Hindawi
Journal of Electrical and Computer Engineering
Volume 2023, Article ID 6466996, 33 pages
https://doi.org/10.1155/2023/6466996

Review Article

Distributed Controller Placement in Software-Defined Networks with Consistency and Interoperability Problems

Muhammed Nura Yusuf,[1,2] **Kamalrulnizam Bin Abu Bakar**,[1] **Babangida Isyaku**,[1,3] **and Fadhil Mukhlif**[1]

[1]*Faculty of Computing, Universiti Teknologi Malaysia, Johor Bahru 81310, Malaysia*
[2]*Abubakar Tafawa Balewa University, PMB 0284, Yelwa, Bauchi State, Nigeria*
[3]*Faculty of Computing and Information Technology, Sule Lamido University, Kafin Hausa, P.M.B 048, Jigawa State, Nigeria*

Correspondence should be addressed to Muhammed Nura Yusuf; ymnura@atbu.edu.ng

Received 8 August 2022; Revised 22 December 2022; Accepted 30 December 2022; Published 30 January 2023

Academic Editor: Bhargav Appasani

Software-defined networking (SDN) brings an innovative approach to networking by adopting a flow-centric model and removing networking decisions from the data plane to provide them centrally from the control plane. A single centralized controller is used in a traditional SDN design. However, the complexity of modern networks, due to their size and requirements' coarseness, has made using a single controller a source of performance bottlenecks. Similarly, the solution found by using multiple controllers in distributed control planes brings forth the profound issue of interoperability, consistency, and the "controller placement problem" (CPP). It is an NP-hard problem that deals with positioning controllers at optimum locations within the network and mapping with resources at the data plane to meet quality of service (QoS) requirements. Over the years, the problem has received significant attention from the research community, and many solutions have been considered. This paper offers an in-depth review of the proposals by providing an updated evolution of the problem concerning the application environment, design objectives, and cost and controller type. Based on our findings, new research ideas were identified and discussed.

1. Introduction

The support for a network of everything and many on-demand applications and services offered by different multitenant cloud computing service providers had pushed the Internet to evolve into a large and complex infrastructure. A prior study reveals that the number of active Internet devices demanding these services will rise from 26.66 billion in 2019 to 41 billion in 2027 [1]. This growth is expected to exponentially increase to 125 billion by 2030 as an average of 127 devices is connected to the Internet daily [2]. The signs of this have become more visible during the COVID-19 pandemic. These devices can generate heterogeneous traffic in cyberspace to the tune of 800 ZBs. Most traffic emanates from applications with conflicting quality of service (QoS) requirements. Communication is no longer exclusive to client-server but involves machine-to-machine (M2M). Managing these devices generating various traffics with different QoS demands is quite challenging in today's traditional networks.

As such, network design and management have become much more difficult. The integration of network control logic and data forwarding entity is considered the major drawback of traditional IP network architecture [3, 4]. The prohibition on customization in its proprietary glued network devices makes network management, policy design and innovation time-consuming, inflexible, and prone to error [1, 5]. These limitations drive the need for a new architecture that can cope with current demand dynamics. Software-defined networking (SDN) [6] is considered a viable option to the traditional architecture because it breaks the distributed vertical integration of network devices to eliminate all their dependencies. This way, the SDN's main revolution to stimulate this possibility is the detachment of

the intelligence aspect of networking from layer three devices and handing it over to a programmable controller. Therefore, the role of these devices is limited to packet forwarding based on the controller's instructions [6]. The controller's global view and access to network statistics enable it to formulate policies to manage the devices flexibly. In addition, the control plane creates, distributes, and enforces the network policies upon request by switches whenever a new flow with no corresponding rule entry in the switch flow table arrives [7]. The architecture of SDN was initially designed to use a single centralized CP to respond to all the events that emanate from the data plane [8]. Although a single controller is sufficient to manage small-scale networks, it introduces overhead, scalability, and reliability concerns due to high failure tendencies in large-scale networks. As such, multiple controllers have shown better performance on large-scale networks. Therefore, it is imperative to use multiple controllers in a distributed architecture (dmCP) to address these issues. However, deploying the multiple controllers also opens contemporary design and performance challenges at the CP. These issues are associated with interoperability, consistency, and the controller placement problem (CPP) [9].

