Flow Setup Time aware Minimum Cost Switch-Controller Association in Software-Defined Networks

Deze Zeng*, Chao Teng*, Lin Gu†, Hong Yao* and Qingzhong Liang*
*School of Computer Science, China University of Geosciences, Wuhan, China
†School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China
Email: deze@cug.edu.cn

***Abstract*—Software Defined Networks (SDN) emerged as a new network paradigm to address the customization and flexibility problems in traditional computer networks. In SDNs, a logical centralized programmable controller manages the whole networks by installing rules onto switches. It is widely regarded that one controller is restricted on both performance and scalability. To address these limitations, pioneering researchers advocate deploying multiple controllers in SDNs where each controller is in charge of a set of switches. This raises the switch-controller association problem on one switch shall be managed by which controller. In this paper, we specially investigate minimum cost switch-controller association (MC-SCA) problem on how to minimize the number of controllers needed in an SDN while guaranteeing the flow setup time. A quadratic integer programming model is first proposed and then transformed into an equivalent integer linear programming model to describe the MC-SCA problem, which is then proved as NP-Hard. We further propose a heuristic algorithm and extensively prove its high efficiency via simulations.**

I. Introduction

Software Defined Networking (SDN) emerged as a new network architectures and become a hot topic in the literature. By providing programmable hardware in SDNs, the networks can be dynamically configured and managed thanks to the separation of control plane and data plane. The control plane is moved to an external entity called controller. The controller is responsible for determining the forwarding rules on the forwarding devices through a control channel. Therefore, the controller is often regarded as the central brain and plays an important role in SDNs.

The most common SDN implementation adopts a centralized network control, where a controller manages and operates the network from a global view. Whenever a switch receives a new flow and finds no matching entry in the flow table, it immediately requires the controller to install appropriate forwarding rules along the desired flow path. However, in a large-scale SDN deployment, this rudimentary centralized approach has several limitations on both performance and scalability. On one hand, a single controller usually has a limited capacity and hence cannot handle large number of flows originating from all the infrastructure switches. In this case, some request packets may have to be dropped, incurring negative effect to the network performance. On the other hand, the latencies between the single controller and the switches situated at geographically distributed locations are highly varied. To some switches far away from the controller, long flow setup time may be introduced. This severely limits the network performance, especially to SDN-based wireless area networks (WANs).

To address these limitations, pioneering researchers advocate deploying multiple controllers that work cooperatively to manage network traffic flows [1], [2]. Much effort has been devoted to addressing various problems related to multiple-controller in SDNs. For example, Heller et al. [3] study the controller placement problem and analyze the impact of the controller locations on the average and worst-case controller-to-switch propagation delay. The controller placement problem is then extensively studied from different aspects, e.g., [4]–[8]. The controller placement solutions are applied in pre-deployment period. When multiple controllers are deployed, it is also essential to consider the association between the controllers and switches, i.e., switch-controller association, as it also affects the performance of the network. Specially, to make the network cost-efficient, one feasible solution is to minimize the number of activated controllers. Of course, no matter how many controllers are actually used, it is first required that the flow setup time is guaranteed. Therefore, we are motivated to investigate the flow setup time aware switch-controller association problem aiming at minimizing the number of controllers, i.e., minimum cost switch-controller association (MC-SCA) problem. The main contributions of this paper are as follows.

- We formulate MC-SCA with the consideration of transmission time between switches and controllers into a quadratic integer programming (QIP) problem, which is then transformed into an equivalent integer linear programming (ILP) problem. We also formally prove that MC-SCA is NP-hard.
- A heuristic algorithm is proposed. Its high efficiency is extensively validated by the fact it much approaches the optimal solution.

The rest of this paper is organized as follows. Section II provides a brief overview of related work. Section III elaborates our system model. Problem formulation and heuristic algorithm are proposed in Sections IV and V, respectively. Section VI shows our performance evaluation results. Finally,

QSHINE 2015, August 19-20, Taipei, Taiwan

DOI 10.4108/eai.19-8-2015.2260893

Section VII concludes our work.

