Received: 22 May 2017 | Revised: 7 December 2017 | Accepted: 10 January 2018

DOI: 10.1002/nem.2018

RESEARCH ARTICLE

WILEY

A survey and classification of controller placement problem in SDN

Ashutosh Kumar Singh | Shashank Srivastava

Department of Computer Science and Engineering, Motilal Nehru National Institute of Technology Allahabad, Allahabad 211004, U.P., India

Correspondence
Ashutosh Kumar Singh, Department of Computer Science and Engineering, Motilal Nehru National Institute of Technology Allahabad, Allahabad 211004, U.P., India.
Email: ashuit89@gmail.com

Summary

Software defined networking emerges as a promising paradigm shift that decouples the control plane from the data plane. It has the ability to centrally monitor and control the network through softwarization, ie, controller. Deploying a single controller is inefficient to handle large network traffic; thereby, making multiple controllers are a necessity of current software defined networking in wide area networks. Placing multiple controllers in an optimum way, ie, controller placement is a vibrant research problem. Controller placement problem (CPP) is of twofold: the minimum number of controllers to be placed in a network and locations of these controllers. Numerous researchers in the last 5 years (2012 to November 2017) have proposed solutions for the CPP, which is an NP-hard problem. In general, solutions are based on objective functions and their optimum solutions considering various factors (such as propagation latency between switches and controllers, and intercontrollers) and constraints (such as the capacity of controllers and switches). To the best of our knowledge, this is the first attempt of state-of-the-art review on CPP. This paper classifies the CPP, critically analyzes the existing solutions, and finds limitations and future scope existing, which will help potential researchers in this area to innovate new solutions for CPP lying on this information.

KEYWORDS

controller placement, CPP, large-sized network, latency, SDN

1 | INTRODUCTION

Today's Internet has become a design of a digital society, in which almost the whole things are connected to each other and also available from anywhere around the globe. It has reduced distance and broken all man-made barriers. Billions of smart objects immersed in the environment are cooperating with each other to provide different services efficiently. The computing trends are driving the need for a new network, as the traditional network is inappropriate to the dynamic computing, communication technologies, and resource capabilities. The traditional network has some limitations. These are as follows: (1) complex management,[1] (2) vendor dependent network, and (3) predefined policies of forwarding devices (like the router and layer 3 switches) that prohibit customization. Software defined networking (SDN) is a promising paradigm, which eliminates these dependencies of the traditional network. The fundamental concept of SDN is achieving programmable network by separating the control plane and data plane[2] to improve the network performance. Data plane is also known as forwarding plane, which is responsible for forwarding traffic as per the decisions made by

the controllers. The control plane is responsible for handling the network traffic and generating rules and policies for the forwarding devices with the help of controller(s). The network administrator can set and generate the rules and policies of the forwarding devices with the help of SDN controller(s), as per the requirements. The network operator has the global view of the whole network and configures individual networking devices separately, dynamically manages the fault as well as adopts the load changes in the networks. Forwarding policies and rules are highly challenging for the dynamic environment. In the traditional network, control plane and data plane are tightly coupled (integrated with each other). The challenges in SDN are security, quality of service (QoS), service management, scalability, and controller placement problem (CPP). In this paper, we only focus on scalability and CPP, and other challenges are out of the scope of our review.

Scalability[3,4] is a major issue in large-sized networks, where deployment of a single controller is not sufficient to deal with the reliability and scalability issues. Multiple controllers can be used to tackle these issues. When more than one controllers are used in the network, then 2 issues arise: a mechanism for inter-SDN–controller communication and locations of the SDN controllers within the networks. During the placement of controllers in the WAN,[5,6] the following questions arise:

1. What should be the minimum number of controllers required for the WAN network?
2. Where should the controller(s) be placed in the network?
3. How many networking devices should be attached to a given controller?

Considering these 3 questions while placing the controllers is called CPP. The CPP may be either uncapacitated or capacitated. All controllers have an unlimited capacity (do not consider capacity as a constraint) in uncapacitated controller placement problem (UCPP) whereas capacitated controller placement problem (CCPP) refers to different controllers having different capacity. The performance of the controller[7] in SDN is a major factor when scalability of SDN is taken into account. In the year 2009, Flow Visor[8] first laid the foundation of multiple controllers in SDN that is based on the partitioning of the large-sized network into several domains and each domain is managed by a single controller. The traditional network has been largely hardware-centric and vertically integrated. The SDN breaks this vertical integration by decoupling the system into control plane (decide "how to handle network traffic") and data plane (forward "traffic according to decisions made by the control plane"). This disassociation of control plane and data plane turns the forwarding devices into dumb devices which forward the incoming port traffic to the output port. Currently, most of the researchers are focusing on the controller placement problem in SDN. This type of scheme increases the availability of the network, makes it more fault tolerant, and enhances the performance of the network.

To increase the reliability and performance of the traditional network, SDN is a better choice for the researchers. Efficient controller placement requires maintaining the scalability, reliability, and performance in the SDN.

The main contributions of this review are summarized as follows:

1. A critical analysis of the state-of-the-art SDN CPP solutions is given along with highlighting the research trends of CPP. These solutions describe the current research status of CPP.
2. Different aspects of our review are the classification of CPP based on the capacity of controllers, network traffic, which may be static or dynamic traffic, fault tolerance, and network partitioning.
3. The existing solutions of CCPP and UCPP are critically analyzed and compared with the existing solutions based on various factors like traffic load balancing, fault tolerance, and network partitioning.
4. Finally, we present open research issues in CPP raised by researchers and industry professionals around the globe.

The remainder of this paper is organized as follows. Section 2 presents a brief introduction of SDN and its architecture with the aim of setting up the background of SDN. In Section 3, we initially discuss CPP and its classification. As we classify the CPP into 2 subclasses, one is UCPP, where the capacity of controllers is not a constraint, and the other subclass is CCPP, where the capacity of controllers is a constraint. In Section 3.1, we describe UCPP and we further subclassify UCPP into different classes, ie, static traffic-aware, fault-aware, and network partitioning based UCPP. In this section, we also analyze already existing solutions of UCPP and try to find out the limitations and open research issues of each type of UCPP. In Section 3.2, we describe CCPP and classify the problem into different subclasses as UCPP, ie, static traffic-aware, dynamic traffic-aware, fault-aware, and network partitioning based CCPP. Finally, Section 4 contains conclusions and future research directions.

2 | BRIEF INTRODUCTION OF SDN

The SDN separates the control plane from the forwarding devices (routers and switches), and therefore, it acts as a logically centralized unit like the network operating system or SDN controller. The SDN has several advantages over the traditional networks.

- It is easier to manage the network and easy to add the new functionalities for the networks as per the network operator requirements.
- The controller has the global view of the whole network and can add or delete the rules and policies dynamically. It is also able to drop the malicious flow that is harmful to the network.
- Only the first packet of every new flow will go to the SDN controller. Based on this packet, the SDN controller defines the rules for the associated forwarding devices, so that the next incoming packets of this particular flow will not go to the SDN controller. It just checks the flow table entries that is stored in TCAM (ternary content addressable memory)[9] and forward the packets to the next forwarding devices, and so on.
- The SDN can reconfigure the network as required by the network operator and manage the dynamic networks. The network operator can add and delete forwarding devices quickly as compared to traditional networks.

The SDN architecture is shown in Figure 1. This architecture can be divided into 6 parts, and each part is explained in detail as follows:

1. **Management plane:** It includes networking applications like routing, monitoring, load balancing, and firewalls. Management plane is responsible for defining rules and policies. Some authors use the term application plane instead of management plane.
2. **Northbound interface:** A northbound interface provides support for communication between management plane and control plane. It provides low-level instructions for the southbound interface. It is also known as management to control plane interface (MCPI). Up till now, no standard protocols have been defined for the northbound interface.
3. **Control plane:** It is responsible for programming the forwarding devices. It thus acts as the brain of the network. Centralized controller(s) resides on this plane. The controller has the complete global view of the networks. It is also known as controller plane. The set of controllers manages control plane or controller plane.

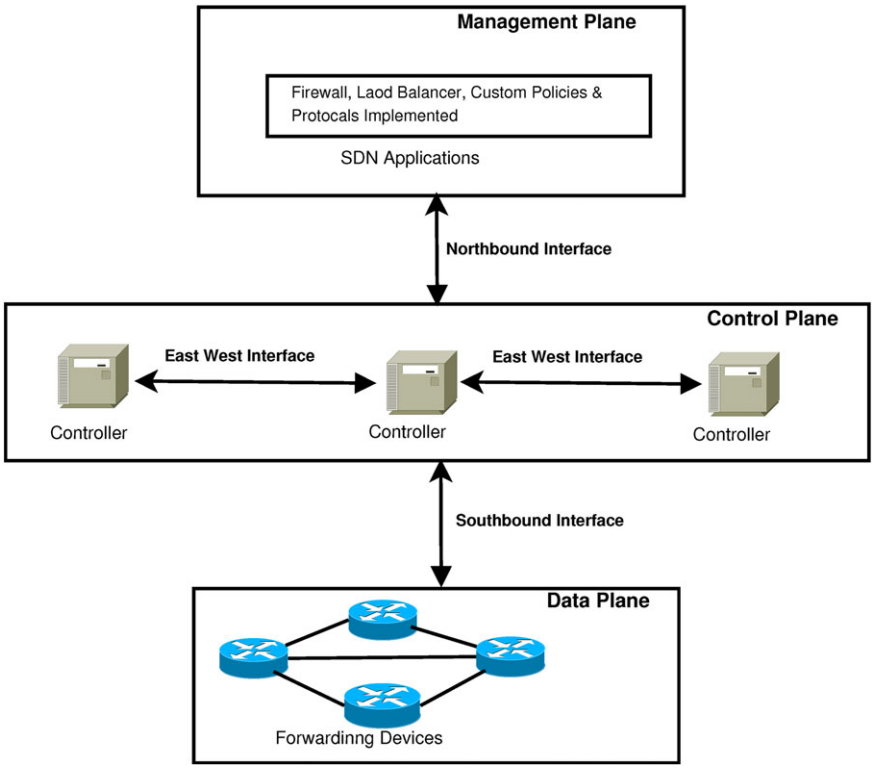

FIGURE 1 Software defined networking architecture

4. **East-west interface protocol:** East-West protocol is used to manage the communication among multiple SDN controllers. Communication among them is a most challenging task. It is also known as a controller to controller interface (CCI). Recently, TCS (Tata Consultancy Services) performed research on inter-SDN–controller communication and presented it as a white paper.[10]
5. **Southbound interface:** The southbound interface provides a protocol for communication between control and data plane. This protocol is nothing but OpenFlow protocol. Nowadays, OpenFlow-enabled switches are commercially available. OpenFlow is a standard protocol for the SDNs. It is also known as a control to data plane interface (CDPI).
6. **Data plane:** Physical network infrastructure is defined in the data plane. Forwarding devices (switches and routers) are interconnected by either wired or wireless medium. Data plane contains forwarding tables that include 3 primary fields: header, match, and actions. The TCAM holds the flow table entries in the data plane. It is also known as a forwarding plane (FP).

