This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2018.2876064, IEEE Transactions on Network and Service Management

SDN Controller Placement with Delay-Overhead Balancing in Wireless Edge Networks

Qiaofeng Qin, Konstantinos Poularakis, George Iosifidis, Sastry Kompella, Leandros Tassiulas

***Abstract*—Fog architectures at the network edge are becoming a popular research trend to provide elastic resources and services to end-users, where the processing capacity resides at the network periphery as opposed to traditional data-centers. Despite their momentum, the control plane of these architectures remains complex and challenging to implement. To enhance control capability, in this work, we propose to use Software Defined Networking. SDN moves the control logic off data plane devices and onto external network entities, the controllers. We provide a proof-of-concept implementation of a multi-controller edge system and measure traffic delay and overheads. The results reveal the sensitivity of delay to the location of controllers and the magnitude of inter-controller and controller-node overheads. Guided by the above, we model the problem of determining the placement of controllers in the edge network. Using linearization and supermodular function techniques, we present approximation solutions which perform close to optimal and substantially better than state-of-the-art methods. Finally, we analyze the interplay between various performance and reliability objectives.**

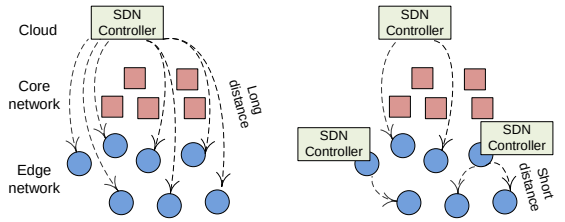
***Index Terms*—Controller placement, edge networking, software defined networking**

I. Introduction

A. Motivation

Emerging architectures, such as fog and mobile edge computing, distribute substantial amounts of data storage, processing and communication resources at the extremes of the network, in proximity to end-users, thereby allowing to bypass fundamental delay issues of traditional centralized cloud platforms [2]. While still at an infancy stage, these architectures are considered to be a key enabler for next-generation wireless (5G) and Internet of Things (IoT) systems [3] for supporting both computation-intensive and delay-sensitive services.

Despite their momentum, resource management in fog/edge architectures remains a very complex task, especially when a diverse set of services with different computation/storage/communication requirements need to be supported [4]. To facilitate resource management, we can benefit from novel softwarization technologies such as Software Defined Networking. SDN is popular in data center and ISP networks [5] and has been recently applied to wireless networks [6], [7], [8]. The main principle of SDN is to shift all the control functions from the data plane nodes to a programmable network entity, the controller. However, *this is a centralized approach, while edge architectures emphasize the distribution of resources and their management*. Therefore, it is challenging to apply SDN ideas at the edge part of the network.



(a) Single controller. (b) Distribution of controllers.

Fig. 1. A remote controller vs many controllers placed at the edge.

To exemplify, in order for SDN protocol to work properly, the state of the data plane nodes[1] should be reported to the controller in a *timely manner* so as to make efficient resource management decisions. This condition is easier to meet in wired networks where the communication between the controller and the nodes is much more stable and faster than in the wireless counterpart.

A method that can be used to solve the above issue is the *placement of many controller instances in proximity to edge nodes*, as it is depicted in Figure 1. The placed controllers, physically distributed but logically centralized, cooperate to manage the edge nodes which can reduce delay due to the shorter distance to them. Such placement strategies are possible today via commercial software implementations of controllers that can be installed in large quantities and operate even in lightweight network devices [10], [11].

The controller placement problem has been extensively studied over the past five years, especially in the context of data center and wired ISP networks (e.g., see the pioneer work in [12] and [13] for a survey). However, this problem obtains an interesting new twist in the context of edge architectures for the following reasons:

1) *Delay of network management*: Certain links between the edge nodes may be wireless in nature, unstable and

Part of this work was appeared in the proceedings of IEEE International Conference on Computer Communications (Infocom), April 2018 [1]. Q. Qin, K. Poularakis and L. Tassiulas are with the Department of Electrical Engineering and Institute for Network Science, Yale University, USA. G. Iosifidis is with the School of Computer Science and Statistics, Trinity College Dublin, the University of Dublin, Ireland. S. Kompella is with the U.S. Naval Research Laboratory, Washington, DC, USA. This publication was supported partly by the US Office of Naval Research (ONR) under award N00014-14-1-2190, the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001, and the Science Foundation Ireland (SFI) under Grant Number 17/CDA/4760. K. Poularakis acknowledges the Bodossaki Foundation, Greece, for a postdoctoral fellowship.

[1]The state of the nodes may include traffic statistics, link metrics and other protocol-specific parameters [9].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2018.2876064, IEEE Transactions on Network and Service Management

of low rate. Moreover, it may happen that many of these links separate a node from a placed controller resulting in slow statistic collection and node re-configuration through a multi-hop path. Hence, the controller placement strategy can drastically affect the delay of network management.

2) *Overhead of control messages*: Multi-controller implementations require the periodic exchange of messages between the controllers and nodes for statistic collection and resource management [9] as well as between the controllers themselves for synchronization purposes [14]. The overheads of these two types typically increase with the number of placed controllers, their distance to the nodes and to each other. Hence, if the controllers are not properly placed, the overheads will be significant, considering the scarcity of edge network resources.
3) *Heterogeneous synchronization strategies:* There are no standard protocols for synchronization among controllers. Therefore, behaviors may vary with different types of controllers. This variety should be considered when deciding the controller placement.

Given the above issues, the key open questions are: *How many controllers to place in the network and where exactly? Should we place many controllers close to the edge nodes to reduce delay of resource management or place fewer controllers close to each other to keep synchronization overheads as low as possible?*

B. Methodology and Contributions

In this paper, we follow a systematic methodology in order to answer the above questions. We focus on 5G networks where edge devices (such as smartphones) are SDN-compatible and can support SDN data paths and controllers. Our results however apply to other types of networks as well, e.g., IoT systems, as long as the edge nodes can support SDN and the network links the coordination of the controllers.

The contributions of this work can be summarized as follows:

- *SDN Controller Placement at the Edge*. We study the placement of controllers in SDN-enabled edge networks. We consider several practical features of these systems such as the different delay values of the wireless links and the impact of control overheads.
- *Experimentation and Emulation*. We analyze the operation and synchronization strategies of two state-of-art SDN controller implementations, namely ONOS and OpenDaylight (ODL). We perform experiments on a testbed of a multi-controller edge network to show that the average delay of managing a device can significantly change for different controller placement solutions. We also perform large-scale emulations to identify two types of overheads (inter-controller and controller-node traffic) and their dependence on the network topology.
- *Optimization Algorithms*. We build upon the emulation findings to formulate the controller placement problem for two different objectives; minimization of delay and overheads. We propose exact solutions of Mixed Integer Programming (MIP), as well as scalable and fast approximate solutions (running in less than 0.1 secs) using linearization and supermodular techniques.
- *Evaluation Results*. We evaluate the proposed controller placement algorithms using two real network topologies. We find that our approach performs close to optimal and better than state-of-the-art methods. We also analyze the interplay between various performance and reliability objectives; minimizing delay can favor the reliability of controller - data plane node communication, but affect the reliability of inter-controller communication.

The rest of the paper is organized as follows. Section II presents our experimentation and emulation results. Guided by these results, we model the controller placement problem in edge networks in Section III. In Section IV, we present optimal and approximate solution algorithms for small- and large-scale problem instances respectively. Section V presents the evaluation of our proposed algorithms, while Section VI reviews our contribution compared to related works. We conclude our work in Section VII.

II. Experimentation & Emulation Analysis

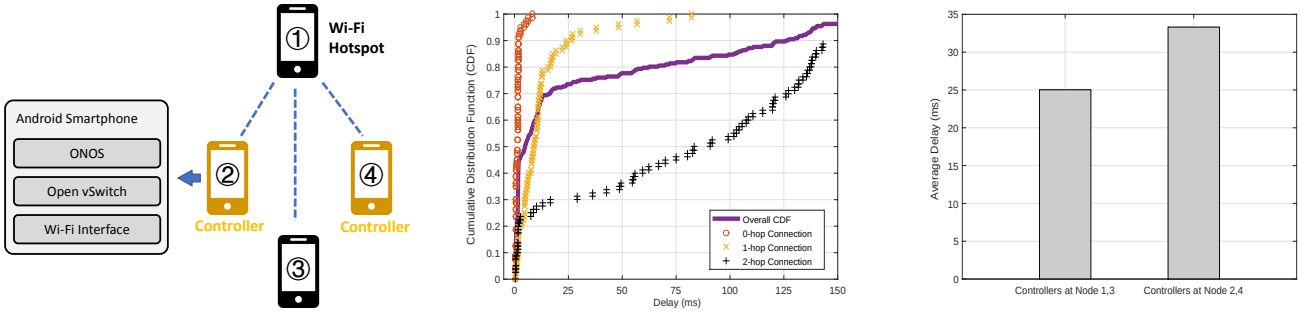
In this section, we present experimentation and emulation results using commercial SDN controller and data plane implementations. The results provide insights about the delay and overheads of multi-controller edge systems which will be used in modeling the controller placement problem in the next section.

A. Experimentation Results for Management Delay

Testbed Set-up. In this subsection, we set-up a testbed of a multi-controller edge system using off-the-shelf network devices. Specifically, we deploy four Nexus 4 Android smartphones to form a wireless network as it is depicted in Figure 2(a). The first smartphone works as an access point (hotspot) to provide the remaining three smartphones with WiFi connections. This represents a common edge network setting, where a node can either establish multihop connections to backbone networks, or exchange data with its peer in a D2D fashion. Besides, smartphones are representative devices widely used in edge networks. Due to their constraints of calculating and storage, our testbed shows a challenging scenario that is worthy of investigation.

We take several steps to make the network SDN-enabled. In each smartphone, we create a *chroot* environment to install the Ubuntu system running with its original Android system at the same time. By this, we are able to install popular SDN-related software in the smartphone. First, we make all devices working as data plane nodes by installing Open vSwitch [15]. This creates a virtual switch that supports SDN in each smartphone. Second, we deploy SDN controllers in Node 2 and Node 4. Though constrained in resources, the smartphone is still capable enough to run a controller instance, such as ONOS. ONOS is designed particularly for scalability and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2018.2876064, IEEE Transactions on Network and Service Management



(a) Testbed topology. (b) CDF of delays. (c) Comparison of two placement schemes.