Several efforts were made to offer solutions to these outlined issues [9–12]. Each has a peculiar application area, parameters, and performance metrics. The method(s) and algorithm(s) proposed to have different degrees of strength and weakness. The state-of-the-art solutions offered are numerous and vary distinctively. This work undertakes a review of these solutions to provide the research community with an insight into these activities although some attempts were made in the past to give this review as reported in [4, 13–21]. However, most of these works focused their studies strictly on CPP alone. Other essential design challenges are not considered. For example, the authors in [4] studied CPP solutions focused on the controller's capacity and traffic conditions. Even at that, they did not consider application environment and cost factors.

The authors in [13] focused on SDN with multicontrollers from a design logic perspective. The paper reviewed the proposed scalability, reliability, and load balance solutions. However, it did not consider issues associated with interoperability between the controllers. Jalili et al. [15] reviewed the control plane deployment mode. The authors compared in-band and out-band solutions. However, optimization objectives, application environment, and controller type were overlooked. The work in [16] focused on CPP design principles and architectures. But like [15], their discussion coverage did not touch on optimization objectives and the deployment environment. This aspect is partially covered in [17–19] as the works categorized the CPP solutions in terms of performance metrics such as latency, reliability, cost, and a combination of many of these in a MOO scenario. However, a controller type and other SDN application areas, such as WSN or IoT, were not considered. The authors restrict their scope to CPP in DCN and WAN. However, wireless scenarios were partially touched in [19].

Similarly, some techniques have applied deep learning and machine learning techniques to integrate SDN with IoT; however, these AI techniques are outside the scope of the present document. On the other hand, Isong et al. [20] considered optimization objectives such as latency, QoS, and resilience. Additionally, they discuss various schemes for solving the CPP and their limitations. They categorized them into optimal and heuristic-based suboptimal solutions. Furthermore, they underlined CPP application areas in next-generation networks such as VANETs, IoT, and telecom. Lastly, the authors of [14, 21] conducted an in-depth analysis of CPP solution strategies applied to optimize the CPP performance metrics such as latency and reliability. However, none of the papers [4, 13–21] considered interoperability and consistency issues among either homogeneous or heterogeneous controllers deployed in a dCP. However, a survey of controllers for scalability, consistency, reliability, and security is provided in [22]. However, it did not consider CPP and interoperability. In other words, none of the papers consider reporting how the multiple controllers placed in the dCP interoperate or synchronize their domain information via the EWi to arrive at a consistent network state for effective service provisioning. These issues and CPP are fundamental in designing a control plane with multiple controllers in SDN.

So, unlike all these other articles, this study looks at how homogeneous and heterogeneous controllers have been used to solve CP design problems in csCP and dmCP. In dmCP, the discussion covers issues on C-C interoperability, information synchronization problems for getting a consistent global network view among the controllers, and CPP. Table 1 provides a comparison summary to bring out these differences. The interoperability and consistency discussion focuses on the EWi and consistency properties, respectively, while the one on CPP solutions is focused on design objectives, load balance application environment, and security. The key contributions of this research are summarised as follows:

(i) A highlight of CP challenges with historical context to trace the source of the problem that evolved into CPP

(ii) Provide the most commonly used CPP mathematical modelling and problem formulation approaches

(iii) Indicate a different solution approach to pursue through DP than the dmCP option

(iv) Provide a critical review of efforts to address dmCP issues related to interoperability, consistency, and CPP solutions proposed in both csCP and dmCP, along with design objectives, the application environment, and security

(v) Finally, the study identifies and discusses potential future research directions

As shown in Figure 1, the rest of the paper is organized as follows: Section 2 briefly overviews SDN and discusses its CP design options and performance issues. Sections 3 and 4 discuss the challenges of interoperability and consistency in dmCP and review the approaches proposed to address them,

Table 1: Comparison of related papers.

Ref	CPP	Controllers interoperability via EWi	Controllers consistency	Study focused, scope, and solution classification
[4]	√	X	X	Controllers' capacity and traffic condition
[13]	√	X	X	The multicontrol plane from a design logic perspective
[14]	√	X	X	
[15]				In-band and out-band solutions
[16]				Design principles and architecture
[17]	√	X	X	Performance metrics, such as latency, reliability, cost, and MOO
[18]	√	X	X	Performance metrics, such as latency, reliability, cost, and MOO
[19]	√	X	X	Classify CPP based on optimization/performance objectives and wireless environment
[20]	√	X	X	Focused on the solution algorithms or approaches used to optimize the well-known CPP performance objectives
[14]	√	X	X	Performance metrics
[21]	√	X	X	Taxonomy of CPP optimization
[22]		X	√	Scalability, consistency, reliability, and security
[23]	X	X	√	The paper focused on works of ensuring consistency at the DP device forwarding state only. It did not cover works focused on CP with multiple controllers. Where the controllers state whether consistent or not at the time of installing the rules entries in the DP switch flow table influence network behaviour
Current document	√	√	√	Controller placement (for resilience, load balancing, application environment, and security), heterogeneous controllers interoperability, and consistency problems

