A New Set Covering Controller Placement Problem Model for Large Scale SDNs

Ahmad Jalili*
Department of Computer Engineering & IT, Shiraz University of Technology, Shiraz, Iran
a.jalili@sutech.ac.ir
Manijeh Keshtgari
Department of Computer Engineering & IT, Shiraz University of Technology, Shiraz, Iran
ahja2005@gmail.com
Reza Akbari
Department of Computer Engineering & IT, Shiraz University of Technology, Shiraz, Iran
Jalili_ah@yahoo.com

Received: 20/Oct/2017 Revised: 07/Jan/2018 Accepted: 19/May/2018

Abstract

Software Defined Network (SDN) is an emerging architecture that can overcome the challenges facing traditional networks. SDN enables administrator/operator to build a simpler and manageable network. New SDN paradigms are encouraged to deploy multiple (rather than centralized) controllers to monitor the entire network. The controller placement problem is one of the key issues in SDN that affect all its aspects including scalability, convergence time, fault tolerance, and node to controller latency. Many researchers focus on solving this problem by trying to optimize the location of an arbitrary number of controllers. The related works in this area get less attention to two following important issues: i) Bidirectional end-to-end latency between the switch and its controller instead of propagation latency and ii) finding the minimal number of controllers, which is a prerequisite for locating them. In this paper, we propose a Set Covering Controller Placement Problem Model (SCCPPM) in order to find the least number of required controllers with respect to carrier-grade latency requirement. The proposed model is carried out on a set of 124 graphs from the Internet Topology Zoo and solve them with IBM ILOG CPLEX Optimization package. Our results indicate that the number of required controllers for high resilient network is dependent on topology and network size. Moreover, to achieve carrier-grade requirement, 86% of topologies must have more than one controller.

Keywords: Software Defined Networks; Controller Placement Problem, Latency Constraint, Carrier-Grade Requirement.

1. Introduction

Software Defined Networks (SDN) is an emerging approach for dealing with the rigidity of traditional network. Unlike traditional networks that both data and control planes are tightly coupled on the same boxes, data and control planes are de-coupled [1]. In SDN, the complexity of data plan rules is off-loaded to the external intelligent modules called controllers. Such separation architecture enables administrator/operator to build a customizable, manageable, adaptable, and simpler network. Many efforts have been made on this separable architecture, which has diverse applications in the realm of data centers (DCNs), cellular networks, cloud, internet of things (IOT), wireless networks, and so on [1].

Recently, a substantial attention has been paid on the SDN concepts extending into wide area networks (WAN) and carrier networks [2]. Utilizing the advantages of logically centralized control of this architecture, it is possible for Carrier Network Infrastructure/WAN organizations to simplify and optimize the management of their network.

Although SDN has wide applications, it suffers from the inherent challenges, which should be probed such as scalability, reliability, and resiliency, specifically in WANs and carrier-grade networks [3].

Todays, WAN/carrier technologies are facing a rapid growth that provides remarkable characteristics and benefits like high availability, high resiliency, scalability, and reliability. For failure recovery, some networks offer carrier-grade quality meaning that a network should recover from failures within 50 ms. For instance, SONET/SDH has a specific protection strategy to provide high availability of service and they can achieve restoration time after failure at the order of 50 ms [4]. Achieving a high resilient communication is one of the major goals of networking. As a replacement for other well-established technologies, SDNs per se are expected to yield the same levels of resiliency as legacy alternative technologies in WAN.

Indeed, SDN must meet the resiliency and availability requirements of today’s production networks to be a reliable alternative to the traditional network architecture.

Resiliency is defined as the persistence of service delivery that can defensibly be trusted when facing changes [5]. In a similar fashion, resilience in SDN is the capability to return to a previous state after the occurrence of some events or actions, which may have changed that

* Corresponding Author

state. In broad terms, an event in SDN implies any kind of occurrence that changes the state of the network to an unstable state such as unexpected failures, arriving a new flow, attacks and any variation in Qos parameters. When an event occurs or an SDN element is faulty, the network should provide a continuous operational service with the same performance [6]. Due to the separation of control and data planes, the issues related to resiliency are more challenging in SDN.

The main reason for this issue is the resilience of such network that depends on fault-tolerance in the data plane (as in traditional networks) and on the high availability of the control plane functions [7].

Furthermore, in SDN, switches are simple and passive devices that cannot perform any computations on their own. Therefore, the distance between the switch and its assigned controller impose more delay on restoration operations causing increasing recovery time.