Fig. 2. (a) Testbed of a multi-controller edge system built from smartphones that enable Open vSwitch and ONOS. (b) CDF of controller-node communication delays. (c) Comparison of average delays under different controller placements.

permits multiple controllers working together in the form of a *controller cluster*. Then, we establish a connection between the two controllers and assign each smartphone to its nearest controller with respect to the hop count length distance metric.

Measurement Methodology. We mainly take measurements on the network delay between the controller and data plane nodes. The devices are placed within an empty lab room, and distance over each wireless link is 2 meters. One frequent and important interaction between a controller and a data plane node is the request and reply of flow statistics. Therefore, we measure the delay at controller nodes, by analyzing the interval between sending such an OpenFlow request message and receiving its corresponding reply. We keep capturing messages since the cluster reaches the steady state and collect 250 measurements. Figure 2(b) shows the cumulative distribution function of the delays we measured. The average value is tens of milliseconds which is comparable or higher than the delay reported in typical wired networks [12]. From the CDF plot, we notice that the variance is large, corresponding to the relatively unstable wireless links. It is common for the delay to go even beyond 100 milliseconds.

We also notice that the placement of controllers is important, because the delay relies highly on the distance between the data plane node and its controller. For example, Node 2 and Node 4 contain controllers locally, while Node 1 and Node 3 have one-hop and two-hop connections to the controller, respectively. As a result, drawn separately in Figure 2(b), local connection shows almost zero delays while the one-hop and two-hop connections show notable delays. To further demonstrate this, we move ONOS controllers from nodes 2 and 4 to nodes 1 and 3. If we still assign each data plane node to its nearest controller, we can find that the average delay is 25% lower, as it is depicted in Figure 2(c).

Main Takeaways. *Modern edge network devices (smartphones) can act as controllers to manage other devices. The management delay highly depends on the number of wireless hops and can significantly change for different controller placement strategies (up to 25% difference in our testbed).*

B. Emulation Results for Control Overheads

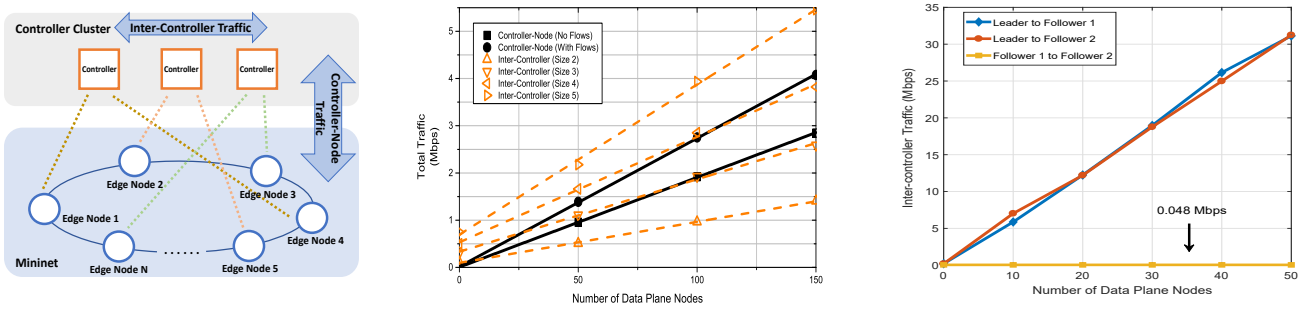
In the previous subsection, we focused on the delay required to manage the edge nodes. In this subsection, we will analyze another important factor; the overheads of SDN control. By measuring the overheads of ONOS and OpenDaylight, the most typical commercial multi-controller solutions, we identify two types of control overheads, controller-node overhead and inter-controller overhead. We also summarize two types of synchronization strategies, leaderless synchronization and leader-based synchronization.

Emulation Set-up. Since control overheads depend heavily on the scale of the network, we need to deploy many more nodes than what we have in our testbed if we wish to analyze them. A more accessible way to take large-scale measurements is by running emulations on a virtual network generated by Mininet [16]. This method allows us to test networks with hundreds of nodes and several controllers using a common CPU machine. Specifically, we create a virtual edge network with ring topology and evenly assign nodes to the placed controllers, as shown in Figure 3(a). All controllers run a simple application; reactive forwarding.

Measurement Methodology - ONOS. First, we run ONOS as the controller. In order to show the impact of the scale of the network, we take measurements on both types of control traffic with different number of nodes in the virtual network. Figure 3(b) verifies that both controller-node traffic and inter-controller traffic grow linearly when the network scales up. What is more, we also consider other factors that have impact. For controller-node traffic, we create a large amount of one-hop *iperf* [17] flows randomly, with a fixed rate (0.1 flows per second for each node). For inter-controller traffic, we deploy different numbers of controllers (cluster sizes). According to Figure 3(b), in all of these situations, the two types of overheads are at the same order of magnitude (up to a few Mbps each). This fact means that both of them are important when deciding the controller placement.

We also analyze the synchronization strategy adopted by ONOS. ONOS has a relatively complicated consensus mechanism, deploying several different algorithms at the same time, which we discuss in detail in the appendix. However, *the*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2018.2876064, IEEE Transactions on Network and Service Management



(a) Emulation setting. (b) Control overheads of ONOS. (c) Control overheads of OpenDaylight.

Fig. 3. (a) Ring topology in emulations. (b) Controller-node overheads (solid lines) and inter-controller overheads (dot lines) of ONOS controllers. (c) Overheads of OpenDaylight controllers.

majority of traffic follows a leaderless manner, i.e., each pair of controllers generates a similar amount of overheads when synchronizing.

Measurement Methodology - OpenDaylight. Next, we replace the controllers with OpenDaylight. Same with ONOS, OpenFlow protocol is adopted for the controller-node communications, leading to similar behaviors. However, the inter-controller traffic pattern is quite different. As shown in Figure 3(c), although the overhead also grows linearly with the number of data plane nodes, the overhead is much larger compared with the same setting in ONOS case. Importantly, the overhead is no longer evenly distributed among each pair of controllers. In the 3-controller case, we can identify one controller as a leader, and the remaining two as followers. We find that non-negligible overhead only exists between a leader and a follower, rather than two followers. Therefore, *controller synchronization of OpenDaylight follows a leader-based manner.*

Main Takeaways. *(i) The two types of overheads (inter-controller and controller-node traffic) are at the same order of magnitude in representative scenarios (up to few Mbps each), increasing linearly to the load of controllers. (ii) There are leader-based and leaderless strategies for controller synchronization, leading to different distributions of inter-controller overheads.*

III. Modeling the Controller Placement Problem

In this section, we build upon the experimentation and emulation results of the previous section to model the controller placement problem in edge networks. We consider a network of a diverse set $\mathcal{N}$ of N edge nodes such as access switches, cellular base stations, set-top boxes, Wi-Fi access points and even mobile devices (e.g., smartphones), as depicted in Figure 4. We use the term edge nodes to describe them as they are in close proximity to the end user, unlike core switches and routers of ISP backbone networks or data centers. Our analysis applies to any kind of edge nodes as long as: (i) they are SDN-compatible; (ii) their links have high enough capacity to support the SDN coordination mechanisms without congestion. Furthermore, the edge nodes can be connected with each other through single or multi-hop paths. Without loss of generality, the nodes generate new flows with uniform rate, normalized to one.

The network is SDN-enabled in the sense that all nodes run virtual switches that support SDN protocol. A controller is placed at the cloud and connects to the edge nodes through in-band or out-of-band control channels. The network operator may decide to place additional controllers in the network. Placing a controller on an edge node requires to locally install and run a controller implementation software such as ONOS [11]. We introduce the binary optimization variable $x_n \in \{0,1\}$ to indicate whether a controller is placed at node $n \in \mathcal{N}$ ($x_n = 1$) or not ($x_n = 0$). These variables constitute the *controller placement policy*:

$$\boldsymbol{x} = (x_n \in \{0,1\} \ : n \in \mathcal{N}) \ . \tag{1}$$

Due to limited resources, not all the edge nodes may be capable of hosting a controller. To model such cases, we denote by $\mathcal{N}_h \subseteq \mathcal{N}$ the subset of nodes that can play the role of host for a controller, where $N_h = |\mathcal{N}_h|$. Then, we require that:

$$x_n = 0, \ \forall n \notin \mathcal{N}_h \ . \tag{2}$$

The network operator will also need to decide the assignment of nodes to controllers, i.e., which controller is responsible for the management of each node. We introduce the binary optimization variable $y_{nm} \in \{0,1\}$ to indicate whether node $n \in \mathcal{N}$ is assigned to the controller at node $m \in \mathcal{N}$ ($y_{nm} = 1$) or not ($y_{nm} = 0$). Similarly, we denote the cloud node by c, and $y_{nc} \in \{0,1\}$ indicates the assignment of node n to the controller located at the cloud. These variables constitute the *assignment policy* of the operator:

$$\boldsymbol{y} = (y_{nm} \in \{0,1\} \ : n \in \mathcal{N}, m \in \mathcal{N} \cup \{c\}) \ . \tag{3}$$

Since every node needs to be assigned to a controller, we require that:

$$\sum_{m \in \mathcal{N} \cup \{c\}} y_{nm} = 1, \ \forall n \in \mathcal{N} \ . \tag{4}$$

In addition, we require that a controller must be placed at node m in order for node n to be able to assign to it:

$$y_{nm} \leq x_m, \ \forall n, m \in \mathcal{N} \ . \tag{5}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2018.2876064, IEEE Transactions on Network and Service Management

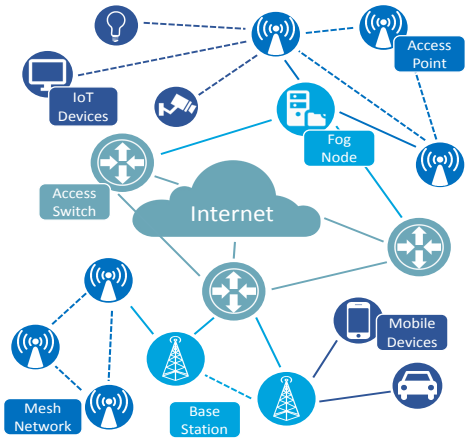
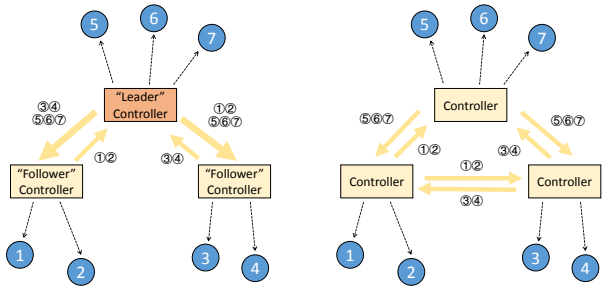


Fig. 4. An example of an SDN-enabled edge network.