II. Related Work

A. Controller in SDNs

As it is widely agreed that single controller SDN suffers from the performance and scalability, researchers therefore advocate multiple controllers in SDNs to address these limitations. When multiple controllers shall be deployed, the controller placement problem is raised. Heller et al. [3] first study the controller placement problem and show the implications of changing the number of controllers for the latency between switches and controllers. Schmid et al. [5] describe a local network view of the controllers and discuss the design of a distributed control plane. In [7], it is demonstrated that the switch-over time depends on the latencies between networking devices and the controllers. Hu et.al [6] present four controller placement algorithms to maximize the reliability of controller placement. Tootoonchian et al. [4] show that long propagation delays among controllers limit the network convergence time, and affect the controller ability to respond to network events. Bari et al. [9] attempt to solve the dynamic controller provisioning problem through an ILP and design algorithms to minimizes flow setup times by dynamically changing the number of controllers and their locations. Recently, Rath et al. [10] use game theory for dynamically mapping the switches to controllers for reducing the average controller-switch latencies and balancing loads on controllers.

B. Modeling, Analysis and Optimization in SDNs

Some efforts are also contributed to the modeling, analysis and optimization in SDNs. Hu et al. [11] build performance models for response time to evaluate the scalability of three controller structures, including centralized, decentralized and hierarchical. Jarschel et al. [12] abstract the OpenFlow architecture as a feedback-oriented queuing system model, divided into an $M/M/1$ forward queuing system and an $M/M/1-S$ feedback queuing system. This model is derived for the forwarding speed and blocking probability of an OpenFlow switch combined with an OpenFlow controller, which can be used to estimate the packet sojourn time and the probability of lost packets in such a system. Their study also gives hints to developers and researchers on how an OpenFlow architecture performs for given parameters. Yao et al. [13] study the issue of controller capacity defined as the number of switches a controller can manage. To this goal, they model the flow set-up requests from switches to controller as a batch arrival process $M^k/M/1$ and derive the expression of average flow service time, which is used to evaluate the controller capacity.

III. System Model

We consider an SDN shown in Fig. 1. Following the OpenFlow model [14], the network consists of a controller plane and a data plane. The controller plane is formed by a set I of controllers. The data plane is composed of a set J of OpenFlow switches which forward data flow according to the flow table. Without loss of generality, we assume that a switch can be reached from any controller. The transmission latency between a controller $i \in I$ and a switch $j \in J$ is denoted as d_{ij}.

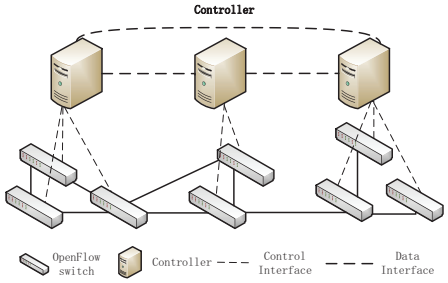

Fig. 1. Network Model

TABLE I
Notations

I	a set of controllers I
J	a set of switches J
i	controller $i \in I$
j	switch $j \in J$
c_i	a binary variable representing whether controller $i \in I$ is activated
e_{ij}	a binary variable representing whether switch j is associated with controller i
T	the maximum allowable flow setup time required by a switch
d_{ij}	the transmission latency between switch j and controller i
λ_j	arrival rate of a packet sent by switch j
μ_i	the processing capacity of controller i

To further state our problem, we first elaborate the flow setup process as follows. In SDNs, each switch has a flow table where each entry corresponds to the operation rule (e.g., forwarding, discarding, packet header altering, etc.) for a flow. When a packet arrives at the OpenFlow switch, the switch extracts its header information and then matches it with the flow table entries. If the matching is successful, the switch executes the forwarding decision instantaneously. If the switch does not contain a matching rule, the packet is sent to the associated controller requesting for an action to execute. The controller will determine the rule to handle the packet and respond to the switchs request with an action to perform on all packets of this flow. A controller $i \in I$ can only manage the switches that associate to it.

From the perspective of controller, the flow setup requests may accumulate at the egress of controller and form a request queue. Following the SDN model presented in [12] and [13], we assume that a switch $j \in J$ generates flow setup requests following Poisson process with rate λ_j. The requests are processed by controller $i \in I$ with exponentially distributed service time with average value $1/\mu_i$, where μ_i is the average service rate.

