On the Capacitated Controller Placement Problem in Software Defined Networks

Article *in* IEEE Communications Letters · August 2014

DOI: 10.1109/LCOMM.2014.2332341

4 authors, including:

Jun Bi
Tsinghua University
206 PUBLICATIONS **1,783** CITATIONS

SEE PROFILE

On the Capacitated Controller Placement Problem in Software Defined Networks

Guang Yao, Jun Bi, *Member, IEEE*, Yuliang Li, and Luyi Guo

***Abstract*—Controller placement is a key problem in software defined networks (SDNs). Previously, the solution to this problem only focused on propagation latency but ignored the load of controllers, which is a critical factor in real networks. In this letter, we define a capacitated controller placement problem (*CCPP*), taking into consideration the load of controllers, and introduce an efficient algorithm to solve the problem. The evaluation shows that the new strategy can significantly reduce the number of required controllers, reduce the load of the maximum-load controller, and reduce the radius stretches compared with the K-center strategy with dynamic controller provision or dynamic scheduling.**

***Index Terms*—SDN, placement, controller.**

I. Introduction

In software defined networks (SDN), the complexity of generating forwarding rules is offloaded to the controllers. Mismatched packets will be buffered or discarded until the corresponding flow entries are installed by the controller. Thus, the propagation latency between the switches and the associated controller is a key point for the performance of SDNs, and the placement of controllers became a critical problem. Currently, the most well-known controller placement strategy, which is introduced in [1], proposes to find the placements of controllers based on the algorithms which solves the K-center problem to minimize the average propagation latency between the switches and controllers. However, we think there are three main reasons why the loads of controllers must be taken into consideration whenever designing a placement strategy:

1) **The server capacity limitation**. Because of the constraints of possessor, memory, access bandwidth and other resources, a commodity server only has the capacity to manage a limited number of routers. Though carrying a controller on a cluster will provide better performance and scalability, it will also significantly increase the cost of deployment and operation.
2) **The latency of message processing**. Whenever the load of a controller reaches a threshold, the message processing latency on the controller will increase substantially based on the result in [2]. In such cases, the controller processing latency will turn out to be a non-negligible factor in the total round-trip latency.
3) **Failure**. Heavy-load controllers always have higher failure probability, because they have less resources to handle various errors and are more likely to be attacked. In some cases, the failure of heavy-load controller may even cause cascading failures of other controllers [3].

Considering these reasons, the K-center algorithm is not always applicable. We think the placement should try to minimize the propagation latency, whereas the load of each controller[1] should not exceed its capacity. To differentiate the well-known Controller Placement Problem (*CPP*) named in [1], we name the new problem as the Capacitated Controller Placement Problem (*CCPP*). Different from *CPP* which is corresponding to the K-center problem, *CCPP* is corresponding to the Capacitated K-center Problem [4]. In this article, after listing the related works in Section II, we give a formal definition of *CCPP* in Section III, and introduce the algorithm to solve the problem. In Section IV, we present the evaluation result. It shows the placements for *CCPP* can significantly reduce the number of required controllers compared with placements for *CPP*. More importantly, we found the latency of placements for *CCPP* is generally smaller than using dynamic controller provision and dynamic scheduling based on placements for *CPP*. This result implies the placement should be considered as an important problem even using dynamic controller provision and dynamic scheduling to avoid overload of controllers.

Our work has the following contributions:

1) To the best knowledge of the authors, this is the first work taking into account the factor of load in controller placement;
2) We formally define the problem and introduce an efficient algorithm to find a solution;
3) We evaluated the new strategy and found it could significantly increase the capacity of the control plane whereas generally introduce less radius[2] stretch than dynamic controller provision and dynamic scheduling.