(a) Leader-based (b) Leaderless

Fig. 5. A comparison between leader-based and leaderless synchronization strategies.

We also consider that in the leader-based synchronization strategy, one controller acts as the leader. We use the optimization variable $z_n \in \{0,1\}$ to indicate whether a node n is the leader or not, and require that:

$$\sum_{n \in \mathcal{N}_h \cup \{c\}} z_n = 1 \ . \tag{6}$$

As we showed in Figure 2(b), the delay of node management increases rapidly with the topological distance (number of hops) between the controller and the node. Therefore, assigning a node to a controller placed at a nearby edge node instead of the cloud controller can greatly expedite its management. In the model, we denote by d_{nm} (milliseconds) the delay when node n is assigned to the controller at node m. Similarly, we denote by d_{nc} the delay when node n is assigned to the cloud controller. The total (across all nodes) delay is given by:

$$D(\boldsymbol{y}) = \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{N} \cup \{c\}} y_{nm} d_{nm} \ . \tag{7}$$

The controllers will continuously exchange messages with the data plane nodes they manage for statistic collection and forwarding table update. As we showed in Figure 3(b), the bandwidth overhead of this message exchange can be significant in practice. Moreover, the cost of this overhead would increase with the topological distance between the controller and the node as resources of more links would be consumed. To model this, we denote by w^a_{nm} the overhead cost of assigning node n to the controller at node m. Similarly, the overhead cost of assigning node n to the cloud controller is denoted by w^a_{nc}. The total (across all nodes) assignment overhead cost is given by:

$$W_a(\boldsymbol{y}) = \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{N} \cup \{c\}} y_{nm} w^a_{nm}. \tag{8}$$

The controllers will also exchange messages to each other for synchronization purposes. As we show in Section II, different types of controllers may adopt either leader-based or leaderless strategy. We should notice that they may even coexist, in case that a controller adopts multiple consensus algorithms, just as ONOS. Therefore, we model the overheads of each strategy separately.

We begin with the leaderless case. As what Figure 3(b) indicates, each pair of controllers exchange messages with *constant rate* as well as messages with *rate that depends on the controller's load*. The latter means that the more nodes are assigned to a controller the more messages it exchanges with the rest controllers. Therefore, we can model the synchronization overheads as follows. For the messages exchanged at a constant rate, we denote by $w^{con}_{ml} \geq 0$ the respective overhead cost between controllers at nodes m and l. For the messages exchanged at a rate that depends on the controller load, we denote by $w^{dep}_{ml} \geq 0$ the respective overhead cost between controllers at nodes m and l for each node assigned to controller m. The total (across all controller pairs) leaderless synchronization overhead cost is given by:

$$W_{s1}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{m \in \mathcal{N} \cup \{c\}} \sum_{l \in \mathcal{N} \cup \{c\}} x_m x_l \big(w^{con}_{ml} + w^{dep}_{ml} \sum_{n \in \mathcal{N}} y_{nm} \big). \tag{9}$$

where, with *a slight abuse of notation*, we used the terms x_m and x_l for m and l equal to c in the above summation. These terms are set to one to indicate that a controller is placed at the cloud[2].

For the leader-based case, we have a similar cost expression:

$$W_{s2}(\boldsymbol{x}, \boldsymbol{z}) = \sum_{m \in \mathcal{N} \cup \{c\}} \sum_{l \in \mathcal{N}_h \cup \{c\}} x_m z_l \big(w^{lb\_con}_{ml} + w^{lb\_dep}_{ml} * N \big). \tag{10}$$

where the difference is that the controllers synchronize only with the leader controller, rather than in a peer-to-peer manner, as depicted in Figure 5. The constants $w^{lb\_con}_{ml}$ and $w^{lb\_dep}_{ml}$ represent the linear relationship between the overhead and the controller load, just like in the leaderless term. However the values can be different.

On the one hand, a *scattered placement* of many controllers across the network would reduce delay and assignment overhead costs since nodes can be managed by controllers at closer

[2]We make the same abuse of notation throughout the paper.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2018.2876064, IEEE Transactions on Network and Service Management

proximity. On the other hand, a more *compact placement* of fewer controllers would reduce the synchronization overhead cost. These metrics are contradicting in general, and therefore cannot be all minimized at the same time. Depending on its preferences, the operator would have to find a way to balance delay and overheads. In general, this will require to optimize a function of the following form, where the weight value $\gamma \geq 0$ is used to balance the different metrics:

$$J(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \gamma D(\boldsymbol{y}) + W_a(\boldsymbol{y}) + W_{s1}(\boldsymbol{x}, \boldsymbol{y}) + W_{s2}(\boldsymbol{x}, \boldsymbol{z}). \tag{11}$$

By setting $\gamma = 0$, the balanced function reduces to the total overhead costs, neglecting delay. However, by increasing the γ value, more priority is given to the delay. The edge controller placement (ECP, for short) can be expressed as follows:

$$\begin{aligned} min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}} \ & J(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \\ s.t. \ & \text{ constraints: } (1),\ (2),\ (3),\ (4),\ (5),\ (6). \end{aligned}$$

The above problem is challenging since it contains discrete variables and a non-linear objective function with cubic and quadratic terms (inside W_{s1} and W_{s2}). In fact, it is not hard to show that ECP is a generalization of the well-studied facility location problem [20], which is NP-Hard, by allowing facility opening (equivalently controller placement) costs to be non-constant (i.e., the synchronization cost depends on the distance between the controllers).

IV. Optimization Algorithms

In this section, we address the ECP problem. We start by presenting an optimal algorithm that can be applied to small-scale problem instances. Following that, we present approximation algorithms that scale well with the size of the problem instance.

A. Small-scale optimal solution

In this subsection, we find an optimal solution by converting ECP to a Mixed Integer Programming (MIP) problem. That is a problem with linear objective function and constraints. This conversion is important since there exist various commercial solvers, such as CPLEX, that can be directly used to solve this type of problems.

To obtain the MIP formulation, we apply *standard linearization techniques* [21]. Specifically, we introduce the following two vectors of additional optimization variables:

$$\boldsymbol{\theta} = (\theta_{ml} \in \{0,1\} \ : m, l \in \mathcal{N} \cup \{c\}) \ . \tag{12}$$

$$\boldsymbol{\delta} = (\delta_{ml} \in \{0,1\} \ : m \in \mathcal{N} \cup \{c\}),\ l \in \mathcal{N}_h \cup \{c\}) \ . \tag{13}$$

$$\boldsymbol{\phi} = (\phi_{mln} \in \{0,1\} \ : m, l \in \mathcal{N} \cup \{c\},\ n \in \mathcal{N}) \ . \tag{14}$$

Then, we add the following linear constraints for θ:

$$\theta_{ml} \leq x_m,\ m, l \in \mathcal{N} \cup \{c\} \ , \tag{15}$$

$$\theta_{ml} \leq x_l,\ m, l \in \mathcal{N} \cup \{c\} \ , \tag{16}$$

$$\theta_{ml} \geq x_m + x_l - 1,\ m, l \in \mathcal{N} \cup \{c\} \ , \tag{17}$$

the following linear constraints for δ:

$$\delta_{ml} \leq x_m,\ m \in \mathcal{N} \cup \{c\},\ l \in \mathcal{N}_h \cup \{c\}) \ , \tag{18}$$

$$\delta_{ml} \leq z_l,\ m \in \mathcal{N} \cup \{c\},\ l \in \mathcal{N}_h \cup \{c\}) \ , \tag{19}$$

$$\delta_{ml} \geq x_m + z_l - 1,\ m \in \mathcal{N} \cup \{c\}),\ l \in \mathcal{N}_h \cup \{c\} \ , \tag{20}$$

and the following linear constraints for ϕ:

$$\phi_{mln} \leq \theta_{ml},\ m, l \in \mathcal{N} \cup \{c\},\ n \in \mathcal{N} \ , \tag{21}$$

$$\phi_{mln} \leq y_{nm},\ m, l \in \mathcal{N} \cup \{c\},\ n \in \mathcal{N} \ , \tag{22}$$

$$\phi_{mln} \geq \theta_{ml} + y_{nm} - 1,\ m, l \in \mathcal{N} \cup \{c\},\ n \in \mathcal{N} \ . \tag{23}$$

The D and W_a functions are already linear, so we only need to linearize the W_{s1} and W_{s2} function. This is possible with the new variables, as they can be written as:

$$\widehat{W_{s1}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{m \in \mathcal{N} \cup \{c\}} \sum_{l \in \mathcal{N} \cup \{c\}} \big(\theta_{ml} w_{ml}^{con} + \sum_{n \in \mathcal{N}} \phi_{mln} w_{ml}^{dep} \ \big). \tag{24}$$

$$\widehat{W_{s2}}(\boldsymbol{\delta}) = \sum_{m \in \mathcal{N} \cup \{c\}} \sum_{l \in \mathcal{N}_h \cup \{c\}} \delta_{ml} \big(w_{ml}^{lb\_con} + w_{ml}^{lb\_dep} * N \ \big). \tag{25}$$

Then, the MIP problem can be expressed as follows:

$$\begin{aligned} min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{\theta},\boldsymbol{\phi}} \ & \gamma D(\boldsymbol{y}) + W_a(\boldsymbol{y}) + \widehat{W_{s1}}(\boldsymbol{\theta}, \boldsymbol{\phi}) + \widehat{W_{s2}}(\boldsymbol{\delta}) \\ s.t. \ & \text{ constraints: } (1) - (6) \text{ and } (12) - (23) \ . \end{aligned}$$

The inequalities in (15)-(17) ensure that θ_{ml} will be zero if at least one (or equivalently the product) of x_m and x_l is zero; otherwise it will be one. It is similar for inequalities in (18)-(20). Combined with the above, the inequalities in (21)-(23) ensure that ϕ_{mln} will be zero if at least one (or equivalently the product) of x_m, x_l and y_{nm} is zero; otherwise it will be one.