The main symbols used in this paper are summarized in Table I.

IV. Problem Formulation

In this section, we first formulate the MC-SCA problem into a quadratic integer programming (QIP) problem. We then linearize the QIP into an equivalent ILP problem. We also prove the NP-hardness of the MC-SCA problem.

A. QIP Formulation

To represent the association relationship, we first define a binary variable e_{ij} to denote whether switch j is associated with controller i as:

$$e_{ij} = \begin{cases} 1, \text{ if switch } j \text{ is associated with controller } i, \\ 0, \text{ otherwise.} \end{cases} \quad (1)$$

1) Completeness Constraints: According to SDN philosophy, a switch must be associated with one controller such that it can be manipulated according to the communication requirements. That is,

$$\sum_{i \in I} e_{ij} = 1, \forall j \in J. \quad (2)$$

Note that a switch can be associated with a controller $i \in I$ provided that i is activated. To this end, we define a binary variable c_i to denote whether controller i is activated or not as follows:

$$c_i = \begin{cases} 1, \text{ if controller } i \text{ is activated,} \\ 0, \text{ otherwise.} \end{cases} \quad (3)$$

We must guarantee that each switch is assigned to an activated controller, i.e.,

$$e_{ij} \leq c_i, \forall i \in I, j \in J. \quad (4)$$

2) Flow Setup Time Constraints: As we know, the first packet (i.e., flow setup request) of a flow arriving at a switch $j \in J$ but with no entry matching in the flow table must go through the associated controller of j, say $i \in I$, to obtain the rule for the flow. The flow setup time has a critical impact on the system performance as the flow can pass j only after the rule is ready. A controller takes charge of the flow setup requests from all the switches in its management domain. A flow setup request queue is formed at each controller as illustrated in Fig. 2. The request from any switch can be regarded as an individual and independent Poisson process. As the sum of a set of independent Poisson processes is still a Poisson process and the request handling time is exponentially distributed, we can describe the process of handling flow setup requests on a controller using $M/M/1$ queuing model. Therefore, the queuing time at a controller i can be calculated as

$$\frac{1}{\mu_i - \sum_{k \in J} \lambda_k e_{ik}},$$

where $\sum_{k \in J} \lambda_k e_{ik}$ denotes the sum Poisson process arrival rate. To ensure that the queue is steady, a hidden condition must be met is that the sum of the arrival rate of all switches shall not be beyond the service rate provided by a controller. Therefore, we have:

$$\mu_i - \sum_{k \in J} \lambda_k e_{ik} > 0 \quad (5)$$

From the perspective of a switch, the flow setup time shall also take the transmission latency between its associated controller. To ensure the network performance, the total flow setup time experienced by any switch shall not exceed T, i.e.,

$$e_{ij} d_{ij} + \frac{1}{\mu_i - \sum_{k \in J} \lambda_k e_{ik}} \leq T, \forall i \in I, j \in J. \quad (6)$$

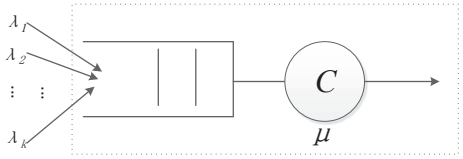

Fig. 2. Request Queue at an SDN Controller

When $\mu_i - \sum_{k \in J} \lambda_k e_{i,k} > 0$, (6) can be transformed into

$$\mu_i e_{ij} d_{ij} + \sum_{k \in J} T \lambda_k e_{ik} - d_{ij} \sum_{k \in J} \lambda_k e_{ij} e_{ik} \leq T\mu_i - 1, \quad \forall i \in I, j \in J. \quad (7)$$

3) QIP Formulation: We intend to minimize the number of activated controllers, which can be expressed as

$$\sum_{i \in I} c_i$$

Summarizing the above together, we can formulate the MC-SCA problem as:

$$\begin{aligned} \min : & \sum_{i \in I} c_i, \\ \text{s.t.} : & (2), (4), (5), (7), \\ & c_i \in \{0,1\}, e_{ij} \in \{0,1\}, \forall i \in I, j \in J, \end{aligned}$$

which is a QIP as there are quadratic terms, e.g., $e_{ij}e_{ik}$.