Manuscript received December 8, 2013; revised May 11, 2014; accepted June 8, 2014. Date of publication June 23, 2014; date of current version August 8, 2014. This work was supported in part by the National High-tech R&D Program ("863" Program) of China under Grant 2013AA013505, by the National Natural Science Foundation of China under Grants 61140454 and 61303194, by the National Science & Technology Pillar Program of China under Grant 2012BAH01B01, and by the China Postdoctoral Science Foundation under Grant 2013M530047. The associate editor coordinating the review of this paper and approving it for publication was B. Bellalta. *(Corresponding author: Jun Bi.)*

G. Yao, J. Bi, and Y. Li are with the Institute for Network Sciences and Cyberspace, Department of Computer Science, Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: yaoguang@cernet.edu.cn; junbi@tsinghua.edu.cn).

L. Guo is with the Beijing University of Posts and Telecommunication, Beijing 100876, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LCOMM.2014.2332341

[1]Hereafter, we use 'controller' to refer to 'the server or the cluster which carries the controller program'. Note that this definition is different from the definition in [1].

[2]'radius' means the maximum latency/distance from each switch to the assigned controller.

II. Related Work

The controller placement problem in SDNs was first proposed by Heller *et al.* [1]. In [1], the controller placement was taken into consideration to minimize the propagation latency; thus the placement problem was corresponding to the K-center (or K-median) problem.

Recently Advait Dixit *et al.* [5] proposed a mechanism which dynamically assigns switches to controllers. This work can better balance the load of controllers in real-time. However, this does not mean the placement strategy proposed in this article is meaningless. At least a more balanced placement can reduce the times of dynamic assignment, and thus the cost. Besides, we found that dynamic scheduling may introduce more radius stretch than the strategy proposed in this article.

A number of controllers e.g., [2], [6], have declared excellent performance, e.g., processing line-rate PACKET_IN messages. However, the performance of the controller plane is also determined by the applications. Besides, the NIC capacity and reserved bandwidth are always bottlenecks of the controllers, especially for WANs.

DevoFlow [7] has disclosed that the bottleneck of SDN is on the switches. However, the device evaluated in DevoFlow is actually an edge switch. For the scenarios where propagation latency is meaningful, typically in WANs, the switches/routers generally have powerful processors to generate high-speed events, as confirmed by vendors.

III. Problem Statement and Solution

A. Analysis on the Load of Controllers

The load of a SDN controller is mainly formed by four components: (1) processing the PACKET_IN events and delivering the events to the applications; (2) maintaining the view of the local network partition; (3) communicating with other controllers to form a global view; (4) installing the flow entries generated by applications. In different networks, the weight of each component may differ greatly. However, the load of processing PACKET_IN events is generally regarded as the most significant part of the total load [2], [6]. Whenever a large quantity of PACKET_IN messages arrive at a controller, the bandwidth, memory and processor of the controller are all potential bottlenecks. Moreover, because the latency of processing PACKET_IN events determines the efficiency and even availability of SDN networks, the controller must provide sufficient resources for processing the events timely. Thus, in the experiment phase of this article, we use the arriving rate of PACKET_IN events on a controller as the measurement of load on the controller.

B. Problem Definition

Denote the network by $G(V, E)$, where V is the set of switches (or routers) and E is the set of physical links between the switches. Denote the set of K controllers by $\Theta = \{\theta_1, \ldots, \theta_K\}$. Each controller θ is placed at a switch's location denoted by $\phi(\theta)$. Then a placement of controllers can be denoted by:

$$\mathbf{P} = (\theta_1, \phi(\theta_1)), (\theta_2, \phi(\theta_2)), \ldots, (\theta_K, \phi(\theta_K)). \quad (1)$$

Denote the controller assigned to switch v by $\varphi(v)$. Let $S_\varphi(\theta)$ denote the switches controlled by the controller θ. Then an assignment from switches to controllers can be denoted by:

$$\mathbf{S} = \{(\theta_1, S_\varphi(\theta_1)), \ldots, (\theta_K, S_\varphi(\theta_K))\}. \quad (2)$$

Each controller θ has a capacity $L(\theta)$. The load to control switch v is denoted by $l(v)$. Then the *CCPP* can be defined as follows.