Various commercial solvers, such as CPLEX, can be directly used to solve a MIP problem. These solvers apply branch-and-bound techniques and can be quite fast for small-scale problem instances. However, in some cases, edge systems can be of extremely large scale, e.g., in IoT architectures, and hence the above MIP problem becomes extremely large, hindering the performance of such branch-and-bound or other computational methods. In the next subsection, we propose a solution method that overcomes this dimensionality problem, and hence extends the range of the systems to which our work can apply.

B. Large-scale approximate solution

We need an algorithm that solves a large-scale ECP instance in a short time, so that the placement strategy can be quickly recalculated periodically or when necessary, e.g., when the network topology changes because nodes join or leave. In this subsection, we present such algorithms for different variants of the problem.

Leader-based case. In some controllers like OpenDaylight, the inter-controller synchronization fully follows leader-based strategy, i.e., $w_{ml}^{con} = 0$ and $w_{ml}^{dep} = 0$. In this case, $W_{s1}(\boldsymbol{x}, \boldsymbol{y})$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2018.2876064, IEEE Transactions on Network and Service Management

Algorithm 1: General Approach to Solve ECP

$\boldsymbol{z} \leftarrow \mathbf{0}$
for $\forall n \in \mathcal{N}_h \cup \{c\}$ **do**
  $z_n \leftarrow 1$
  $\boldsymbol{z}^n \leftarrow \boldsymbol{z}$,
  $\boldsymbol{x}^n \leftarrow Place(\boldsymbol{z}^n),\ \boldsymbol{y}^n \leftarrow Assign(\boldsymbol{x}^n, \boldsymbol{z}^n)$
  $J^n \leftarrow J(\boldsymbol{x}^n, \boldsymbol{y}^n, \boldsymbol{z}^n)$
  $z_n \leftarrow 0$
end
$n^o \leftarrow \text{argmin}_n\, J^n$
return $\boldsymbol{x}^{n^o}, \boldsymbol{y}^{n^o}, \boldsymbol{z}^{n^o}$

TABLE I
APPROXIMATION RATIOS FOR INCAPACITATED FACILITY LOCATION PROBLEM.

Bound	Main technique	Reference
3	primal-dual	[22]
3	local search	[23]
1.488	LP rounding	[24]

can be ignored. Here we describe the steps to solve this problem in Algorithm 1. We note that there exist $N_h + 1$ possible locations to place the leader; the cloud or the edge nodes with sufficient resources to play that role. Let us assume for a moment that we could find a way to solve the subproblem of controller placement and assignment $(\boldsymbol{x}, \boldsymbol{y})$ for a given leader selection $(\boldsymbol{z})$. In that case, we could simply iterate the value of leader for all the $N_h + 1$ possible choices, solve the subproblem for each choice, and pick the solution with the lowest balanced cost. In the rest of this section, we will show how to solve the above subproblem.

Consider the subproblem in which the leader is fixed at node $e \in \mathcal{N}_h \cup \{c\}$. Then, the synchronization overhead cost depends only on the controller placement decisions $(\boldsymbol{x})$:

$$W_{s2}(\boldsymbol{x}, \boldsymbol{z}) = W_{s2}(\boldsymbol{x}) = \sum_{m \in \mathcal{N} \cup \{c\}} x_m \big(w_{me}^{lb\_con} + w_{me}^{lb\_dep} * N\big) \tag{26}$$

while we recall that the delay and assignment overhead costs depend only on the assignment decisions $(\boldsymbol{y})$. By defining $f_m = w_{me}^{lb\_con} + w_{me}^{lb\_dep} * N$ and $c_{nm} = \gamma d_{nm} + w_{nm}^{a}$ the balanced objective can be written as:

$$\sum_{m \in \mathcal{N}_h \cup \{c\}} x_m f_m + \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{N}_h \cup \{c\}} y_{nm} c_{nm}\ . \tag{27}$$

We notice that this is the standard form of the well-studied *uncapacitated facility location problem*. The cost of locating a facility at node m is given by f_m, while the cost of assigning a client at node n to a facility at node m is given by c_{nm}. Various algorithms have been developed to solve this problem in polynomial time with provable approximation ratios, as we summarized in Table I. These include LP rounding [20], primal-dual [22] and local-search [23] methods. So far, the best performance is the 1.488-approximation proposed in [24].

TABLE II
APPROXIMATION RATIOS FOR NON-NEGATIVE SUBMODULAR FUNCTION MAXIMIZATION.

Bound	Main technique	Reference
$\frac{1}{4}$	uniformly random	[26]
$\frac{1}{3}$	local search (deterministic version)	[26]
$\frac{2}{5}$	local search (randomized version)	[26]
0.41	simulated annealing	[27]
0.42	structural continuous greedy	[28]
$\frac{1}{3}$	greedy (deterministic version)	[29]
$\frac{1}{2}$	greedy (randomized version)	[29]

Leaderless case and hybrid case. Next, we turn to a more general case, where any term in the original problem is not necessary to be zero. Similarly, we follow the steps of Algorithm 1, and consider a different approach to solve the subproblem of placement and assignment. We begin by showing that for a given controller placement $\boldsymbol{x}$, the optimal assignment policy $\boldsymbol{y}$ can be easily found. Specifically, we show the following lemma (proved in the appendix).

Lemma 1: For a given controller placement $\boldsymbol{x}$, the optimal assignment policy can be described by:

$$y_{nm} = \begin{cases} 1, \text{ if } m = \underset{m': x_{m'}=1}{\text{argmin}} \ [\ \gamma d_{nm'} + w_{nm'}^{a} + \sum_{l: x_l = 1} w_{m'l}^{dep}\ ] \\ 0, \text{ otherwise} \end{cases} \tag{28}$$

for each $n \in \mathcal{N},\ m \in \mathcal{N} \cup \{c\}$.

The above lemma indicates that the complexity of the subproblem primarily lies on the optimization of the controller placement policy. Following this intuition, we introduce the *element* X_n to denote the placement of a controller at node n which is equivalent of deciding $x_n = 1$. The set of all possible elements, also called the *ground set*, can be defined as follows:

$$G = (X_n : n \in \mathcal{N}_h)\ . \tag{29}$$

A subset of elements $X \subseteq G$ corresponds to a controller placement policy $\boldsymbol{x}$, such that $x_n = 1$ if and only if $X_n \in X$. Besides, let $\boldsymbol{x}_X$ be the binary representation of the set of elements X. Then, the objective function B can be expressed as a set function $f:\ 2^G \rightarrow \mathbb{R}$:

$$f(X) = J(\boldsymbol{x}_X, y(\boldsymbol{x}_X)) \tag{30}$$

where $y(\boldsymbol{x}_X)$ denotes the optimal assignment policy given the controller placement policy $\boldsymbol{x}_X$ based on equation (28).

Next, we consider a well-studied class of set functions called *supermodular* [25].

Definition 1: Let G be a finite set of elements (ground set). A set function $f:\ 2^G \rightarrow \mathbb{R}$ is called supermodular if for all subsets $A, B \subseteq G$ with $A \subseteq B$ and every element $i \in G \setminus B$ it holds that:

$$f(A \cup \{i\}) - f(A) \leq f(B \cup \{i\}) - f(B) \tag{31}$$

The above definition indicates that the marginal value for adding an element i in a set increases as the respective set expands. We will show that, under certain conditions on the cost values, our objective $f(X)$ can be expressed as a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2018.2876064, IEEE Transactions on Network and Service Management

supermodular function. Specifically, we have the following lemma (proved in the appendix).

Lemma 2: The set function $f(X)$ defined in (30) is supermodular for the case of uniform costs $w_{ml}^{dep} = w_{m'l'}^{dep} = w^{dep}$, $\forall m, l, m', l' \in \mathcal{N}_h \cup \{c\}$.

Based on Lemma 2, the subproblem can be casted as the *minimization of a supermodular function* f. This type of problems are usually addressed by considering their equivalent *submodular function maximization* version. That is, minimizing the supermodular function f (i.e., delay and overhead costs) is equivalent to maximizing the submodular function $\widehat{f}(X) = f^{ub} - f(X)$ (respective delay and overhead cost savings). Here, the constant f^{ub} indicates an upper bound to the highest possible value of $f(X)$.

Given that $\widehat{f}(X)$ is non-negative, there exist various approximation algorithms to maximize it (Table II). Here, an approximation bound β means that the ratio of the value of the approximate solution over the optimal solution value is always at least β, namely $\widehat{f}^{apx}/\widehat{f}^{opt} \geq \beta$. Therefore, we obtain the following theorem.

Theorem 1: There exists a solution to the subproblem such that $\widehat{f}^{apx}/\widehat{f}^{opt} \geq \beta$, where $\beta \in \{\frac{1}{4}, \frac{1}{3}, \frac{2}{5}, 0.41, 0.42, \frac{1}{3}, \frac{1}{2}\}$.

The Algorithm in the last row of Table II has the best approximation bound. As it is summarized in Algorithm 2, this algorithm proceeds in N_h iterations which correspond to some arbitrary order $r_1, \ldots r_{N_h}$ of the ground set G. At each iteration, two solutions A and B are maintained, initially set to $\emptyset$ and G respectively. At the n^{th} iteration, the algorithm either adds r_n to A or removes r_n from B. This decision is done randomly and greedily based on the marginal gain of each of the two options. After N_h iterations both solutions coincide, i.e., $A = B$. This is the output of the algorithm.

With the solution to this subproblem with a fixed leader, we follow the same steps in Algorithm 1 as in the leader-based case, to have exhaustive search on all possible leader choices. Finally we get a solution to ECP problem with approximation bounds listed in Theorem 1.

We emphasize that although the approximation bounds of Theorem 1 are shown for the case that w^{dep} values are identical, we make no assumptions on the values of d and w^{con} vectors. Moreover, in the next section we will show that in practice Algorithm 2 achieves a near-optimal solution even for heterogeneous w^{dep} values.

