions to minimize the latency. However, the propagation latency is only one of the contributors for the overall latency. Other contributors include packet transmission latency, switch processing latency, and queuing latency of controllers. Therefore, it requires a more extensive research which includes all of these components. In this paper, we explore all of the possible latencies between controllers and switches, and seek solutions to shorten the latencies.

III. Problem Formulation

The latency between controllers and their associated switches is critical for SDN, because all of the functions in an SDN-enabled network are achieved by frequent message exchanges between controllers and switches. In this section, we investigate the possible components of the latency between controllers and switches and formulate the controller placement problem.

A. End-to-end latency

For an SDN-enabled network with given nodes and links, let $S = \{s_1, s_2, \cdots, s_m\}$ denote the switches in the network, and $C = \{c_1, c_2, \cdots, c_n\}$ denote the controllers. Assume the hop number from switch s_m to controller c_n is denoted by $h(s_m, c_n)$. Fig. 2 depicts the end-to-end latency when transmitting a packet from switch s_1 to controller c_1.

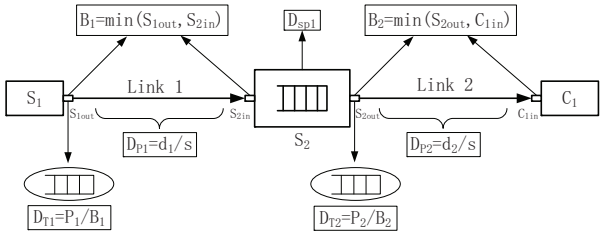


Fig. 2. The end-to-end latency of packet transmission in SDN

The end-to-end latency of transmitting a packet from switch s_1 to controller c_1 is composed of three components: packet transmission latency (D_{Ti}), packet propagation latency (D_{Pi}) and switch processing latency (D_{SPi}). The transmission latency refers to the time taken to push the bits of a packet onto the link. It is given by $D_{Ti} = P_i/B_i$, where P_i is the amount of bits of a packet in link i and B_i is the bandwidth of the link. Note that the bandwidth of a specific link is determined by the minimum transmission rate of the network interfaces, e.g., $B_1 = min(S_{1out}, S_{2in})$ as depicted in Fig. 2. The packet propagation latency refers to the time taken for a packet to reach the destination. The propagation delay is given by $D_{Pi} = d_i/s$, where d_i is the distance of link i and s is the signal speed at which data travels

through the medium. The switch processing latency is denoted to D_{spi}, which is affected by the load of switch i. Therefore, the end-to-end latency for a packet traveling from switch s_m to controller c_n is formulated by

$$D_{e2e}(s_m, c_n) = \sum_{i=1}^{h(s_m,c_n)} (\frac{P_i}{B_i} + \frac{d_i}{S} + D_{SP_{i-1}}) \quad (1)$$

Note that those three components of the end-to-end latency do not weigh equally in different networks. For instance, for a Local Area Network (LAN) with low speed bandwidth and limited transmission range, the packet transmission latency (D_T) is dominant in the end-to-end latency, while the propagation latency (D_P) can be ignored. In contrast, for a backbone network which is equipped with 10 Gbps or even 100 Gbps switches [32] [33] and crosses thousands of miles, the packet transmission latency is negligible, while the packet propagation latency dominates. The switch processing delay (D_{SP}) is mainly determined by the performance of switches. High performance switches with wire-speed switching, low latency and small jitter are usually adopted in backbone networks. Recently, great developments have been achieved in SDN switches. For example, as reported in [34], Corsa DP2100 [33] can achieve 100 Gbps line-rate throughput, which means the switch processing delay (D_{SP}) can be ignored if those switches are adopted in an SDN-enabled back bone network.

B. Queuing latency in controllers

In an SDN-enabled network, if a packet arrives at a switch with unmatched flow table to this packet, the switch will generate a request packet to controllers. To fully understand how a request is sent to controllers, the service model between controllers and switches is depicted in Fig. 3. This model is based on a working principle of the cluster-based multiple controller approach: ONOS [35]. As one of the most famous SDN control plane architectures, ONOS is attracting more and more attention in the academia and industry [36] [37], and has been deployed in both production and research networks [38], including GEANT pan European Network, FIU/AMLight network, Internet2 AL2S network, etc. In ONOS, switches are connected to multiple ONOS controllers for load-balancing and high availability. The scheduler adopted in the model is referred from the ONOS application management subsystem, which is responsible for distributing load among multiple controllers.

Assume there are j switches which are connected to m controllers, and both the switches and controllers are

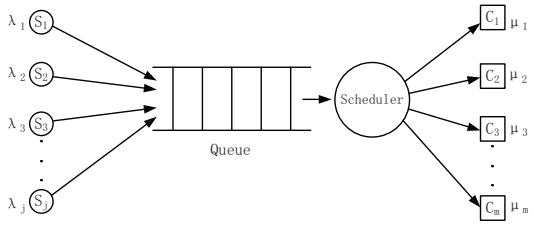


Fig. 3. The service model between controllers and switches in SDN

independent of each other. The consecutive requests may be randomly sent from any switches and the arrival time is exponentially distributed. Therefore, those requests follow a Poisson Process and the arrival rate is assumed to λ_i. The total requests' arrival rate in the system is $\lambda = \sum_{i=1}^{j} \lambda_i$. Those requests are stored in the 'Queue' before sending to the 'Scheduler', which is utilized to distribute requests to controllers in the system. Suppose there are m controllers in the system and their service rate is μ_i, so the total service rate is $\sum_{i=1}^{m} \mu_j$. Since both the arrival and service times are exponentially distributed and m controllers exist in the system, the SDN system can be regarded as a $M/M/m$ queuing model [39] [40]. Here m stands for the number of servers in the system. To simplify the analysis, all controllers are assumed to have the same service rate μ. Then the traffic intensity is termed as $\rho = \lambda/m\mu$. The steady-state limiting probability p_k is formulated by

$$p_k = p_0(\frac{\lambda}{\mu})^k \frac{1}{m!m^{k-m}} \quad for \quad k \leq m \quad (2)$$

Because $\sum_{k=0}^{\infty} P_k = 1$, we can derive p_0 from Eq. (2) as

$$p_0 = [\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!(1-\rho)}]^{-1} \quad (3)$$