To elucidate this subject, a scenario is presented to depict what happens in the network whenever an event occurs. When a switch detects an event[1], a notification message is sent to the controller, which then takes the required actions and installs updated flow entries in the required switches to conduct the incident occurred. Such reactive strategies imply high restoration time due to the necessary interaction with the controller. One experimental work on Open Flow for carrier-grade networks investigated the restoration process and measured a restoration time in the order of 100 ms [8].

The delay introduced by the controller, intermediate switches, and load of the network may be prohibitive, especially when in-band control is in play.

Generally, various metrics affect the resilience and restoration time. Some of these metrics include performance and resilience of control and data planes, the processing power of controller, the platform of the switch, the distance between switch and controller, recovery scheme, controller workload, and so on. In the literature, various ideas have been proposed to achieve high resiliency and improve restoration time. For instance, compressing the packet in messages [9], delegating some control decisions to the forwarding devices [10], protection schemes [8], designing and deploying high-performance controllers [11] [12], and the distributed control plane [13] [14] are elaborated in Section II.

Here, we start to look at the problem of resiliency in SDNs from a different perspective. As previously mentioned, the distance between switches and their assigned controller affects restoration time and resiliency, especially for wide area SDN deployments. Furthermore, one of the most important problems in SDN is controller placement problem (CPP), which is defined as how many controllers needed and where they should be placed to satisfy the optimal network performance [15] [16].

[1] In this paper, an event implies any variation or any observable occurrence in the network such as arriving a new flow, congestion detection, link failures, controller failures, TCP timeouts, port-down event, adjacency switch failure and any variation in traffic parameters.

There are many papers published on CPP. Heller et al. established the first study in this area. In addition, many papers have been published trying to extend this work [15]. Using a brute-force method, authors evaluated the impact of controller placement on average and maximum latency metrics for real network topologies. Unsurprisingly, they show that in most topologies one single controller is enough to fulfill "existing reaction-time requirements". In this work, in order to measure latency between switches and controllers, only propagation latency is considered. As well, the other following works also continue this imperfect scheme, except that they consider other objectives such as latency between controllers and load balancing.

Generally, the related works in this area have received less attention with respect to the following two issues. i) In order to calculate the distance between switches and controllers or between controllers, they only consider propagation latency in their measurement; while, this metric is widely variable and it depends on many parameters. We delve more deeply into this matter in Section III-A. Moreover, we show how to calculate end-to-end latency between switches and controllers. ii) They focus more on optimizing the location of an arbitrary number of controllers to satisfy some objectives such as load balancing, latency and so on.

They do not argue the number of controllers and the reason for its selection. Meanwhile, finding the required number of controllers with respect to resiliency is of highest importance especially in the case of distributed control planes; because if you do not know how many controllers you need, you cannot locate them optimally. Indeed, determining the number of required controllers is a prerequisite matter for locating them.

To find an optimum number of controllers, several strategies may be considered. 1) Since the main goal of architectures based on SDN is centralizing the control plane, at the first glance one controller is enough [15] [17].

However, fully physically centralized control is inadequate because it limits (i) responsiveness, (ii) reliability, and (iii) scalability. 2) In the second strategy, each switch is allocated to a dedicated controller that is directly attached to it. Although this way reduces latency and response time very much, it is not effective at all; as it is the same as the architecture of traditional networks with their inherent challenges [4]. 3) The last strategy suggests determining the number of required controllers in some network by a specific scheme (in section III-B). Nevertheless, it is to be noted that the number of required controllers depends on various constraints regarding the requirement of network administration. This is not to say that there are many requirements in the network, which can be considered as constraints such as latency, Qos metrics, administration aspects, load, and various domains in autonomous systems. In this strategy, some essential constraints are considered based on the defined requirements of the network, and then calculations are practiced to find the least number of controllers.

Here, only latency constraint is considered in the modeling, due to focusing on restoration time as a most important requirement in network resiliency. In order to meet carrier grade requirements, this constraint must be up to 50 ms. Our analysis is based on the condition that whenever an event occurs within the network there is tens of mili-seconds opportunity for the network to return to the proper state (or to handle it). Furthermore, in order to yield the same levels of resiliency as carrier technologies, it is assumed this time must not exceed 50 ms.

For this purpose, a Set Covering Controller Placement Problem Model (SCCPPM) is proposed to determine the least number of required controllers regarding carrier grade latency requirement. The new model is carried out on a set of 124 graphs from the Internet Topology Zoo, which are solved using the IBM ILOG CPLEX Optimization package.

As expected, the minimum number of required controllers varies greatly and is more related to the network topology than the network size. Besides, in order to achieve carrier-grade requirement, 86% of topologies must have more than one controller. However, we absolutely need more than 86% of topologies in case of consideration of some other administration constraints.