Algorithm 2: Randomized Greedy Algorithm

$A \leftarrow \emptyset, B \leftarrow G$
for $n = 1$ *to* N_h **do**
  $\Delta A \leftarrow f(A) - f(A \cup \{r_n\})$
  $\Delta B \leftarrow f(B) - f(B \setminus \{r_n\})$
  $\Delta A \leftarrow \max(\Delta A, 0)$, $\Delta B \leftarrow \max(\Delta B, 0)$
  with probability $\Delta A/(\Delta A + \Delta B)^*$ **do:**
  $A \leftarrow A \cup \{r_n\}$
  else (with probability$\Delta B/(\Delta A + \Delta B)$) **do:**
  $B \leftarrow B \setminus \{r_n\}$
end
return A (or equivalently B)

* If $\Delta A = \Delta B = 0$, then $\Delta A/(\Delta A + \Delta B) = 1$.

V. Evaluation Results

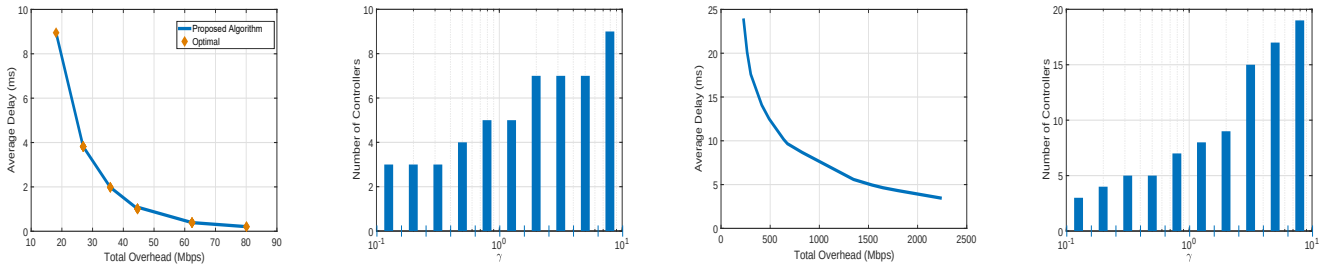
In this section, we evaluate the performance of the proposed algorithms using two real wireless network topologies. Overall, we find that our algorithms achieve excellent performance and multi-fold gains over state-of-the-art methods, especially when the priority is given on the delay rather than the overhead optimization (large γ value). Such tendency exists regardless of the synchronization strategy of controllers. We repeat the same evaluations on both OpenDaylight (Figure 6) and ONOS (Figure 7), in order to show the results in a comparative way. We also find that the optimization of delay and overheads indirectly favors two reliability objectives, namely controller-node and inter-controller path loss. Therefore, by balancing delay and overheads, we can also balance the above two reliability metrics.

Evaluation Setup. Throughout the evaluation, we use the MANIAC mobile ad hoc network in [30] and the Barcelona wireless mesh network in [31]. MANIAC contains only 14 nodes, which allows us to execute MIP algorithm and find the optimal solution in reasonable time. On the other hand, Barcelona contains 60 nodes, the evaluation of which verifies the scalability of the algorithms and enriches the results. In each topology, we augment an extra node representing the cloud controller, connected to all other nodes with a large distance. We set this distance value as the half of the graph's diameter. We define delay and overhead costs based on our measurements in previous sections. The overhead cost is the product of the measured traffic volumes in Figure 3 and the network distances between the nodes. The specific values depend on the type of controllers. The delay cost d_{nm} is the aggregate delay of the links of the respective shortest path. We set the delay of each link randomly with average value $12.23\ ms$, in accordance with our experiment result in Figure 2(b). Our simulation codes, written in MATLAB, are publicly available online in [32]. We hope that the reproducibility of our results will facilitate future research efforts for the benefit of research community.

Evaluations on OpenDaylight. First, we choose OpenDaylight as the controller, which has totally leader-based synchronization strategy, and the facility location problem based algorithm is available. In accordance with measurement of Section II, we set $w_{ml}^{a} = 0.019 \cdot hops_{ml}$, $w_{ml}^{lb\_con} = 0.207 \cdot hops_{ml}$ and $w_{ml}^{lb\_dep} = 0.62 \cdot hops_{ml}$, where $hops$ indicates the number of hops of the shortest path between the respective nodes. The unit here is $Mbps$.

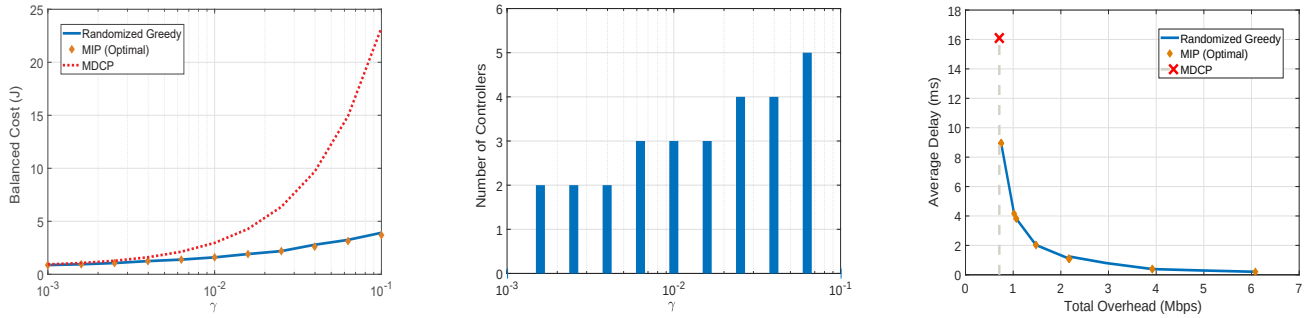
Figures 6(a)-6(b) depict the delay-overhead cost trade-off and the number of controllers placed in MANIAC network. In spite of different weights, our algorithm always has a close to optimal performance. The algorithm is capable to balance the delay and overhead cost as well as to place a proper

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2018.2876064, IEEE Transactions on Network and Service Management



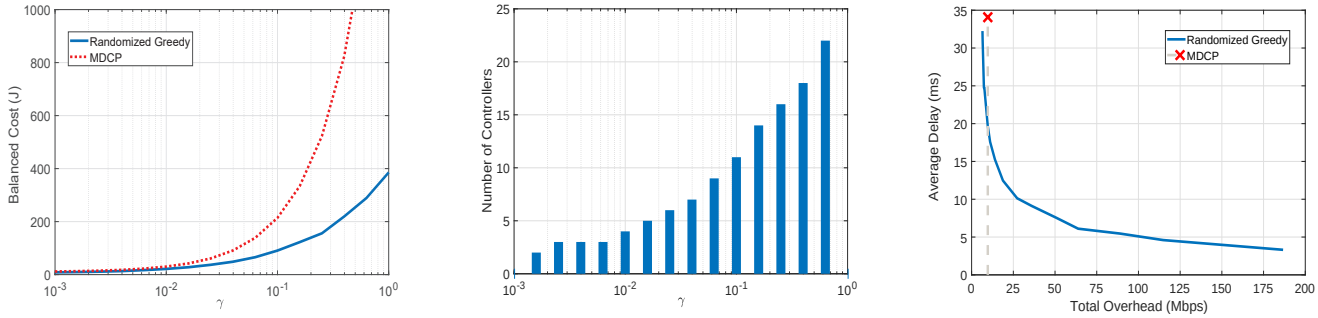
(a) Tradeoff in MANIAC network. (b) Controllers in MANIAC network. (c) Tradeoff in Barcelona network. (d) Controllers in Barcelona network.

Fig. 6. Evaluation results for the extended model, with parameters extracted from the measurements on OpenDaylight.



(a) Balanced cost for different weight γ. (b) Number of controllers for different weight γ. (c) Delay-overhead tradeoff.

Fig. 7. Evaluations on MANIAC network [30]. (a) Balanced cost, (b) number of placed controllers and (c) tradeoff between delay and overhead for two different algorithms.



(a) Balanced cost for different weight γ. (b) Number of controllers for different weight γ. (c) Delay-overhead tradeoff.

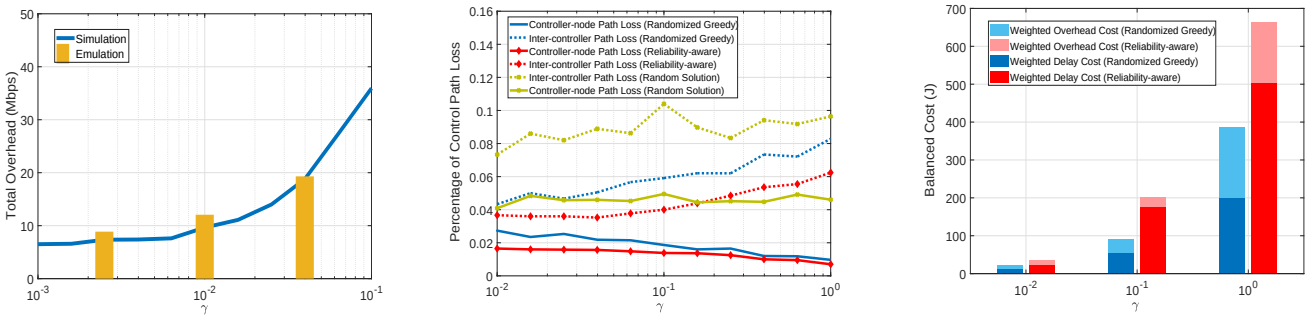
Fig. 8. Evaluations on Barcelona network [31]. (a) Balanced cost, (b) number of placed controllers and (c) tradeoff between delay and overhead for two different algorithms.

number of controllers. Figures 6(c)-6(d) depict the results for the Barcelona network, demonstrating the scalability of the algorithm.