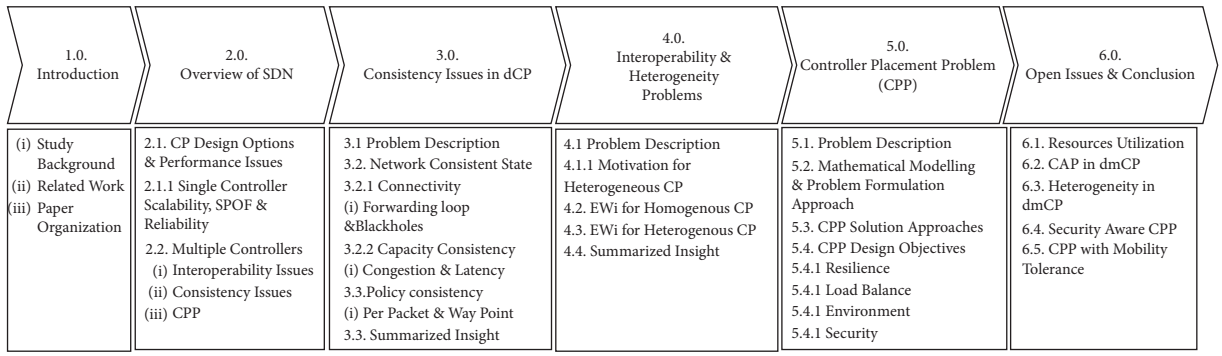

Figure 1: Organization of the review.

respectively. Section 5 presents a review of the state-of-the-art solutions to CPP. Finally, the conclusion and future research direction are provided in Section 6.

2. Overview of SDN

SDN comprises three (3) planes with two interfaces to manage the communication between them, as shown in Figure 2: the application plane (AP), the control plane (CP), and the data plane (DP). The communication between AP and CP is managed through the northbound (NB) interface. The APs are a set of network applications such as network virtualization, firewalls, intrusion detection systems (IDSs), routing, QoS, and mobility management. These applications are translated into high-level networking policies and exposed to CP. The CP is considered the network's brain that manipulates the network forwarding entities based on the design of network policies. It is responsible for routing

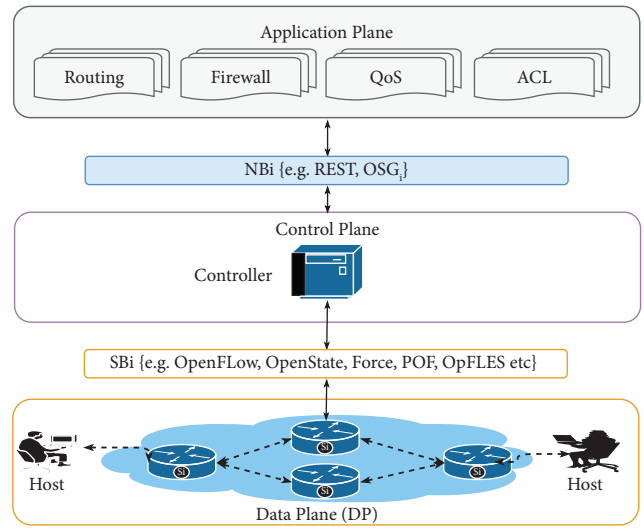

Figure 2: SDN architecture.

computation, network monitoring, balancing the load on the network, enforcing security policies, and many more. The DP is released from all control functions and focuses on forwarding traffic based on the decision made by the control plane. The CP manipulates the network's behaviour by installing flow rules in the DP flow table. It is a logical data structure that stores the flow entries for each corresponding traffic flow. The communication between the CP and DP is managed through the southbound interface. Protocol Oblivious Forwarding (POF) [1] and Open vSwitch Database (OVSDB) [6] are among the early southbound management protocol used in SDN. However, OpenFlow is considered the most popular southbound standard. Over the last decade, SDN has attracted interest from both academia and industry in terms of research and deployment. Many institutions widely apply it in data centre networks (DCN) and wide area networks (WAN). It is also observed to be influencing the design of IoT [24] applications such as VANET, WBAN [25], and next-generation technologies such as 5 G [26].

