Approximation Algorithms for Controller Placement Problems in Software Defined Networks

Tianshu Li
Guangzhou University
Divine tech. Ltd., co.
Guangzhou, China
Helloleolee@gmail.com

Zhaoquan Gu
Guangzhou University
Guangzhou, China
zqgu@gzhu.edu.cn

Xiao Lin
Tsinghua University
Beijing, China
jackielinxiao@gmail.com

Shudong Li
Guangzhou University
Guangzhou, China
lishudong@gzhu.edu.cn

Qingfeng Tan
Guangzhou University
Guangzhou, China
tqf528@gzhu.edu.cn

***Abstract*—Software Defined Networks (SDNs) have been a new paradigm to separate the network control plane from the data forwarding plane. Controller placement is one fundamental problem which identifies the number of controllers and the placement of these controllers. Time to reach and control each node in the network is denoted as *latency* and no provably-efficient algorithms have been proposed under various latency constraints. In this paper, we initialize the study of controller placement problems in SDNs under two different latency constraints. The first one is to design efficient placement algorithms when the maximum or average latency is bounded. We derive that the controller placement problem becomes NP-hard under such latency constraints and propose approximation algorithms to identify the minimum number of controllers. The other one is balanced cost-latency placement, which seeks to minimize a function considering both the controller cost (setup and maintain) and the latency. We also propose a constant approximation algorithm based on the primal dual approach. We conducted these algorithm on real network topologies and the simulation results validate our theoretical analyses.**

I. Introduction

Software Defined Network (SDN) is a new paradigm that enables great innovations in current networks [4], [18]. By separating the control plane from the data forwarding plane, the network behaviors can be monitored and controlled in a centralized manner, which provides strong capability to implement traditional network polices, such as routing [5], [15] and reliability [10]. A centralized Routing Control System was proposed in [5] to separate the interdomain routing from individual routers and it makes the routing system more manageable and accommodative of various service needs. The SoftRouter architecture was proposed in [15] to disaggregate the routers into packet transmitting elements and the routers are controlled by a centralized control plane with open standardized interfaces. In extant SDN design, there are two main components to compose an SDN:

- **SDN Controllers** can acquire and control the network behaviors, such as determining the forwarding path for each flow in the network. They compose the control plane of the SDN and they are to implement various centralized functions.
- **SDN Forwarding Elements** compose the network data plane, through which each element executes the data forwarding operations after the controllers' decision. It's similar as the traditional routers except that they don't decide where to transmit the packets.

There are many works focusing on the design of controllers or the control plane. The SDN controller designed in [6] can handle 30000 requests from the forwarding elements per second. With parallelism and multi-core system, the controllers can support a larger network under a considerable latency and sufficient bandwidth. Distributed control plane have been designed in [13], [21] to maintain a global network view and synchronize the network states among the controllers. A root controller with a global network view is designed in [7] while the other controllers are left to maintain a local view of a part of the network. However, they don't solve the fundamental problems: where should the controllers be placed and how many controllers are needed.

In order to tackle these problems, we study the **controller placement problem** in this paper, which is firstly proposed in [8]. Given a network of n nodes (here each node represents a data forwarding element, similar as a router), the controller placement problem is to find a subset of nodes to place controllers satisfying some predefined constraints. Note that, the controllers are assumed to be placed at the nodes since it's costly to set up a new position to place a controller. There are several related works about this problem. Reliability-aware controller placement is proposed in [10], which aims at maximizing the reliability of the network. Pareto-based optimal resilient controller placement is studied in [11], which uses different types of resilience and failure tolerance as the measurement. A framework for deploying multiple controllers is proposed in [3], which changes the controllers' placement dynamically with changing network conditions.

Different from these works, we use latency of controlling the network as the main measurement. Several simulation results are shown in [8] to illustrate the relationship between the latency and the number of controllers. However, it does not provide a provably efficient algorithm, but adopts an exhausted searching method to find the optimal placement when the number of controllers is fixed. In this paper, we study the controller placement problem from two different views:

- *Bounded Latency Placement* is to find the minimum number of controllers such that the latency to control the

network is bounded. Both maximum latency and average latency are our considerations.

- *Balanced Cost-Latency Placement* is to find a number of controllers such that the cost to set up and maintain the controllers and the latency to control the network is balanced. We define a trade-off function as Eqn. (1) (refer to Section III) and our goal is to minimize it.

Our contributions of this paper are summarized as follows.

1) We prove that the bounded latency placement problem is NP-hard under both maximum and average latency constraints;
2) When the maximum latency is bounded by a given value, we propose two approximation algorithms to find the lower bound and upper bound of the minimum number of controllers;
3) When the average latency is bounded by a given value, we propose an $\frac{2(1+\epsilon)^2}{\epsilon}$-approximation algorithm for any given ϵ;
4) In order to balance the cost to maintain the controller and the latency to control the network, we minimize the trad-off function by an approximation algorithm which is no more than $1.86OPT$ where OPT is the optimal value.

The remainder of the paper is organized as follows. The next section introduces some related works. Bounded latency placement problem is studied in Section IV, considering both maximum and average latency constraints. In order to balance the cost to maintain the controllers and the latency to control the network, we minimize the trade-off function in Section V. We introduce simulations briefly in Section VI and we conclude our paper in Section VII.

II. Related Works

There have been extensive research efforts on designing the control plane of a SDN to maintain a global view of the whole network. Onix is a platform designed in [13] where the network control plane can be implemented as a distributed system. Onix provides a general API for the control plane implementations and the control plane can operate and decide on a global view of the network. HyperFlow is a distributed event-based control plane for OpenFlow in [21]. HyperFlow provides scalability while maintaining network control centralization. The localized decision is made by individual controllers to minimized the response time to the requests from the data plane, and it's also resilient to several network failures, such as network partitioning or some nodes' failures. Dividing large SDN into domains for distributed control is also studied in [19]. However, these works don't solve the fundamental problems: where to place the controllers and how many controllers are needed.

The controller placement problem is firstly proposed in [8] to answer these questions. There are many important and interesting issues to consider, such as the latency to control the network, the reliability of the network, and fault tolerance. When the number of controllers is fixed, the optimal placement is found through an exhausted search in [8] and the latency (both maximum and average latency) can be computed correspondingly. It shows how the latency changes when the number of controllers varies. Reliability is another important metric and some placement algorithms are proposed in [10] to maximize the network reliability. It also evaluates the impact of the controller number on the reliability of the network. A framework for resilient pareto-based optimal controller placement is proposed in [11] and it reveals that more than 20% of the nodes need to be placed controllers such that the network is tolerant of different types of failures. Controller provisioning in SDN is studied in [3] to minimize the flow setup time and communication overhead. In this paper, we also focus on the latency to control the network. Different from all the extant results, our goal is to design efficient algorithms to identify the controllers with theoretical guarantee.