The overall comparison between traditional network and SDN is given in Table 1.

3 | CONTROLLER PLACEMENT PROBLEM

The CPP is a fundamental research problem in SDN. The CPP has come into the picture since 2012. The CPP is not an issue for small- and medium-sized networks because one controller is sufficient to manage these kinds of networks. But in the case of large-sized networks, there would be significant aspects to consider multiple controllers and how to place these controllers in the SDN-based infrastructure. The CPP is a well-known NP-hard problem and is similar to the facility location problem.[11] Placing the controllers in any of the available locations would only increase the overhead delay of service. This would also lead to a decrease in overall performance of SDN-based infrastructure. The primary objective of the CPP is to place the optimum number of controllers on the best locations of SDN-based infrastructure to achieve better performance. Figure 2 shows the overall classification of our state-of-the-art of CPP. The CPP is first classified into 2 sections: (1) UCPP and (2) CCPP. The UCPP is further classified into 3 subsections: (*a*) static traffic-aware UCPP, (*b*) fault-aware UCPP, and (*c*) network partitioning–based UCPP. The CCPP is divided into 4 subsections: (*a*) static traffic-aware CCPP, (*b*) dynamic traffic-aware CCPP, (*c*) fault-aware CCPP, and (*d*) network partitioning–based CCPP.

Controller placement problem may be either uncapacitated or capacitated. Literally, uncapacitated means unlimited capacity. All controllers have unlimited capacity in UCPP whereas CCPP refers to different controllers having different capacity. The UCPP and CCPP can be considered as failure-free CPP (FFCPP) and fault-tolerant CPP (FTCPP). In

TABLE 1 Software defined networking (SDN) vs traditional network

Sr. No.	Criteria	Traditional Network	SDN
1	Network management	Difficult because changes are implemented separately at each device	Easier with the help of controller(s)
2	Global network view	Difficult	Central view at controller
3	Maintenance cost	Higher	Less
4	Time required for update/error handling	Sometimes it takes months	Quite easy because of central controller(s)
5	Controller utilization	Not relevant	Important
6	Authenticity, Integrity and consistency of controller(s)	Not important	Important
7	Integrity and consistency of forwarding tables and network state	Important	Important
8	Availability of controller	Not relevant	Important
9	Resource utilization	Less	High

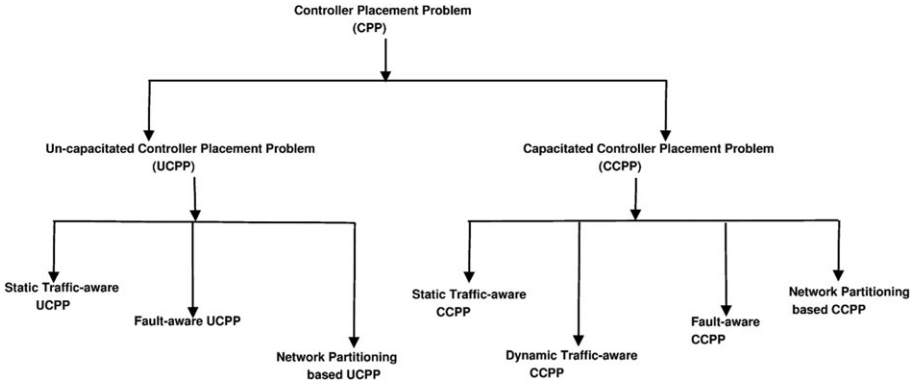

FIGURE 2 Classification of controller placement problem

FFCPP, failure scenarios are not considered while focusing on controller placement in the SDN, whereas FTCPP considers failure scenarios.

3.1 | Uncapacitated CPP

In UCPP, each SDN controller has unlimited capacity, and load on the controllers are ignored. Load on the controllers is a significant factor for the CPP. Some research works[12-16] have not considered load and capacity of the controllers and assuming that the switches have fixed load. This kind of problem can be restricted to the static environment because the burden on the controllers cannot be distributed to other controllers, which are less loaded.

In 2012, Heller et al[12] have introduced SDN CPP. Here, authors formulated UCPP as a minimum k-median problem and minimum k-center problem to minimize the average and maximum (worst case) switch-to-controller latency, respectively.

3.1.1 | Static traffic-aware UCPP

Static traffic means fixed traffic. Here, traffic refers to the number of packets transferred between switches and controllers. In Table 2, we list down the mathematical symbols used in static traffic-aware UCPP.

Heller et al[12] present the CPP as to find the set of k controllers and their respective placements so that the average latency between switches and controllers $\left(\pi^{average-latency}(P)\right)$ is minimized. The average latency between switches and controllers is given in Equation 1. This optimization problem is known as the minimum k-median problem.

$$\pi^{average-latency}(P) = \frac{1}{n}\sum_{v\epsilon S}\min_{u\epsilon P} d(u,v). \tag{1}$$

Heller et al[12] also propose to solve the CPP by minimizing the maximum latency between switches and controllers. This maximum latency $\left(\pi^{maximum-latency}(P)\right)$ is given in Equation 2 and the minimization consists of solving the minimum k-center problem, being k again the number of controllers. The k-center problem is a location analysis problem related to the optimization problem in the area of operation research.

TABLE 2 Symbols and definitions of static traffic-aware uncapacitated controller placement problem

Symbols	Definitions
$G(S,E)$	Graph G, where S is a set of switches and E is set of edges between switches
$d(u,v)$	Shortest distance between switch $u\epsilon S$ and $v\epsilon S$
n	Total number of switches, ie, $\lvert S\rvert$
$P\subseteq S$	Set of all possible placements for controllers

$$\pi^{maximum-latency}(P) = \max_{v \epsilon S} \min_{u \epsilon P} d\ (u, v). \quad (2)$$

The *k*-center scheme provides a solution for maximum latency among nodes. A node may be either switch or may be controller and controller can be deployed in a location of the switch. Heller et al[12] concentrate only on the static environment and ignores the load on the controllers, intercontroller latency, and failure scenarios.

3.1.2 | Fault-aware UCPP

A fault-tolerant system is one that functions correctly even in the face of faults. Ensuring the reliability a crucial aspect to achieving better performance of SDN. Fault-tolerant systems are mostly based on the concept of redundancy. For instance, a five nines system would statistically provide 99.999% availability. Fault-tolerant mechanism removes the single point of failure. There are 3 key features of fault-tolerant systems: (1) replication "provides multiple identical instances of the same system or subsystem, directing tasks or requests to all of them in parallel, and chooses the correct result on the basis of a quorum," (2) redundancy "provides multiple identical instances of the same system and switching to one of the remaining instances in case failure occurs," and diversity "provides multiple different implementations of the same specification, and using them like replicated systems to cope with errors in a specific implementation." Zhang et al[17] take the initial step to maximize the resilience, and this work assumes that switch to controller latency can be changed in case of failures. In Table 3, we list down the mathematical symbols used in fault-aware UCPP.

In the year 2012, Yan-nan et al[15] have introduced reliability as a placement metric in the SDN environment where a number of controllers are deployed. According to authors, reliability of SDN control network increases the performance of network (network availability). In the large-sized network, the network may fail due to either controller or switch failure or link goes down. To prevent a network from such failure, the controller needs to be fault-tolerant. The FTCPP problem is similar to fault-tolerant facility location problem.[18] If the controller itself fails, then a network without the brain is meaningless. Considering all the above cases, fault-tolerant controller placement is required to maintain the reliability in the network. Authors proposed random placement, brute force and *l*-*w*-greedy algorithm for the reliable controller placement in the SDN. The *l*-*w*-greedy algorithm provides better results as compared to random placement and brute force algorithm. It uses Internet2 OS3E network and real ISP topologies from Rocketfuel repository. Here, authors calculate the percentage of expected valid control paths. Valid control paths are nothing but logical links between networking

TABLE 3 Symbols and definitions of fault-aware uncapacitated controller placement problem

Symbols	Definitions
$G(S,E)$	Graph G, where S is a set of switches and E is set of edges between switches
$P \subseteq S$	Set of all possible placements for controllers
m	Total number of control paths in the networks
d^l	Represents the control paths
p^l	Failure probability of each components
δ	Expected percentage of valid control paths
μ	Expected percentage of control path loss
h_{ijl}	Binary variables (0,1)
	1-indicates control path between i and j passes l
	0-otherwise
y_i	Binary variables (0,1)
	1-indicates controllers at location j
	0-otherwise
x_{ij}	Binary variables (0,1)
	1-indicates control path between i and j
	0-otherwise

devices to controllers or between the controllers. If control path fails, then disconnection occurs between networking devices and controllers or among the controllers. FTCPP or RCP (reliability-aware controller placement) metrics can easily handle this problem. Equation 3 is used to find the expected percentage of valid control paths.

$$\delta = 1 - \frac{1}{m} \sum_{l \epsilon (S \cup E)} \sum_{i \epsilon S, j \epsilon P} h_{ijl} x_{ij} p^l. \tag{3}$$

RCP problem formulation as integer linear programming is given as follows:

$$\max \delta, \tag{4}$$

subject to

$$\sum_{j \epsilon P} y_j = k \tag{5}$$

$$\sum_{j \epsilon P} x_{ij} = 1, \quad \forall i \epsilon S | y_j = 0 \tag{6}$$

$$\sum_{i, j \epsilon P, i < j} x_{ij} = k \frac{k-1}{v}, \quad v \epsilon S \tag{7}$$

$$\sum_{j \epsilon P, j < i} y_{ij} - k \frac{(m-n-1) y_i}{2} \leq 0, \quad \forall i \epsilon P. \tag{8}$$

An objective of Hu et al[15] is to maximize the expected percentage of valid control paths (see Equation 4). Equation 5 represents the total number of controllers that are equal to k. Equation 6 declares that a switch is attached to exactly one controller, and Equation 7 guarantees the appropriate number of intercontroller control paths. Equation 8 implies that adjacencies (intercontroller control paths) for areas where an absence of controller is forbidden. The SDN contains an optimal number of controllers, and switches have connected these controllers. However, the authors ignore switch-to-controller and intercontroller latency impact.

Maximizing the reliability of the control network is focused in Hu et al.[13] It provides optimal results for controller placement. The disconnection between data planes and control planes may lead to packet loss as well as degrade the performance. Given a network and failure probability of switches and links, they are interested in finding the solution of CPP (how many controllers are required and where it is to be placed). Research works[13,15] focus only on failure scenario for small- and medium-sized networks, and not for the large-sized network as they assume that not more than one physical component may fail at a time. The control paths also traverse the failed physical component. In Hu et al,[13] the author proposed the reliability metric as "expected percentage of control path loss," where control path loss is the number of broken control paths due to network failure. Expected percentage control path loss when failure scenario considered is given in Equation 9.

$$\mu = \frac{1}{m} \sum_{l \epsilon S \cup E} d^l p^l. \tag{9}$$

An objective of Hu et al[13] is to minimize the percentage control path loss. There is no such difference in Hu et al[13] and Hu et al.[15] The purpose of both research works is to maximize the reliability of the network even though the fault occurs. They ignored the switch-to-controller latency, controller-to-controller latency, and load balancing among the controllers.