Evaluations on ONOS. Next, we replace the controllers with ONOS. According to our analysis in Section II and the Appendix, ONOS's synchronization strategy can be regarded as a leaderless one. Therefore we verify both our small-scale MIP method and the large-scale randomized greedy algorithm. We set $w^a_{ml} = 0.019 \cdot hops_{ml}$, $w^{con}_{ml} = 0.04579 \cdot hops_{ml}$ and $w^{dep}_{ml} = 0.00793 \cdot hops_{ml}$ Most of the state-of-the-art methods require that the number of placed controllers is fixed and known in advance (e.g., see the review of related work in Section VI). Therefore, it would not be fair to compare the above with our methods which optimize both the number and location of controllers. Interestingly, there is a method in the literature that explores the same solution space with our work. Namely, the MDCP method (Algorithm 3 in [33]) is a greedy procedure that places controllers based on the degrees of the nodes and the inter-controller traffic. The latter is estimated by the diameter of the topology graph, rather than based on actual measurements. In next subsections, we will show a detailed comparison among randomized greedy algorithm, MIP and MDCP.

Impact of weight γ. Figure 7(a) depicts the evaluation results for the MANIAC network. Running on a common CPU machine, MIP typically takes 30 seconds to return the optimal solution using the built-in optimizer of MATLAB. On the other hand, Randomized Greedy algorithm takes much shorter time.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2018.2876064, IEEE Transactions on Network and Service Management



(a) Overhead measured by Mininet emulation. (b) Control path loss for different weight γ. (c) Balanced cost for different weight γ.

Fig. 9. (a) Verification on the overhead values by Mininet emulations. (b), (c) Performance comparison with a state-of-the-art reliability-aware algorithm [34] on the Barcelona network.

Within 0.1 second, we are able to repeat the algorithm for 200 times and pick the best result. Overall, *Randomized Greedy has a performance very close to the optimal for all values of* γ, while the performance of MDCP drops when γ is large (i.e. more weights are put on the delay rather than overhead cost). In these cases, *our algorithm has up to 3 quarters lower balanced cost than MDCP*.

We need to emphasize that many of the previous controller placement works used clustering methods such as k-median [12]. An issue is that these methods do not optimize the number of placed controllers, but take it as an input. In contrast, Randomized Greedy algorithm can adjust the number of placed controllers based on the weight (Figure 7(b)). When overhead cost is preferred, the algorithm tends to deploy fewer controllers so that the inter-controller communications reduce. When delay cost is preferred, however, it places more controllers so that the data plane nodes can reach a controller with smaller delay.

By iterating a large range of weight values, in Figure 7(c), we depict the *trade-off curve between the average delay and the total traffic overhead*, in order to show more details of the performance. The usage of measured data of overhead and delay empowers Randomized Greedy algorithm with two improvements over MDCP, which makes decisions based on some topology metrics only, such as network diameter and nodes degree. First, with similar total overhead, the average delay achieved by Randomized Greedy is lower than MDCP. Besides, Randomized Greedy algorithm is able to further decrease the delay by tolerating some additional overhead, and vice versa.

Figure 8(a)-8(c) repeat the evaluations for the Barcelona network. While MIP cannot run in reasonable time for this larger network, Randomized Greedy algorithm is scalable. It takes 0.8 second in average to finish the algorithm for 200 times. The qualitative results are similar with the MANIAC network. When γ is relatively large, the gains of our algorithm over MDCP are more pronounced. For the same total overhead, the Randomized Greedy algorithm achieves smaller average delay than MDCP.

Emulations vs simulations. To verify the accuracy of our model and evaluation results, we use Mininet to run emulations on the same topology we used for the MATLAB simulations. Mininet allows us to take measurements on the real traffic overhead in the same way as we did in Section II. In Figure 9(a), a consistency is shown in both the value and tendency of overheads between the simulations and emulations. This implies that our model successfully captures the pattern of the control traffic and it is capable for providing guidance in realistic cases.

Interplay between different objectives. Though concentrating on delay and overhead costs, our algorithm will have an impact on other metrics as well. One important metric is the reliability of control paths. Compared with wired networks like data centers, one significant difference in wireless edge networks is that links are generally less reliable. The operator must consider failure scenarios where a portion of links cease working. The work in [34] defines *control path loss*, which is the expected number of disconnected control paths (including both controller-node and inter-controller connections) given the probability of network link failure. This work proposes a simulated annealing algorithm achieving very good reliability. In order to investigate the reliability of our solution, we use this algorithm for comparison. To simplify, we assume every link in the Barcelona network may fail independently with the same probability, e.g., 0.01. The connections to the cloud are usually in another channel and with longer distances, therefore it is reasonable to assume the links have a lower quality. Here we assume they fail with doubled probability. Since the algorithm in [34] is not able to determine the amount of placed controllers, we set the same value as our algorithm.

Figure 9(b) depicts the percentage of control path loss. Compared with the reference (Reliability-aware) algorithm, our algorithm achieves a close performance in the controller-node path loss. Although we have larger inter-controller path loss when γ is large, by calculating the performance of random placement solutions, it shows that our algorithm still gets a significantly lower inter-controller path loss than average. Moreover, in practice the Reliability-aware algorithm may not succeed in finding the optimal amount of placed controllers, which may incur a deteriorated performance. Importantly, our algorithm has the ability to balance the two reliability metrics. When γ is large, it tends to decrease the distance between

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2018.2876064, IEEE Transactions on Network and Service Management

TABLE III
RELATED WORKS ON CONTROLLER PLACEMENT.

Reference	Objective	Number of controllers	Method	Testbed
Delay-centric approaches				
[12]	min. average & worst-case delay	given	k-median & k-center	✗
[35]	min. worst-case delay s.t. controller capacity	given	capacitated k-center	✗
[36]	min. controller-node & inter-controller delay	given	pareto evolutionary	✓
[37]	min. number of controllers s.t. delay probability	optimized	integer programming	✗
[38]	min. number of controllers & delay & load imbalance	optimized	quadratic integer programming	✗
Reliability-centric approaches				
[34]	min. control path loss	given	greedy & simulated annealing	✗
[39]	min. number of controllers s.t. a robust tree	given	k-critical	✗
[40]	max. number of nodes survived after failures	given	clustering	✗
[41]	max. path diversity between nodes & controllers	given	proximity-based heuristics	✗
[42]	min. delay, load imbalance & controller-less nodes after failures	given	pareto simulated annealing	✗
Cost-centric approaches				
[43]	min. deployment & connection cost for different types of controllers	optimized	integer programming	✗
[44]	min. energy cost of controller-node communication	given	genetic	✗
[33]	min. controller-node & inter-controller overheads	optimized	facility location	✗
[45]	min. controller-node & inter-controller overheads	optimized	bargaining game	✗
Adaptive approaches				
[46]	min. node reassignment cost & control overheads	optimized	greedy & simulated annealing	✗
[47]	min. number of controllers s.t. worst-case delay	optimized	network partitioning	✗
[48]	min. CAPEX & OPEX associated to controllers	optimized	non-zero-sum game	✗
This work	min. delay, controller-node & inter-controller overheads	optimized	linearization & submodularity	✓

controllers and data plane nodes (so as to reduce delay). As a result, controller-node connections acquire higher reliability. In a similar way, when γ is small, inter-controller connections acquire higher reliability. Meanwhile, Figure 9(c) shows that our algorithm achieves significantly lower delay and overhead cost than the reliability-aware reference algorithm.

VI. RELATED WORK

In this section we present the related work on the SDN controller placement problem. We also put our work in perspective in Table III that lists the main papers that tackle this problem, categorized based on their contributions.

Delay-centric approaches: The delay between controllers and data plane nodes is especially critical for quickly exchanging messages required by the SDN protocol. Most of the related works targeted wide area networks that extend over a large geographical distance with large capacity links. In this context, Heller et al. [12] recognized that the problem of placing a given number k of controllers to minimize the average and worst-case delay can be modeled as a k-median and a k-center problem, respectively. Yao et al. [35] explored the impact of controller capacity limitations on the worst-case delay, and proposed an algorithm inspired by the capacitated k-center problem. Zhang et al. [36] experimentally showed that the interaction between the controllers can affect the reaction time perceived by the data plane nodes. Motivated by this, an evolutionary algorithm that finds a Pareto optimal solution with respect to the average controller-node and inter-controller delay metrics was proposed. In the context of a wireless network, Abdel-Rahman et al. [37] modeled the controller-node delay probability assuming the TDMA strategy, and formulated an integer program for minimizing the number of placed controllers. Sudheera et al. [38] augmented to the above objective function components which represent controller-node delay and controller load imbalance costs in a vehicular ad hoc network, and formulated the controller placement problem as a quadratic integer program.

Reliability-centric approaches: Since disruptions in the network can isolate data plane nodes from controllers, it is of great importance to improve the reliability of SDN. Previous works defined various reliability metrics. Hu et al. [34] focused on the minimization of the expected percentage of control path loss which was solved using greedy and simulated annealing based heuristics. The tradeoff between reliability and delay was also explored through simulations. Jimenez et al. [39] proposed an algorithm of different flavor, named k-critical, which creates a robust topology with the minimum number of controllers to deal with network failures and balance the load among the controllers. Guo et al. [40] performed a cascading failure analysis and applied a clustering algorithm to minimize the number of nodes survived at the steady stage. Müller et al. [41] proposed heuristic algorithms to maximize the number of node-disjoint paths between controllers and data plane nodes (path diversity). Lange at al. [42] investigated the delay, load imbalance and number of controller-less nodes (i.e., nodes isolated from controllers) caused by network failures. A simulated annealing algorithm that returns a pareto optimal solution was presented.

Cost-centric approaches: Another aspect of the problem lies on the costs associated to the deployment and operation of controllers. These include the expenses to buy the controllers, if specialized equipment is required, manually install them and connect them in the network. Sallahi et al. [43] proposed an optimal model aiming to calculate the optimal number, location, and type of controllers that minimizes the overall cost.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2018.2876064, IEEE Transactions on Network and Service Management

A limitation of this approach is that it requires the solution to an integer program which does not scale well for large networks. Hu et al. [44] studied the problem of minimizing the energy cost associated to the communication between the controllers and data plane nodes and proposed a genetic heuristic algorithm. However, the energy cost associated to the inter-controller communication was not considered. Another operational cost component, which is central to our work, is the bandwidth overhead of inter-controller and controller-node communication. Su et al. [33] and Ksentini et al. [45] considered the minimization of the above two types of overheads and proposed a facility-location inspired heuristic and a bargaining game based pareto optimal solution, respectively. While these are perhaps the closest to our work, they were based on a simpler model for overheads and did not balance overheads with delay. In contrast, our model is driven by empirical measurements on delay and overheads from a multi-controller edge system we developed.