Traffic engineering is a similar problem which is studied in [1]. By placing controllers in the network, the throughput of the network is improved and the convenient traffic engineering is enabled. Facility location problem is also similar as the controller placement problem [9], [17]. The first approximation algorithm for the facility location problem is proposed in [9], which has an $log(n)$ approximation ratio (here n is the number of facilities). The first constant approximation algorithm is shown in [20], which is 3.16 more than the optimal value. An improved result achieves 1.86 approximation ratio by primal dual approach [22] and we adopt this algorithm as a cornerstone to implement a better approximation algorithm as described in Theorem 4 (please refer to Section V).

III. Preliminaries

A. System Model

Considering a network $N = (V, E, w)$ where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, $E = \{e_1, e_2, \ldots, e_m\}$ is the set of edges. For each edge $e \in E$, the weight $w(e)$ represents the propagation latency caused by a message transmitting along the edge. In traditional design, each node in the network decides *where* and *how* to forward packets. In contrast, a more recent trend is to move the control-plane logic to a set of dedicated controllers, and the other nodes only perform the actual packet forwarding. Define the latency between any two nodes v_i, v_j (denote as $l(v_i, v_j)$) as the sum of weights of the edges along the *shortest weighted path* between them. Specially, we set $l(v_i, v_j) = 0$ if $v_i = v_j$. We adopt this definition since many practical protocols are designed to transmit packets through the shortest path, such as the Routing Information Protocol (RIP) and the Open Shortest Path First (OSPF) [14].

In this paper, we focus on selecting a set of nodes $F \subseteq V$ to place controllers, and they form a distributed control plane to control the whole network. In order to keep track of the network's states for further control, each node in V should be connected to at least one controller in F. For simplicity, we consider the situation that each node $v_i \in V$ is connected to one controller $v_j \in F$. (For fault tolerance, one node can be connected to many controllers and this is our future step

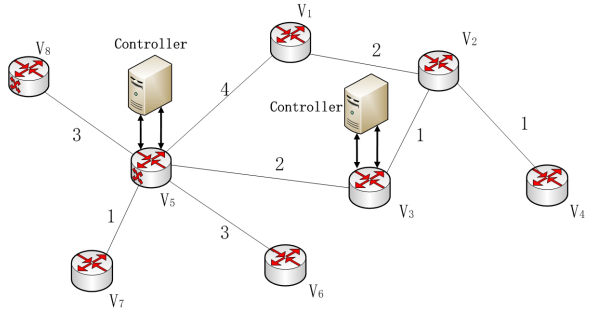

Fig. 1. An example of placement problem

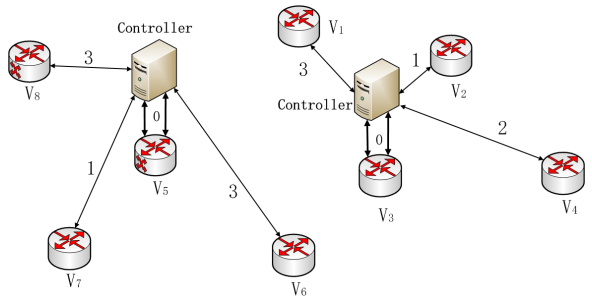

Fig. 2. An example of connection map between the controllers and the nodes

to design robust control system.) We formulate the placement problem as:

Definition 3.1: **Placement Problem** is to find a subset $F \subseteq V$ to place controllers and to design a connection map between V and F such that each node $v_i \in V$ is connected to node $v_j \in F$.

In this paper, we consider the latency to control the network as the measurement since it reflects the time to response the requests from the data plane and the efficiency to acquire the network's states. We define the latency to control each node $v_i \in V$ from the controller set F as:

Definition 3.2: The latency to control node $v_i \in V$ from the set of controllers F is $l_F(v_i) = \min_{v_j \in F} l(v_i, v_j)$.

Obviously, v_i is connected to the controller $v_j \in F$ such that $l(v_i, v_j) = l_F(v_i)$. As depicted in Fig. 1, the network consists of 8 nodes $V = \{v_1, v_2, \ldots, v_8\}$ and 8 edges with corresponding latency. $F = \{v_3, v_5\}$ are selected to place controllers and the nodes in V are connected to them as Fig. 2. As shown in the figure, the latency between v_3 and the controller is 0 since it is where the controller is placed; v_4 is connected to the controller at v_3 and the latency is $l(v_4, v_2) + l(v_2, v_3) = 2$; v_1 is connected to the controller at v_3 since the latency is 3, which is smaller than the latency to be connected to the controller at v_4.

B. Bounded Latency Placement

In designing an efficient SDN, the latency to control the nodes in the network should satisfy some requirements. In this paper, we use two common metrics as the measurement.

1) *Maximum Latency* is the worst situation to control the nodes in the network, which is formulated as $L_m = \max_{v_i \in V} l_F(v_i)$.
2) *Average Latency* is the average latency to control all nodes, which is formulated as $L_a = \frac{1}{|V|} \sum_{v_i \in V} l_F(v_i)$;

In Fig. 2, the maximum latency and the average latency are

$$L_m = \max_{v_i \in V}\{l_F(v_i)\} = \max\{3, 1, 2, 3, 1, 3\} = 3$$

$$L_a = \tfrac{1}{6} \sum_{v_i \in V} l_F(v_i) = \tfrac{1}{6}(3 + 1 + 2 + 3 + 1 + 3) = 2.167.$$

We define the controller placement problem under both maximum latency and average latency requirements.

Problem 1: Bounded Maximum Latency Placement (BMLP) is to find a subset $F \subseteq V$ of minimum number of controllers such that the maximum latency is bounded by a given value B_1 (i.e. $L_m \leq B_1$).

Problem 2: Bounded Average Latency Placement (BALP) is to find a subset $F \subseteq V$ of minimum number of controllers such that the average latency is bounded by a given value B_2 (i.e. $L_a \leq B_2$).

BMLP is to find the minimum number of controllers such that the maximum latency to control each node can be guaranteed in B_1 time, which implies that the controllers can acquire the network's states in B_1 time and all the nodes can receive the instruction from the controllers in a guaranteed latency. BALP is to find the minimum number of controllers such that the time to control all nodes can be bounded in B_2 in an average sense.

C. Balanced Cost-Latency Placement

Both BMLP and BALP aim at identifying the controllers when the latency requirements are fixed. In practical SDN design, placing a controller is costly and we need to consider both cost and latency requirements. Without the cost requirement, we can place a controller at each node, which implies all nodes are "self-controlled" and the latency to control each node is only 0. However, it's impractical since it's costly to place controllers. Therefore, the placement problem should consider both latency and cost requirements at the same time. In this paper, we denote the cost to setup and maintain a controller at $v_i \in V$ as a constant value $c(v_i)$. Since we can't minimize the latency to control the network and minimize the cost of controllers simultaneously, we introduce a trade-off factor λ ($\lambda > 0$) to balance the two conditions. We formulate the problem as:

Problem 3: Balanced Cost-Latency Placement (BCLP) is to find a subset $F \subseteq V$ such that the trade-off function of the cost and the latency

$$C(F, V, \lambda) = \lambda \sum_{v_i \in F} c(v_i) + \sum_{v_j \in V} l_F(v_j) \tag{1}$$

is minimized, where $\lambda > 0$ is a given constant.