Theorem 1: The flow setup time aware minimum cost switch-controller association problem is NP-Hard.

Proof: Let us consider a special case of the MC-SCA problem by excluding the queuing time at each controller. In this case, whether a switch can be associated with a controller is only determined by the transmission time between them. For example, for a controller $j \in J$ and a switch $i \in I$, only when $d_{ij} \leq T$, i can be associated with j. We shall find out the number of activated controllers that are able to ensure the completeness constraints in (2) that any switch is associated to a controller. This is exactly a minimum set cover problem [15], which has been proved as NP-Hard. ■

B. QIP to ILP Transformation

We notice that it is possible to linearize the quadratic terms $e_{ij}e_{ik}$ by introducing new auxiliary variables

$$x_{ijk} = e_{ij}e_{ik}, \forall i \in I, j, k \in J$$

which can be equivalently replaced by the following linear constraints

$$x_{ijk} \leq e_{ij}, \forall i \in I, j, k \in J \quad (8)$$

$$x_{ijk} \leq e_{ik}, \forall i \in I, j, k \in J \quad (9)$$

$$x_{ij} \geq e_{ij} + e_{ik} - 1, \forall i \in I, j, k \in J \quad (10)$$

The constraint (7) can be then rewritten in linear form as

$$\mu_i e_{ij} d_{ij} + \sum_{k \in J} T \lambda_k e_{ik} - \sum_{k \in J} \lambda_k x_{ijk} d_{ij} \leq T\mu_i - 1, \quad \forall i \in I, j \in J \quad (11)$$

Algorithm 1 Allocation-Merge Algorithm

Input: controller set I, switch set J, the transmission delay $d_{ij}, i \in I, j \in J$, the maximum allowable setup time T, flow request rate $\lambda_j, j \in J$, the processing rate $\mu_i, i \in I$

Output: the number of activated controllers

1: Initialize three arrays: $e[i,j], c[i], s[j], i \in I, j \in J$ as 0
2: Select the controller with the smallest transmission latency d_{ij} for each switch
3: $e[i,j] \leftarrow 1$, $c[i] \leftarrow 1$, $s[j] \leftarrow 1$
4: **for all** $i \in I$ **do**
5: Calculate $m = \mu_i - \sum_{k \in J} \lambda_k e_{i,k}$
6: **if** $m > 0$ **then**
7: **for all** $j \in J$ **do**
8: Calculate $t = e_{i,j} d_{i,j} + \frac{1}{m}$
9: **if** $t > T$ **then**
10: $e[i,j] \leftarrow 0$, $s[j] \leftarrow 0$
11: **end if**
12: **end for**
13: **else**
14: Exclude the switches with the smallest arrival rates from i until $m > 0$
15: **end if**
16: **end for**
17: $\forall i \in I$, if no switch assigned to i, $c[i] \leftarrow 0$.
18: **for all** $j \in J$ and $s[j] = 0$ **do**
19: Associate j to an activated controller provided that (5) and (6) are satisfied.
20: **end for**
21: Merge the activated controllers provided that (5) and (6) are satisfied.

Thus, an ILP formulation for the MC-SCA problem can be obtained as

$$\min : \sum_{i \in I} c_i,$$
$$\text{s.t. : } (2), (4), (8) - (11),$$
$$c_i \in \{0,1\}, e_{ij} \in \{0,1\}, x_{ijk} \in \{0,1\}, \forall i \in I, j, k \in J.$$

V. Heuristic Algorithm

It is still computationally prohibitive to solve the ILP problem to get the optimal solution in large-scale SDNs. To address this problem, we propose a heuristic algorithm named "Allocation-Merge Algorithm" in this section.

Allocation-Merge Algorithm assigns switches to controllers according to the transmission latency under the constraints. The overall algorithm is presented in Algorithm 1, which mainly consists of three phases, initial association (lines 1-17), reassociation (lines 18-20) and merging (line 21). In the initial allocation phase, we first greedily associate each switch to the controller with the smallest transmission latency, provided that the setup time constraints are not violated with the consideration of queueing time by checking constraints (5) and (6). Therefore, after the initial association, there are some switches remaining in un-associated state. We shall then try to associate each of them onto an appropriate controller. In order to minimize the number of activated controllers, we first try to associate an un-associated switch to an activated controller. If no activated controller can accommodate the switch without violating constraints (5) and (6), we activate an inactive controller and associate the switch to it. After the second phase, we ensure that all the switches are successfully associated to appropriate controllers with guaranteed flow setup time. However, we notice that the performance can be further improved as some controllers are with small request load after the initial two phases. Therefore, we then try to merge the activated controllers with light loads to reduce the number of the controller. We iteratively merge two controllers into one provided that the flow setup time constraints are not violated. Finally, we obtain the controller activation and switch-controller association decisions.