The queuing latency, denoted by D_{que}, can be formulated by Eq. (4) to depict the time a request packet remains in the queue before being processed. More details of the M/M/m model and the related derivations can be referred to [39] [40].

$$D_{que} = \frac{L_q}{\lambda} = \frac{(m\rho)^m \rho p_0}{m!(1-\rho)^2 \lambda} \quad (4)$$

The queuing latency analysed above also applies to the situation that requests do not send to the controller. Due to the memoryless property of underlying exponential distribution of Poisson process, the splitting of Poisson process gives rise to a Poisson process. That is to say, given that the packet arriving at a switch follows Poisson process with mean arrival rate λ, if a packet is dropped with the probability p, the packet arrival process from

switch to controller is the splitting of the Poisson process with the splitting probability $(1-p)$, i.e., the Poisson process with the mean arrival rate $(1-p)\times\lambda$. Therefore, the service model between switches and controllers still follows the Poisson process. The only difference is the reduced mean arrival rate: $(1-p)\times\lambda$.

C. Problem formulation

In this subsection, we formulate the network partition problem in SDN. For an SDN-enabled network with given nodes and links, the physical topology of the network is denoted by an undirected graph $G=(V,E)$, where V is the set of nodes, representing both of the switches and controllers in the network. This is based on the assumption that controllers are placed to the locations where switches exist so that controllers and switches can be connected conveniently [11] [12] [28]. E is the set of physical links among those nodes. Assume the number of subgraphs is given by K, denoting the number of subnetworks that the network will be partitioned. When SDN is considered, the network partition can be defined by $SDN_i(V_i,E_i)$, which subjects to:

$$\bigcup_{i=1}^{k} V_i = V; \quad \bigcup_{i=1}^{k} E_i = E \tag{5}$$

$$SDN_i \cap SDN_j = \phi, \quad \forall\ i \neq j, i, j \in k \tag{6}$$

$$Similarity(SDN_i) = TRUE, \quad \forall\ i \in k \tag{7}$$

$$Similarity(SDN_i \cap SDN_j) = FALSE, \quad \forall\ i \neq j,\ i,j \in k \tag{8}$$

$$SDN_i\ is\ a\ connected\ region \quad \forall\ i \in k \tag{9}$$

A successful network partition should divide a network, including all of the nodes and links in the network, into subnetworks without introducing over-allocation (one element is allocated to more than one subnetworks) and miss-allocation (elements are not allocated to any subnetwork). Therefore, Eq. (5) is introduced to indicate that the total subnetworks need to cover all the network elements: nodes and links. In Eq. (6), ϕ stands for an empty set, which means that a node or link can be only allocated to one subnetwork. Eq. (7) implies that the elements in one sub-network have the same similarity. Eq. (8) suggests that the elements allocated to different subnetworks have different similarity. Here the similarity is defined as latency in this paper. Specifically, the network is expected to be divided in a way that the nodes in one cluster have a smaller latency (Eq. (7)), while the nodes in different clusters have a larger latency (Eq. (8)). Eq. (9) indicates that all the vertexes in one subnetwork are connected by links.

The objective of this paper is to place controllers in an SDN-enabled wide area network to minimize the maximum latency between controllers and switches. The latency is denoted to $D_{total} = D_{e2e}(s_m, c_n) + D_{que}$, which includes both the end-to-end latency and queuing latency of controllers. Therefore, the objective is formulated below.

$$\min\{\max\{D_{e2e}(s_m, c_n) + D_{que}\}\}$$

such that

$$s_m,\ c_n \in SDN_i \quad (\forall i \in k)$$

Note that the ultimate goal of this objective is to reduce the total latency so that a latency requirement can be satisfied for a specific application. Assuming the requirement of latency is denoted by T_{th}, the maximum total latency is expected to be less than T_{th}.

IV. Network partition and multi-controller placement algorithm

In this section, a Clustering-based Network Partition Algorithm (CNPA) is proposed to address the controller placement problem for an SDN-enabled wide area network. The objective of CNPA is to shorten the maximum total latency between controllers and their associated switches. The total latency is composed of the end-to-end latency and the queuing latency of controllers. The end-to-end latency is relatively stable, while the queuing latency varies. In addition, for a wide area network, the end-to-end latency is the majority contributor of the overall latency (e.g., the biggest round end-to-end latency is 50 milliseconds in OS3E), while the queuing latency is in the order of several milliseconds or fewer. Therefore, it is necessary to tackle these two latencies separately. Specifically, the first step of our solution is to explore methods to partition a network into subnetworks to shorten the end-to-end latency. Due to distinct geographic distributions, the density of switches in each subnetwork varies significantly. The subnetworks with a large number of switches may experience excessive queuing latency. Therefore, in the second step, multiple controllers are placed inside subnetworks with a large number of switches.

A. A clustering-based network partition algorithm

In this subsection, we propose a clustering-based network partition algorithm and elaborate how to partition a network into subnetworks to shorten the maximum end-to-end latency.

Essentially, the network partition problem resembles the clustering problem, and solutions can be borrowed from contexts of clustering algorithms. However, the standard clustering algorithms, e.g., K-means and K-center, cannot be directly adopted to partition a network into subnetworks. To facilitate understanding, we first introduce the standard K-means method and point out its shortcomings in partitioning a network topology. Note that there are two critical parameters in clustering algorithms: 'center' and 'centroid'. For clarification, the *initial* nodes which are selected to perform clustering algorithms are termed as 'center'. The *actual center* of each cluster which is eventually found by clustering algorithms is termed as 'centroid'. The standard K-means clustering algorithm includes four main steps:

1) initialize k clusters and allocate one center for each cluster using random sampling;
2) allocate nodes into one of the clusters based on Euclidean distance;
3) recalculate centroid for each cluster;
4) repeat step 2 and 3 until there is no change in each cluster.

The standard K-means algorithm cannot be directly applied to partition network topologies due to the following reasons. First, randomly choosing initial centers will not guarantee that each partition can shorten the maximum latency between centroid and its associate nodes in the subnetwork. Second, the Euclidean distance cannot be used in calculating the distance between two nodes, as physical links may not exist in the path of the Euclidean distance. Third, the centroid, where controller is placed, should be chosen from V ($centroid \in V$) to guarantee there are physical connections between the centroid and its associate nodes. Those shortcomings also exist in the K-center algorithm. To overcome those shortcomings, CNPA is proposed to address the network partition problem and is summarized in Algorithm 1.