IV. Bounded Latency Placement

In this section, we propose algorithms for both BMLP and BALP problems. When the maximum latency $L_m \leq B_1$, we

show the NP-hardness of the BMLP problem and propose two approximate algorithms to compute the lower bound and upper bound of the minimum number of controllers. When the average latency $L_a \leq B_2$, the BALP problem is also NP-hard and we present an $(1+\epsilon)$-approximation algorithm to guarantee $L_a \leq 2(1+1/\epsilon)B_2$, where ϵ is a small positive constant. Then we can modify it to an $(\frac{2(1+\epsilon)^2}{\epsilon})$-approximation algorithm which accords with the latency constraint ($L_a \leq B_2$).

A. Bounded Maximum Latency Placement

When the maximum latency is bounded by B_1, any node in the network can access the nearest controller in no more than B_1 time and it ensures efficient communication speed between the nodes and the controllers. We show that the BMLP problem is NP-hard and we propose approximation algorithms to compute the lower bound and upper bound.

Theorem 1: The BMLP problem is NP-Hard.

Proof: In the network $G = (V, E, w)$, each edge $e = (v_i, v_j)$ has a weight $w(e) = l(v_i, v_j)$, where $l(v_i, v_j)$ is the latency between node v_i and v_j. We construct a complete graph $G' = (V, E', w')$ based on G where $|E| = m = \frac{|V|(|V|-1)}{2}$, and each edge $e = (v_i, v_j)$ has a weight $w'(e) = l(v_i, v_j)$, where $l(v_i, v_j)$ is the latency between node v_i and v_j in G. Sort the edges in an increasing order of their weights as $w'(e_1) \leq w'(e_2) \leq \ldots \leq w'(e_m)$. We construct a series of graphs: $G_1, G_2, ..., G_m$, where $G_i = (V, E_i)$,$E_i = \{e_1, ..., e_i\}$. Denote the set of graphs $GS_k = \{G_1, G_2, .., G_k\}$ such that $w'(e_k) \leq B_1, w'(e_{k+1}) > B_1$. As depicted in Fig. 3, G contains 5 nodes and the latency on each edge is illustrated in the figure (the latency on the blue edge is $1+\beta$ and the latency on the black edge is 2, here $0 < \beta < 1$), G' is the constructed complete graph. When $B_1 = 1+\beta$, $k = 5$ and we construct G_5 correspondingly.

Consider an instance of dominating set DS_i of graph $G_i \in GS_k$, we have $\forall v_1 \in DS_i, v_2 \notin DS_i, l(v_1, v_2) \leq B_1$. Denote the union of all the dominating sets in all the graphs of GS_k as U. Consider a set of nodes $F \subseteq V$, if $F \bigcap (V - U) \neq \emptyset$, by the definition of dominating set and the construction of G_i, $\exists e \in G, v_1 \in F \bigcap (V-U), v_2 \in V \setminus F$ such that $w(e) = c(v_1, v_2) \geq B_1$. So the locations of the controllers must be a subset of U. Denote the minimum dominating set of G_i as DS_{im} and let $|DS_{jm}| = \min\{|DS_{1m}|, |DS_{2m}|, ..., |DS_{km}|\}$, we have $\forall v_1 \in DS_{jm}, v_2 \in (V \setminus DS_{jm}) \bigcap DS_{jm}, c(v_1, v_2) \leq B_1$. Therefore, an instance of the minimum dominating set is also an instance of a minimum set of controllers. So the BMLP problem is NP-hard. ■

Through the proof, the BMLP problem is similar as finding the minimum dominating set, which is NP-hard. We derive both lower bound and upper bound of the number of controllers. Before that, we introduce the square graph G^2.

Definition 4.1: G^2 is the square of graph G by adding edges between nodes whose distance is no more than 2 in G.

Note that the distance between two nodes is different from the latency. If two nodes are connected directly, their distance is 1. Fig. 3 is an illustration of the construction of G_5^2 based on G_5 (G_5 is defined in the proof of Theorem 1).

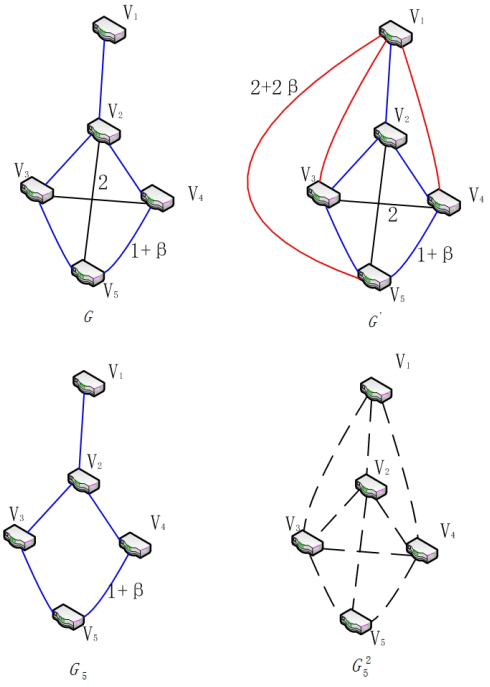

Fig. 3. An illustration of the construction of G', G^2. $B_1 = 1+\beta$ and $\beta \in (0, 1)$. G_5 is defined as in Theorem 1.

Lemma 4.1: Denote a dominating set of G as $D(G)$ and an independent set of G^2 as $I(G^2)|$, $|D(G)| \geq |I(G^2)$.

Proof: Suppose G has a minimum dominating set D^*, then the graph contains $|D^*|$ stars which consists of a single node from D and adjacent nodes to it. By the construction of G^2, each star can be considered as a clique where each node inside it is adjacent to each other. When choosing nodes to compose an independent set, at most one node is chosen from each clique. Therefore, the number of elements in the independent set ($I(G^2)$) is bounded by the number of cliques which is the same as the size of D^*. Thus $|D(G)| \geq |D^*| \geq |I(G^2)|$. ■

Note that, $I(G^2)$ doesn't have to be the maximum independent set and $D(G)$ doesn't have to be the minimum dominating set. A maximal independent $I(G^2)$ which can be computed in polynomial time is a lower bound of $D(G)$. In Alg. 1, we describe the method to find a lower bound. We sort the edges in the graph by non-decreasing order of the weight as $\{e'_1, e'_2, \ldots, e'_m\}$. A series of graphs are constructed correspondingly, where $G_i = (V, E_i)$ and $E_i = \{e'_1, e'_2, \ldots, e'_i\}$. After the construction, we find the maximum value i such that $w(e'_i) \leq B_1 = L_m$ and we apply the maximal independent set algorithm on G_i^2 [2]. From Lemma 4.1, $I(G_i^2)$ is a lower bound of the minimum number of controllers under the maximum latency constraint.