Definition: Capacitated Controller Placement Problem

$$\mathbf{min} \quad \mathbf{max}_{v \in V} \quad d(v, \varphi(v))$$

such that

$$\sum_{v \in S_\varphi(\theta)} l(v) \leq L(\theta), \ \forall \theta \in \Theta$$

$$\mathbf{P} \in \|\mathbf{P}\|, \ \mathbf{S} \in \|\mathbf{S}\|,$$

where $d(v, u)$ is the minimal propagation latency from v to u, $\|\mathbf{P}\|$ is the set of all the placements, and $\|\mathbf{S}\|$ is the set of all the assignments. For simplicity, we use *radius* to denote $\max_{v \in V} d(v, \varphi(v))$.

CCPP can be regarded as a variant of the capacitated K-center problem [4]. The difference from the classical problem is, the load of each vertex is not constant based on the analysis in the last subsection. A variant of the problem is to minimize the average latency instead of the maximum latency. In this article, we mainly discuss minimizing the maximum latency. This is to avoid the situation that some switches are too long aways from their assigned controllers.

C. Solution

Obviously *CCPP* is a NP-Hard problem. The complexity of the enumeration algorithm is $O(|V|^K K^{|V|})$. For a network with tens of nodes, it will be hard to find the solution based on the enumeration algorithm. However, the size of a Software-Defined network may reach thousands of switches. Besides, different from the classic capacitated K-center problem, vertexes in *CCPP* have various load. Thus, the approximate algorithms in [4] cannot be used.

In this article, we make use of the algorithm proposed in [8]. An Integer Programming model $SP(r)$ is used to find the minimal number of controllers with a specified radius r. $SP^{LR}(r)$ is the linear relaxation of $SP(r)$. This algorithm has two phases: in phase I, the lower bound of radius is obtained by solving SP^{LR} in binary search; in phase II, the radius is increased from lower bound until a placement is found. We made a modification to the algorithm: in phase I, binary search only considers possible distances, i.e., the distance between any pair of locations, rather than all integer numbers in a given range. Though this modification may look trivial, it ensures faster convergence. When r is small, it will cost a lot of time to solve $SP^{LR}(r)$. However, even measured in kilometers, the distances between each pair of locations are typically large numbers. Because the binary search in integer numbers converges until the step is less than 1, it requires more iterations than searching in possible distances. Because

```
1: function CapControllerPlacement
2:     disArray ← distances between all pairs in ascending order
3:     lower ← 0, upper ← |disArray.length| − 1
4:     while lower < upper do
5:         mid ← (lower + upper)/2
6:         r ← disArray[mid]
7:         Solve SP^LR(r)
8:         if num of controllers > K then
9:             lower ← mid + 1
10:        else
11:            upper ← mid
12:        end if
13:    end while
14:    index ← lower
15:    Solve SP(disArray[index])
16:    while num of controllers > K do
17:        index ← index + 1
18:        Solve SP(disArray[index])
19:    end while
20:    return disArray[index]
21: end function
```

Fig. 1. The solution algorithm for CCPP.

the possible radius must be one of the distances, searching in possible distances will not omit the result radius. The algorithm is illustrated in Fig. 1. This algorithm can always find the exact solution. For typical WANs which include tens to thousands of routers, the computation time for *CCPP* is acceptable.

IV. Evaluation

A. Experiment Setup

There has been a large number of variants of SDN in the real world. Though they can be covered by the proposed placement strategy, it is impossible to evaluate all of them. Thus, we only evaluate the proposed placement strategy in the most common scenarios.

1) Topologies: We choose the topology set from the Topology Zoo [9] which contains topologies of hundreds of WANs. 82 out of 261 topologies, whose nodes have latitude and longitude marked, are used in the evaluation. The reason why we choose this dataset is that the propagation latency in WANs is typically the most significant part of the total latency.

2) Flow and Rule Generation Model: The flow generation rate of the whole network is set to $400K * N$ per second, where N is the number of routers. This rate is estimated based on measurements in CERNET (5-tuples). We considered two flow generation patterns in the experiments. The first is *Random*, in which the source and destination of flows and the generated number of flows in each second are all evenly distributed; the second is *HOT*, in which 20% of routers are randomly selected as popular destinations. The *HOT* pattern is used to catch the traffic feature that a small fraction of prefixes are popular in WANs.