3.1.3 | Network partitioning–based UCPP

The SDN network is partitioned into several small SDN network domains, and only one controller controls each domain.[19] Network partitioning can reduce the management complexity of resulting partitions of large networks. It could be related to various aspects of the network such as scalability, manageability, privacy, and deployment. Here,

we discuss network partitioning based on only latency between switches, not on the traffic load of the switches. There are various techniques (local search, spectral clustering, density-based clustering, and multilevel partitioning) by which network partitioning can be done. [16]

FlowVisor has partitioned the large-sized network into several SDN domains, and each domain has its controller. Partitioning the network is based on average-case latency and worst case latency as shown[20] in Figures 3 and 4, and these figures do not indicate the actual placement of controllers. It is just for the sake of understanding what the controller placement problem is for SDN in large-sized network.

Hu et al[14] extended their previous work by relaxing the previous assumptions (the control paths also traverse the failed physical component) and gave the detailed analysis of reliability for SDN. Authors proposed reliable CPP with network partitioning. Even if controllers are placed in the SDN, the number of controllers should be appropriately chosen. Reliability of the network reduces if the number of controllers "k," are too few or too many. However, if the value of k is not too few or too many, then reliable controller placement can be achieved. Hu et al[14] have used the OS3E and the Rocketfuel topologies for their simulation. Average latency and reliability metrics have discussed for all the possible placements of controllers (up to 5). Results of Hu et al[14] show that optimizing the reliability and average latency is not possible at the same time. For example, if only one controller ($k = 1$) placed then both the average latency and reliability is optimized. Reliability is reduced by 15. 1% when the value of k is 4. Authors have formally proved that RCP is NP-hard. However, they do not give more focus on partitioning the network. Their objective is to minimize the percentage control path loss.

Resilience-based CPP ("controller placement for improving resilience of software defined networks") is discussed by Gua and Bhattacharya[16] in which the authors' aim is to provide reliability, scalability, and security for the SDN. This research work focused on (a) controller placement on SDN resilience while taking interdependent networks into account, (b) analyzing cascading failure on the interdependence networks and they proposed a network partition strategy for the controller placement to improve the resilience of SDN.

In research works,[12-15] the authors considered only propagation latency as an input, which gives an optimal number of controllers and their corresponding locations as an output. They do not consider traffic load balancing for the controller placement problem. Table 4 shows the comparative analysis of UCPP.

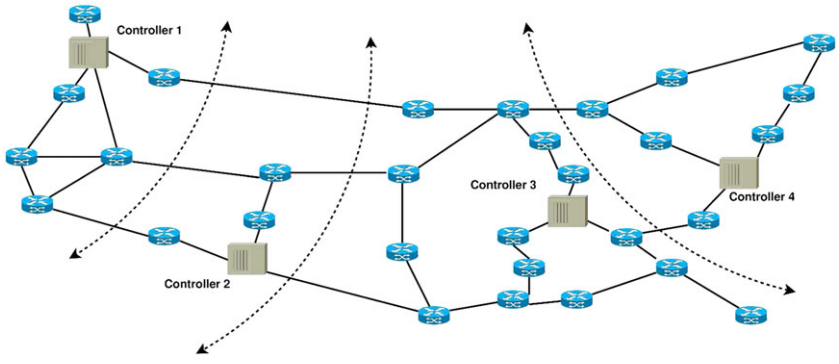

FIGURE 3 Partitions based on average case latency

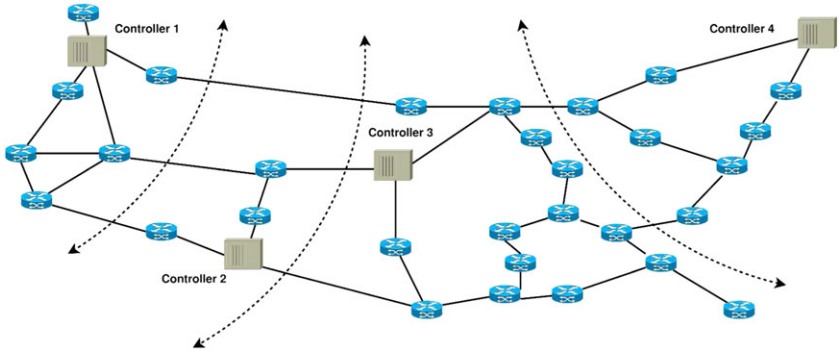

FIGURE 4 Partitions based on worst case latency

TABLE 4 The key points considered from literature for Un-capacitated controller placement problem

	Parameters Considered				Solution Based on	
Ref.	**WAN**	***S2C***	**Network Partitioning**	**Fault Tolerant**	***ILP***	**Others**
Heller et al[12]	✓	✓	✗	✗	✗	✓
Yan-nan et al[13]	✗	✓	✗	✓	✓	✗
Hu et al[14]	✓	✓	✓	✓	✗	✓
Yan-nan et al[15]	✗	✓	✗	✓	✓	✗
Gua and Bhattacharya[16]	✓	✗	✓	✓	✗	✓

S2C= switch-to-controller latency and *ILP*= Integer Linear Programming.

3.2 | Capacitated CPP

In CCPP, load and capacity of controllers are considered during the placement of controllers. There is a chance of controller failure if load and capacity are not taken into account. In other words, we can say that some controllers may have overloaded and some controllers are negligibly loaded. Thus, load balancing is required for the load distribution to controllers having negligible load. Most of the researchers have considered capacity and load of the controller for placing the controllers in SDN. Load on the controller can be measured in terms of the number of switches associated with that controller and a number of flow initiations per second. Most popular NOX controller can handle around 30k flow initiations per second.[7] The load and capacity can be taken into account in the following ways during placement of controllers.

3.2.1 | Static traffic-aware CCPP

As already discussed, most of the earlier research works ignore the load balancing between nodes and controllers, which is a major factor for the better performance of SDN. Generally, traffic (latency) between nodes and controllers, ie, static latency is predefined. The load on the controllers and intercontroller latency is considered by Hock et al.[21] This work is based on Pareto-based optimal placement (POCO toolset) in MATLAB to achieve optimal placement of controllers. The POCO provides better load balancing between switch and controllers and balanced placement of controllers in the SDN. This work uses load imbalance ($\pi^{imbalance}$)as a metric, which is the difference between controller having more number of switches and controller having less number of switches. Load imbalance ($\pi^{imbalance}$) and intercontroller latency ($\pi^{controller-latency}$) are defined for failure-free scenarios in Equations 10 and 11, respectively.

$$\pi^{imbalance} = \max_{c\epsilon C} n_c - \min_{c\epsilon C} n_c, \tag{10}$$

$$\pi^{controller-latency} = \max_{c_1,c_2\epsilon C} d(c_1, c_2), \tag{11}$$

where, C=$\{c_1,c_2,c_3,.....c_k\}$ is the set of controllers to be placed and n_c is the number of switches connected to controller ($c\epsilon C$) for failure-free scenarios, n_c^s is the number of switches connected to controller ($c\epsilon C$) for failure scenarios ($s\epsilon f$), $f\epsilon\{node\ failure, controller\ failure\}$. Load imbalance metric is defined for failure scenarios (ie, $\pi_f^{imbalance}$) in Equation 12. This metric calculates the totally balanced distribution for all possible failure scenarios.

$$\pi_f^{imbalance} = \max_{s\epsilon f} \left(\max_{c\epsilon C} n_c^s - \min_{c\epsilon C} n_c^s\right). \tag{12}$$

The objective of Hock et al[21] is to cover important possible failure scenarios for reliable CPP. Other objectives are (1) load balancing between controllers, (2) intercontroller latency, and (3) trade-offs between failure-free cases and failure cases. It is restricted to small- and medium-sized networks and does not work well for the large-sized networks. In Hock et al,[21] maximum switch-to-controller latency is discussed, and the average switch-to-controller latency is ignored.

Bargaining game-based controller placement of the SDN is discussed by Ksentini et al.[22] They have considered 3 objectives: (1) latency between switches and controllers, (2) latency among controllers, and (3) load balancing between controllers simultaneously. The authors claim that their simulation results are better than other mono-objective–based CPP results.

3.2.2 | Dynamic traffic-aware CCPP

In changing network traffic scenarios, if switch-to-controller latency is taken as constant, then the result is not good for load balancing among the controllers. To overcome this limitation, dynamic traffic load is considered while placing the controllers on the SDN. Various research works have been done in this category.

Controller placement problem in dynamic environments is proposed by Dixit et al.[4] The authors proposed a mechanism for the dynamic environment so that switches can be dynamically assigned to the controller(s). Using this mechanism load can be distributed among the controllers and thereby reducing the chance of controller failure. The authors used Mininet[23] as the experimental test-bed and designed an elastic controller called EstiCon by adding an additional component in the SDN controller. According to traffic load, controller pool is dynamically shrunk or grown and dynamically shifts the load among the controllers. Intercontroller synchronization overhead is reduced. Placement metrics such as load on the controller and switch-to-controller latency are ignored, and these metrics are considered by Latz et al.[24] Dynamic load distribution among controllers and switch migration from one controller to another has been discussed, but the placement of controllers in SDN has been ignored. Dynamic mapping is not suitable for the large-sized network because the number of switch migrations leads to more communication overhead.

In 2014, Hock et al[25] extended their work (SDN network under static conditions) to dynamic conditions. Authors present the best placement of controllers based on the propagation latency and a load on the controller(s). Hock et al[25] considered dynamic parameter (such as changing switch-to-controller latency of the network and changing loads of the controllers) and analyze different types of use cases. These use cases are as follows: (1) static switch-to-controller latencies and dynamic load on switches, (2) dynamic switch-to-controller latencies and constant load on switches, and (3) dynamic switch-to-controller latencies and dynamic load on switches. This work is limited to small and medium-sized networks. In Hock et al,[26] authors further extended their work to improve the reliability of the SDN networks.