Adaptive approaches: For completeness, we should stress that dynamic algorithms which periodically adapt the controller placement solution were proposed in [46], [47] and [48]. In this way, the required controllers can be dynamically added or deleted and data plane nodes can be reassigned to different controllers based on traffic dynamics. The above algorithms are optimized to minimize the number of reassignments, the number of added controllers and other CAPEX and OPEX costs associated to controllers.

Our previous approaches: We have previous work [1] discussing the controller placement at the edge, whose model and conclusion are limited to ONOS controller, though. In this paper, we consider different controller synchronization strategies and present corresponding models, algorithms and experiment results. We also conduct evaluations with more different objectives, such as reliability.

VII. Conclusion

In this paper, we studied the SDN controller placement problem in edge network architectures. Our work combines strong experimentation results along with valid theoretical modeling and analysis. Namely, we built a testbed of a multi-controller edge system and described the sensitivity of delay on the controller placement as well as the magnitude and shape of traffic overheads. Guided by these findings, we presented a methodology that yields a set of optimal controller locations and assignment of nodes to controllers. Evaluation results demonstrated significant performance gains over state-of-the-art methods, and provided insights about the interplay between various performance and reliability objectives.

Appendix
Controller Traffic Analysis

Controller-node Traffic. When mentioning controller-node traffic, we refer to the traffic between the controllers and data plane nodes. Namely, controllers and nodes exchange various messages through a specific protocol, which is OpenFlow in most cases, including periodic heartbeat messages and statistic

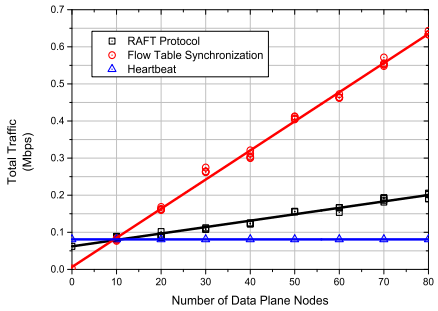


Fig. 10. Different inter-controller messages in ONOS.

requests/replies. It is intuitive that such overheads grow when the network scales up. Moreover, there are also overheads related to the routing of packet flows. When a new flow is generated and a node receives packets that cannot be matched in its forwarding table, it will report to the controller through *PacketIn* messages. On the contrary, the controller may install new forwarding rules to nodes using *FlowMod* messages. Therefore, the overhead is also influenced by the number of flows. The more frequently new flows emerge, the larger overhead is needed to install forwarding rules.

Inter-controller Traffic. In ONOS, multiple protocols for controller synchronization coexist [14]. The first one is the anti-entropy gossip protocol [18]. With an interval, a controller sends the information (network topology, flow tables, etc.) within its domain (nodes assigned to it) to a random peer controller. The second algorithm is RAFT. ONOS uses it to synchronize controller-node assignments within the cluster, generating an overhead also related to the controller load. In Figure 10, we classify the captured inter-controller traffic in the same setting as Section II. Compared with RAFT traffic, the flow tables synchronization following gossip protocol and the heartbeat massages between each pair of controllers generate the majority of overheads. Besides, although RAFT is leader-based, ONOS deployed multiple RAFT clusters. Every controller acts as leader in specific clusters. Taking all of above into consideration, we can regard the synchronization of ONOS controllers as a leaderless approach. In OpenDaylight, RAFT [19] is the only protocol for controller synchronization, therefore all the traffics are leader-based.

Appendix
Proof of Lemma 1

Consider an optimal assignment policy $\mathbf{y}^o$ with a node n^o being assigned to the controller at a node m^o, i.e., $y_{n^o m^o} = 1$. Clearly, it should be $x_{m^o} = 1$. Let us assume that there is another node $m^h \neq m^o$ such that $x_{m^h} = 1$ and:

$$\gamma d_{nm^h} + w^a_{nm^h} + \sum_{l:x_l=1} w^{dep}_{m^h l} < \gamma d_{nm^o} + w^a_{nm^o} + \sum_{l:x_l=1} w^{dep}_{m^o l} . \tag{32}$$

By reassigning node n^o to m^h instead of m^o the delay and assignment overhead cost are reduced by $d_{nm^o} + w^a_{nm^o} -$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2018.2876064, IEEE Transactions on Network and Service Management

$d_{nm^h} - w^a_{nm^h}$. At the same time, the synchronization cost is reduced by:

$$\sum_{l:x_l=1} w^{dep}_{m^o l} - \sum_{l:x_l=1} w^{dep}_{m^h l} \tag{33}$$

since the load-independent part of the synchronization cost is not affected by the above reassignment. Hence, the value of the objective function J is reduced by:

$$\gamma d_{nm^o} + w^a_{nm^o} + \sum_{l:x_l=1} w^{dep}_{m^o l} - \gamma d_{nm^h} - w^a_{nm^h} - \sum_{l:x_l=1} w^{dep}_{m^h l} \overset{(32)}{>} 0 \tag{34}$$

This contradicts our assumption of optimality of $\mathbf{y}^o$.

APPENDIX
PROOF OF LEMMA 2

Since the positively weighted sum of supermodular functions is also supermodular it suffices to show that each of the following three functions is supermodular.

$$f_{s,con}(X) = \sum_{m\in\mathcal{N}\cup\{c\}} \sum_{l\in\mathcal{N}\cup\{c\}} \mathbb{1}_{\{X_m\in X\}}\mathbb{1}_{\{X_l\in X\}} w^{con}_{ml} \tag{35}$$

$$f^n_{s,dep}(X) = \sum_{m\in\mathcal{N}\cup\{c\}} \sum_{l\in\mathcal{N}\cup\{c\}} \mathbb{1}_{\{X_m\in X\}}\mathbb{1}_{\{X_l\in X\}} w^{dep} y(\boldsymbol{x}_X)_{nm} \tag{36}$$

$$f^n_a(X) = \sum_{m\in\mathcal{N}\cup\{c\}} y(\boldsymbol{x}_X)_{nm}(\gamma d_{nm} + w^a_{nm}) \tag{37}$$

Here, $f_{s,con}(X)$ and $f^n_{s,dep}(X)$ denote the constant and load-dependent synchronization overhead cost, respectively. For a given node $n \in \mathcal{N}$, the function $f^n_a(X)$ denotes the total (delay and overhead) assignment cost.

Let us consider two placement sets A and B where $A \subset B \subset G$. We add an element $X_k \in G \setminus B$ to both placement sets. In other words, we place a controller at node k.

1) For the function $f_{s,con}$ and the placement set A, the marginal value of element X_k is:

$$\sum_{l\in\{l':X_{l'}\in A\}\cup\{c\}} w^{con}_{kl} + \sum_{m\in\{m':X_{m'}\in A\}\cup\{c\}} w^{con}_{mk} \ . \tag{38}$$

Since the above value increases if we replace A with $B \supset A$, the function $f_{s,con}$ is supermodular.

2) For the function $f^n_{s,dep}(X)$ and any placement set, the marginal value of element X_k is w^{dep} (which is independent of the assignment policy). Hence, the function $f^n_{s,dep}(X)$ is modular, which is a special class of supermodular functions.

3) For the function $f^n_a(X)$, we distinguish between two cases. **In the first case**, according to the placement set B, node n is assigned to the controller at node j where $\gamma d_{nj} + w^a_{nj} \leq \gamma d_{nk} + w^a_{nk}$. Then, the marginal value of element X_k is zero. For the placement set A, node n is assigned to the controller at node j'. If $\gamma d_{nj'} + w^a_{nj'} \leq \gamma d_{nk} + w^a_{nk}$ then the marginal value is again zero. However, if $\gamma d_{nj'} + w^a_{nj'} > \gamma d_{nk} + w^a_{nk}$ then the marginal value is $\gamma d_{nk} + w^a_{nk} - \gamma d_{nj'} - w^a_{nj'} < 0$. **In the second case**, according to the placement B, node n is assigned to a controller at node j where $\gamma d_{nj} + w^a_{nj} > \gamma d_{nk} + w^a_{nk}$. Then, the marginal value of element X_k is $\gamma d_{nk} + w^a_{nk} - \gamma d_{nj} - w^a_{nj}$. For the placement set A, node n is assigned to a controller at node j' where it must be $\gamma d_{nj'} + w^a_{nj'} \geq \gamma d_{nj} + w^a_{nj}$ since $A \subset B$. Hence, the marginal value is $\gamma d_{nk} + w^a_{nk} - \gamma d_{nj'} - w^a_{nj'} \leq \gamma d_{nk} + w^a_{nk} - \gamma d_{nj} - w^a_{nj}$.

REFERENCES

[1] Q. Qin, K. Poularakis, G. Iosifidis, L. Tassiulas, "SDN Controller Placement at the Edge: Optimizing Delay and Overheads", *IEEE Infocom*, 2018.

[2] C. Mims, "Forget 'The Cloud';'The Fog'Is Tech's Future", *The Wall Street Journal*, May 2014.

[3] M. Peng, S. Yan, K. Zhang, C. Wang, "Fog Computing based Radio Access Networks: Issues and Challenges", *IEEE Network, vol. 30, no. 4, pp. 46-53*, 2016.

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Edge Computing: Vision and Challenges", *IEEE Internet of Things Journal, vol. 3, no. 5*, 2016.

[5] K. Poularakis, G. Iosifidis, G. Smaragdakis, L. Tassiulas, "One Step at a Time: Optimizing Incremental SDN Upgrades in ISP Networks", *IEEE Infocom*, 2017.

[6] J. Wang, Y. Miao, P. Zhou, M. S. Hossain, and S. Mizanur, "A Software Defined Network Routing in Wireless Multihop Network", *Journal of Network and Computer Applications, vol. 85, pp. 76-83*, 2017.

[7] K. Poularakis, G. Iosifidis, L. Tassiulas, "SDN-enabled Tactical Ad Hoc Networks: Extending Programmable Control to the Edge", *IEEE Communications Magazine, vol. 56, no. 7, pp. 132-138*, July 2018.

[8] K. Poularakis, Q. Qin, E. Nahum, M. Rio, L. Tassiulas, "Bringing SDN to the Mobile Edge", *IEEE Smart World Congress, DAIS workshop*, 2017.