In the experiment, each mismatched flow will trigger a PACKET_IN message and the controller will generate an exact match flow entry. Though there can be a number of variants, currently such a PACKET_IN and rule generation pattern are the most common. Though this fully passive mode and exact match rules may not be used in real WANs, evaluation in the worst cases tells the upper bound of requirement on controller capacity.

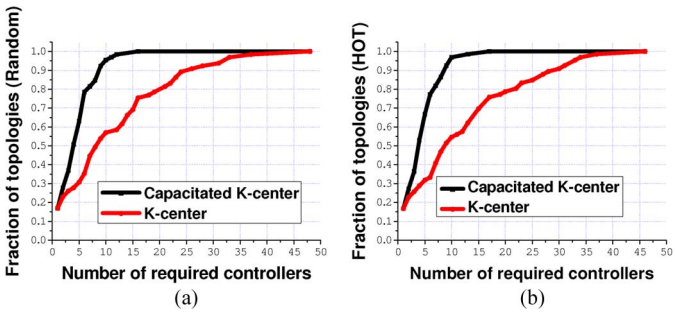

Fig. 2. The CDF of minimal required controllers for each topology to avoid overload. (a) Random. (b) HOT.

3) Controller Capacity: The capacity of a controller can be very flexible, because the available resources for a controller can range from a slice of a server to a whole cluster. Besides, the capacity of the controller to process events from data plane heavily depends on the applications. Thus, it is very hard to quantify the capacity of controllers. In the experiment, we use a critical bottleneck for the controllers: the access bandwidth of controllers. Currently, it is rare for servers to have NIC with more than 10 Gbps. Thus, we set the access bandwidth for each controller to be 10 Gbps in the simulations, i.e., each controller runs on a single server. The corresponding PACKET_IN message rate is approximate 7.8 Mpps.[3] The bandwidth can be set to other values but similar results are got. Though our algorithm support controllers with different capacities, we only considered controllers are of the same capacity in the simulations.

B. Experiment Results

We evaluate two aspects of placements for *CCPP*: 1) the load of controllers; 2) the radius of the network. We use 'Capacitated K-center' to denote the placements for *CCPP*, and 'K-center' to denote the placements for *CCP*. K_min is used to denote the minimal required number of controllers for *CCPP* in each topology.

1) Load: Based on the experiments, we made the following discoveries:

Capacitated K-center strategy can reduce the number of controllers required to avoid overload: In the experiments, we gradually increase the number of controllers for each placement strategy until no overload happens. As the results illustrated in Fig. 2, to avoid overload, 5 controllers are required using capacitated K-center strategy for 50% of topologies, and at most 15 controllers for all the topologies, whereas using K-center strategy requires 8 and 40 controllers correspondingly.

Capacitated K-center strategy can reduce the load of heaviest-load controller: As illustrated in Fig. 3, when deploying K_min controllers based on the capacitated K-center strategy, the load of the heaviest-load controller in each topology is

[3] $10G/(8 \times 160)$ pps, where 160-byte is the average PACKET_IN packet length measured in our experiments using Mininet.

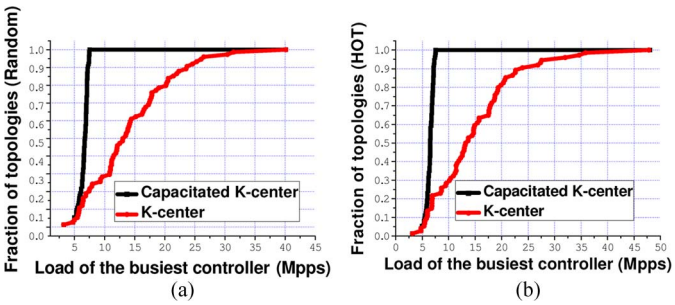

Fig. 3. The CDF of the load of heaviest-load controller in each topologies when deploying K_min controllers. (a) Random. (b) HOT.