2.1. SDN Control Plane Design Options and Performance Issues. The CP of SDN can be designed with a single controller (csCP) or multiple controllers (dmCP). While csCP can meet performance requirements in small networks, it struggles to scale to large and dynamic network scenarios due to high control message processing overhead. It also exhibits reliability concerns because of the single failure point (SPOF) because failure tendencies are higher when the network is large. Figure 3 provides a visual representation of performance issues in each design option. The CP uses link layer discovery protocol (LLDP) to discover the DP devices. After the discovery, it is responsible for ensuring real-time fine-grained maintenance of its state. For that, it monitors the topology at regular intervals to collect network statistics for the operation of network applications at AP. Some of the activities it monitors are traffic arrival patterns, traffic types, or topology changes due to events such as failures, which have increased lately [27]. This enables it to recalculate new instructions and install them on the switch's flow table at DP upon the occurrence of any of these events. It centrally performs these functions reactively or proactively.

2.1.1. Centralize Single Control Plane: Performance Issues. As the name implies, in csCP, a lone controller is configured to control the entire network. The single controller centrally manages all the devices at the DP (see Figure 4). Although it can satisfy performance obligations if the network is of average size, it easily suffers performance degradation if the network begins to grow or span over a wide geographical area. Table 2 provides the summarised features of some of these controllers.

2.1.2. Scalability and Overhead. One of the performance issues suffered by a CP designed with a csCP is its inability to handle a network with DP devices extending to a geographically large area or rapidly generating a high number of events. The network's diameter influences the flow setup time in SDN; the time tends to be higher when the switch is further away from the controller. Furthermore, a network can grow so that numerous flow setup requests from many switches become a source of performance bottlenecks. This is because the number of flow setup requests is directly proportional to the number of switches, implying that the overall cost to configure a flow route for n-switches concerning network load is about $94 + 144\,N$. The authors in [28] report that a large network can have switches that can generate up to 10 million flow requests per second. This is beyond the capacity of a single controller, as some controllers can only accommodate 6000 flow requests per second [13].

On the other hand, a DCN might have a dynamic environment where high-volume network events are generated rapidly within a short period. In such a situation, the csCP suffers from communication and processing overhead that can prolong response time, causing a serious delay that can hurt some time-constrained applications. This is likely to happen because the controller may not have adequate CPU, memory, and bandwidth capacity to process and respond to this many DP events [6].

2.1.3. Reliability: Single Point of Failure. A csCP is vulnerable to a single point of failure (SPOF) [29] because if the single controller is down, all the switches under it will not have another controller to fall back on. This is crucial, as it will affect service availability and security if the network is compromised under any attack, like a DoS attack by flooding the network with numerous fake flow setup requests. Furthermore, the singularity of the CP makes it more vulnerable to attacks such as spectrum sensing data falsification (SSDF), also known as the Byzantine attack. And defence against such an attack on an SDN controller is tough. If successful, the adversaries will acquire full control of all network devices and behave arbitrarily to disrupt the network. The only known defence against such a threat is a $3f + 1$ switch-to-controller mapping.

There are two approaches to mitigating these challenges. One of them is a DP modification approach in which some levels of decision-making are relaxed and allowed to be taken by the forwarding devices. The other is to redesign the CP with multiple controllers. Using the former approach, DevoFlow [30] proposed decreasing the intercommunication rate between the CP and DP by implementing a wildcard rules mechanism at switches. The mechanism empowers switches to be able to make some local routing decisions involving matching mice flows. This frees up the controller to focus solely on elephant flows. This way, significant overhead is reduced while in DIFANE [31], a distributed DP framework for handling all data packets is proposed. In this architecture, if traffic flows that do not resemble a precached rule arrive at an ingress switch, the ingress switch is instructed to re-encapsulate the flows in packets and redirect them to a designated switch with route forwarding determination authority. This is done based on rules for partitioning information. The authority switch

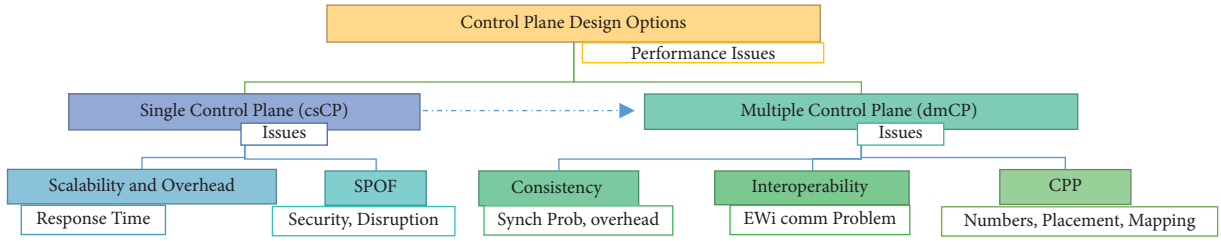

Figure 3: Taxonomy of SDN control plane performance issues.