A mathematical model for optimal controller placement is proposed by Sallahi and St-Hilaire.[27] Controllers and links can be activated or deactivated to enhance the performance of the network. They find an optimal number of controllers, locations where controllers should be placed and the kind of controllers. This model tries to minimize the cost of the controllers. This is the first mathematical model for the SDN controller placement. They use CPLEX Optimizer 12.5 to find all optimized results for controller placement in SDN.[28] In Table 5, we list down the mathematical symbols used in this model. Equations 13, 14, and 15 define the controller's installation cost, linking cost of controllers to switches, and linking cost of intercontroller, respectively. Controller placement problem can be modelled to minimize the total cost of the network as given in Equation 16.

$$C_c(x) = \sum_{c\epsilon C} k^c \sum_{p\epsilon P} x_{cp}, \tag{13}$$

$$C_l(v) = \sum_{l\epsilon L} \phi^l \sum_{s\epsilon S} \sum_{p\epsilon P} d(s,p) v^l_{sp}, \tag{14}$$

$$C_t(z) = \sum_{l\epsilon L} \phi^l \sum_{s\epsilon S} \sum_{p\epsilon P} d(s,p) z^l_{sp}. \tag{15}$$

Controller placement problem can be formulated as follows:

$$\min(C_c(x) + C_l(v) + C_t(z)), \tag{16}$$

subject to

$$\sum_{c\epsilon C} x_{cp} \le 1, \qquad p\epsilon P \tag{17}$$

$$\sum_{q\epsilon P} \sum_{l\epsilon L} (z^l_{pq} + z^l_{qp}) + \sum_{s\epsilon S} \sum_{l\epsilon L} v^l_{sp} \le \sum_{c\epsilon C} \alpha^c x_{cp}, \quad p\epsilon P \tag{18}$$

$$\sum_{l\epsilon L} \sum_{p\epsilon P} v^l_{sp} = 1, \quad s\epsilon S \tag{19}$$

TABLE 5 Symbols and definitions used in Sallahi and St-Hilaire[27]

Symbols	Definitions
S	Set of switches
$P \subseteq S$	Set of all possible placements for controllers
C	Set of controllers
$d(s,p)$	Shortest distance between switch $s \epsilon S$ and placement $p \epsilon P$
k^c	Price (in \$) of controller $c \epsilon C$
$L=\{l_1,l_2,l_3.....\}$	Set of links to connect the switches and controllers
ϕ^l	Price (in \$ per meter) of link
α^c	Number of ports available for the controller $c \epsilon C$
σ^s	Number of packets
μ^c	Number of packets/second that can processed by controller $c \epsilon C$
φ^c	Number of available controllers $c \epsilon C$
x_{cp}	Binary variables (0,1)
	Equals to 1, if controller $c \epsilon C$ deployed at location $p \epsilon P$
	Equals to 0, otherwise
v_{sp}^l	Binary variables (0,1)
	Equals to 1, if link $l \epsilon L$ established between switch $s \epsilon S$ and controller $c \epsilon C$ at location $p \epsilon P$
	Equals to 0, otherwise
z_{pq}^l	Binary variables (0,1)
	Equals to 1, if location $p \epsilon P$ connected to location $q \epsilon P$
	Equals to 0, otherwise
$C_c(x)$	Cost of deploying controllers
$C_l(v)$	Linking cost of controllers and switches
$C_t(z)$	Linking cost of controllers

$$\sum_{s \epsilon S} \sum_{l \epsilon L} \sigma^s v_{sp}^l \leq \sum_{c \epsilon C} \mu^c x_{cp}, \quad p \epsilon P \tag{20}$$

$$\sum_{s \epsilon S} x_{cp} \leq \varphi^c, \quad c \epsilon C. \tag{21}$$

The objective of this model is to minimize the optimization problem as given in Equation 16. Equation 17 represents the uniqueness constraint for controllers. The number of connected controllers and switches to a controller is less than the number of available port on the controller (see Equation 18). Equation 19 implies that only one link is given between a switch and the controller. Controller can process the number of packets of the switches as given in Equation 20. Equation 21 represents the limiting constraint for controllers. This model is valid only for small area networks (1 km × 1 km) in SDN. Authors extended their work by considering expansion problem.[29] Expansion problem can be defined as the expansion of current SDN infrastructure such that investment cost is minimized.

An optimal controller placement technique by using the concept of the Non-Zero-Sum game is presented by Rath et al.[30] Controllers can be either added or deleted dynamically and can also go to the sleep mode periodically based on the load on the controllers. It only provides maximum utilization of the controllers in the SDN but does not focus on placement of controllers in SDN. This technique improves the QoS and also minimizes the deployment cost of the controllers. Both Sallahi and St-Hilaire[27] and Rath et al[30] have considered propagation latency between the switches

and controllers and traffic load of the switches as an input for placement of controllers. Both research works are limited to small-sized networks. In Table 6, we list down the mathematical symbols used in the Non-Zero-Sum game–based CPP.

The minimum number of controllers (k) can be computed using Equation 22. If dynamic traffic load is considered, the value of k can be changed. Controller(s) can be added or removed based on the current load on the network. Rath et al[30] assume that l_i number of new flows and average load are received by switch i and that can be handled by one controller C_l.

$$\lceil k \rceil = \frac{\sum_{i=1}^{n} l_i}{C_l}. \tag{22}$$

Addition and deletion of a controller(s) in the network are decided by the optimization problem which is given in Equation 23. If load keeps on changing, the unique value of k is not possible for placement problem in SDN.

$$\min f(k, c), \tag{23}$$

subject to

$$\begin{aligned} \Delta_i &\leq \Delta_{th}, \quad \forall i \epsilon I \\ U_\alpha \leq U_i &\leq U_{th}, \quad \forall i \epsilon I. \end{aligned} \tag{24}$$

Global optimization problem is given in Equation 23 can be solved by using local optimization problem, which is given in Equation 25.

$$\min c_i, \tag{25}$$

subject to

$$\begin{aligned} \Delta_i &\leq \Delta_{th}, \quad \forall i \epsilon I \\ U_\alpha \leq U_i &\leq U_{th}, \quad \forall i \epsilon I. \end{aligned} \tag{26}$$

Their objective is to solve Equations 23, 24, 25, and 26 by using a non-zero-sum game.

Kyung et al[31] have proposed load distribution scheme based on per-flow for controllers in SDN. If the capacity of the controllers reaches its threshold, the incoming message is routed to other controllers to avoid blocked state of the controller. On the arrival of a new flow, the switch sends *Packet-In* message to the default controller. If the controller is not overloaded, it sends back Packet-Out message to the switch. However, if the controller is overloaded, it then sends *Packet-In* message to another controller and then finally it sends back *Packet-Out* message to the switch. Here, authors use Markov Chain model for formulating this model.

Research work of Jimenez et al[32] is not limited to the small area SDN. They have discovered the minimum number of controllers and their locations to create a robust control topology for the controller placement in SDN. Authors have concentrated on designing scalable control layer for SDN and then finding the minimum number of controllers. Load balancing among the selected controllers is also considered for SDN. Expected data loss can be easily calculated using predefined approach. Network topology can be constructed in the form of tree topology and the root of the topology act as a global controller (logically centralized controller), which has a global view of the whole network. And next children serve as the local controller (no interconnections among local controllers), which have a local view of the network. It means that they have only information of attached networking devices.

TABLE 6 Symbols and definitions used in Rath et al[30]

Symbols	Definitions
f	Function of controllers and cost associated with controllers
U_i	Utilization index
U_α	Maximum Utilization threshold
U_{th}	Minimum Utilization threshold
Δ_i	Delay associated with controller at time T

Yao et al[33] have proposed capacitated to CPP in 2014 and considered propagation latency and dynamic traffic load of switches for placements of controllers. Failure scenario can be handled by adding additional controllers in the SDN environment. It minimizes the maximum switch-to-controller latency and ignores the average switch-to-controllers latency because author avoids situations where some switches are far from the assigned controllers. In Yao et al,[33] authors assume heterogeneous traffic load whereas[32] assume homogeneous traffic load. Both works are applicable for large-sized network. An objective is to select the minimum number of controllers to build the robust control layer. The optimal number of controllers required in the wireless network is discussed in Johnston and Modiano. [34]

Ruiz-Rivera et al[35] introduce energy-aware (GreCo) controller placement strategy for SDN. The authors try to reduce the energy consumption to maximize the link utilization and switch off as many links as possible. Here, they assumed that the locations of controllers are already known. They discuss energy consumption and load on the controller. Binary integer programming is used to find out the optimal solution for the controller placement problem in SDN. GreCo is capable of switching off almost 55% of the total number of links in networks, and it showed that 20% links are used in the peak hours. And here they are also reducing the active links between the switches and controllers, and intercontrollers. Hu et al[36] also addressed the energy-aware SDN CPP and discussed the energy consumption of network, which was modeled by using binary integer programming. Improved genetic controller placement algorithm (IGCPA) is used to solve the CPP. After finding the locations of controllers, allocation process of switches is done by using GreCo.

The comparison of key points considered from literature for dynamic traffic-aware CCPP is given in Table 7

3.2.3 | Fault-aware CCPP

We have already discussed fault-aware CPP for which capacity is not a constraint. In this section, we analyze the solution of fault-aware CPP where capacity is a constraint.

In 2014, Ros and Ruiz et al[37] introduced fault-tolerant CPP in SDN environment. To achieve five nines (99.999%) of reliability, the authors have found a solution where each node on an average needs to establish a connection with 2 controllers or at most 3 controllers. So that, in the case of failure, southbound reliability is assured. To achieve the aforementioned reliability constraint, authors have proposed a heuristic algorithm where connectivity between controllers and switches act as a surrogate for reliability. For finding connectivity, the authors find all disjoint paths that help to improve reliability by solving the max-flow problem. The authors provide a mathematical proof of the existence of some nondisjoint paths in the max-flow network, which results in degraded reliability and performance. The extended version of heuristic algorithm[38] forms all set of disjoint paths by analyzing the properties of topology. In previous works,[37,38] real Internet topologies are taken from publicly available topology Zoo,[20] and they test their algorithm for all publicly available topologies. The objective of other works[37,38] is to minimize the total deployment cost of the controllers. They ignore the switch-to-controller, inter-controller latency, and load imbalance among controllers.

Survivability-based CPP is proposed by Muller et al.[39] In survivability, they find the number of disjoint paths (used for the failure scenarios) between the switches and controllers. This work is divided into 2 parts: (1) placement of controllers in the SDN environment, where they find the optimal controllers for placement in such a way that connectivity between

TABLE 7 Comparison of literature of dynamic-traffic CCPP

		Parameters Considered			Solutions Based on		
Ref.	**WAN**	***Lty***	***Cost***	***Load***	***Heuristics***	***Integer programming***	***Others***
Dxit et al[4]	✗	✗	✗	✓	✗	✗	✓
Hock et al[25]	✗	✗	✗	✓	✗	✗	✓
Kyung et al[31]	✗	✗	✗	✓	✗	✗	✓
Sallahi and Hilaire[27,29]	✗	✗	✓	✓	✗	✗	✓
Rath et al[30]	✗	✗	✓	✓	✗	✗	✓
Jimenez et al[32]	✓	✓	✗	✓	✗	✗	✓
Yao et al[33]	✓	✓	✗	✓	✓	✓	✗
Johnton and Modiano[34]	✗	✗	✗	✗	✗	✓	✓
Rivera et al[35]	✗	✗	✗	✓	✓	✓	✗

Lty=switch-to-controller or intercontroller latency and *Cost*=minimize the total cost of network.

switch and controller is maximized and capacity of controllers is also taken into account and (2) maintain the list of backup controllers, this category finds the list of backup controllers as these are useful in case of controller failure. Each controller reserves some percentage of capacity as a backup. The objective is to maximize the connectivity (average disjoint paths) between controller and switches. This mechanism does not specify how auxiliary connections and the list of backup controllers are defined.