Algorithm 1: Clustering-based Network Partition Algorithm

Input:
(1) A network topology $G = (V, E)$.
(2) The number of subnetworks (k).
Output:
(1) Centroids of k clusters.
(2) Allocation of v switches to one of the k clusters.
Preparatory work: Calculate the end-to-end latency between any two nodes $l_{e2e}(u, v)$.
step 1: Randomly select one node from V as the first initial center of G.
step 2: Calculate the actual center (centroid) of the network. The node which has the smallest sum end-to-end latency to other nodes is selected as the centroid (c_1).
step 3: Find out the second initial center of the network. The node which has the biggest end-to-end latency to the centroid is selected as the second initial center (c_2).
step 4: Distribute vertex v ($v \in V$) to one of the clusters using the below relation:

$$v \in cluster_i,\ if\ l_{e2e}(v, c_i) < l_{e2e}(v, c_j),\ \forall i, j \in \{1, 2, \cdots, k\}$$

step 5: Update centroids $C' = \{c'_1, c'_2, \cdots c'_k\}$ such that the sum end-to-end latency from all nodes in $cluster_i$ to the new centroid c'_i is minimized.

$$c'_i = v_m,\ if\ l_{e2e}(v_m, v) = minimum,\ \forall\ m \in size(cluster_i), v\ \in cluster_i,\ i = \{1, 2, \cdots, k\}$$

step 6: Repeat steps 3, 4 and 5 until the network is partition into K subnetworks.

The preparatory work of CNPA is to calculate the end-to-end latency ($l_{e2e}(u, v)$) between any two nodes. The shortest path distance is adopted in CNPA to calculate the end-to-end latency. Specifically, given a network topology with numerous links and nodes, the adjacent matrix of this topology is firstly calculated. Coordinates of those nodes are obtained from Google Map and distances of those links are calculated by using the 'haversine' formula [41] [42]. The shortest path as well as the shortest path distance between any two nodes is calculated based on the coordinates and the adjacent matrix by using the Dijkstra's algorithm [43]. Note that the Euclidean distance is usually adopted in the standard clustering algorithms, such as K-center and K-means. However, the Euclidean distance is not applicable for partitioning a network topology, as the nodes in a network topology are only connected by physical links. Therefore, the shortest path distance rather than Euclidean distance is adopted in CNPA to calculate the propagation latency as well as the end-to-end latency. Accordingly, this method is also adopted in associated clustering algorithms, including K-means and K-center, which are introduced to compare with the CNPA.

The processes of CNPA are summarized as follows. The first step of CNPA is to randomly select one node as the center of the network. As the center is randomly selected, CNPA will further find out the actual center (centroid) of the network in the second step. Specifically, CNPA calculates each node's sum end-to-end latency to

other nodes and selects the one that has the minimum sum latency as the centroid. In the third step, CNPA will find out the second center of the network. The second center is selected as the node which has the biggest end-to-end latency to the centroid. In the fourth step, CNPA treats the centroid (c_1) and the second center (c_2) as two initial centers and allocate associated nodes to those centers. Specifically, for each node n_i, calculate its latency to those two centers and get two latencies, $l_1 = l_{e2e}(n_i, c_1)$ and $l_2 = l_{e2e}(n_i, c_2)$. Compare those two latencies and allocate the node to the center which is closer to it. For example, if $l_1 < l_2$, node n_i will be assigned to c_1. Once the node is assigned to one center, the centroid of this cluster will be recalculated based on the minimum sum end-to-end latency described in the second step. The process is continuing until the network is eventually divided into *K* subnetworks. It is worthy to mention that unlike regular clustering algorithms, which randomly select all *K* centers at once and then optimize all of them during the following iterations, the proposed CNPA first divides the entire network into two subnetworks, then three till *K* subnetworks. During each partition, it is able to shorten the maximum end-to-end latency between centroids their associated nodes and hence the maximum latency is significantly decreased compared with the regular clustering algorithms such as K-center and K-means. The essence of the CNPA is to use the node that is most different from existing nodes as the new center. The effectiveness of this idea has been proved in [44]–[46].

B. Multi-controller placement algorithm

As analyzed in Section III, the latency between controllers and their associated switches is primarily composed of the end-to-end latency and the queuing latency in controllers. In Section IV-A, a network partition strategy is proposed to partition a network into subnetworks and minimize the maximum end-to-end latency between controllers and switches. In this subsection, we will propose solutions to address the queuing latency of controllers.

Since the density of switches in different subnetworks varies due to their geographic distributions, controllers in subnetworks which have dense switches may experience heavier traffic load and hence result in longer queuing latency. This will negatively affect the quality of service and even destroy some delay-sensitive applications, e.g., live streaming video and voice over IP. Therefore, we propose to assign multiple controllers to balance their loads so that the queueing latency will be reduced. Specifically, we first calculate the appropriate number of controllers for each subnetwork and then place them in those subnetworks. A multi-controller selection and placement algorithm is summarized in Algorithm 2.

Algorithm 2: Multi-controller selection and placement algorithm

step 1: Select a subnetwork and calculate all the concurrent packets requesting rate $\lambda = \sum_{i=1}^{n} \lambda_i$.
step 2: Set $m = 1$ and calculate D_{que} by using Eq. 3 and Eq. 4.
step 3: Calculate the maximum total latency $\max\{D_{total}\} = \max\{D_{e2e}\} + D_{que}$.
step 4: Increase m until $\max\{D_{total}\}$ is smaller than T_{th}.
step 5: set $K = m$ and execute CNPA to partition the associated subnetwork into m subnetworks and find their centroids.
step 6: Place m controllers into these centroids.
step 7: Repeat step 1 to 6 until all the subnetworks are placed with appropriate controllers.