Finally, we conclude this paper by discussing the main results of our analysis, which indicates that resilience in SDNs is achievable by carefully choosing the number of controllers within the target network topology.

The proposed model is of great importance from several aspects: i) By this mathematical model, decision makers and authors can determine the minimal number of controllers for their specific networks with any kind of structures or any size. They do not need to arbitrary choose or estimate it anymore. ii) Because of diverse requirements in networking, this model can be applied to the various networks regarding defined constraints for them. iii) With acquiring the number of controllers required, it is possible to estimate the cost allocation of designing and deploying of a specific SDN.

Therefore, it is necessary to know many controllers we need. Some papers have introduced the distributed control plane but they have not investigated the amount of the minimum number of controllers. In summary, in the present work, the following contributions are made:

– To the best of our knowledge, this is the first study that illustrates the concept of restoration time regarding bidirectional end-to-end delay between switches and controllers.
– The mathematical model proposed in this work first evaluates the resiliency constraint, such that it can be considered as a position paper for other works that have been addressing CPP. For this purpose, initially, the least number of controllers is determined through the proposed model. Then, in the next phase, utilizing the other approaches in the literature, controllers are optimally deployed in the network.

The remainder of this paper is structured as follows. The needed background and an overview of related work are presented in Section II. A scenario is provided in Section III to illustrate how to analysis distance between switch and controller. Also, the proposed model is presented regarding the new definition of latency. Evaluation and results are presented in Section IV. Conclusions and future work are outlined in Section V.

2. Background and Related work

A. Background

1) Facility Location Problem (Mathematically):

The facility location problem (FLP) is the general problem behind the SDN controller placement problem [15]. FLP appears in many contexts such as manufacturing plants, storage facilities, depots, warehouses, libraries, fire stations, hospitals, and base stations for wireless services. Before going into more details, first, a review is presented about location problem and its classification.

Facility location problems deal with selecting the placement of a facility to best meet the demanded constraints. Location problems and models can be classified in a number of ways. The classification may be based on the topography that is used, the number of facilities to be located, the nature of the inputs, whether there is one objective or multiple objectives, whether the facilities are of unlimited capacity or are capacitated, and a variety of other classification criteria [18].

Another way of characterizing facility location problems is by the number of facilities to be located. In some problems (e.g., the P-median, f-center, and maximum covering problems), the number of facilities to locate is exogenously specified. In other cases (e.g., the set covering problem and the fixed charge facility location problem), the number of facilities is endogenous to the problem and is a model output. For those problem statements in which the number of facilities to locate is exogenously specified.

We also distinguish between single-facility location problems and those in which multiple facilities are to be sited. Often, single-facility location problems are dramatically easier than are their multifacility counterparts [18].

In many location contexts, service to customers depends on the distance or time between the customer and the facility to which the customer is assigned. Often, service is considered adequate if the customer is within a given distance of the facility and is considered inadequate if the distance exceeds some critical value. Such problems are called “covering” problems, which require each demand to be served or “covered” within some maximum time or distance standard [18].

A demand is defined as covered if one or more facilities are located within the maximum distance or time standard of that demand [19]. Our SCCPPM model can be related to set covering problem. Similarly, the task of the model is to find the minimal number of controllers such that each switch is no farther than a pre-specified distance away from its closest facility.

B. Related Work

We review main research areas related to the resilience of SDN networks in this section and distinguish our present study on SDN controller placement from related work.

1) Resilience in SDN:

One of the major goals of networking is to achieve resilient communication. Because of the split architecture of SDN and multiple possible failures in different pieces ones (of the architecture), this issue needs to be investigated further. A number of related work have started to tackle the concerns around resilient in SDN.

Jivorasetkul et al. proposed an end-to-end header compression mechanism for reducing latency in SDN networks and thus improving convergence time [9].

Sharma et al. [8] focused on fault tolerance of SDN to deploy it in carrier-grade networks. In order to meet carrier grade requirements, they utilized two recovery mechanisms (restoration and protection) and added the recovery action in the switches themselves. Besides, the delegated some control functions from the controller to switches so that the switches can do recovery without contacting the controller. They demonstrate that such approach can achieve recovery in the order of 100 ms in a large-scale network.

Another related line of work is SlickFlow [20], leveraging the idea of using packet header space to carry alternative path information to implement resilient source routing in Open-Flow networks. Under the presence of failures along a primary path, packets can be re-routed to alternative paths by the switches themselves without involving the controller.