VI. Performance Evaluation

In this section, we present our simulation-based performance evaluation results on the efficiency of our proposed algorithm, by comparing it (i.e., "AMA") against the optimal solution and another greedy-based heuristic algorithm "GA".The optimal results "Optimal" are obtained by solving the ILP using commercial solver Gurobi optimizer [16]. The basic idea of the greedy algorithm is to assign as many switches as possible to an activated controller without violating the resource capacity constraints and one controller is activated in each iteration. The whole process works as follows. At first, we decreasingly sort both the switches and the controllers according to their request rates and processing rates, respectively. We then iteratively activate the controllers and assign as many switches as possible to the activated controller in each iteration. At the same time, constraints (5) and (6) are checked to see whether the setup time constraints are satisfied or not. The iterative process stops when all switches are assigned.

To extensively investigate the performance of our heuristic algorithms, we vary the values of the number of controllers $|I|$, the number of switches $|J|$, the arrival rate of the flow request λ (here we consider uniform request rate for all switches), the maximum allowable flow setup time T in different group of simulations. 20 simulation instances in each group are conducted to get the average number of activated controllers. In each simulation instance, the transmission time between switches and controllers is randomly generated.

We first check how the maximum allowable setup time T affects the number of controllers. Fig. 3 presents the simulation results. In this group of simulations, we fix $|I| = 10$, $|J| = 20$, $\lambda = 8$, $\mu = 50$ and vary T from 1 to 30. The results are presented in Fig. 3(a). In Fig. 3(b), we set $|I| = 10$, $|J| = 20$, $\lambda \in [5, 10]$, $\mu \in [40, 60]$ and T from 1 to 30. From both Fig. 3(a) and Fig. 3(b), we can see that our proposed algorithm much approaches the optimal one and outperforms the greedy one, under any settings. Besides, it can be also noticed that the minimum cost shows as a decreasing function of the maximum allowable setup time. One more interesting thing is that all three algorithms converge and have the same number of controllers when the maximum allowable setup time is big enough. This is because, longer queuing time is tolerable with larger value of T. In this case, more requests can be