The essence of this algorithm is to calculate the required number of controllers by monitoring the maximum total delay ($\max\{D_{total}\}$), and place appropriate controllers to the centroids of those subnetworks. Specifically, the Algorithm 2 first selects a subnetwork and calculates the overall packets arrival rate aggregated by its switches. Afterwards, the number of controllers for a subnetwork, m, is continually increased until the maximum total latency is smaller than the delay threshold T_{th}. Once the appropriate number of controllers is obtained, the CNPA is adopted to further partition and find m clusters for one specific subnetwork. The required controllers are finally placed to the location of those m centroids, and associated switches are allocated to those m clusters. The above steps are repeated until all the subnetworks are placed with appropriate controllers.

V. Performance evaluation

In this section, we evaluate the performance of CNPA and compare it with solutions in the literature under two real topologies from the Internet Topology Zoo [47] that constitutes hundreds of network topologies from network providers. The first topology is the Internet2 OS3E [48], which is widely adopted in the literature to evaluate the controller placement problem. The second one is the ChinaNet, which is the largest network in China [49].

The proposed algorithms are implemented in MATLAB, a powerful multi-paradigm numerical computing environment, which has been widely used in the research

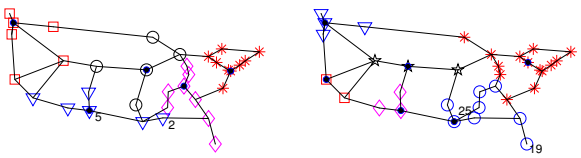
of controller placement of SDN [27] [28]. The simulation includes the following steps. First, the topologies of these two networks are referred from the Topology Zoo, where we can get coordinates of nodes and links among them. Second, distances between any two nodes are calculated by using the Haversine formula [41] [42] and the shortest path distance is calculated by using the Dijkstras algorithm [43]. Third, the end-to-end latency and queuing latency are calculated by Eq. 1 and Eq. 4, respectively. Fourth, the controller placement demonstrations are plotted in the MATLAB and the specific latencies are calculated, processed and stored in *.txt files accordingly.

In order to evaluate the performance of the proposed approach, we compare the proposed approach with two representative solutions in the literature: K-center and K-means. K-center is adopted widely in the literature to reduce the latency between controllers and switches [11] [12]. K-means is the most well-known partition-based clustering algorithm, which aims to minimize the deviations of points in the same cluster [50]. We first evaluate the CNPA in comparison with the current existing solutions and demonstrate how the overall latency is decreased after deploying multiple controllers in each subnetwork.

A. Demonstration of the network partition conducted by standard clustering algorithms

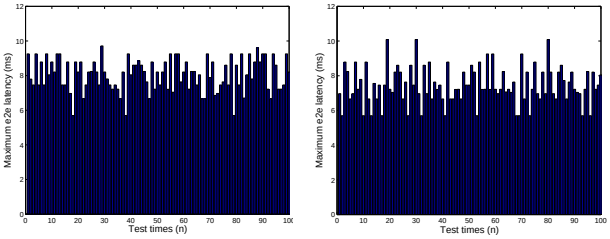
In this subsection, we demonstrate the network partition that is conducted by standard clustering algorithms, e.g., K-means, to illustrate the shortcomings of those algorithms. The initial centers of those two algorithms are randomly selected, so the result of network partition varies for each execution. Therefore, it is quite possible that the maximum end-to-end latency between controllers and their associated switches in the subnetworks increases even when the network is partitioned into more subnetworks, which is not desirable in the network partition of SDN. For example, as depicted in Fig. 4 (a), the OS3E network is partitioned into 5 subnetworks by the K-means. The maximum end-to-end latency appears in the path from node 5 to 2 and the value is 7.459 ms. In contrast, when the network is partitioned into 6 subnetworks, the maximum latency does not decrease but increases to 9.254 ms from node 25 to 19 as depicted in Fig. 4 (b). To reflect the real performance of the K-means, we have conducted 100 times of the K-means algorithm to partition both the OS3E and the ChinaNet topology. Associated results are presented in Fig. 5.

Fig. 5 depicts the maximum end-to-end latency of the Internet2 OS3E which is partitioned by the K-means.



(a) K = 5 (b) K = 6

Fig. 4. Demonstration of the K-means network partition of the Internet2 OS3E



(a) K = 5 (b) K = 6

Fig. 5. The maximum latency of the Internet2 OS3E which is partitioned by the K-means (100 times of partition)

The K-means is executed by 100 times so that we obtain different partitions as well as different maximum end-to-end latencies. When $K = 5$, the maximum end-to-end latency of the subnetwork varies from 6 ms to 10 ms as depicted in Fig. 5 (a). Intuitively, the maximum latency of the subnetwork should be decreased if the network is partitioned into more subnetworks. However, as depicted in Fig. 5 (b), when the network is partitioned into 6 subnetworks, K-means fails to meet this objective, as the maximum latency still fluctuates from 6 ms to 10 ms. The difference between the 5 subnetworks partition and the 6 subnetworks partition is that the probability of maximum end-to-end latency in the 6 subnetworks partition occurs more frequently than the one in the 5 subnetworks partition, while the maximum end-to-end latency remains the same. The same phenomenon are also observed in the ChinaNet network partition which is adopted by the K-means.

B. Demonstration of the CNPA network partition

In this subsection, we demonstrate the network partition which is conducted by the CNPA. Fig. 6 depicts the Internet2 OS3E network which is partitioned by the CNPA step by step. We go over the process and demonstrate how the network is partitioned. The first step is to find the centroid of the entire network. The centroid of the network can be found easily, because no matter where the initial center is selected, the CNPA can ultimately find the unique centroid of the network. As