In 2015, Lange et al[40] extended their previous works[25] and eliminate the limitations (only valid for small- and medium-sized networks) of previous work. Additionally, the authors have considered failure scenario, load balancing as well as load on the controllers. They provide heuristics based controller placement problem for large-sized networks. They have also considered the dynamic SDN environment (variations in switch-to-controller and inter-controllers latencies) for the placement of controllers. Research works[12,21,40] ignore the expected data loss in case of the network element (or controller) failure. In Lange et al,[40] the authors focus on all the metrics simultaneously switch-to-controller latencies, inter-controller latencies, load balancing on the controllers, and the failure of a controller(s), switches, and links. That is why CPP is described as MOCO (multiobjective combinatorial optimization). In previous works,[25] the authors focus only on finding maximum latency for switch-to-controller and inter-controller latency but here average latency is also taken into account. An objective of Lange et al[40] is to minimize the average and maximum switch-to-controller latency, load imbalance, and intercontroller latency.

Jalili et al[41] provide a multiobjective genetic algorithm–based solution for CPP. A genetic algorithm is used for the single-objective optimization problem. Here, CPP is treated as a multiobjective combinatorial optimization (MOCO) problem. They have proposed the heuristics based NSGA-II to solve the controller placement problem. Load on the controller(s), switch-to-controller latency, and intercontroller latency are considered. Internet2 OS3E network topology is used and simulated in MATLAB 2013b. This approach provides a minimum number of controllers as compared to Lange et al. [40]

Guo et al[42] have considered SDN failure cases. Firstly, the authors have formulated controller placement problem into two optimization problems: "Controller Placement under the Comprehensive Network States" (CPCNS) problem and "Controller Placement under Single Link Failure" (CPSLF) problem. Secondly, they have proposed optimal placement algorithm and greedy algorithm to solve the CPSLF problem. In Guo et al,[42] authors are reported that this is the first research work of CPP, where single link failure case has been taken into account. Almost 70% networks fail due to single link failure. They have improved the network performance when link failures occur and have ignored the load on the controller as well as the expected percentage of data loss during the network failure.

In 2015, Su and Hamdi[43] proposed low-cost measurement framework for the CPP. This approach minimizes the synchronization cost, communication cost, and measurement overhead. The authors have formulated measurement aware CPP as a quadratic integer programming that takes both flow statistics collection cost and synchronization cost into consideration. They built their simulator which is written in python. It reduces 40% of the measurement overhead on an average. The primary objective of Su and Hamdi[43] is to optimize the measurement overhead. The first term of this objective function is responsible for minimizing the synchronization cost among controllers and the second term reduces the communication cost between the controller(s) and switches.

Gao et al[44] defined a global latency for the CPP. In the literature, the load on the controller (capacity of the controller) and intercontroller latency have been ignored. Here, authors have focused on the capacity of the controller as well as latency among the controllers. This is PSO (particle swarm optimization) based first work for CPP that provides a solution for the placements of controllers when the number of controllers is known in advanced. The authors try to minimize the global latency of the SDN-based network. The global latency of the SDN is the combination of switch-to-controller latency and inter-controller latency. PSO-based solution provides a better result for the large-sized network as compared to the integer linear programming (ILP) and greedy algorithms. An objective of Gao et al[44] is to minimize the global latency of the network. But they ignored the failure scenarios for the controller(s) and switch(es).

Killi and Rao[45] proposed a mathematical model for fault-aware CCPP. They are assuming the failure foresight of switches. They have considered failure scenarios with capacity constraints, which means that if there are k controllers then this model will work well for up to $(k-1)$ controllers failure. In this situation, only one controller takes the responsibility of all $(k-1)$ failed controllers without any drastic changes in latency and performance. This problem is known as "Failure Foresight Capacitated Controller Placement Problem (FFCCPP)". The objective of FFCCPP is to minimize the worst case latency between switched and controllers while considering both capacity and reliability. They have also proposed a mathematical model for the combined objective for fault-aware CCPP which is named as "Combined Objective-Failure Foresight Capacitated Controller Placement Problem (CO-FFCCPP)." The aim of CO-FFCCPP is to minimize the combined worst-case latency of all k controllers and its attached switches. FFCCPP and CO-FFCCPP provide better

results as compared with CCPP while considering failure. They ignore average-case latency between switches and controllers, and latency among controllers.

Zhong et al[46] propose a min-cover based reliable controller placement. Reliability of the SDN totally depends on the size of the network and positions of the controllers deployed within the network. They focus on link failure and management of the failed link(s). It requires less number of controllers for managing a reliable SDN as compared to standard controller placement strategy, and at the same time, the latency between switches and controllers is also minimized. It maintains the list of backups links, to deal with the case where links get failed due to some reasons. The objective is to improve the reliability and minimize the number of required controllers simultaneously.

Another reliability based algorithm for CPP is proposed by Liu et al.[47] Simulations are tested in Internet2 OS3E and Internet Topology Zoo. The results show that its performance is better than other regular placement algorithms. The objective is only to improve the reliability. Both works[46,47] ignore the switch-to-controller latency, inter-controller latency and load on the controller(s).

Previous research works only focused and minimized the latency between the controller(s) and its attached switches and ignored the intercontroller latency, which is important for maintaining the consistency among the controllers. Zhang et al[48] have discussed the distributed SDN controller placement to improve the network performance and reliability. Coordination is required for the distributed SDN controllers (for example, OpenDaylight). Consistency plays a critical role in case of any updates are done by the master (leader) controller, and these updates propagate to all follower controllers.

Perrot and Reynaud[49] discuss controller placement problem for the resilient SDN. Resilient controller placement problem is solved by proposed integer linear programming (ILP). This solution provides a minimum number of controllers to achieve the QoS, load balancing between controllers and maintain the consistency even when there is a fault. POCO framework is used for the evaluation of the solutions. SDN Lib and CPLEX are used for their evaluation of numerical results. One of the objectives is to achieve "good" placement regarding quality of services (QoS) and also to minimize the number of active controllers within the latency bounds.

Killi and Rao[50] presented Controller Placement With Planning for Failures (CPWP). CPWP is limited to the small and medium-sized network. They discuss Capacitated Next Controller Placement (CNCP) in Killi and Rao[51] to remove the limitation of their previous work.[45,50] Here, they consider capacity and all possible failure scenarios without assuming the failure foresight of switches. Each switch is connected to all the reference controllers. A switch is assigned to the controller with minimum latency which is called the nearest controller of a particular switch. The nearest controller then finds its closest controller from all set of controllers. Its objective is to minimize the maximum of the sum of the switch-to-nearest controller and nearest controller-to-closest controller. This problem is formulated as mixed integer linear programming (MILP) using two-indexed and three indexed decision variables. Simulated annealing is used to salve the problem. CNCP provides better results as compared with CCPP while considering all failure scenarios.

The key points considered from literature for fault-aware capacitated controller placement problem are as given in Table 8

3.2.4 | Network partitioning–based CCPP

Initially, researchers reported that the response time of the controller depends only on switch-to-controller latency. According to recent claims, the response time of controller depends on both switch-to-controller latency and load on the controller. SDN performance depends on the time of reply between switch and controller, load balancing, and reliability. Controller performance is evaluated by Tootoochian et al[7] and the authors found that the controller handles a limited number of the service requests. If controller gets overloaded and there exists large propagation latency between switches, then partitioning comes into the picture.

In 2013, Bari et al[52] introduced dynamic controller provisioning problem (DCPP) in SDN for the first time. Multiple controllers are deployed in the large-sized network and simultaneously manage and control this network. Optimal DCPP is formulated by integer linear programming (ILP). They have proposed algorithms to minimize the communication overhead as well as flow setup time with the help of integer linear programming. In Bari et al,[52] networking frame dynamically adjusts the active controllers and inactive controllers (active controller is represented by 1, and the inactive controller is represented by 0). In other works,[7,52] authors remove the static version problem of Heller et al.[12] Greedy Knapsack (GK) and Simulated Annealing (SA) are proposed to solve the controller placement problem. First, authors try to simulate their work in Mininet with POX[53] controllers. Both mininet and POX controller are deployed on the same physical machine and after the experiment, they found that mininet cannot support very large traffic.

TABLE 8 The key points considered from literature for controller placement problem

		Latencies/Load			Solutions based on		
Reference	**WAN**	***S2C***	***C2C***	**Load**	**Heuristics**	**Integer Programming**	**Others**
Ros and Ruiz[37,38]	✓	✗	✗	✓	✓	✗	✗
Muller et al[39]	✓	✗	✗	✓	✓	✗	✗
Lange et al[40]	✓	✓	✓	✓	✓	✗	✗
Jalili et al[41]	✓	✓	✓	✓	✓	✗	✗
Gao et al[42]	✓	✓	✗	✓	✗	✗	✓
Su and Hamdi[43]	✓	✗	✗	✓	✗	✓	✗
Gao et al[44]	✓	✓	✓	✓	✓	✗	✗
Killi and Rao[45]	✗	✓	✗	✗	✗	✗	✓
Killi and Rao[50]	✗	✓	✓	✗	✓	✓	✗
Killi and Rao[51]	✓	✓	✓	✓	✓	✓	✗
Zhong et al[46]	✓	✗	✗	✗	✓	✗	✗
Liu et al[47]	✓	✗	✗	✗	✗	✗	✓
Zhong et al[48]	✓	✗	✗	✓	✓	✗	✗
Perrot and Reynaud[49]	✓	✓	✓	✗	✗	✓	✗

S2C=switch-to-controller and *C2C*= controller-to-controller.

Switch-to-controller latency goes through only single LoopBack interface of the physical machine and as a result switching capability of LoopBack interface is limited. Second, deploying mininet on one physical machine and POX controller in another physical machine decreased the load on the LoopBack interface. However, this modification is failed to provide support for large traffic. Finally, authors have developed in-house simulator by which it can easily handle such a large traffic.

The objective of Bari et al[52] is to minimize the communication overhead as well as flow setup time. But the controller load imbalance, switch-to-controller latency, and inter-controller latency metrics are ignored. Authors do not explain how to find which switches are reassigned to the controller.

Cheng et al[54] introduced QoS-guaranteed controller placement problem. In the literature, many researchers have highlighted load distribution, propagation latency, fault resilience but they ignore the response time of the controller which is an important QoS parameter. Here, the authors have proposed three heuristic based algorithms for the controller placement problem. These algorithms are incremental greedy algorithm, primal-dual based algorithms, and network partitioning algorithm. Simulations have been performed in Internet Topology Zoo. In their simulations, incremental greedy, primal-dual and network partitioning algorithms have used 4, 7 and 5 controllers on an average respectively.