In order to derive an upper bound, we transform the problem to find the minimum dominating set of G_i into a set cover problem as follows. For each node v_i in the network $G = (V, E)$, denote the degree as d_i and the neighbors as $N_i = \{v_i^1, v_i^2, \ldots, v_i^{d_i}\}$. Therefore, we can get a series of neighbors'

Algorithm 1 Lower Bound Algorithm for BMLP

1: $m = |E|$;
2: Sort the edges in a non-decreasing order of their weights as: $\{e'_1, e'_2, ..., e'_m\}$;
3: Construct a series of graphs: $G_1, G_2, ..., G_{|E|}$ where graph $G_i = (V, E_i)$ where $E_i = \{e'_1, e'_2, ..., e'_i\}$;
4: Find the maximum i such that $w(e'_i) \leq B_1$;
5: Find a Maximal Independent Set in G_i^2 and denote it as $I(G_i^2)$;

Algorithm 2 Upper Bound Algorithm for BMLP

1: $\Delta = \max_{v_i \in V} d_i$, $C = \emptyset$;
2: **if** $\log n > \Delta + 1$ **then**
3: Relax the integer program to a linear program and denote the solution as $\overrightarrow{x}$;
4: Choose nodes v_i to C if $x_i \geq \frac{1}{\Delta}$;
5: **else**
6: **while** C is not a dominating set **do**
7: For each node $v_i \in V \setminus C$, define the efficiency of v_i as $eff_i = \frac{1}{|N_i \setminus C|}$;
8: Choose v_i with the smallest efficiency value and let $C = C \cup \{v_i\}$;
9: **end while**
10: **end if**

sets $\{N_1, N_2, .., N_n\}$. The minimum collection of sets N_i that covers all the nodes in the network corresponds to a collection of nodes in V that forms the minimum dominating set. We formulate the set cover problem as an integer programming:

$$
\begin{aligned}
& minimize : \sum_{N_i} x_i \\
s.t. \quad & \forall v \in V, \sum_{N_i : v \in N_i} x_i \geq 1 \quad (1) \\
& \forall N_i, x_i \in \{0, 1\} \quad (2)
\end{aligned}
$$

where x_i is a variable indicating whether N_i is selected (if N_i is selected, $x_i = 1$; otherwise $x_i = 0$). For any node $v \in V$, the first constraint implies that it is covered by at least one chosen set. We present an approximation algorithm to find an upper bound based on the integer programming in Alg. 2.

Theorem 2: Alg. 2 guarantees an approximation ratio of $min\{log(n), \Delta + 1\}$, where Δ is the maximum degree of a node in the network.

Proof: We show the approximation ratio of Alg. 2 from two situations.

Case 1: When $\log n > \Delta + 1$, Alg. 2 is based on the LP rounding technique and we show that the approximation ratio is Δ. For any node $v_i \in V$, it appears in at most Δ sets, thus there must exists one set N_j with $x \geq 1/\Delta$ and N_j is chosen to cover v_i. So the algorithm gives a feasible solution. For each set N_i, x_i is augmented by at most a factor of Δ. So the approximation ratio is Δ.

Case 2: When $\log n \leq \Delta + 1$, the controllers (sets) are chosen in at most n iterations. In each iteration, the remaining nodes can be covered by at most OPT sets (we use OPT to be the minimum number of controller satisfying the maximum latency constraint). Then there must exist one node in the remaining nodes which has an efficiency at least $\frac{OPT}{|N_i \setminus C|}$ (the efficiency means the number of sets to cover each node). For the k-th iteration of Alg. 2, there are $n - k + 1$ nodes left. Since the algorithm chooses the node with lowest efficiency, the needed number of sets to cover each remaining node is at most $\frac{OPT}{n-k+1}$. By adding up all the efficiency to choose the nodes, we have the total number of selected sets is at most $H_n \cdot OPT = (1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n})OPT$. So we have the approximation ratio is $H_n = 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n} = O(log(n))$

Combining the two cases, the theorem holds. ■

Based on the computed lower bound and upper bound from Alg. 1-2, we can improve the upper bound through a binary search. Denote the lower bound and upper bound as n_l, n_u respectively, supposing there are n_u controllers, we plug it into Alg. 1 (Line 4 where $B_1 = n_u$) to construct G_{n_u} and find a maximal independent set as Line 5. If the size is larger than B_1, it implies the upper bound can be further improved and we let $n_u = (n_u + n_l)/2$ to continue this process.

B. Bounded Average Latency Placement

Different from the BMLP problem, we minimize the number of controllers when the average latency is bounded by a given value B_2. We formulate this problem as an integer programming as:

$$
\begin{aligned}
& minimize : \sum_{i \in V} y_i \\
s.t. \quad & \frac{1}{n} \sum_{v_j \in V, v_i \in V} l(v_i, v_j) \cdot x_{ij} \leq B_2 \quad (1) \\
& \forall v_j : \sum_{v_i \in V} x_{ij} = 1 \quad (2) \\
& \forall v_i, v_j : x_{ij} \leq y_i \quad (3) \\
& \forall v_i, v_j : x_{ij}, y_i \in \{0, 1\} \quad (4)
\end{aligned}
$$

where y_i is a variable indicating whether node v_i is selected to place a controller (if it's selected, $y_i = 1$; otherwise $y_i = 0$), x_{ij} is a variable indicating whether node v_j is connected to the controller at v_i through a shortest path (if it's connected, $x_{ij} = 1$, otherwise $x_{ij} = 0$). The first constraint indicates that the average latency is bounded by a given value B_2. Constraint (2) means each node is connected to one controller through a (minimum weighted) path. Constraint (3) guarantees that if $x_{ij} = 1$, $y_i = 1$. Obviously, the BALP can be reduced by the set cover problem and it is also NP-hard. Due to the page limits, we omit the details. We present our algorithm for the BALP problem based on the LP relaxation as Alg. 3.

Alg. 3 solves the linear programming relaxing of the problem and a feasible solution is denoted as $\overrightarrow{x}$ and $\overrightarrow{y}$ (denote $\bar{x}_{ij}, \bar{y}_i$ as the values). Based on the solution, each node v_i has a corresponding value which is computed as Line 3. Then two sets $V_i, \overline{V}_i$ are constructed in Line 4-5 (ϵ is a small constant). The algorithm sorts the sets $\{\overline{V}_1, \overline{V}_2, \ldots, \overline{V}_n\}$ in non-decreasing order of C_i value and constructs the set

Algorithm 3 Approximation Algorithm for BALP

1: $F = \emptyset$;
2: Solve the linear programming relaxation of the problem, denote the solution as $\overrightarrow{x}$ and $\overrightarrow{y}$;
3: For each node v_i, compute $C_i = \sum_{j \in V} \frac{1}{n} l(v_i, v_j) \cdot \bar{x}_{ij}$;
4: For each node v_i, construct set $V_i = \{v_j | \frac{1}{n} l(v_i, v_j) \leq (1 + 1/\epsilon) C_i\}$;
5: For each node v_i, construct set $\overline{V}_i = \{v_j | v_j \in V_i \text{ or } (V_i \cap V_j \neq \emptyset, v_i, v_j \notin V_i \cap V_j)\}$;
6: Sort $\{\overline{V}_1, \overline{V}_2, \ldots, \overline{V}_n\}$ in non-decreasing order of C_i;
7: Choose set $\overline{V}_i$ with the smallest C_i value and $F = F \cup \{v_i\}$. Then delete sets $\overline{V}_j$ for all $j \in \overline{V}_i$. Repeat this step until no set C_i remains;
8: Output set F as the nodes to place controllers;

F iteratively. From the definition of $\overline{V}_i$, we introduce an important property.