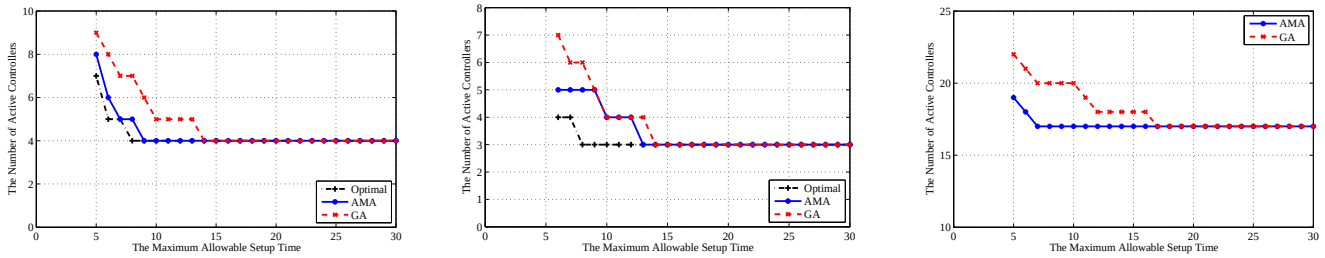

Fig. 3. On the effect of the maximum allowable flow setup time

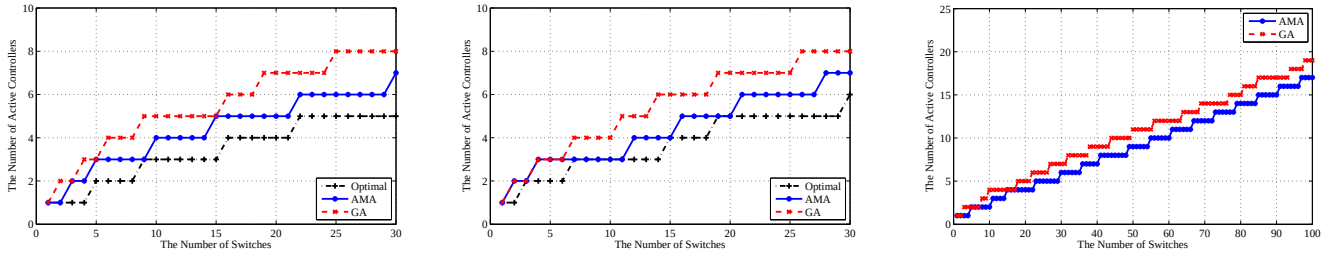

Fig. 4. On the effect of the number of switches

allocated to one controller and hence less activated controllers are needed. In order to further study the performance of our algorithms, we extend the scale of the network under the setting of $|I| = 25$, $|J| = 100$, $\lambda = 8$, $\mu = 50$ and T from 1 to 30. Due to the complexity of obtaining the optimal results, only "AMA" ad "GA" are reported in Fig. 3(c). We can still see that "AMA" still performs better than "GA" in large-scale networks.

Then, we study the effect of the number of switches to the minimum number of the activated controllers using different settings. Similar to the study on the value of maximum allowable setup time, three group of settings are also considered. In the first group of simulations, we fix $|I| = 10$, $\lambda = 8$, $\mu = 50$, $T = 8$ and vary the number of switches from 1 to 30 and show the results in Fig. 4(a). In the second group of simulations, we further vary the values of the flow request arrival rate $\lambda \in [5, 10]$, the processing rate of controller $\mu \in [40, 60]$ and show the results in Fig.4(b). Fig.4(c) gives the results under the settings of $|I| = 25$, $\lambda = 8$, $\mu = 50$, $T = 10$ and $|J|$ from 1 to 100. Once again, we can see that our algorithm performs much close to the optimal one and has obvious advantage over the greedy one. We also notice that the number of activated controllers needed increase with the number of switches. This is attributed to the fact that more controllers must be activated to ensure the setup time if there are more switches.

Next, Fig. 5 gives the results on the effect of the arrival rate under the setting of $T = 8$, $|I| = 10$, $|J| = 20$, $\mu = 50$ and λ from 1 to 15. From the figure, we can still see the high efficiency of our proposed Allocation-Merge Algorithm. In addition, it also shows that the number of the activated controllers increases with the increasing of the arrival rate. It is straightforward to know that more requests imply that more

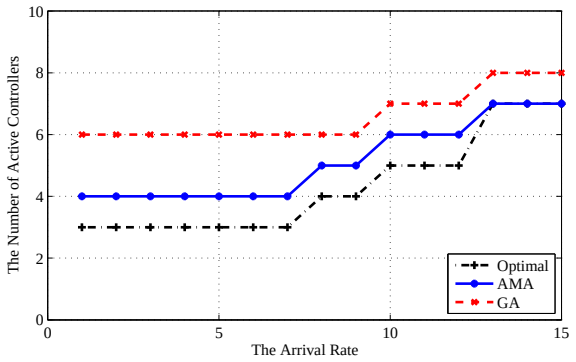

Fig. 5. On the effect of flow setup request arrival rate

controllers shall be activated to handle them.

Finally, in order to extensively show the the efficiency of our proposed algorithm, we carry out a group of experiments by randomly setting the request arrival rates on all switches in $[5, 10]$, the processing rates on the controllers in $[40, 60]$ and maximum allowable setup time in $[5, 15]$ in different simulation instances. We plot the cumulative distribution function (CDF) of the number of activated controllers for 200 instances in Fig. 6, from which we still see the high efficiency of our algorithm under any random settings. After extensively validating the efficiency of our algorithm, we further conduct a group of experiments to check the computation time of different algorithms. The results are reported in Table II. Obviously, both greedy and our heuristic algorithm require much less running time than solving the ILP to get the optimal solution. "AMA" needs only a little more running time than "Greedy".