LiDy (location independent dynamic flow management) is a new scheme proposed by Huque et al.[53] It provides best algorithms to find the optimum controllers and locations where controllers can be deployed to minimize the propagation switch-to-controller latency, flow setup cost as well as maintenance cost in the SDN. Authors try to achieve better controller utilization. Utilization of the controller is in terms of a number of switches and number of flows. Authors extended their work[53] by proposing a modified LiDy scheme, ie, LiDy+[55] which gives a better result as compared to LiDy. LiDy+ runs in $O(n^2)$ where LiDy takes $O(n^2 logn)$ time complexity. LiDy+ does not only provide the minimum number of controllers and maximum controller utilization but also takes less energy consumption and maintenance costs as compared to LiDy. Both schemes provide better results as compared to previous studies.[4,12,13,21,25-27,30,32,33,40,52,56] In previous research work, controllers were deployed only to switch locations, but here, the authors' aim is to deploy the controller in a location other than of the switch locations so that the propagation latency gets reduced.

The controller placement scheme with flow-based switch migration algorithm is proposed by Rivera et al.[56] In literature, researchers have mainly focused on static controllers and switches, which are not better for the controllers because in the static case, the load may be imbalanced. So, load balancing among controllers requires dynamicity. This work primarily partitions the SDN network into domains and then defines the controller placement metric for the single domain, and finally, it tries to distribute the load on multiple domains of SDN. The objective is to minimize the cost of the deploying controller at the candidate location.

The network clustering based particle swarm optimization algorithm for the controller placement in SDN is proposed by Wang et al.[57] It is an extended work of Gao et al.[44] The authors divide the network into k clusters and each cluster has its controller for controlling and management. Load on the controllers is taken into consideration, which is the critical factor for large-sized network. The load on controllers, switch-to-controller latency, inter-controller latency, and load balancing are considered. Results show better performance as compared to k-center and capacitated k-center strategy. The objective is to achieve maximum utilization by every controller while taking load balancing into account. Another network partitioning based algorithm for CPP is proposed by Ishigaki et al,[58] and they discuss the load on the communication nodes (switches or controllers). When network size is too large, loads on the communication nodes play an important role.

CPP for multidomain SDN is formulated by Aloki and Shinomiya.[59] With the help of partitioning algorithm, the large-sized network is divided into different SDN domains, and each domain has its controller for managing the domain. This problem is twofold: (1) how to partition the WAN network and (2) where to place the controller in each SDN domain. The symbols and definitions related to this scheme are given in Table 9.

The maximum switch-to-controllers latency of each SDN domain is defined as follows:

$$\min \sum_{G_j^l \epsilon G^l} L(G_j^l), \tag{27}$$

$$L(G_j^l) = \max_{v_i \epsilon S_i^l \setminus \{p_i\}} d(v_i, p_i), p_i \epsilon P^{'}. \tag{28}$$

In Aoki and Shinomiya,[59] the objective is to minimize the latency (see Equation 27) of the network which is given in Equation 28 and find the $P^{'}$ from all possible placements P such that $|P^{'}|=k$.

Spectral Clustering based Controller Placement (SCCP) algorithm is used by Xiao et al[60] to divide the large-sized network into different small networks and then find the optimum number of controllers. Min-max cut function is used for partitioning the large-sized network into SDN domains, and each SDN domain has its controller. Here, the question arises as to where the controller should be placed in SDN domain, to improve the performance of the SDN. Reliability and efficiency of the controller can be enhanced by reducing the size of the network. Performance metrics such as average switch-to-controller latency, load balancing among the controllers and reliability have been considered. In Xiao et al,[61] authors extended their previous research work[60] to ensuring security, reliability, management and so on. An objective is to minimize switch-to-controller latency. Additionally, authors focused on balanced partitions of the large-sized network.

In Liao et al,[62] Density-based Controller Placement (DBCP) for large-sized network has been introduced. A large-sized network is partitioned into small-sized networks. It provides a better result for CPP as compared to Lange et al.[40] The authors discuss CPP with capacity and without capacity and provide solutions for each of them and also focus possible failure scenarios. In most of the research works, a known number of controllers are assumed for large-sized networks, because it is not easy to determine the minimum number of controllers. Without traversing all possible locations, we cannot find this number, which is not feasible for large-sized networks.

Optimized k-means network partitioning algorithm is proposed by Wang et al.[63] Latency is the only parameter considered in controller placement and network partitioning problem. CPP is discussed as a twofold problem: (1) partitioning the network based on the k-means algorithm and (2) placing the controllers in the partitioned networks. Again partitioning of the network relies on two subtasks: (1) minimizing the response time between switches and controllers and (2) implementing load balancing and fault tolerance directly implemented onto the partitioned networks instead of the whole networks. The optimized k-means algorithm provides the better result as compared to

TABLE 9 Symbols and definitions of used in Aoki and Shinomiya[59]

Symbols	Definitions
$G(S,E)$	Graph G, where S is the Set of switches and E is the link between switches
$G_i^l(S_i^l, E_i^l)$	SDN domains for i=1,2,3,.....k
$G^l = \{G_1^l, G_2^l .....G_k^l\}$	Set of SDN domains

TABLE 10 Comparative analysis of controller placement problem

S.No.	Placement Metrics	Environment	Topology Used	Solutions/ Algorithms	Network Size	Traffic Load Balancing	Network Partitioning	Ref.
1	1. k-center	Static	Internet2 OS3E	Random Placement	Large-sized	No	No	Heller et al[12]
	2. k-median							
2	Reliability aware CPP	Static	1. Internet2 OS3E	1. l-w greedy	Small and Medium-sized	No	No	Hu et al[15]
			2. Rocketfuel	2. Random Placement				
3	FTCP	Static	1. Ring	Placement Algorithms	Large-sized	No	Yes	Ksentini et al[22]
			2. Binary tree	for improving resilience				
			3. Erdos-Renyi random					
4	Reliability aware CPP	Static	1. Internet2 OS3E	1. Random placement	Small and Medium size	No	No	Hu et al[13]
			2. Rocketfuel	2. l-w greedy				
				3. Simulated annealing				
				4. Brute Force				
5	Reliability	Static	Internet2 OS3E	1. l-w greedy	Large-sized	No	No	Hu et al[14]
	aware CPP			2. Simulated annealing				
6	k-center	Static	Internet2 OS3E	POCO	Small and Medium-sized	Yes	No	Hock et al[21]
7	Not discussed	Dynamic	Network with 4 hosts	ElastiCon	Small-sized	Yes	No	Dixit et al[4]
8	Mathematical model	Dynamic	Not discussed	Markov Chain Model	Small and Medium-sized	Yes	No	Kyung et al[31]
9	Reliability aware CPP	Dynamic	Internet2 OS3E	POCO-PLC	Small and Medium-sized	Yes	No	Hock et al[25]
10	Mathematical model	Dynamic	Network with switches 10, 20, 30, 40, 50,75, 100, 150, 200 and controllers 10, 15, 20	CPLEX	Small-sized ($1km*1km$)	Yes	No	Sallahi and St-Hilaire[27]
11	Game Theoretic based CPP	Dynamic	Random network with 28 switches	Non-zero Sum Game	Small-sized	Yes	No	Rath et al[30]

(Continues)

TABLE 10 (Continued)

S.No.	Placement Metrics	Environment	Topology Used	Solutions/ Algorithms	Network Size	Traffic Load Balancing	Network Partitioning	Ref.
12	1. k-center	Dynamic	Sparse, Medium and Dense	K-critical	Large-sized	Yes	No	Jimenez et al[32]
	2. k-median							
13	Capacitated	Dynamic	Internet Zoo	Linear Relaxation	Large-sized	Yes	No	Yao et al[33]
	k-center							
14	Binary Integer	Dynamic	1. Abilene(11nodes, 28links)	Heuristic Algorithm	Small and Medium-sized	Yes	No	Ruiz-Rivera et al[35]
	Programming		2. AT&T(25nodes, 112links)					
			3. GEANT(40nodes,112links)					
			4. SURFnet(50nodes, 138 links)					
15	FTCP	Dynamic	Internet Zoo	Heuristic Algorithm	Large-sized	Yes	No	Previous works[37,38]
16	FTCP	Dynamic	2. Internet2 OS3E	Heuristic Algorithm	Large-sized	Yes	No	Muller et al[39]
			2. GEANT					
			3. RNP					
17	1. k-center	Dynamic	2. Internet2 OS3E	Pareto Simulated	Large-sized	Yes	No	Lange et al[40]
	2. k-median		2. Internet Zoo	Annealing				
18	1. k-center	Dynamic	Internet2 OS3E	NSGA-II	Large-sized	Yes	No	Jalili et al[41]
	2. k-median							
19	1. CPCNS	Dynamic	2. Internet2 OS3E	1. Optimal Placement	Large-sized	Yes	No	Guo et al[42]
	2. CPSLF		2. Internet Zoo	2. Greedy Placement				
			3. Cernet2					
20	QoS-	Dynamic	Internet Zoo	1. Incremental Greedy	Large-sized	Yes	Yes	Cheng et al[54]
	Guaranteed			2. Primal Dual based				
	CPP			3. Network partition-based				
21	Quadratic	Dynamic	Internet Zoo	1. Discrete Approximation	Large-sized	Yes	No	Su and Hamdi[43]
	Integer			2. Connectivity Ranking				
	Programming							

(Continues)

TABLE 10 (Continued)

S.No.	Placement Metrics	Environment	Topology Used	Solutions/ Algorithms	Network Size	Traffic Load Balancing	Network Partitioning	Ref.
22	k-median	Dynamic	Internet2 OS3E	PSO	Large-sized	Yes	No	Gao et al[44]
23	ILP	Dynamic	1. RF-I(79 nodes, 294 links)	2. DCP-GK	Large-sized	Yes	Yes	Bari et al[52]
			2. RF-II(108nodes, 306links)	2. DCP-SA				
24	Divide and Conquer	Dynamic	NSFNET	Flow based dynamic	Large-sized	Yes	Yes	Yao et al[56]
				scheduling				
25	1. Capacitated	Dynamic	Different size of topology	NCPSO	Large-sized	Yes	Yes	Liu et al[57]
	k-center							
	2. k-median							
26	k-median	Dynamic	Different size of topology	PSO-CGLCPP	Large-sized	Yes	Yes	Aoki and Shinomiya[59]
27	CNUM	Dynamic	Rocketfuel	GAME-SM	Large-sized	Yes	Yes	Cheng et al[66]
28	Reliability based	Dynamic	1. Internet2 OS3E	Greedy algorithms	Large-sized	Yes	Yes	Liug et al[47]
	CPP		2. Internet Zoo					
29	SCCP	Dynamic	Internet2 OS3E	Spectral clustering	Large-sized	Yes	Yes	Other works[60,61]
30	DBCP	Dynamic	Internet Zoo	Density based clustering	Large Scale	Yes	Yes	Liao et al[62]
31	1.LICMP	Dynamic	Large-sized	LiDy+	Large-sized	Yes	Yes	Ul Huque MTI et al[55]
	2. LiDy+							
32	Mathematical model	Dynamic	Network with switches 20,25,30	CPLEX	Small-sized	Yes	No	Sallahi and St-Hilaire[29]
32	Mathematical model	Dynamic	Network with switches 20,25,30	CPLEX	Small-sized	Yes	No	Sallahi and St-Hilaire[29]
33	CPWP	Dynamic	Internet Topology Zoo	MILP	Small and Medium-sized	No	No	Killi and Rao[50]
34	CNCP	Dynamic	Internet Topology Zoo	MILP/simulated annealing	Large-sized	Yes	No	Killi and Rao[51]
35	ILP	Dynamic	Network with switches 11 and 26 links	Heuristic DSP	Large-sized	No	Yes	Zhao and Wu[64]

normal k-means based algorithm. The worst case (maximum) switch-to-controller latency is 2.437 times lesser than the average latency.