Property 4.1: If $v_j \in \overline{V}_i$, $v_i \in \overline{V}_j$.

It's easy to verify that $\forall v_i, v_j \in F$, $V_i \bigcap V_j = \emptyset$. We show that Alg. 3 has an approximation ratio of $1+\epsilon$ while exceeding the latency constraint by at most $2(1 + 1/\epsilon)$. Before that, we present several important lemmas.

Lemma 4.2: For each set $\overline{V}_j$ deleted in Line 7, suppose $\overline{V}_i$ is the chosen set when $\overline{V}_j$ is deleted, then $C_i \leq C_j$.

Proof: Assuming that $C_i > C_j$, then $\overline{V}_j$ is selected by the algorithm before $\overline{V}_i$. As shown above, it leads to $i \in \overline{V}_j$. Thus $\overline{V}_i$ would have been deleted, which is a contradiction. Otherwise, consider the time just before $\overline{V}_i$ is selected, if $\overline{V}_j$ is not deleted, this implies that $\overline{V}_i$ is not the set selected later, which is a contradiction. So we have $C_i \leq C_j$. ∎

Lemma 4.3: For each node $v_i \in F$ and for any node $v_j \in \overline{V}_i$, $l(v_j, v_i) \leq 2(1 + 1/\epsilon) C_j$;

Proof: Since node $v_j \in \overline{V}_i$, there are three situations:

1) If $v_j = v_i$, then $l(v_j, v_i) = 0$;
2) If $v_j \in V_i$, which implies $l(v_i, v_j) \leq (1 + 1/\epsilon) C_i$, then $l(v_j, v_i) = l(v_i, v_j) \leq (1 + 1/\epsilon) C_j$.
3) If $v_j \notin V_i$, from $V_i \cap V_j \neq \emptyset$ in Line 5, there exists node $v_{j'}$ such that $v_{j'} \in V_i \cap V_j$. Then $l(v_i, v_{j'}) \leq (1+1/\epsilon) C_i$ and $(v_j, v_{j'}) \leq (1+1/\epsilon) C_j$. From the triangle inequality, we have $l(v_j, v_i) \leq l(v_j, v_{j'}) + l(v_{j'}, v_i) \leq (1+1/\epsilon) C_j + (1 + 1/\epsilon) C_i \leq 2(1 + 1/\epsilon) C_j$.

Therefore, the lemma can be derived. ∎

Lemma 4.4: For each node $v_i \in V$, $\sum_{j \in V_i} \bar{y}_j \geq \sum_{j \in V_i} \bar{x}_{ij} \geq \frac{1}{1+\epsilon}$

Proof: If $\sum_{j \in V_i} \bar{x}_{ij} \leq \frac{1}{1+\epsilon}$, we have

$$\begin{aligned} C_i &= \sum_{j \in V} l(v_i, v_j) \bar{x}_{ij} \geq \sum_{j \notin V_i} l(v_i, v_j) \bar{x}_{ij} \\ &> (1 + \epsilon) C_i \sum_{j \notin V_i} \bar{x}_{ij} \\ &\geq (1 + \epsilon) C_i (1 - \frac{\epsilon}{1 + \epsilon}) = C_i \end{aligned}$$

which is a contradiction, thus $\sum_{j \in V_i} \bar{x}_{ij} \geq \frac{1}{1+\epsilon}$. ∎

Combining Lemma 4.2-4.4, we conclude the approximation of the algorithm as:

Theorem 3: Alg. 3 has an approximation ratio of $1+\epsilon$ which exceeds the average latency by at most $2(1 + 1/\epsilon)$.

Proof: For any two nodes $v_i, v_j \in F$, the corresponding sets $V_i \cap V_j = \emptyset$, and the number of nodes in F is bounded by $\frac{OPT}{\frac{1}{1+\epsilon}} = (1 + \epsilon) OPT$ (Lemma 4.4). From Lemma 4.3, the latency is exceeded by at most a factor of $2(1 + 1/\epsilon)$. ∎

Remark 4.1: From Theorem 3, we can modify the first constraint in the integer programming as $\sum_{j \in V, i \in V} l(v_i, v_j) \cdot x_{ij} \leq \frac{B_2}{2(1+1/\epsilon)}$, then the algorithm satisfies the average latency constraint with an approximation ratio $\frac{2(1+\epsilon)^2}{\epsilon}$.

Remark 4.2: Alg. 1-3 can terminate in polynomial time and we omit the details to derive the time complexity.

V. Balanced Cost-Latency Placement

Section IV presents the methods to find the minimum number of controllers under both maximum and average latency constraints. In real scenarios, there are more issues to consider, such as the energy consumption and the cost to place a controller. In this section, we take both the latency to control the network and the cost to setup and maintain the controllers into consideration. We introduce a trade-off function to balance the two objectives: minimize the latency to control the network and minimize the cost of the controllers.

A. Approximation Algorithm Description

We formulate the BCLP problem as an integer programming. There are several approaches to get an approximation solution for this problem. The techniques include LP rounding and Primal-Dual approaches. We choose the Primal-Dual approach because it does not require to solve the integer programming which might be time consuming. The BCLP problem (see definition in Problem 3) is formulated as:

$$Minimize: \sum_{v_i, v_j \in V} c(v_i, v_j) \cdot x_{ij} + \lambda \sum_{v_i \in V} c(v_i) \cdot y_i$$

$$s.t. \quad \forall v_j \in V : \sum_{i \in V} x_{ij} \geq 1 \tag{1}$$

$$\forall v_j \in V, v_i \in V : x_{ij} \leq y_i \tag{2}$$

$$\forall v_j \in V, v_i \in V : x_{ij}, y_i \in \{0, 1\} \tag{3}$$

The first constraint indicates that the node is connected to at least one controller; constraint (2) means that if v_j is connected to v_i, v_i has to be chosen to place a controller. The relaxation of the integer programming is:

Primal $$Minimize: \sum_{v_j \in V, v_i \in V} c(v_i, v_j) \cdot x_{ij} + \lambda \sum_{v_i \in V} c(v_i) \cdot y_i$$

$$s.t. \quad \forall v_j \in V, \sum_{v_i \in V} x_{ij} \geq 1 \tag{1}$$

$$\forall v_j \in V, v_i \in V : x_{ij} \leq y_i \tag{2}$$

$$\forall v_j \in V, v_i \in V : x_{ij}, y_i \geq 0 \tag{3}$$

The dual formulation of the relaxation is:

$$\textbf{Dual} \quad Maximize: \sum_{j \in V} \alpha_j$$

$$s.t. \quad \forall v_j \in V, v_i \in V : \alpha_j - \beta_{ij} \leq l(v_i, v_j) \quad (1)$$

$$\forall v_i \in V : \sum_{j \in V} \beta_{ij} \leq \lambda c(v_i) \quad (2)$$

$$\forall v_j \in V, v_i \in V : \alpha_j, \beta_{ij} \geq 0 \quad (3)$$

The approximation algorithm for the BCLP problem based on the primal-dual technique is described in Alg. 4 (Alg. 4 terminates in a polynomial time and we omit the details).