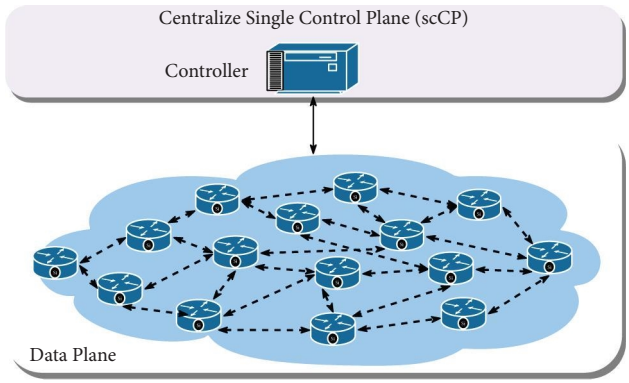

Figure 4: Single control plane architecture.

Table 2: Comparison of controllers.

Controller	Organization	Language	Issue			Multi threading	Capacity	Flow request	Interface
			Consistency	Reliability	Scalability				
NOX	Nicira	C++	High	Low	Low	Yes	Small	30 k/s	X
POX	Nicira	Python	High	Low	Low	X	Small	X	X
Maestro	Rice uni	X	X	X	X	X	Small	300–600 k/s	X
Ryu	NTT lab	Python	X	X	X	X	Small	6 k/s	OpenFlow
Floodlight	Big switch	Java	High	Low	Low	X	Small	250 k	X
Beacon	Stanford	Java	X	X	X	Yes	Small	12.8 m/s	OpenFlow

must deal with all packets in the data plane and update the ingress switch to cache the decided rules locally. The approach employs a link-state routing technique that detects changes in network topology without the controller's involvement to curtail its traffic overhead. However, both techniques violate the SDN basic principle of freeing DP elements from any function but packet forwarding.

Moreover, they may require a complex modification or replacement of the SBi API, such as OpenFlow compliance DP elements, which may increase the design cost. Alternatively, a technique in [32] uses a partitioning algorithm to proactively generate wildcard rules and install them in the switches to handle mice flow free of the controller. The technique performs the "action" of rewriting the server's IP address for packet forwarding to the egress port. On the contrary, the approach in [33] proposed a technique that modified the packet handling process such that only the packet that arrives first is sent to the CP, while later packets are handled locally. This is achieved via the blackhole mechanism. In a similar approach, the authors of [34] proposed a technique that reduces CP overhead via packet-in filtering. It screens packets with duplicate information and drops them. However, the techniques incur high packet loss due to the architecture of the black hole mechanism and filtering techniques, respectively, while the authors of [35] proposed a scheme that reduces the number of control messages between CP and DP using an out-of-band controller to avoid hybrid architecture.

Conversely, Isyaku [36] proposed a technique that reduced CP overhead using flow timeouts and eviction mechanisms. The authors consider traffic characteristics to select flows to evict from the flow table whose reinstallation would cause the least overhead on the CP. Similarly, to improve csCP scalability, ethane [37] and NOX [38] enhance enterprise networks by allowing administrators to define policies such that mismatched requests pass through the controller to have centralized control. However, both approaches suffer from SPOF and support a relatively small network. Other approaches [39, 40] adopt parallelism-based optimization using a multicore system and multithreading to reduce flow setup latency in CP. On the other hand, the authors of [41, 42] embraced routing scheme based-optimization and entry aggregation with early match using the hidden Markov model to scale and reduce the number of

events processed by the CP. The approaches aim to optimize this process in terms of flowable.

2.2. Control Plane with Multiple Controllers. A CP with multiple controllers is designed to solve the csCP limitations. To achieve that, the csCP is modified to deploy multiple controllers in a distributed architecture to manage the network. Figure 5 depicts an example of this design option. The dmCP uses a load-sharing mechanism to allocate the DP switches among different controller instances as appropriate. The controllers communicate via east-to-west interface (EWi) to synchronise their information for global network knowledge. In addition, the interface provides a channel to coordinate activities such as data transmission, leader selection, failover, and load balancing among controllers. Architecturally, the mCP with multiple controllers is designed either as a logically centralized CP or a logically distributed CP.