Zhao and Wu[64] discussed the distributed controller placement for large-sized SDN. Authors partitioned the large-sized SDN into small-sized SDN domains and placed the controller for each domain in order to minimize the switch-to-controllers latency. Unfortunately, this scheme introduced extra latency, ie, inter-controller latency. Controller placement problem is formulated as Integer Linear Programming (ILP). They provide an efficient heuristic based algorithm (named as distributed SDN placement) to solve this problem. Authors solve the issue of scalability and minimization of network cost. Simulation results of Zhao and Wu[64] show that heuristic based algorithm gives better solutions as compared to integer linear program results.

Distributed SDN Controller System is discussed by Oktian et al.[65] Distributed controller removes the limitation of the single controller. Many network issues such as load balancing, scalability, and reliability are easily handled with the help of distributed controller system. They have provided two design models: (1) *Flat model* in which all controllers in a cluster maintain the global network state and (2) *Hierarchical model* where some controllers (but not all) in a cluster maintain the global network state. The overall comparative analysis of CPP is given in Table 10.

4 | CONCLUSIONS AND FUTURE DIRECTIONS

First, we discuss the conclusive results of CPP, which are obtained from the discussions of this review. Second, we highlight the future directions of CPP, based on which the potential researchers can innovate new solutions for CPP.

4.1 | Conclusions

Placement of controller(s) is an important aspect in the large-sized SDN. Efficient controller placement tries to improve the performance metrics like propagation latency, reliability, load distribution, failure resilience, and so on. CPP was introduced in 2012. In the last 5 years, while many researchers have focused on this problem, no related review has been conducted. Thus, a state-of-the-art on CPP is required. To the best of our knowledge, this is the first state-of-the-art review on CPP. In this paper, first, we have presented the brief introduction of SDN and its architecture with the aim of setting up the background of SDN. To ensure scalability and reliability, multiple controllers are required for large-scale SDN. Optimum placement of controllers is needed to improve the performance of SDN. Reviewed research work of CPP is classified based on their solutions (mathematical models and algorithms) provided by recent researchers.

Next, we have surveyed the CPP research initiatives and already identified issues in CPP. Finally, we have investigated the open research issues of CPP which are not explicitly addressed in the literature. So there is a need to address these open research issues through future research efforts. A potential research effort in this direction could gain a truly reliable solution to SDN controller placement problem.

4.2 | Future directions

In this section, we present the future directions of CPP. However, CPP is in the infancy stage of the research to give openings for academician, industrialist, and vendors. In following paragraphs, we present the open research issues that cover the complete life cycle of SDN controller placement problem.

In general, network partitioning based solution for CPP is the type of clustering problem. K-means, spectral, density-based clustering are some of the commonly used clustering algorithms in the literature.[61-63] Numerous researchers have discussed the network partitioning based solution for CPP, but they do not emphasize load balancing among controllers and all possible failure scenarios.

Placing one controller per switch is not a better choice, and this is not the case of CPP. Using this concept, the performance of network will improve, but deployment cost of the controllers will be very high. Splitting the large-sized network into several clusters (domains) and placing the controllers or controller pool (set of controllers)[53] in each cluster seems to be a promising solution for CPP.[57,60] But it is a proactive solution which does not consider the dynamic scenarios such as occurrence of the random pattern in network traffic (ie, a momentary increase followed by idle traffic). In this case, a proper reactive solution is required. The main objective of CPP is to place the required number of controllers or controller pool in a large-sized SDN and also maximize the controller utilization. The controller utilization is poorly addressed in the literature and hence mandates further research.

During its developmental phase, SDN was deployed on a minimal network like a university campus. In recent years, a lot of research has been directed towards the feasibility of SDN deployment in a home, wireless, cellular network, and so on. Researchers conclude that SDN deployment is not a big challenge for small and medium-sized networks. However, implementation and deployment of SDN for a large-sized network is an open research issue.

Function Placement Problem (FPP) is a similar problem of CPP in the context of Network Function Virtualization (NFV).[40] FPP is formulated as an optimization problem[67-69] and is also NP-Hard.[70] Deploying Virtual Network Functions (VNF) in a specific order is called Service Function Chaining (SFC). Network bandwidth is fixed during the searching of SFC path,[71] and flow routes are also fixed.[72] Further, research is required on it to maintain the QoS.

Controller placement problem is similar to facility location problem[11] which is an old mathematical problem. Recent research works provide a solution for CPP in two different directions: (1) by using clustering mechanism and (2) by using ILP, Greedy, Heuristics approach, and so on. But no one has focused on relative merits and demerits of one over others. CPP thus remains an open research issue.

Currently, most of the researchers are working on controller placement problem, but they do not provide a very efficient solution for the large-sized network with dynamic traffic load and fault. So further research is required to increase the availability, performance of the network and making the network more reliable.

ORCID


Ashutosh Kumar Singh http://orcid.org/0000-0003-3922-4389

REFERENCES

1. Benson T, Akella A, Maltz D. Unraveling the complexity of network management. In: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI09) USENIX Association; 2009; Berkeley, CA:335-348.
2. Jammal M, Singh T, Shami A, Asal R, Li Y. Software defined networking: State of the art and research challenges. *Comput Networks*. 2014;72:74-98.
3. Yeganeh SH, Tootoonchian A, Ganjali Y. On scalability of software-defined networking. *IEEE Commun Mag*. 2013;51(2):136-141.
4. Dixit A, Hao F, Mukherjee S, Lakshman TV, Kompella R. Towards an elastic distributed sdn controller. In: *ACM SIGCOMM Computer Communication Review*. Hong Kong, China: ACM; 2013: Vol. 43:7-12.
5. Ahmed R, Boutaba R. Design considerations for managing wide area software defined networks. *IEEE Commun Mag*. 2014;52(7):116-123.
6. Lange S, Gebert S, Spoerhase J, et al.. Specialized heuristics for the controller placement problem in large scale sdn networks. In: Proceedings of the 27th International Teletraffic Congress (ITC'27) IEEE; 2015; Ghent:210-218.
7. Tootoonchian A, Gorbunov S, Ganjali Y, Casado M, Sherwood R. On controller performance in software-defined networks. *Hot-ICE*. 2012;12:1-6.
8. Sherwood R, Gibb G, Yap K, et al.. Flowvisor: a network virtualization layer. In: *OpenFlow Switch Consortium*. Stanford, California: Tech Rep; 2009:1-13.
9. Kannan K, Banerjee S. Compact tcam: Flow entry compaction in tcam for power aware sdn. *Proceedings of International Conference on Distributed Computing and Networking (ICDCN'13)*. Berlin Heidelberg: Springer; 2013:439-444.
10. Gupta D, Jahan R. Inter-sdn controller communication: Using border gateway protocol. White Paper by Tata Consultancy Services (TCS); 2014.
11. Arya V, Garg N, Khandekar R, Meyerson A, Munagala K, Pandit V. Local search heuristics for k-median and facility location problems. *SIAM J Comput*. 2004;33(3):544-562.
12. Heller B, Sherwood R, McKeown N. The controller placement problem. In: *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*. Helsinki, Finland: ACM; 2012:7-12.
13. Hu Y, Wendong W, Gon X, Que X, Shiduan C. Reliability-aware controller placement for software-defined networks. In: *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM'13)*. Ghent, Belgium: IEEE; 2013:672-675.
14. Hu Y, Wendong W, Gon X, Que X, Shiduan C. On reliability-optimized controller placement for software-defined networks. *China Commun*. 2014;11(2):38-54.
15. Hu Y, Wendong W, Gon X, Que X, Shiduan C. On the placement of controllers in software-defined networks. *The J China Univ Posts Telecommun*. 2012;19:92171-97.
16. Guo M, Bhattacharya P. Controller placement for improving resilience of software-defined networks. In: Proceedings of the 4thInternational Conference on Networking and Distributed Computing (ICNDC'13) IEEE; 2013; Los Angeles:23-27.