Algorithm 4 Approximation Algorithm for BCLP

$V = \{v_1, v_2, \ldots, v_n\}$, $F = \emptyset$;
$\forall v_j \in V$, $\alpha_j = 0$, $e_{1j} = false, e_{2j} = false$
while $e_{1j} = false$ and $e_{2j} = false$ for node $v_j \in V$ **do**
  for Each node $v_i \in V \setminus F$ **do**
    if $\sum_{v_j \in R} \max(\alpha_j - l(v_j, v_i), 0) \geq \lambda c(v_i)$ **then**
      $F = F \cup \{v_i\}$, $V = V \setminus \{v_i\}$, $e_{1i} = true$;
      For each node $v_j \in V$, if $\alpha_j \geq l(v_j, v_i)$, connect v_j to v_i and remove it from V as $V = V \setminus \{v_j\}$, $e_{1j} = true$;
    end if
  end for
  for Each node $v_i \in F$ **do**
    For each node $v_j \in V$, if $\alpha_j \geq l(v_j, v_i)$, connect v_j to v_i and remove it from V as $V = V \setminus \{v_j\}$, $e_{2j} = true$;
  end for
  $\alpha_j = \alpha_j + 1$;
end while

B. Correctness and Efficiency

When the algorithm terminates, order the α_i value for all $v_i \in V$. Without loss of generality, we assume that $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_n$. To show the correctness and efficiency, we introduce several important lemmas.

Lemma 5.1: For two nodes $v_j, v_{j'}$, and a controller placed at v_i, $\alpha_j \leq \alpha_{j'} + l(v_j, v_i) + l(v_{j'}, v_i)$.

Proof: Note that if v_j and $v_{j'}$ are selected to place controllers, the lemma should also hold. When $\alpha_{j'} \geq \alpha_j$, the inequality holds. We consider the situation when $\alpha_{j'} \leq \alpha_j$.

1) Both $v_{j'}$ and v_j are selected to place controllers: since $\alpha_{j'} \leq \alpha_j$, when $v_{j'}$ is selected, node v_j does not satisfy the constraint $\alpha_j \geq l(v_j, v_{j'})$. By triangle inequality, $l(v_j, v_{j'}) \leq l(v_j, v_i) + l(v_i, v_{j'})$.

2) Neither $v_{j'}$ nor v_j is selected to place controllers: supposing $v_{j'}$ is connected to the controller at node $v_{i'}$ (here $v_{i'}$ can be v_i), if v_j is also connected to the controller at $v_{i'}$, $l(v_j, v_{i'}) \leq \alpha_j \leq l(v_j, v_i)$ should hold, otherwise v_j would have been connected to v_i instead. So the lemma holds. For the other case v_j is not connected to the controller at $v_{i'}$, we have $\alpha_j \leq l(v_j, v_{i'})$, otherwise v_j will be connected to the controller at $v_{i'}$ before α_j increasing from 0 to its current value. Thus we have $\alpha_j \leq l(v_j, v_{i'})$. By triangle inequality, $\alpha_j \leq l(v_j, v_{i'}) \leq l(v_{j'}, v_{i'}) + l(v_{j'}, v_i) + l(v_j, v_i) \leq \alpha_{j'} + l(v_j, v_i) + l(v_{j'}, v_i)$.

3) Only one node is selected to place a controller: without loss of generality, supposing v_j is the selected node to place a controller and $v_{j'}$ is connected to a controller at $v_{i'}$. If $v_{i'} = v_i$, $\alpha_{j'} \leq \alpha_j$, and thus $\alpha_{i'} = \alpha_i \leq \alpha_j$, which implies when v_i is selected, v_j is not connected to node v_i. Therefore we have $\alpha_j \leq l(v_j, v_i)$ and the lemma holds. If $v_{i'} \neq v_i$, $\alpha_{j'} \leq \alpha_j$, and $\alpha_{i'} \leq \alpha_{j'}$ since $v_{j'}$ is connected to $v_{i'}$, thus $\alpha_{i'} \leq \alpha_j$. Similarly we have $\alpha_j \leq l(v_j, v_{i'}) \leq l(v_{j'}, v_{i'}) + l(v_{j'}, v_i) + l(v_j, v_i) \leq \alpha_{j'} + l(v_j, v_i) + l(v_{j'}, v_i)$.

Combining these situations, the lemma holds. ■

Lemma 5.2: For any node $v_j \in V$ and any node $v_i \in F$ to place controllers, $\sum_{k=j}^{n} \max(\alpha_j - l(v_k, v_i), 0) \leq \lambda c(v_i)$.

Proof: Supposing there exist $v_j \in V, v_i \in F$ such that $\sum_{k=j}^{n} max(\alpha_j - l(v_k, v_i), 0) > \lambda c(v_i)$. Order the nodes from j to n such that $\forall k \geq j, \alpha_k \geq \alpha_j$. Notice that some node $v_k \in \{v_j, v_{j+1}, \ldots, v_n\}$ can be selected to place a controller and thus $\alpha_k \leq l(v_i, v_k)$. Since $\alpha_j \leq \alpha_k$, $\alpha_j - l(v_i, v_k) \leq 0$, it is easy to check that v_i is chosen to the set F before time $t = \alpha_j$ (since α_j increases by 1 for every iteration, we use time t to represent the iterations). There must exist $v_k \in V \setminus F$ such that $\alpha_j - l(v_k, v_i) > 0$. Then α_k reaches $l(v_k, v_i)$ before $t = \alpha_j$, which means $\alpha_k \leq \alpha_j$ and this is a contradiction. Therefore, the lemma holds. ■

Lemma 5.3: Denote V_i as the set of nodes connected to the controller placed at v_i, when the algorithm terminates, $\forall v_i \in F$, $\alpha_i + \sum_{v_{ij} \in V_i} \alpha_{ij} \leq 3(\lambda c(v_i) + \sum_{v_{ij} \in V_i} l(v_{ij}, v_i))$.

Proof: Supposing $V_i = \{v_{i1}, v_{i2}, \ldots, v_{ik}\}$ is ordered by α_{ij} value such that $\alpha_{ij} \leq \alpha_{ij'}$ if $j \leq j'$. We apply Lemma 5.1 to v_{ij}, v_{i1} $(j > 1)$ as: $\alpha_{ij} \leq \alpha_{i1} + l(v_{ij}, v_i) + l(v_{i1}, v_i)$. Since node v_{i1} is connected to v_i no earlier than v_i is selected to place a controller, which means $\alpha_i \leq \alpha_{i1}$. Because node v_{i1} is connected to v_i, we have $l(v_{i1}, v_i) \leq \alpha_{i1}$. Thus we have: $\alpha_{ij} \leq 2\alpha_{i1} + l(v_{ij}, v_i)$.