[9] H. Xu, Z. Yu, C. Qian, X. Li, Z. Liu, "Minimizing Flow Statistics Collection Cost of SDN Using Wildcard Requests", *in Proc. IEEE Infocom*, 2017.

[10] https://www.opendaylight.org/

[11] http://onosproject.org/

[12] B. Heller, R. Sherwood, N. McKeown, "The Controller Placement Problem", *in Proc. HotSDN*, 2012.

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

[14] A. Muqaddas, A. Bianco, P. Giaccone, G. Maier, "Inter-controller Traffic in ONOS Clusters for SDN Networks", *in Proc. IEEE ICC*, 2016.

[15] https://openvswitch.org

[16] B. Lantz, B. Heller, N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-defined Networks", *in Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, ACM, pp. 19*, 2010.

[17] https://iperf.fr/

[18] R. Van Renesse, D. Dumitriu, V. Gough, C. Thomas, "Efficient Reconciliation and Flow Control for Anti-entropy Protocols", *in Proc. ACM Workshop on Large-Scale Distributed Systems and Middleware*, 2008.

[19] D. Ongaro and J. K. Ousterhout, "In Search of an Understandable Consensus Algorithm", *in USENIX Annual Technical Conference*, 2014.

[20] D. Shmoys, E. Tardos, and K. Aardal, "Approximation Algorithms for Facility Location Problems", *in Proc. ACM STOC*, 1997.

[21] F. Glover, E. Woolsey, "Converting the 0-1 Polynomial Programming Problem to a 0-1 Linear Program", *Operations Research, vol 22, no. 1, pp. 180-182*, 1974.

[22] K. Jain, V. V. Vazirani, "Approximation Algorithms for Metric Facility Location and K-median Problems using the Primal-Dual Schema and Lagrangian Relaxation", *in Proc. Journal of the ACM (JACM), vol. 48, no. 2, pp. 274-296*, 2001.

[23] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, V. Pandit, "Local Search Heuristics for K-median and Facility Location Problems", *in Proc. SIAM Journal on Computing, vol. 33, no. 3, pp. 544-562*, 2004.

[24] S. Li, "A 1.488 Approximation Algorithm for the Uncapacitated Facility Location Problem", *in Proc. Information and Computation, vol. 222, pp. 45-58*, 2013.

[25] V.P. Il'ev, "An Approximation Guarantee of the Greedy Descent Algorithm for Minimizing a Supermodular Set Function", *Discrete Applied Mathematics, vol. 114, no. 1-3, pp. 131-146*, 2001.

[26] U. Feige, V. Mirrokni, J. Vondrak, "Maximizing Non-monotone Submodular Functions", *in Proc. IEEE FOCS*, 2011.

[27] S.O. Gharan, J. Vondrak, "Submodular Maximization by Simulated Annealing", *in Proc. ACM/SIAM SODA*, 2011.

[28] M. Feldman, J. Naor, R. Schwartz, "Nonmonotone Submodular Maximization via a Structural Continuous Greedy Algorithm", *in Proc. ICALP*, 2011.
[29] N. Buchbinder, M. Feldman, J. Naor, R. Schwartz, "A Tight Linear Time (1/2)-Approximation for Unconstrained Submodular Maximization", *in Proc. IEEE FOCS*, 2012.
[30] A. Hilal, J.N. Chattha, V. Srivastava, M.S. Thompson, A.B. MacKenzie, L.A. DaSilva, P. Saraswati, CRAWDAD dataset vt/maniac: http://crawdad.org/vt/maniac/20081101, https://doi.org/10.15783/C78W2V
[31] A. Neumann, E. López, L. Navarro, "An evaluation of BMX6 for Community Wireless Networks", *CNBuB*, 2012.
[32] Publicly available code: https://www.dropbox.com/s/rdhu8esut7lq6ct/public_code-cp.zip?dl=0
[33] Z. Su, M. Hamdi, "MDCP: Measurement-Aware Distributed Controller Placement for Software Defined Networks", *in Proc. IEEE ICPADS*, 2015.
[34] Y. Hu, W. Wendong, X. Gong, X. Que, C. Shiduan, "Reliability-aware Controller Placement for Software-defined Networks", *in Proc. IFIP/IEEE IM*, 2013.
[35] G. Yao, J. Bi, Y. Li, L. Guo, "On the Capacitated Controller Placement Problem in Software Defined Networks", *IEEE Communications Letters, vol. 18, no. 8, pp. 1339-1342*, 2014.
[36] T. Zhang, P. Giaccone, A. Bianco, S.D. Domenico, "The role of the Inter-controller Consensus in the Placement of Distributed SDN Controllers", *Computer Communications, vol. 113, no. 15, pp. 1-13*, 2017.
[37] M.J. Abdel-Rahman, E.A. Mazied,, A. MacKenzie, S. Midkiff, M.R. Rizk, M. El-Nainay, "On Stochastic Controller Placement in Software-defined Wireless Networks", *IEEE WCNC*, 2017.
[38] K. Sudheera, K. Liyanagea, M. Maa, P.H.J. Chong, "Controller Placement Optimization in Hierarchical Distributed Software Defined Vehicular Networks", *Computer Networks, vol. 135, pp. 226-239*, 2018.
[39] Y. Jimenez, C. Cervello-Pastor, A.J. Garcia, "On the Controller Placement for Designing a Distributed SDN Control Layer", *in Proc. IFIP Networking*, 2014.
[40] M. Guo, P. Bhattacharya, "Controller Placement for Improving Resilience of Software-defined Networks", *IEEE ICNDC*, 2013.
[41] L. F. Müller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, M. P. Barcellos, "Survivor: an Enhanced Controller Placement Strategy for Improving SDN Survivability", *IEEE GLOBECOM*, 2014.
[42] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, 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.
[43] A. Sallahi, 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.
[44] Y. Hu, T. Luo, N.C. Beaulieu, C. Deng, "The Energy-Aware Controller Placement Problem in Software Defined Networks", *IEEE Communications Letters, vol. 21, no.4, pp. 741-744*, 2017.
[45] A. Ksentini, M. Bagaa, T. Taleb, I. Balasingham, "On Using Bargaining Game for Optimal Placement of SDN Controllers", *in Proc. IEEE ICC*, 2016.
[46] M.F. Bari, A.R. Roy, S.R. Chowdhury, Q. Zhang, M.F. Zhani, R. Ahmed, R. Boutaba, "Dynamic Controller Provisioning in Software Defined Networks", *in Proc. IEEE CNSM*, 2013.
[47] Md.T.I. ul Huque, W. Si, G. Jourjon, V. Gramoli, "Large-Scale Dynamic Controller Placement", *IEEE Transactions on Network and Service Management vol. 14, no. 1, pp. 63-76*, 2017.
[48] H. K. Rath, V. Revoori, S. M. Nadaf, A. Simha, "Optimal Controller Placement in Software Defined Networks (SDN) Using a Non-zero-sum Game", *IEEE WoWMoM*, 2014.

Qiaofeng Qin received his B.S. degree in electrical engineering from Peking University, China, in 2015. He is currently a PhD student in the Department of Electrical Engineering at Yale University. His research focuses on the area of communication networks, including Software Defined Networking and Internet of Things.

Konstantinos Poularakis obtained the Diploma, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Thessaly, Greece, in 2011, 2013 and 2015 respectively. Currently, he is a Post-doc researcher at Yale University. His research interests lie in the broad area of network optimization. He has been honored with several awards and scholarships during his studies, from sources including the Greek State Scholarships foundation (IKY), the Center for Research and Technology Hellas (CERTH), the Alexander S. Onassis Public Benefit Foundation, and the Bodossaki Foundation. He also received the Best Paper Award at IEEE Infocom 2017.

George Iosifidis is the Ussher Assistant Professor in Future Networks, at Trinity College Dublin, Ireland. He received a Diploma in Electronics and Communications, from the Greek Air Force Academy (Athens, 2000) and a PhD degree from the Department of Electrical and Computer Engineering, University of Thessaly in 2012. He was a Postdoctoral researcher ('12-'14) at CERTH-ITI in Greece, and Postdoctoral/Associate research scientist at Yale University ('14-'17). He is a co-recipient of the best paper awards in WiOPT 2013 and IEEE INFOCOM 2017 conferences, and has received an SFI Career Development Award in 2018.

Sastry Kompella (S'04-M'06-SM'12) received the B.E. degree in electronics and communication engineering from Andhra University, Visakhapatnam, India, in 1996, the M.S. degree in electrical engineering from Texas Tech University, Lubbock, TX, USA, in 1998, and the Ph.D. degree in electrical and computer engineering from the Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, in 2006. Currently, he is the Section Head for the Wireless Network Research Section under the Information Technology Division, U.S. Naval Research Laboratory, Washington, DC, USA. His research interests include various aspects of wireless networks, from mobile ad hoc to underwater acoustic networks, with specific focus towards cognitive and cooperative network optimization and programmable networking.

Leandros Tassiulas (S'89-M'91-SM'05-F'07) is the John C. Malone Professor of Electrical Engineering and member of the Institute for Network Science at Yale University. His research interests are in the field of computer and communication networks with emphasis on fundamental mathematical models and algorithms of complex networks, architectures and protocols of wireless systems, sensor networks, novel internet architectures and experimental platforms for network research. His most notable contributions include the max-weight scheduling algorithm and the back-pressure network control policy, opportunistic scheduling in wireless, the maximum lifetime approach for wireless network energy management, and the consideration of joint access control and antenna transmission management in multiple antenna wireless systems. Dr. Tassiulas is a Fellow of IEEE (2007). His research has been recognized by several awards including the IEEE Koji Kobayashi computer and communications award (2016), the inaugural INFOCOM 2007 Achievement Award for fundamental contributions to resource allocation in communication networks, the INFOCOM 1994 and 2017 best paper award, a National Science Foundation (NSF) Research Initiation Award (1992), an NSF CAREER Award (1995), an Office of Naval Research Young Investigator Award (1997) and a Bodossaki Foundation award (1999). He holds a Ph.D. in Electrical Engineering from the University of Maryland, College Park (1991). He has held faculty positions at Polytechnic University, New York, University of Maryland, College Park, and University of Thessaly, Greece.