17. Zhang Y, Beheshti N, Tatipamula M. On resilience of split-architecture networks. In: *Proceedings of the International Conference on Global Telecommunications (GLOBECOM'11)*. Kathmandu, Nepal: IEEE; 2011:1-6.
18. Swamy C, Shmoys DB. Fault-tolerant facility location. *ACM Trans Algorithms (TALG)*. 2008;4(4):51:1-51:27.
19. Schmid S, Suomela J. Exploiting locality in distributed SDN control. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking ACM; 2013; Hong Kong, China:121-126.
20. Knight S, Nguyen HX, Falkner N, Bowden R, Roughan M. The internet topology zoo. *IEEE J Sel Areas Commun*. 2011;29(9):1765-1775.
21. Hock D, Hartmann M, Gebert S, Jarschel M, Zinner T, Tran-Gia P. Pareto-optimal resilient controller placement in sdn-based core networks. In: Proceedings of the 25th International Teletraffic Congress (ITC'13) IEEE; 2013; Shanghai, China:1-9.
22. Ksentini A, Bagaa M, Taleb T, Balasingham I. On using bargaining game for optimal placement of sdn controllers. In: Proceedings of the International Conference on Communications (ICC'16) IEEE; 2016; Kuala Lampur, Malaysia:1-6.
23. De Oliveira RLS, Shinoda AA, Schweitzer CM, Ligia PLR. Using mininet for emulation and prototyping software-defined networks. In: Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on IEEE; 2014; Bogota (Colombia):1-6.
24. Lantz B, Heller B, McKeown N. A network in a laptop: rapid prototyping for software-defined networks. In: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks ACM; 2010; Monterey, CA:19.
25. Hock D, Hartmann M, Gebert S, Jarschel M, Zinner T, Tran-Gia P. Poco-plc: Enabling dynamic Pareto-optimal resilient controller placement in sdn networks. In: *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS'14)*. Toronto, ON, Canada: IEEE; 2014:115-116.
26. Hock D, Gebert S, Hartmanns M, Zinner T, Tran-Gia P. Poco-framework for Pareto-optimal resilient controller placement in SDN-based core networks. In: *Proceedings of the International Conference on Network Operations and Management Symposium (NOMS'14)*. Krakow, Poland: IEEE; 2014:1-2.
27. Sallahi A, St-Hilaire M. Optimal model for the controller placement problem in software defined networks. *IEEE Commun Lett*. 2015;19(1):30-33.
28. IBM ILOG. Cplex optimizer. http://www-01.ibm.com/software/commerce/optimization/ cplex-optimizer; 2012.
29. Afrim Sallahi A, St-Hilaire M. Expansion model for the controller placement problem in software defined networks. *IEEE Commun Lett*. 2017;21(2):274-277.
30. Rath HK, Revoori V, Nadaf SM, Simha A. Optimal controller placement in software defined networks (SDN) using a non-zero-sum game. In: Proceedings of the 15th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'14) IEEE; 2014; Sydney, NSW:1-6.
31. Kyung Y, Hong K, Nguyen TM, Park S, Park J. A load distribution scheme over multiple controllers for scalable sdn. In: Proceedings of the 7thInternational Conference on Ubiquitous and Future Networks (ICUFN'15) IEEE; 2015; Sapporo, Japan:808-810.
32. Jimenez Y, Cervello-Pastor C, Garcia AJ. On the controller placement for designing a distributed sdn control layer. In: *Proceedings of the IFIP Networking Conference*. Trondheim, Norway: IEEE; 2014:1-9.
33. Yao G, Bi J, Li Y, Guo L. On the capacitated controller placement problem in software defined networks. *IEEE Commun Lett*. 2014;18(8):1339-1342.
34. Johnston M, Modiano E. Controller placement for maximum throughput under delayed CSI. In: *Proceedings of the 13th International Conference on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt'15)*. Mumbai, India: IEEE; 2015:521-528.
35. Ruiz-Rivera A, Chin K, Soh S. Greco: an energy aware controller association algorithm for software defined networks. *IEEE Commun Lett*. 2015;19(4):541-544.
36. Hu Y, Luo T, Beaulieu NC, Deng C. The energy-aware controller placement problem in software defined networks. *IEEE Commun Lett*. 2017;21(4):741-744.
37. Ros FJ, Ruiz PM. Five nines of southbound reliability in software-defined networks. In: *Proceedings of the 3rd Workshop on Hot Topics in Software Defined Networking*. Chicago, Illinois, USA: ACM; 2014:31-36.
38. Ros FJ, Ruiz PM. On reliable controller placements in software-defined networks. *Comput Commun*. 2016;77:41-51.
39. Muller LF, Oliveira RR, Luizelli MC, Gaspary LP, Barcellos MP. Survivor: an enhanced controller placement strategy for improving sdn survivability. In: Proceedings of the Conference on Global Communications (GLOBECOM'14) IEEE; 2014; Austin, Tex, USA:1909-1915.
40. Lange S, Gebert S, Zinner T, et al.. Heuristic approaches to the controller placement problem in large scale sdn networks. *IEEE Trans Netw Service Manage*. 2015;12(1):4-17.
41. Jalili A, Ahmadi V, Keshtgari M, Kazemi M. Controller placement in software-defined wan using multi objective genetic algorithm. In: *Proceedings of the 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI'15)*. Tehran, Iran: IEEE; 2015:656-662.
42. Guo S, Yang S, Li Q, Jiang Y. Towards controller placement for robust software-defined networks. In: Proceedings of the 34thInternational Conference on Computing and Communications Conference (IPCCC'15) IEEE; 2015; Nanjing, China:1-8.
43. Su Z, Hamdi M. Mdcp: Measurement-aware distributed controller placement for software defined networks. In: Proceedings of the 21st International Conference on Parallel and Distributed Systems (ICPADS'15) IEEE; 2015; Melbourne, Australia:380-387.

44. Gao C, Wang H, Zhu F, Zhai L, Yi S. A particle swarm optimization algorithm for controller placement problem in software defined network. *Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'15)*. Zhangjiajie, China: Springer; 2015:44-54.
45. KillI BPR, Rao SV. Optimal model for failure foresight capacitated controller placement in software-defined networks. *IEEE Commun Lett*. 2016;20(6):1108-1111.
46. Zhong Q, Wang Y, Li W, Qiu X. A min-cover based controller placement approach to build reliable control network in sdn. In: *Proceedings of the IEEE/IFIP International Conference on Network Operations and Management Symposium (NOMS16)*. Istanbul, Turkey: IEEE; 2016:481-487.
47. Liug J, Liu J, Xie R. Reliability-based controller placement algorithm in software defined networking. *Comput Sci Inf Syst*. 2016;13(2):547-560.
48. Zhang T, Bianco A, Giaccone S, Domenicoand P. The role of inter-controller traffic for placement of distributed SDN controllers. arXiv preprint arXiv:1605.09268; 2016.
49. Perrot N, Reynaud T. Optimal placement of controllers in a resilient sdn architecture. In: Proceedings of the 12th International Conference on Design of Reliable Communication Networks (DRCN'16) IEEE; 2016; Paris:145-151.
50. Killi BPR, Rao SV. Controller placement with planning for failures in software defined networks. In: *Proceedings of the International Conference on Advanced Networks and Telecommunications Systems (ANTS'16)*. Bangalore, India: IEEE; 2016:1-6.
51. Killi BPR, Rao SV. Capacitated next controller placement in software defined networks. *IEEE Trans Netw Serv Manage*. 2017;14(3):514-527.
52. Bari MF, Roy AR, Chowdhury SR, et al.. Dynamic controller provisioning in software defined networks. In: Proceedings of the 9th International Conference on Network and Service Management (CNSM'13) IEEE; 2013; Zürich, Switzerland:18-25.
53. T.I. Md H, UI, Jourjon G, Gramoli V. Revisiting the controller placement problem. In: Proceedings of the 40th International Conference on Local Computer Networks (LCN'15) IEEE; 2015; Clearwater Beach, FL, USA:450-453.
54. Cheng TY, Wang M, Jia X. Qos-guaranteed controller placement in sdn. In: *Proceedings of the International Conference on Global Communications Conference (GLOBECOM'15)*. San Diego, CA, USA: IEEE; 2015:1-6.
55. UL Huque MTI, Si W, Jourjon G, Gramoli V. Large-scale dynamic controller placement. *IEEE Trans Netw Serv Manage*. 2017;14(1):63-76.
56. Yao L, Hong P, Zhang W, Li J, Ni D. Controller placement and flow based dynamic management problem towards sdn. In: *Proceedings of the International Conference on Communication Workshop (ICCW'15)*. London, UK: IEEE; 2015:363-368.
57. Liu S, Wang H, Yi S, Zhu F. Ncpso: a solution of the controller placement problem in software defined networks. *Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP15)*. Zhangjiajie, China: Springer; 2015:213-225.
58. Ishigaki G, Shinomiya N. Controller placement algorithm to alleviate burdens on communication nodes. In: Proceedings of the International Conference on Computing, Networking and Communications (ICNC'16) IEEE; 2016; Kauai, HI, USA:1-5.
59. Aoki A, Shinomiya N. Controller placement problem to enhance performance in multi-domain sdn networks. In: ICN 2016; 2016; Lisbon, Portugal:120.
60. Xiao P, Qu W, Qi H, Li Z, Xu Y. The sdn controller placement problem for wan. In: *Proceedings of the International Conference on Communications in China (ICCC'14)*. Shanghai, China: IEEE; 2014:220-224.
61. Xiao P, Li Z, Guo S, Qi H, Qu W, Yu H. A k self-adaptive sdn controller placement for wide area networks. *Front Inf Technol Electron Eng*. 2016;17:620-633.
62. Liao J, Sun H, Wang J, Qi Q, Li K, Li T. Density cluster based approach for controller placement problem in large-scale software defined networkings. *Comput Networks*. 2017;112:24-35.
63. Wang G, Zhao y, Huang J, Duan Q, Li J. A k-means-based network partition algorithm for controller placement in software defined network. In: *Proceedings of the International Conference on Communications (ICC'16)*. Kuala Lumpur, Malaysia: IEEE; 2016:1-6.
64. Zhao Z, Wu B. Scalable sdn architecture with distributed placement of controllers for wan. *Concurr Comput: Pract Experience*. 2017;29(16):1-9.
65. Oktian YE, Lee S, Lee H, Lam J. Distributed SDN controller system: A survey on design choice. *Comput Networks*. 2017;121:100-111.
66. Cheng G, Chen H, Hu H, Lan J. Dynamic switch migration towards a scalable sdn control plane. *Int J Commun Syst*. 2016;29(9):1482-1499.
67. Basta A, Kellerer W, Hoffmann M, Morper HJ, Hoffmann K. Applying nfv and sdn to lte mobile core gateways, the functions placement problem. In: *Proceedings of the 4*th *Workshop on All Things Cellular: Operations, Applications, & Challenges*. Chicago, Illinois, USA: ACM; 2014:33-38.
68. Moens H, De Turck F. Vnf-p: A model for efficient placement of virtualized network functions. In: *Proceedings of the 10*th *International Conference on Network and Service Management (CNSM'14)*. Rio de Janeiro, Brazil: IEEE; 2014:418-423.
69. Bagaa M, Taleb T, Ksentini A. Service-aware network function placement for efficient traffic handling in carrier cloud. In: Proceedings of the International Conference on Wireless Communications and Networking Conference (WCNC'14) IEEE; 2014; Istanbul, Turkey:2402-2407.
70. Schrijver A. *Theory of Linear and Integer Programming*. Chichester: John Wiley & Sons; 1998.
71. Kim S, Kim S, Parkand Y, Kim S, Lee K. Vnf-eq: dynamic placement of virtual network functions for energy efficiency and qos guarantee in nfv. *Cluster Comput*. 2017;20(3):2107-2117.

72. Sang Y, Ji B, Gupta GR, Du X, Ye L. Provably efficient algorithms for joint placement and allocation of virtual network functions. arXiv preprint arXiv:1702.01154; 2017.

Ashutosh Kumar Singh obtained his B.Tech degree in Information Technology from Uttar Pradesh Technical University Lucknow, India, in 2011 and M.Tech degree in Computer Science and Engineering from Indian Institute of Information Technology and Management Gwalior, India, in 2014. Now, he is currently a PhD student in the Department of CSE, Motilal Nehru National Institute of Technology Allahabad, 211004, India. He is having a membership of IEEE. His research interest includes network optimization and software defined networking.

Shashank Srivastava obtained his PhD degree at Indian Institute of Information Technology Allahabad, India, in 2014. He is currently working as Assistant Professor in the Department of CSE, Motilal Nehru National Institute of Technology Allahabad, 211004, India. He possesses an experience of more than 6 years in the field of teaching and research. He has published various research papers in the area of Network and security. At present he is guiding 5 PhD students in the field of software defined networking (SDN), named data networking (NDN), network flow optimization and security. He is having the Membership of IEEE, ACM, CSI, and CRSI (Cryptographic Research Society of India). His areas of expertise are SDN, NDN, information security, and future Internet technologies.

How to cite this article: Singh AK, Srivastava S. A survey and classification of controller placement problem in SDN. *Int J Network Mgmt*. 2018;e2018. https://doi.org/10.1002/nem.2018