Apply Lemma 5.2 to the set V_i as: $\sum_{v_{ij} \in V_i} \max(\alpha_{i1} - l(v_{ij}, v_i), 0) \leq \lambda c(v_i)$. So we have $\sum_{v_{ij} \in V_i} \alpha_{i1} \leq \lambda c(v_i) + \sum_{v_{ij} \in V_i} l(v_{ij}, v_i)$. Since node v_i is selected to place a controller and at least one node is connected to it, $\alpha_i \leq \lambda c(v_i)$. Therefore, by adding up the inequalities together, we have: $\alpha_i + \sum_{v_{ij} \in V_i} \alpha_{ij} \leq 3(\sum_{v_{ij} \in V_i} l(v_{ij}, v_i) + \lambda c(v_i))$ which concludes the lemma. ■

Obviously, Alg. 4 can find a subset of nodes F to place controllers such that the trade-off function value is no more than $3OPT$ where OPT is the optimal value of the BCLP problem. However, this is not the best approximation ratio. Actually, we can derive a better approximation ratio as follows.

Lemma 5.4: If $\forall v_i \in F$, there exists a constant d such that: $\sum_{v_{ij} \in V_i} \alpha_{ij} \leq d \cdot (\lambda c(v_i) + \sum_{v_{ij} \in V_i} l(v_{ij}, v_i))$. The approximation ratio of Alg. 4 is d.

Proof: We construct a solution for the dual formulation like this: $\beta_{ji} = max(\alpha_{ij} - d \cdot l(v_{ij}, v_i), 0)$. It is clear that $(\frac{\alpha_{ij}}{d}, \frac{\beta_{ji}}{d})$ is a feasible solution to the dual problem.

When the algorithm terminates, the nodes in V are either selected to place controllers or get connected to the controllers.

Denote the controller at v_i and its connected nodes V_i as a pair (v_i, V_i). By adding up the inequalities for each pair of (v_i, V_i), we have $\sum_{v_i \in F} \sum_{v_{ij} \in V_i} \alpha_{ij} \leq \sum_{v_i \in F} d \cdot (\lambda c(v_i) + \sum_{v_{ij} \in V_i} l(v_{ij}, v_i))$. Since it's the duality of the programming problem, we can derive that the approximation ratio is d. ■

Theorem 4: Alg. 4 finds a subset of controllers F such that the trade-off function value is bounded by $1.86OPT$.

Proof: From Lemma 5.4, we can get a better approximation ratio by finding a good d value. Therefore, we want to minimize d such that a minimized approximation ratio is achieved. Denote any subset of V as $\tilde{V}$, $\forall v_i \in F$, the process of calculating d can be modeled as follows:

$$Maximize: \frac{\sum_{v_j \in \tilde{V}} \alpha_j}{\sum_{v_j \in \tilde{V}} l(v_j, v_i) + \lambda c(v_i)}$$

$$s.t. \quad \forall v_j \in \tilde{V}: \alpha_j \leq \alpha_{j+1} \quad (1)$$

$$\forall v_j, v_{j'} \in \tilde{V}: \alpha_j \leq \alpha_{j'} + l(v_j, v_i) + c(v_{j'}, v_i) \quad (2)$$

$$\forall v_j \in \tilde{V}: \sum_{v_j \in \tilde{V}} \max(\alpha_j - l(v_j, v_i), 0) \leq \lambda c(v_i) \quad (3)$$

$$\forall v_j \in \tilde{V}: \alpha_j, l(v_j, v_i), c(v_i) \geq 0$$

It is easy to see that for any node $v_i \in F$ and any node $v_j \in \tilde{V}$, when we plug in $c(v_i)$ and $l(v_j, v_i)$, it is a linear programming. The objective value of this program has been shown to be at most 1.861 [22]. So the approximation ratio of the algorithm is at most 1.861. Thus, the theorem holds. ■

VI. Simulation

A. Simulation Setup

We select the real network topology from the Internet Topology Zoo which collects annotated network topologies from all over the world. The network we choose is "Garr201103" [12] which consists of 59 nodes. The propagation latency for each edge is calculated with the geographical distance and a propagation speed randomly selected from $0.1c$ to $0.5c$ (c stands for the speed of light).

B. Controller Placement with Bounded Latency

We compare the number of controllers in the maximum latency metric given by our algorithms. Denote the maximum latency between the nodes in the network as $l_{max} = 10ms$, we choose the latency bound increasingly from $0.5l_{max}$ to l_{max} with a step of $0.05l_{max}$. As shown in Fig. 4, the latency bound limits the number of controllers to be placed (in the figure, we use the ratio of the latency and l_m as the x-coordinate). When the latency bound approaches the maximum latency l_m, the number of controllers is close to 1. It is worth noticing that the upper bound and lower bound do provide an interval for the optimal number of controllers needed. Meanwhile, the upper bound is within $O(log(n))$ (n is the number of nodes in the network) times of the optimal number, which corroborates our analyses.

When the average latency is bounded by B_2 (we set $B_2 = 1ms$ and different λ value will cause Alg. 3 exceeds the constraint by a factor of at most $2(1 + 1/\epsilon)$), Fig. 5 describes the trends of the average latency and the number of controllers when ϵ ranges from 0.1 to 1. As shown in the figure, the average latency decreases with a increasing number of controllers that Alg. 3 selects. A lower ϵ value means a stricter requirement on the number of controllers to place, which results in a higher average latency. However, it means a higher risk of exceeding the latency bound requirement. As shown in the figure, the average latency exceeds B_1 within $2(1 + \frac{1}{\epsilon})$ factor, which also corroborates our analyses.

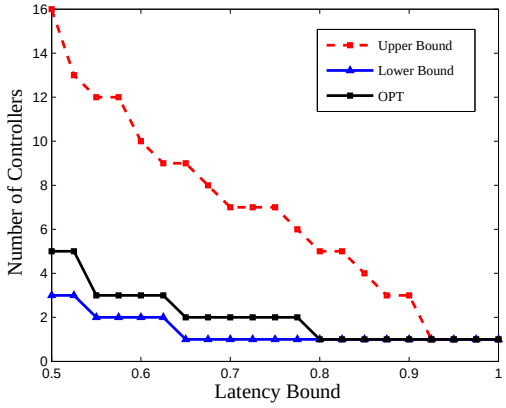

Fig. 4. The lower bound (Alg. 1), upper bound (Alg. 2) and the optimal number of controllers for BMLP

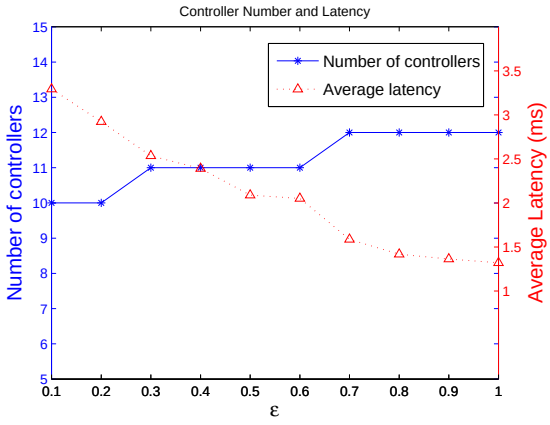

Fig. 5. The controller number of Alg. 3 when ϵ ranges from 0.1 to 1.0, and the average latency bound is $B_2 = 10$.