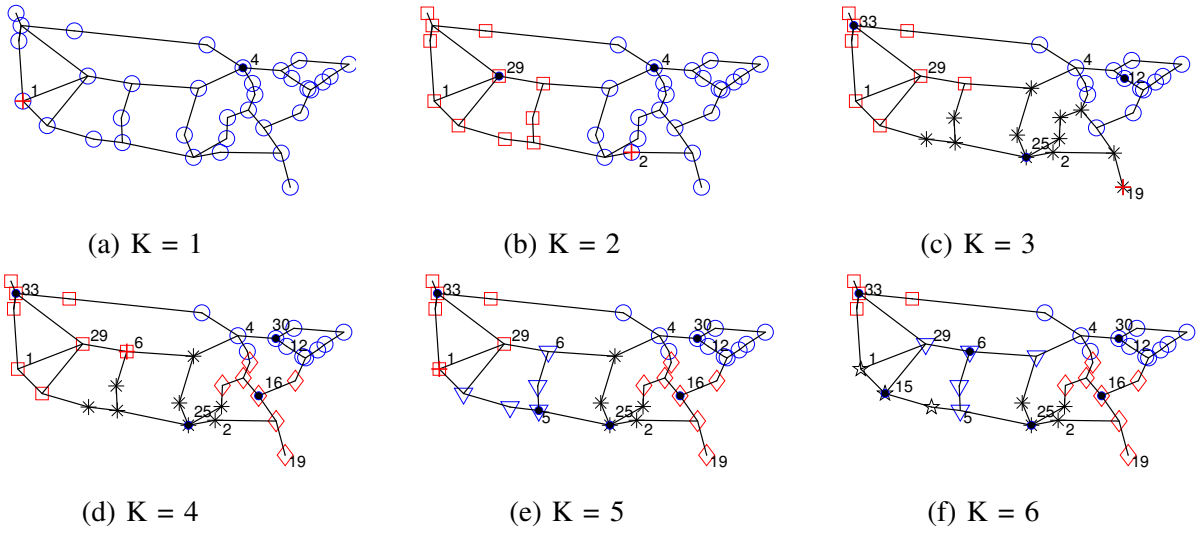


Fig. 6. Demonstration of the CNPA network partition of the Internet2 OS3E

depicted in Fig. 6 (a), the centroid of the network is found at node 4 and the node which has the biggest latency to the centroid is node 1. Therefore, node 1 is selected as the second initial center to perform next partition. Fig. 6 (b) shows the result that the network is partitioned into two subnetworks with the initial centers 4 and 1. Note that although one of the initial centers is selected to node 1, the final centroid found by CNPA is at node 29, because it is the closest node to all of other nodes in the subnetwork. The biggest latency in the two subnetworks is from node 4 to node 2, so the subnetwork depicted by '◯' needs to be further partitioned and the initial centers of the CNPA are selected to node 4, 29 and 2. The rest partitions are depicted in Fig. 6 (c), (d), (e) and (f). The nodes which have the biggest latency to their centroids are found at node 19, 6, 1, respectively, so they are selected as initial centers of the CNPA. The final result of the network partition is depicted in Fig. 6 (f) where the network is partitioned into 6 subnetworks which are distinguished by '○', '∗', '◇', '⋆', etc.

C. Comparison of the end-to-end latency

In this subsection, we evaluate the performance of CNPA in terms of end-to-end latency and compare it with K-means and K-center.

Fig. 7 depicts the maximum end-to-end latency Cumulative Distribution Functions (CDFs) from all possible partitions of the OS3E network. Both the K-means and K-center are executed by 100 times while CNPA is only executed once, as CNPA will always obtain the same result for each execution. The reason is that appropriate initial centers have been calculated and selected in the

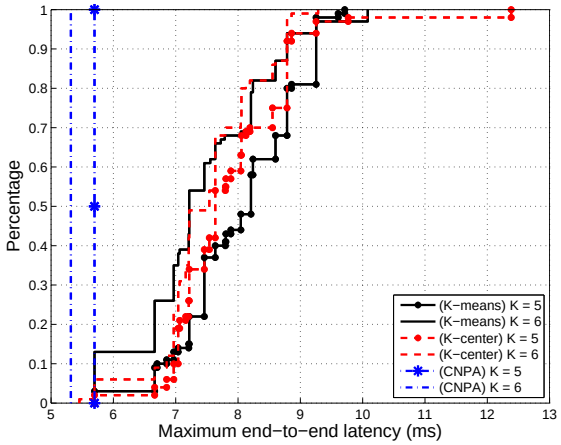


Fig. 7. Maximum end-to-end latency CDFs of the Internet2 OS3E

CNPA and the fixed initial centers will lead to the fixed partition.

The maximum end-to-end latency CDFs of the K-means, K-center and CNPA are depicted in colors of green, red, and blue, respectively. The dashed curve represents the maximum end-to-end latency CDFs where the OS3E is partitioned into 5 subnetworks, and the solid curve represents the results where the network is partitioned into 6 subnetworks. It can be observed that among the 100 times of partitions, there is a slim chance for both K-means and K-center to reach the smallest maximum end-to-end latency, while the CNPA can constantly obtain the smallest one. As depicted by the dashed red curve, when the OS3E is partitioned into 5 subnetworks, K-center even results in over 12 ms latency

between the controllers and their associated switches, which is over 2 times larger than the results achieved by the CNPA. When the OS3E is partitioned into 6 subnetworks, the maximum end-to-end latency achieved by the CNPA is even smaller than the smallest one achieved by both the K-means and K-center, although they are executed by 100 times.

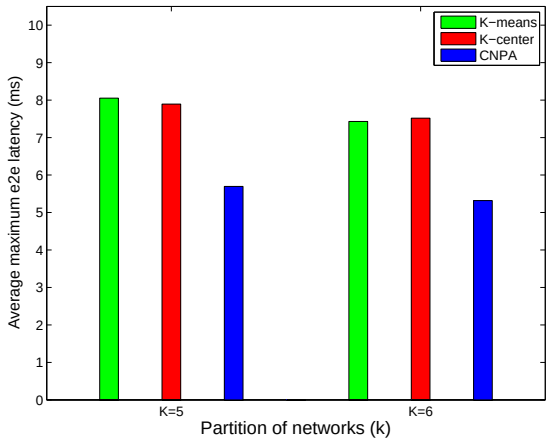


Fig. 8. Average maximum end-to-end latency of the Internet2 OS3E

The average end-to-end latencies of the 100 runs of both the K-means and K-center are further calculated and depicted in Fig. 8. The average maximum end-to-end latencies depicted in Fig. 8 further verify that the CNPA is able to remarkably decrease the latency between controllers and their associated switches in comparison with both the K-means and K-center.