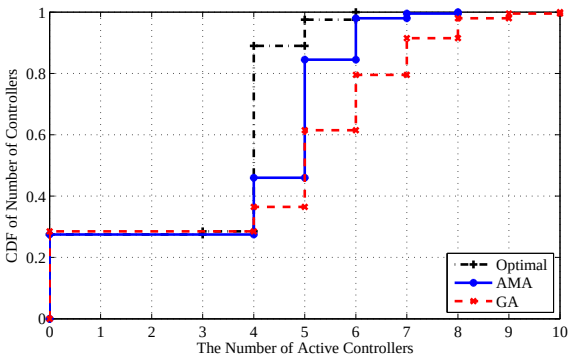

Fig. 6. CDF of the minimum number of controllers

TABLE II
RUNNING TIME

Instances	Optimal	Heuristic Algorithm	Greedy Algorithm
50	206.400718887	0.00139172755833	0.000393690880978
100	400.420835177	0.0026170858729	0.000931541952999
200	718.891972255	0.00520450670253	0.00142874264548
300	1279.7988438	0.00856356298393	0.00310085783613

VII. Conclusion

In this paper, we investigate the MC-SCA problem on how to minimize of the number of activated controllers with guaranteed flow setup time. By modeling the flow setup request process on a controller using an $M/M/1$ queue, we first formulate the MC-SCA into a QIP problem and further linearize it into an ILP problem. We also prove that the MC-SCA problem is NP-Hard. To address the computation complexity, we propose a heuristic algorithm. Through extensive simulations, the high efficiency of our algorithm is validated as it much approaches the optimal performance.

Acknowledgment

This research was supported in part by the NSF of China (Grant No. 61402425, 61272470), the Fundamental Research Funds for National University, China University of Geosciences (Wuhan) (Grant No. CUG14065, CUGL150829), the Provincial Natural Science Foundation of Hubei (Grant No. 2015CFA065).

References

[1] A. Tootoonchian and Y. Ganjali, "HyperFlow: A Distributed Control Plane for OpenFlow," in *Proceedings of the Internet Network Management Conference on Research on Enterprise Networking*. USENIX Association, 2010, pp. 3–3.

[2] 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*. ACM, 2012, pp. 19–24.

[3] B. Heller, R. Sherwood, and N. McKeown, "The Controller Placement Problem," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*. ACM, 2012, pp. 7–12.

[4] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On Controller Performance in Software-Defined Networks," in *USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE)*, vol. 54, 2012.

[5] 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*. ACM, 2013, pp. 121–126.

[6] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability-aware Controller Placement for Software-Defined Networks," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. IEEE, 2013, pp. 672–675.

[7] K. Nguyen, Q. T. Minh, and S. Yamada, "A Software-Defined Networking Approach for Disaster-Resilient WANs," in *22nd International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2013, pp. 1–5.

[8] Y. Jimenez, C. Cervello-Pastor, and A. J. Garcia, "On the Controller Placement for Designing a Distributed SDN Control Layer," in *Networking Conference, 2014 IFIP*. IEEE, 2014, pp. 1–9.

[9] 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, pp. 18–25.

[10] H. K. Rath, V. Revoori, S. Nadaf, and A. Simha, "Optimal Controller Placement in Software Defined Networks (SDN) using a Non-Zero-Sum Game," in *A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on*. IEEE, 2014, pp. 1–6.

[11] J. Hu, C. Lin, X. Li, and J. Huang, "Scalability of Control Planes for Software Defined Networks: Modeling and Evaluation."

[12] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, "Modeling and Performance Evaluation of an OpenFlow Architecture," in *Teletraffic Congress (ITC), 2011 23rd International*, Sept 2011, pp. 1–7.

[13] L. Yao, P. Hong, and W. Zhou, "Evaluating the Controller Capacity in Software Defined Networking," in *23rd International Conference on Computer Communication and Networks (ICCCN), 2014*, Aug 2014, pp. 1–6.

[14] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein *et al.*, *Introduction to algorithms*. MIT Press Cambridge, 2001, vol. 2.

[16] G. Optimization, "Gurobi optimizer reference manual," *URL: http://www. gurobi. com*, 2012.