C. Balanced Cost-Latency Placement

We also evaluate the BCLP problem on the same topology. The cost to setup and maintain a controller is chosen uniformly from the interval $[10, 20]$. We conduct the simulations when λ ranges from 0.5 to 10. The values of the trade-off function with different λ are shown in Fig. 6 while the average latency and the number of controllers to place are shown in Fig. 7. We compare Alg. 4 with the linear program relaxation result as defined in Section V-A. As shown in Fig. 6, when λ increase from 0.5 to 10, both the LP relaxation method and Alg. 4 increase and they are very close to each other. This

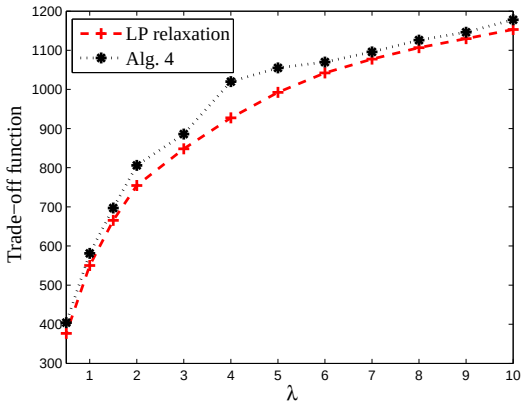

Fig. 6. The trade-off function of Alg. 4 with λ ranging from 0.5 to 10

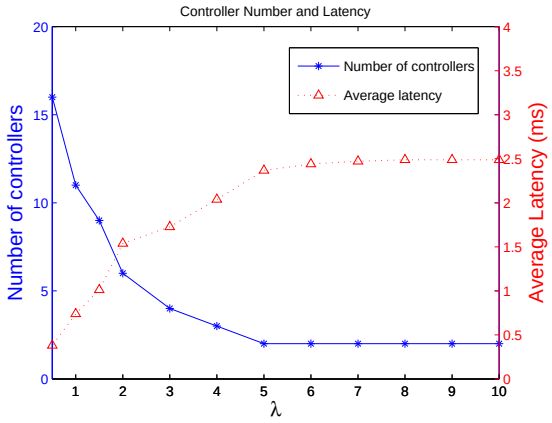

Fig. 7. The number of controllers and average latency of Alg. 4 with λ ranging from 0.5 to 10

result shows that our polynomial algorithm can achieve good performance. Since λ can be regarded as a trade-off factor between minimizing the number of controllers and minimizing the cost of the controllers, when λ increase, the number of controllers decrease while the average latency increases as depicted in Fig. 7. A higher λ value means the algorithm cares more about the cost of the controllers which sacrificing the average latency that can be achieved. A lower λ value pays more attention on the average latency and gives a less strict requirement on the cost of the controllers. Therefore, more controllers can be selected, which corroborate our analyses. Combing them, Alg. 4 has good performance in balancing the latency to control the network and the cost of the controllers.

VII. Conclusion

The controller placement problems have been gaining increasing attention in software defined networks. However, none of them consider the problem under the latency and/or controller cost constraints which are fundamental issues. In addition, all the previous work only give exhaustive search like methods which only work on very limited network sizes. In this paper, by taking the latency and controller cost constraints into consideration, We initiate two controller placement problems called Bounded (Maximum/Average) Latency Placement and Balanced Cost-Latency Placement. For all these problems, we give provably-efficient polynomial algorithms that can achieve acceptable approximation ratio compared to the optimal value. We also conducted simulations on real network topologies and the simulation results also corroborate our theoretical analyses.

References

[1] S. Agarwal, M. Kodialam, and T. V. Lakshman. Traffic Engineering in Software Defined Networks. In *INFOCOM*, 2013.

[2] N. Alon, L. Babai and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567C583, 1986.

[3] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and R. Boutaba. Dynamic Controller Provisioning in Software Defined Networks. In *CNSM*, 2013.

[4] M.Casado, M.Freedman, J.Petit, J.Luo, N. McKeown, and S.Shenker. Ethane: Taking Control of the Enterprise. *ACM SIGCOMM CCR*, 37(4):1-12, 2007

[5] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and K. van der Merwe. The Case for Separating Routing from Routers. In *FDNA*, 2004.

[6] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. NOX: Towards an Operating System for Networks. *ACM SIGCOMM Computer Communication Review*, 38(3): 105-110, 2008.

[7] S. Hassas Yeganeh, and Y. Ganjali. Kandoo: a Framework for Efficient and Scalable Offloading of Control Applications. In *Proceedings of the first workshop on Hot topics in software defined networks*, 2012.

[8] B. Heller, R. Sherwood, and N. McKeown. The Controller Placement Problem. In *Proceedings of the first workshop on Hot topics in software defined networks*, 2012: 7-12.

[9] D. S. Hochbaum. Heuristics for the Fixed Cost Median Problem. *Mathematical programming*, 22(1): 148-162, 1982.

[10] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng. Reliability-aware Controller Placement for Software-Defined Networks. In *Integrated Network Management (IM)*, 2013.

[11] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia. Pareto-Optimal Resilient Controller Placement in SDN-based Core Networks. In *International Teletraffic Congress (ITC)*, 2013.

[12] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The Internet Topology Zoo.

[13] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker. Onix: A Distributed Control Platform for Large-scale Production Networks. In *OSDI*, 2010.

[14] J. Kurose, and K. Ross. Computer Networks: A Top Down Approach Featuring the Internet. Peorsoim Addison Wesley, 2006.

[15] T. V. Lakshman, T. Nandagopal, R. Ramjee, K. Sabnani, and T. Woo. The SoftRouter Architecture. In *Proceeding of Hotnets*, 2004.

[16] T. Li, Z. Gu, X. Lin, S. Li, and Q. Tan. Approximation Algorithms for Controller Placement Problems in Software Defined Networks (full version). www.XXX.com

[17] N. Megiddo, and A. Tamir. On the Complexity of Locating Linear Facilities in the Plane. *Operations Research Letters*, 1(5): 194-197, 1982.

[18] 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 CCR*, April, 2008.

[19] S. Schmid, and J. Suomela. Exploiting Locality in Distributed SDN Control. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, 2013.

[20] D. B. Shmoys, E. Tardos, and K. Aardal. Approximation Algorithms for Facility Location Problems. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 1997.

[21] A. Tootoonchian A, and Y. Ganjali. HyperFlow: A Distributed Control Plane for OpenFlow. In *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, 2010.

[22] C. Swamy, and D. B. Shmoys. Fault-Tolerant Facility Location. *ACM Transactions on Algorithms (TALG)*, 4(4): 1-27, 2008.