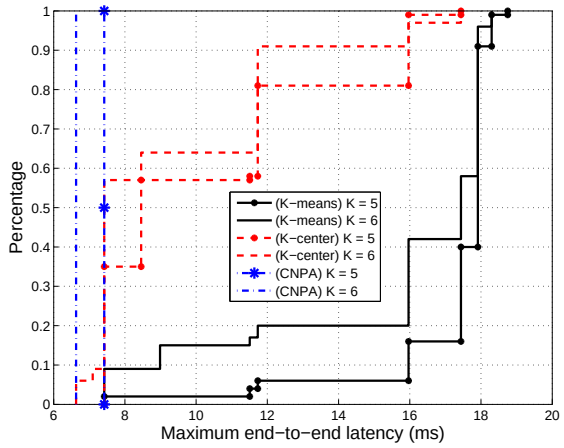


Fig. 9. Maximum end-to-end latency CDFs of the ChinaNet

The performance of CNPA is also evaluated under the ChinaNet topology. Fig. 9 depicts the maximum end-to-end latency CDFs from all possible partitions of the ChinaNet topology. It can be observed that CNPA achieves smaller end-to-end latency between controllers and their associated switches than K-means and K-center, which are executed by 100 times.

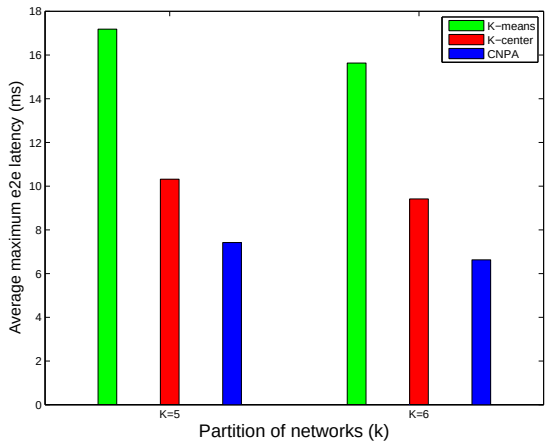


Fig. 10. Average maximum end-to-end latency of the ChinaNet

Fig. 10 further depicts that the CNPA has considerably decreased the average maximum end-to-end latency between controllers and their associated switches in comparison with both the K-means and K-center. Specifically, the average maximum end-to-end latency achieved by the CNPA is 2.312 times and 2.437 times smaller than that achieved by K-means when the network is partitioned into 5 and 6 subnetworks, respectively.

D. Comparison of the total latency

In this subsection, we evaluate how the total latency is decreased when multiple controllers are deployed by CNPA. Note that after deploying multiple controllers by CNPA, the total number of controllers in the network increases accordingly. In order to fairly compare the performance of CNPA, we let K-means and K-center partition the targeting network into more subnetworks, and each subnetwork deploys one controller, so that all the algorithms deploy the same number of controllers in the network. In the following simulations, the service rate of each controller is set to 1Kpps (packets per second) to guarantee that a request can be processed within 1ms [51]. The maximum request rate of switches is set to 0.1Kpps. Considering each switch may generate different requests, a random factor varied from 0.5 to 1 is adopted to indicate the load degree of those switches.

Fig. 11 depicts the total latency CDFs in the OS3E network. As depicted in Fig. 11, there is a slim chance for both K-means and K-center to reach the smallest total latency, while the CNPA can constantly obtain the smallest one. When 8 controllers are deployed in the

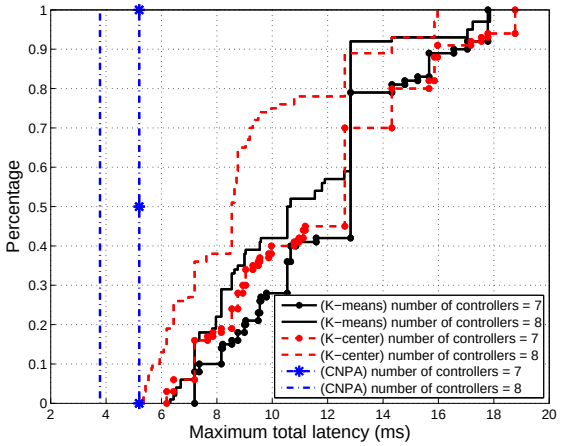


Fig. 11. Total latency CDFs of the Internet2 OS3E

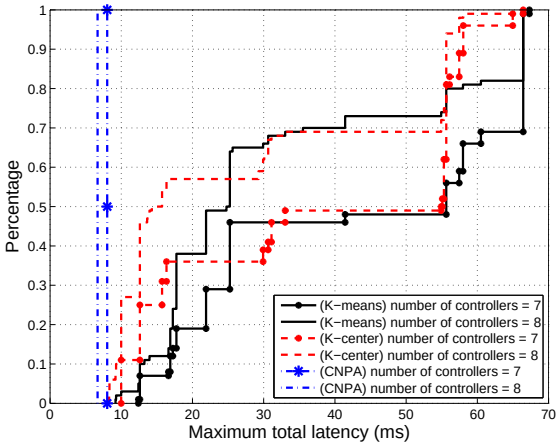


Fig. 13. Total latency CDFs of the ChinaNet

OS3E network, the maximum total latency achieved by the CNPA is still shorter than the smallest one of 100 results from the K-means and K-center.

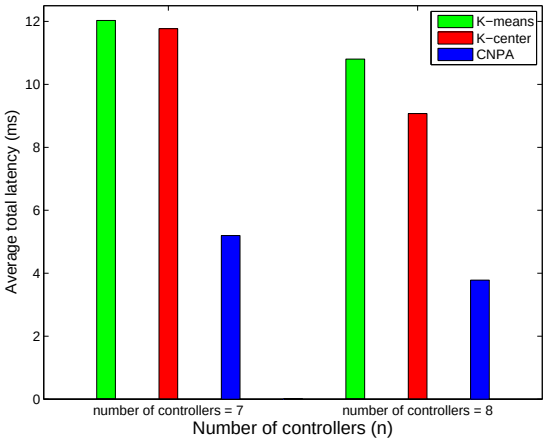


Fig. 12. Average total latency of the Internet2 OS3E

The average total latencies of the 100 runs of K-means and K-center are further calculated and depicted in Fig. 12. When 7 controllers are deployed in the OS3E network, the average total latency achieved by the CNPA is 2.31 and 2.26 times smaller than those achieved by K-means and K-center, respectively. When 8 controllers are deployed in the OS3E network, the average total latency achieved by the CNPA is 2.86 and 2.40 times smaller than those achieved by K-means and K-center, respectively. Those results again verify that the CNPA can significantly decrease the total latency between controllers and their associated switches compared with both of the K-means and K-center.