In [10], a new model based on modifying Open-Flow was proposed. Devo-Flow improves resiliency by delegating some work to the forwarding devices. For instance, instead of requesting a decision from the controller for every flow, switches can selectively identify the flows (e.g., elephant flows) that may need higher-level decisions from the control plane applications.

DIFANE is another scalable and efficient solution that reduces a load of a centralized controller by distributing network state among switches [21]. This scheme keeps all traffic in the data plane by selectively directing packets through intermediate switches that store the necessary rules. DIFANE relegates the controller to the simpler task of partitioning these rules over the switches.

Furthermore, several efforts have been made to tackling performance and distributed control plane, including Maestro [11], SDX [12], Onix [14], and NOX-MT [22]. Overall, one cannot say that distributed control plane and high-performance controllers cause high resiliency and quickly respond to network events.

2) Controller Placement Problem:

The controller placement problem has been discussed in a couple of papers. This problem is comparable with facility location problem in many aspects. Heller et al. established the first study in this area [15]. Using a brute-force method, they evaluated the impact of controller placement on average and maximum latency metrics for real network topologies. They show that in most topologies one single controller is enough to fulfill ‘existing reaction-time requirements’.

In [23], a mathematical model was presented for the capacitated controller placement that predicts failures to prevent a significant growth in worst-case latency and disconnections. Indeed, if there are multiple controllers, reallocating switches of the failed controller may considerably raise the worst-case latency. The model aims at minimizing the worst-case latency between switches and their Kth reference controllers such that the capacity and closest assignment constraints are satisfied.

Kshira and et al. proposed two population-based meta-heuristic algorithms, Firefly and Particle Swarm Optimization (PSO), for optimal placement of the controllers. These algorithms take a particular set of objective functions and return the best possible position in comparison with previous works [24].

In [25], authors presented a dynamic controller placement model that consists of determining the locations of controller modules to optimize latencies, and the number of controllers per module to support the load. The provisioning of controllers at each of these controller modules is to handle the dynamic load.

In [26], authors propose an energy-aware traffic engineering solution, called GreCo. They proposed a controller association algorithm to address the assignment of switches to controllers under an energy saving consideration, where they assumed that the controller placement was already known.

As can be seen, most work on the topic of controller placement in literature concentrates on the fact that the problem is NP-hard and depending on some objectives often provide only approaches to the location of controllers; while determining the number of required controllers regarding some requirements is a prerequisite input for the former.

3. Problem Definition

A. Analysis of Latency (Restoration Time)

In order to calculate bidirectional end-to-end delay between switches and their controllers (or restoration time), we first need to assess imposed latencies in network devices whenever an event occurs. Following scenario explains serial steps to handle an event regarding their latencies.

1. When a switch detects an event, it performs required processes and sends a request to the controller to get instructions on how to handle the event or any other variation. Here, processing delay in the switch includes the required time to process the event and generate a notification message.

2. The notification message is sent to the controller through allocated links between the switch and its controller. This step consists of multiple delays: processing delay and queuing delay in intermediate switches, transmission delay, and propagation delay.
3. Then, the controller will decide how to process/handle that event. It performs required processes at the behest of the switch. It sends requiring instructions back to the switch, through processing delay and queuing delay in the controller.
4. Required rules are sent back to the switch. Similar to Step 2, we have multiple delays: processing delay and queuing delay in intermediate switches, transmission delay, and propagation delay.
5. The switch updates its entries or installs new rules in the flow table to conduct the incident occurred. In this way, it processes the delay and queue delay in the switch including the required time for updating the switch as the flow changes.

As can be seen, restoration time is the sum of the processing times and queuing times in the switch and the controller and intermediate switches and the time for transmitting and propagating messages through links in both directions.

Let d_{Sproc}, d_{prop}, d_{tran}, $d_{interSproc}$, $d_{interSque}$, d_{Cproc}, d_{Cque}, and $d_{Supdate}$ denote the processing delay in the intended switch, propagation delay, transmission delay, processing and queuing delays in intermediate switches, processing and queuing delays in the controller and flow table update delay in the intended switch, respectively. Then the total end-to-end bidirectional delay between switch and controller is given by:

$$\mathrm{D_{C2N}} = \mathrm{D_{Sproc}} + 2\mathrm{D_{prop}} + 2\mathrm{D_{tran}} + 2\mathrm{D_{InterSproc}} + 2\mathrm{D_{InterSque}} + \mathrm{D_{Cproc}} + \mathrm{D_{Cque}} + \mathrm{D_{Supdate}} \quad (1)$$

As can be seen, the restoration time in SDN is totally different compared to that in traditional networks. Because returning to a previous state after the occurrence of an event in an Open-Flow switch requires instructions from the Open-Flow controller, it usually results in longer restoration time compared with legacy network. Indeed, many processes exist that affect the restoration time or bidirectional end-to-end delay[1] in SDN.