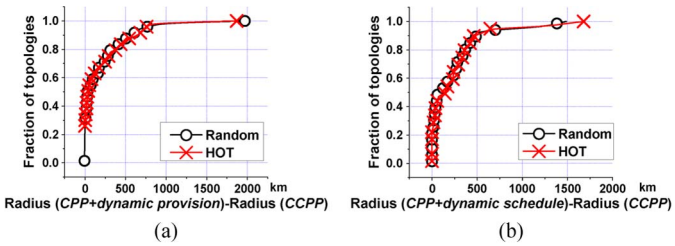

Fig. 4. The CDF of differences in radius. (a) Dynamic controller provosion. (b) Dynamic schedule.

no more than 7.8 Mpps, whereas based on the K-center strategy, the load of the heaviest-load controller in more than 70% topologies is higher than 7.8 Mpps. The load of the heaviest-load controller can even reach 40 Mpps based on K-center strategy.

2) Radius: We compare the radius in the placement strategies when the number of controllers is less than the required controller number with K-center strategy. To avoid overload, two advanced strategies are used in K-center placement respectively: *dynamic controller provision* and *dynamic scheduling*. We made the following discoveries.

Compared with K-center placement using dynamic controller provision, capacitated K-center strategy generally results in smaller radius: In the experiments, at first K_min controllers are placed based on K-center strategy. If a controller is overloaded, additional controllers will be placed in the same location until the load can be handled; if the controllers are over-provisioned, the excessive controllers will be removed. After getting the average number of deployed controllers, controllers of that number are deployed based on capacitated K-center strategy. The difference of radius in these two scenarios is plotted in Fig. 4(a). It can be found, for most of the topologies, the radius of capacitated K-center strategy is smaller than the radius of K-center strategy with dynamic controller provision.

Compared with K-center placement using dynamic scheduling, capacitated K-center strategy generally results in smaller radius: In the K-center placements, the routers are assigned to a further controller which has remaining capacity to handle the load when the nearest controller is of full load. As illustrated in Fig. 4(b), when dynamic scheduling is used to avoid overload in K-center placements, the radius is found to be generally larger than the radius in capacitated K-center placements. We found in K-center placements, a part of controllers are always overloaded; thus the relevant routers have to always be assigned to a further controller. Because the distances on WANs are large, the incremental in radius is significant. This result may imply dynamic scheduling in WANs should be carefully designed.

V. Conclusion

In this article, we proposed to include the factor of controllers' load into the problem of controller placement. We defined the Capacitated Controller Placement Problem (*CCPP*) and introduced an effective algorithm to solve the problem. According to the results, the new strategy can reduce the number of controllers required to avoid overload, and reduce the load of busiest controllers. Also, it has smaller radius than using dynamic controller provisioning or dynamic scheduling strategy in K-center placement.

Acknowledgment

The authors gratefully acknowledge many helpful comments and suggestions received from the anonymous reviewers and the editors.

References

[1] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. HotSDN*, 2012, pp. 7–12.
[2] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Proc. HotICE*, 2012, pp. 7–10.
[3] G. Yao, J. Bi, and L. Guo, "On the cascading failures of multi-controllers in software defined networks," in *Proc. IEEE ICNP*, 2013, pp. 1–2.
[4] S. Khuller and Y. J. Sussmann, "The capacitated k-center problem," *SIAM J. Discr. Math.*, vol. 13, no. 3, pp. 403–418, May 2000.
[5] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, "Towards an elastic distributed sdn controller," in *Proc. HotSDN*, 2013, pp. 7–12.
[6] D. Erickson, "The beacon openflow controller," in *Proc. HotSDN*, 2013, pp. 13–18.
[7] A. R. Curtis *et al.*, "Devoflow: Scaling flow management for high-performance networks," in *Proc. ACM SIGCOMM Conf.*, 2011, pp. 254–265.
[8] F. A. Ozsoy and M. C. Pinar, "An exact algorithm for the capacitated vertex k-center problem," *Comput. Oper. Res.*, vol. 33, no. 5, pp. 1420–1436, May 2006.
[9] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.