The performance of CNPA is also evaluated under the ChinaNet topology. Fig. 13 depicts the total latency CDFs from all possible partitions of the ChinaNet topology. It can be observed that CNPA can achieve smaller latency between controllers and their associated switches than K-means and K-center.

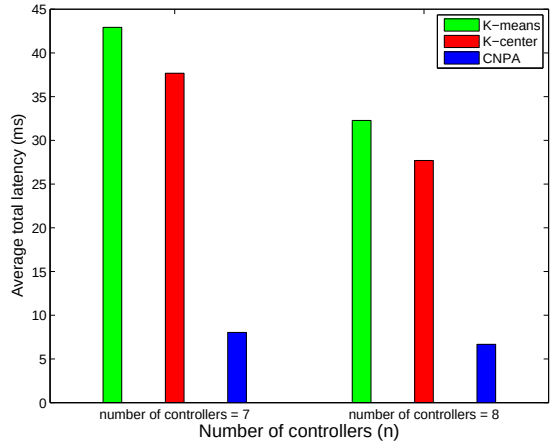


Fig. 14. Average total latency of the ChinaNet

The average total latencies of the 100 runs of the K-means and K-center are further calculated and depicted in Fig. 14. This figure demonstrates that the CNPA outperforms both K-means and K-center in terms of total latency between controllers and their associated switches.

E. Comparison of the switch and controller ratio

It is known that the density of switch distribution in a network is determined by the network topology. In order to shorten the latency between controllers and switches, switches which are closer to each other should be divided into one subnetwork, even if this subnetwork

has a larger number of switches. This may lead to imbalanced switches to controllers. Since we allocate multiple controllers to each subnetwork, the imbalance can be well addressed. In this subsection, we evaluate whether the number of switches and controller in each subnetwork is balanced.

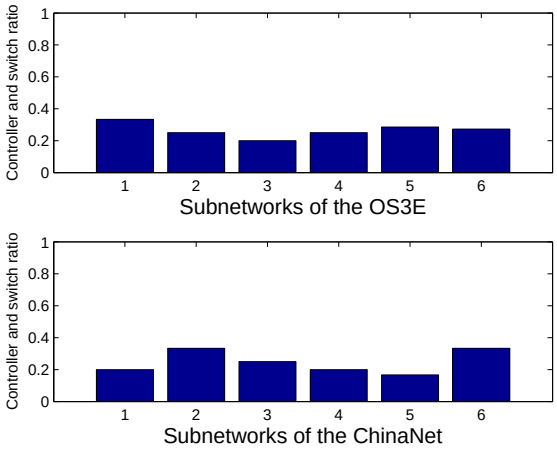


Fig. 15. The controller and switch ratio in each subnetwork

Fig. 15 depicts the controller and switch ratio in each subnetwork of both the OS3E and the ChinaNet. We can find that although the objective of this paper is to shorten the latency between controllers and switches in SDN-enabled wide area networks, the proposed solution can also achieve an acceptable controller and switch ratio.

VI. Conclusion

In this paper, we have investigated new methods to shorten the latency between controllers and their associated switches. First of all, the overall latency between controllers and switches is investigated and formulated qualitatively. Then, a clustering-based network partition algorithm is proposed to partition a network into subnetworks to shorten the end-to-end latency. Since the density of switches in each subnetwork may vary differently due to geographic distributions, we propose to deploy multiple controllers into each subnetwork to decrease the queuing latency resulted by excessive packet requests from switches. In order to evaluate the performance of the proposed algorithm, extensive simulations are conducted under two real topologies from the Internet Topology Zoo. Simulation results verify that CNPA can effectively decrease the maximum end-to-end latency between controllers and their associated switches. When multiple controllers are deployed in subnetworks, the overall latency has also been greatly decreased by CNPA in comparison with K-means and K-center.

References

[1] G. Wang, Y. Zhao, J. Huang, Q. Duan, and J. Li, "A K-means-based Network Partition Algorithm for Controller Placement in Software Defined Network," *International Conference on Communications*, 2016.

[2] 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.

[3] W. Miao, G. Min, Y. Wu, H. Wang, and J. Hu, "Performance Modelling and Analysis of Software-Defined Networking under Bursty Multimedia Traffic," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 12, no. 5s, p. 77, 2016.

[4] H. Huang, H. Yin, G. Min, H. Jiang, J. Zhang, and Y. Wu, "Data-Driven Information Plane in Software-Defined Networking," *IEEE Communications Magazine*, 2017.

[5] D. Levin, M. Canini, S. Schmid, F. Schaffert, A. Feldmann *et al.*, "Panopticon: Reaping the Benefits of Incremental SDN Deployment in Enterprise Networks," *USENIX Annual Technical Conference*, pp. 333–345, 2014.

[6] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.

[7] OpenFlow Switch Specification Version 1.5.1. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf

[8] G. Wang, Y. Zhao, J. Huang, and W. Wang, "The Controller Placement Problem in Software Defined Networking: A Survey," *IEEE Network*, vol. 31, no. 5, pp. 21–27, 2017.

[9] K. Phemius, M. Bouet, and J. Leguay, "DISCO: Distributed Multi-domain SDN Controllers," *arXiv preprint arXiv:1308.6138*, 2013.

[10] 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." *OSDI*, vol. 10, pp. 1–6, 2010.

[11] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," *The first workshop on Hot topics in software defined networks*, pp. 7–12, 2012.

[12] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," *IEEE Communications Letters*, vol. 18, no. 8, pp. 1339–1342, 2014.

[13] S. Khuller and Y. J. Sussmann, "The capacitated k-center problem," *SIAM Journal on Discrete Mathematics*, vol. 13, no. 3, pp. 403–418, 2000.

[14] Y. Hu, T. Luo, W. Wang, and C. Deng, "On the load balanced controller placement problem in Software defined networks," *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pp. 2430–2434, 2016.

[15] L. Han, Z. Li, W. Liu, K. Dai, and W. Qu, "Minimum Control Latency of SDN Controller Placement," *Trustcom/BigDataSE/ISPA, 2016 IEEE*, pp. 2175–2180, 2016.