Each of these latency metrics depends on various factors. For instance, the processing and queuing latencies in intermediate switches are widely varied depending on network load and the hardware switch platform. The time for flow table update and processing delay, $d_{Supdate}$ and d_{Sproc} respectively, in the intended switch can be quite high and varying. They depend on many factors such as flow table size, the switch platform, flow setup rate, event rate, rule priority, and a load of the network [27] [28]. However, transmission delay can be negligible if notification messages and forwarding rules are small. Propagation delay can be determined by topology graph.

[1] In some paper this called (well known as) flow setup time or convergence time.

As can be seen, the total end-to-end bidirectional delay between switch and controller is highly varying, depending on many factors. Few researchers have investigated some of these delay metrics separately [27] [28]. They carried out various setup experiments to measure convergence time. They changed the protection schemes, number of switches, number of threads, occupancy of flow table, hardware platform switch, controller workload, and many other parameters. Based on their experiments and simulations, they concluded that the restoration time varies between 10 and 40 ms or even more [8] [29] [30]. It is important to note that most of the experiments are done in out of band control plane.

The calculation of each one of these latencies is outside the scope of this article and is a direction for future work. It can be modeled by queuing theory, calculus network, and other mathematical models. Here, we optimistically consider this delay within the range of 15 to 25 ms randomly. Therefore, the total end-to-end bidirectional delay between switch and controller is given by:

$$\mathrm{D_{C2N(end-to-end)}} = 2\mathrm{D_{prop}} + \mathrm{D_{variable}} \quad (2)$$

Propagation latency between the switch and the controller is denoted by d_{prop}, which is extracted from internet topology ZOO and $d_{variable}$ indicates other mentioned latencies. In the evaluation phase, we optimistically consider this delay is between 15 to 25ms randomly in our model.

B. Proposed Model (SCCPP Model)

So far, we have been familiar with the concept of resiliency and the calculation of latency between each switch and its assigned controller in SDN. The aim of this study is to find the minimum number of controllers required in the specific topology so that the maximum amount of the latency becomes less than or equal to 50 milliseconds. For this purpose, we model this problem as an Set Covering Controller Placement Problem (SCCPP) regarding latency constraint. A network is given. Let:

Data

G : the network that the decision maker locates controllers;

n : the number of nodes (or switches);

V : set of vertices in the network;

E : the set of physical links between the nodes, and the weight of each edge between two nodes represent the propagation latencies;

D_{ij} : the shortest path from node $i \in V$ to $j \in V$ according to the propagation latency;

D_{proc} : the time required it takes controllers and switches to process needed;

D_c : maximum coverage distance (let $D_c = 50$ milliseconds be coverage distance);

f_i : cost of locating a controller at candidate site i;

N_j : the set of controllers eligible to provide "cover" to switch j:

$$\mathrm{N_j} = \{\mathrm{i} \in \mathrm{V} \big| \mathrm{D_{ij}} + \mathrm{D_{proc}} \leq \mathrm{D_c}\}$$

Decision Variables

$$Z_i = \begin{cases} 1 & \text{if a controller is located at site i,} \\ 0 & \text{otherwise,} \end{cases}$$

$$X_{ij} = \begin{cases} 1 & \text{if switch j is assigned to controller i,} \\ 0 & \text{otherwise,} \end{cases}$$

With this notation, we can formulate the problem as follows:

$$\text{minimize} \sum_i f_i Z_i \tag{3}$$

Subject to:

$$\sum_{i \in N_j} X_{ij} = 1 \quad \forall j \tag{3-a}$$

$$\sum_j X_{ij} \le nZ_i \quad \forall i \tag{3-b}$$

$$Z_i \in \{0,1\} \tag{3-c}$$

$$X_{ij} \in \{0,1\} \tag{3-d}$$

The objective function (3) minimizes the total cost of the selected controllers. Constraints (3-a) state that each switch *j* must be covered by exactly one controller within the maximum time or distance standard D_c. Constraints (3-b) require that if site (node) *i* is selected to establish a controller, its maximum assigned switches would be *n*. Otherwise, no switches could be dedicated to this, as it is not a controller. Constraints (3-c) and (3-d) are the integrality constraints. Cost of all coefficients f_i can be measured in the terms more related to the network operator; for instance, they could be considered as economic cost. If all the cost coefficients f_i are equal, the problem is called the uni-cost SCP and the objective function may be simplified as follow:

$$\text{minimize} \quad \sum_i Z_i \tag{4}$$

Before proceeding further, we should note that since this problem is a set covering problem, it proves that the problem on a general graph is NP-complete. This is true for either objective function (1) or objective function (2) [18].