[16] Y. Zhang, N. Beheshti, and M. Tatipamula, "On resilience of split-architecture networks," *IEEE Global Telecommunications Conference (GLOBECOM)*, pp. 1–6, 2011.

[17] Y.-n. HU, W.-d. WANG, X.-y. GONG, X.-r. QUE, and S.-d. CHENG, "On the placement of controllers in software-defined networks," *The Journal of China Universities of Posts and Telecommunications*, vol. 19, pp. 92–171, 2012.

[18] L. F. Müller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, and M. P. Barcellos, "Survivor: an Enhanced Controller Placement

Strategy for Improving SDN Survivability," *IEEE Global Communications Conference (GLOBECOM)*, 2014.

[19] Y. Jimenez, C. Cervello-Pastor, and A. J. Garcia, "On the controller placement for designing a distributed SDN control layer," *IFIP Networking Conference*, pp. 1–9, 2014.

[20] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "On reliability-optimized controller placement for Software-Defined Networks," *Communications, China*, vol. 11, no. 2, pp. 38–54, 2014.

[21] M. Tanha, D. Sajjadi, and J. Pan, "Enduring Node Failures through Resilient Controller Placement for Software Defined Networks," *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, 2016.

[22] A. Sallahi and M. St-Hilaire, "Optimal Model for the Controller Placement Problem in Software Defined Networks," *IEEE Communications Letters*, vol. 19, no. 1, pp. 30–33, 2015.

[23] H. K. Rath, V. Revoori, S. Nadaf, and A. Simha, "Optimal controller placement in Software Defined Networks (SDN) using a non-zero-sum game," *IEEE 15th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014*, pp. 1–6, 2014.

[24] A. Ruiz-Rivera, K.-W. Chin, and S. Soh, "GreCo: An Energy Aware Controller Association Algorithm for Software Defined Networks," *IEEE Communications Letters*, vol. 19, no. 4, pp. 541–544, 2015.

[25] M. T. I. ul Huque, G. Jourjon, and V. Gramoli, "Revisiting the Controller Placement Problem," *NICTA, Australia. Tech. Rep.*, 2015.

[26] A. Sallahi and M. St-Hilaire, "Expansion Model for the Controller Placement Problem in Software Defined Networks," *IEEE Communications Letters*, vol. 21, no. 2, pp. 274–277, 2017.

[27] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-optimal resilient controller placement in SDN-based core networks," *25th International Teletraffic Congress (ITC)*, pp. 1–9, 2013.

[28] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, "Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 4–17, 2015.

[29] D. Tuncer, M. Charalambides, S. Clayman, and G. Pavlou, "Adaptive resource management and control in software defined networks," *Network and Service Management, IEEE Transactions on*, vol. 12, no. 1, pp. 18–33, 2015.

[30] J. Liao, H. Sun, J. Wang, Q. Qi, K. Li, and T. Li, "Density cluster based approach for controller placement problem in large-scale software defined networkings," *Computer Networks*, vol. 112, pp. 24–35, 2017.

[31] B. Zhang, X. Wang, L. Ma, and M. Huang, "Optimal Controller Placement Problem in Internet-Oriented Software Defined Network," *2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 481–488, 2016.

[32] SDN-Enabled Programmatic Control of the Network. [Online]. Available: http://www.brocade.com/en/backend-content/pdf-page.html?/content/dam/common/documents/content-types/solution-brief/brocade-mlx-service-provider-sb.pdf

[33] "Corsa's DP2100 SDN switching and routing platform." [Online]. Available: http://www.corsa.com/products/dp2100/

[34] R. Daniels and D. Whittaker, "Benchmarking the SDN Switch," in *Open Networking Foundation's SDN Solution Showcase*, 2015. [Online]. Available: https://www.opennetworking.org/images/stories/sdn-solution-showcase/germany2015/Spirent%20-%20Benchmarking%20the%20SDN%20Switch.pdf

[35] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "ONOS: towards an open, distributed SDN OS," *The third workshop on Hot topics in software defined networking*, pp. 1–6, 2014.

[36] J. Xie, D. Guo, Z. Hu, T. Qu, and P. Lv, "Control plane of software defined networks: A survey," *Computer communications*, vol. 67, pp. 1–10, 2015.

[37] D. Kreutz, F. M. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[38] Global ONOS and SDN-IP deployment. [Online]. Available: http://onosproject.org/wp-content/uploads/2015/06/PoC_global-deploy.pdf

[39] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, "Modeling and performance evaluation of an OpenFlow architecture," *The 23rd International Teletraffic Congress*, pp. 1–7, 2011.

[40] V. V. Kalashnikov, *Mathematical methods in queuing theory*. Springer Science & Business Media, 2013, vol. 271.

[41] C. Veness, "Calculate distance and bearing between two Latitude/Longitude points using Haversine formula in JavaScript," *Movable Type Scripts*, 2011.

[42] G. Wang, Y. Zhao, J. Huang, and R. M. Winter, "On the Data Aggregation Point Placement in Smart Meter Networks," *26th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–6, 2017.

[43] S. Skiena, "Dijkstras algorithm," *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley*, pp. 225–227, 1990.

[44] I. Katsavounidis, C.-C. J. Kuo, and Z. Zhang, "A new initialization technique for generalized Lloyd iteration," *IEEE Signal processing letters*, vol. 1, no. 10, pp. 144–146, 1994.

[45] M. E. Celebi, H. A. Kingravi, and P. A. Vela, "A comparative study of efficient initialization methods for the k-means clustering algorithm," *Expert Systems with Applications*, vol. 40, no. 1, pp. 200–210, 2013.

[46] I. Palomares, L. Martinez, and F. Herrera, "A consensus model to detect and manage noncooperative behaviors in large-scale group decision making," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 3, pp. 516–530, 2014.

[47] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[48] *Internet2 Open science, scholarship and services exchange.* [Online]. Available: http://www.internet2.edu/network/ose/

[49] *Chinanet, http://en.chinatelecom.com.cn/products/t20060116_48406.html*.

[50] T. Su and J. G. Dy, "In search of deterministic methods for initializing K-means and Gaussian mixture clustering," *Intelligent Data Analysis*, vol. 11, no. 4, pp. 319–338, 2007.

[51] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," *USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE)*, 2012.