4. Evaluation and Result

A. Analysis of NTT topology - case study

Our interest in developing models (3) and (4) lies in their ability to analyze existing network topologies from the perspective of deploying carrier-grade SDNs with regard latency. Given a topology, if it will be implemented based on SDN architecture, there is a set of candidate sites where controllers can be deployed. At the outset, we need to know how many controllers are required.

For this purpose, first, we examine the proposed model for NTT communication network and analyze it, followed by investigating further a larger number of topologies from internet topology ZOO in the next section. NTT is one of the largest carrier-grade infrastructure providers in the world, with its services reaching 160 countries/regions including the most extensive coverage in the Asia Pacific.

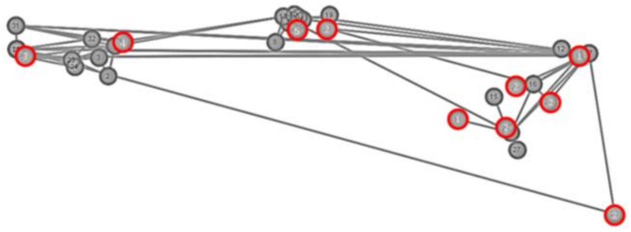

Fig. 1. The minimum number of required controllers in NTT network topology

Moreover, it has been a pioneer in the arena of software-defined networking. NTT is going to take the advantage of architectures based on SDN for delivering its services, and it has been deploying SDN/Open-Flow to connect 17 of its global data centers for almost three years [31].

The network topology related to the infrastructure of this carrier is presented in Figure 1. This topology contains 32 nodes and it has spread throughout the world. Considering the characteristics of this topology, which is stretched in large scale, controller placement problem is critical for it. Model (3) was applied to this topology to find out the minimum number of the required controllers.

To provide some intuition for placement considerations, Figure 1 shows the minimum number of controllers and their placements. By solving this model in CPLEX 12.6, it can be found that to achieve carrier-grade requirements, the required controllers must be at least 10 controllers with their locations being at 4, 5, 11, 18, 20, 21, 22,2 5, 28, and 30.

B. Analysis of more Topologies - Case Study

In this section, we expand our analysis to 124 topologies from Internet Topology Zoo with graph sizes ranging from 25 to 65 nodes randomly. This dataset includes a collection of network graphs derived from public network maps covering a diverse range of geographic areas, network sizes, and topologies. The proposed model, solved through CPLEX 12.6, is applied to these topologies. The obtained results are presented in Figure 2 with a confidence interval of α=0.05. The eight values written on the horizontal axis represent bins' thresholds. The first bin contains all topologies for which the network size contains up to 30 nodes, the second bin comprises of all topologies for which $30 < \textit{number of nodes} \le 35$, and so on. The vertical axis illustrates the minimum number of required controllers regarding the latency constraint. The graph shows that the number of required controllers for a high resiliency varies highly. Besides, it strongly depends on the network topology than network size.

Figure 3 presents the cumulative distribution of minimal required controllers for each topology. From these results, we can see that to achieve carrier-grade requirement, 86% of topologies must have more than one controller.

Furthermore, one controller is only enough for 14% of topologies, contrary to the results of Heller et al. reported [15] who reported that "one controller location is often sufficient to meet existing reaction-time requirements".

Nonetheless, 5 controllers are enough for 95% of scenarios even looking for a high resiliency.

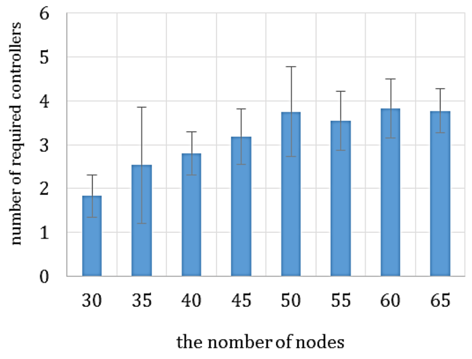

Fig. 2. Size of topology and the corresponding minimum number of required controllers

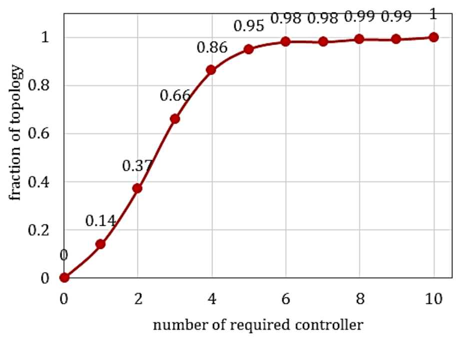

Fig. 3. The CDF of minimal required controllers for each topology

5. Conclusion and Future Work

Achieving a high resilient communication is one of the most important goals in networking. In this paper, we initiated the study of the resilience in SDN from the controller placement standpoint. Furthermore, we focus more on two important issues in CPP-related works that have received less attention. First, we exhausted the subject of latency between the switch and its controller and found this metric is highly varying and it depends on various conditions. For future works, this scheme can be extended by other network modeling such that to estimate the distance between switch and controller more accurately. Moreover, such approach should be followed to calculate the latency between controllers especially when in-band control is in play. Secondly, a SCCPPM is proposed to find the minimal number of controllers with regard to latency constraint. Normally, it is essential to determine the number of required controllers with respect to administration requirement followed by finding their optimal location. Indeed, the former is a prerequisite act before applying the latter. Finally, we conclude that 86% of topologies must have more than one controller to achieve carrier grade requirement. However, we absolutely need more than this, if we consider some other administration constraints.

References

[1] Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 103(1), 14-76.

[2] Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., and Zolla, J. (2013). B4: Experience with a globally-deployed software defined WAN. ACM SIGCOMM Computer Communication Review, 43(4), 3-14.

[3] Sezer, S., Scott-Hayward, S., Chouhan, P. K., Fraser, B., Lake, D., Finnegan, J., and Rao, N. (2013). Are we ready for SDN? Implementation challenges for software-defined networks. IEEE Communications Magazine, 51(7), 36-43.

[4] Kano, S., Miyazaki, K., Nagata, A., and Chugo, A. (2005, November). Shared segment recovery mechanism in optical networks. In 6th Asia-Pacific Symposium on Information and Telecommunication Technologies (pp. 415-420). IEEE.

[5] Laprie, J. (2005, July). Resilience for the scalability of dependability. In Fourth IEEE International Symposium on Network Computing and Applications (pp. 5-6). IEEE.

[6] Benzekki, K., El Fergougui, A., and Elbelrhiti Elalaoui, A. (2017). Software- defined networking (SDN): a survey. Security and Communication Networks.

[7] Oliveira, D., Pourvali, M., Bai, H., Ghani, N., Lehman, T., Yang, X., and Hayat, M. (2017, March). A novel automated SDN architecture and orchestration framework for resilient large-scale networks. In SoutheastCon, 2017 (pp. 1-6). IEEE.

[8] Sharma, S., Staessens, D., Colle, D., Pickavet, M., and Demeester, P. (2013). OpenFlow: Meeting carrier-grade recovery requirements. Computer Communications, 36(6), 656-665.

[9] Jivorasetkul, S., Shimamura, M., and Iida, K. (2013, August). Better network latency with end-to-end header compression in SDN architecture. In Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on (pp. 183-188). IEEE.

[10] Curtis, A. R., Mogul, J. C., Tourrilhes, J., Yalagandula, P., Sharma, P., and Banerjee, S. (2011). DevoFlow: Scaling flow management for high-performance networks. ACM SIGCOMM Computer Communication Review, 41(4), 254-265.

[11] Ng, E., Cai, Z., and Cox, A. L. (2010). Maestro: A system for scalable openflow control. Rice University, Houston, TX, USA, TSEN Maestro-Techn. Rep, TR10-08.

[12] Mambretti, J., Chen, J., Yeh, F., Grossman, R., Nash, P., Heath, A., and Zhang, Z. (2017, March). Designing and deploying a bioinformatics software-defined network exchange (SDX): Architecture, services, capabilities, and foundation technologies. In Innovations in Clouds, Internet and Networks (ICIN), 2017 20th Conference on (pp. 135-142). IEEE.

[13] Canini, M., Salem, I., Schiff, L., Schiller, E. M., and Schmid, S. (2017, June). A self-organizing distributed and in-band SDN control plane. In Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on (pp. 2656-2657). IEEE.

[14] Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., and Shenker, S. (2010, October). Onix: A distributed control platform for large-scale production networks. In OSDI (Vol. 10, pp. 1-6).
[15] Heller, B., Sherwood, R., and McKeown, N. (2012, August). The controller placement problem. In Proceedings of the first workshop on Hot topics in software defined networks (pp. 7-12). ACM.
[16] Zhang, Y., Cui, L., Wang, W., and Zhang, Y. (2017). A Survey on Software Defined Networking with Multiple Controllers. Journal of Network and Computer Applications.
[17] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., and Turner, J. (2008). OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 38(2), 69-74.
[18] Farahani, R. Z., Asgari, N., Heidari, N., Hosseininia, M., and Goh, M. (2012). Covering problems in facility location: A review. Computers & Industrial Engineering, 62(1), 368-407.
[19] Li, S., and Huang, Y. (2014). Heuristic approaches for the flow-based set covering problem with deviation paths. Transportation Research Part E: Logistics and Transportation Review, 72, 144-158.
[20] Ramos, R. M., Martinello, M., and Rothenberg, C. E. (2013, October). Slickflow: Resilient source routing in data center networks unlocked by openflow. In Local Computer Networks (LCN), 2013 IEEE 38th Conference on (pp. 606-613). IEEE.
[21] Yu, M., Rexford, J., Freedman, M. J., and Wang, J. (2010). Scalable flow-based networking with DIFANE. ACM SIGCOMM Computer Communication Review, 40(4), 351-362.
[22] Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., and Sherwood, R. (2012). On Controller Performance in Software-Defined Networks. Hot-ICE, 12, 1-6.
[23] Killi, B., and Rao, S. Optimal Model for Failure Foresight Capacitated Controller Placement in Software Defined Networks, (2016, June), Communications Letters, IEEE, 20(6), 1108 - 1111.
[24] Sahoo, K. S., Sarkar, A., Mishra, S. K., Sahoo, B., Puthal, D., Obaidat, M. S., and Sadun, B. (2017). Metaheuristic Solutions for Solving Controller Placement Problem in SDN-based WAN Architecture
[25] ul Huque, M. T. I., Si, W., Jourjon, G., and Gramoli, V. (2017). Large-Scale Dynamic Controller Placement. IEEE Transactions on Network and Service Management, 14(1), 63-76.
[26] Ruiz-Rivera, A., Chin, K. W., and Soh, S. (2015). Greco: an energy aware controller association algorithm for software defined networks. IEEE Communications Letters, 19(4), 541-544.
[27] He, K., Khalid, J., Das, S., Gember-Jacobson, A., Prakash, C., Akella, A., and Thottan, M. (2015, June). Latency in software defined networks: Measurements and mitigation techniques. In ACM SIGMETRICS Performance Evaluation Review (Vol. 43, No. 1, pp. 435-436). ACM.
[28] Kuźniar, M., Perešíni, P., and Kostić, D. (2015, March). What you need to know about SDN flow tables. In International Conference on Passive and Active Network Measurement (pp. 347-359). Springer International Publishing.
[29] Sood, K., Yu, S., Xiang, Y., and Cheng, H. (2016). A General QoS Aware Flow-Balancing and Resource Management Scheme in Distributed Software-Defined Networks. IEEE Access, 4, 7176-7185.
[30] Rechia, F. S. (2016). An Evaluation of SDN Based Network Virtualization Techniques (Doctoral dissertation, ARIZONA STATE UNIVERSITY).
[31] P. Bernier, “NTT Recognized with IBC Award for SDN-based HDTV Service,” September 2013. [Online]. Available: http://www.sdnzone.com/topics/software-defined-network/articles/353466-ntt-recognized-with-ibc-award-sdn-based-hdtv.html.

Ahmad Jalili born in 1987. Now he is a Ph.D. candidate on Computer Networks in Shiraz University of Technology. He has many publications in international conferences and journals regarding Wireless Sensor Networks (WSNs) and Software Defined Networks (SDNs). Currently he focused on Software Defined Networks (SDN) as a new trend in computer networks. His major fields of interest are Software Defined Networks (SDNs), heuristic algorithms, Wireless Sensor Networks (UWSNs), Ad hoc Networks and Modeling.

Manijeh Keshtgari received her B.Sc. in Computer engineering from Shiraz University in 1986, her Master in Electrical and Computer Eng. from Colorado State University, USA in 1992 and Ph.D. degree in Computer Eng. from Sharif University of Technology in 2004. Her research interests include wireless Networks, Fiber Optic Networks, Software Defined Networking (SDN) and Named Data Networking (NDN). He is now member of faculty and lecturer in Computer Engineering and IT Department in Shiraz University of Technology. In addition she is lecturer of Computer Science in Department of Computer Science in University of Georgia, USA.

Reza Akbari received his M.Sc. in artificial intelligence from Isfahan University of Technology (2006). He has received his Ph.D. from Shiraz University (2011). He is an Assistant Professor at the department of computer engineering and information technology of the Shiraz University of Technology. His areas of research include evolutionary computation, engineering optimization, and search based software engineering.