The multiple controllers work together to perform the functions of a single controller in a logically centralised CP. This is accomplished by constantly synchronising their network state and policies to provide a consistent network view. However, intensive state synchronisation among controllers to maintain a logically central CP can result in significant bandwidth consumption and high latency in large networks whereas in logically distributed CP, each controller only has a view of the domain for which it is responsible; it makes decisions for its local domain alone and only dispenses the information that is needed to other controllers. This is as opposed to logically centralising designs where each controller must have a global view of the entire network to take decisions. Consensus algorithms are applied in most distributed control plane designs to achieve eventual correctness and consistency [43]. The consistency level achieved by these designs may be strong or weak. Strong consistency requires that the state of each controller instance be replicated and transmitted to all controllers through consensus. This implies that an appropriate and consistent network state is only achieved through consensus. The procedure introduces overhead and delays, limiting responsiveness, and potentially resulting in suboptimal performance. While the eventual consistency model omits consensus and assures at least one delivery invariant, the approach only integrates information as it becomes easily available and reconciles updates when each domain knows them. This supports faster reaction with the ability to handle higher update rates, but at the cost of a temporarily inconsistent network view. Thus, it may cause inappropriate network behaviours.

ONIX [44] is an example of these CP architectures with distributed controller instances deployed on one or more physical machines. The control architecture of ONIX maintains a global network view within a network information base (NIB) data structure with two unique update and distribution mechanisms. ONIS guarantees consistency of network state using distributed locking and Paxos consensus algorithms. It also incorporates replication and transactional database modes to ensure that consistency attend is reliable. In addition, it contains a distributed hash

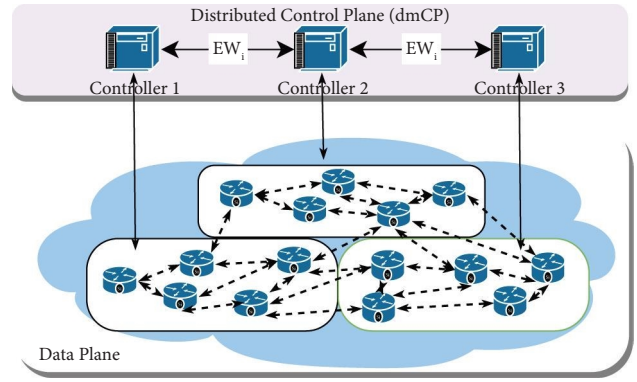

Figure 5: Multiple control plane architecture.

table (DHT) mode that provides an extensive API to verify the consistency. If you need a solution with high availability and your network experiences frequent events, Onix is a great option. However, despite all these consistency checks, it lacks confidentiality and integrity mechanisms to ensure secure state exchange among the controller instances.

Nevertheless, ONOS [45] and ODL [46] employed stringent access control techniques and security services to prevent repudiation and elevation of privilege risks if security is crucial to you. ONOS [45], distributed control architecture, operates different instances of floodlight controllers on multiple servers, with each server responsible for a subset of OpenFlow switches. The controllers broadcast network events using a publish/subscribe method, and intercontroller communication is handled via various routes. ONOS leverages Titan's transactional semantics on top of Cassandra's consistent data store to ensure the consistency and integrity of the network state. In addition, the secure mode (SM) of ONOS provides protected access and granular control over internal data structures and libraries.

OpenDaylight (ODL) [46] is another controller with distributed architecture designed through clustering multiple controllers. ODL can keep a centralized, logical network view using the Akka framework and the RAFT consensus algorithm. The consensus algorithm is incorporated to enable the clustered ODLs to achieve network consistency. The algorithm randomly selects one of the cluster members to serve as the leader and then transmits all the most recent data changes to that leader for update processing. It is an open-source controller that can accommodate various specialized security modules such as secure network bootstrapping infrastructure (SNBI), AAA service, and Defense4All. As a direct result, ODL can preserve the integrity of the data, as such; it is recognized as one of the most secure dCPs. These features facilitate SDN integration with conventional network architecture.

Similarly, like ODL [46], DISCO [47] is another horizontally distributed CP architecture. But unlike ODL, it has limited security mechanisms because of an inherent Floodlight controller's vulnerability. DISCO employs an advanced message queuing protocol (AMQP) to design an expandable dCP suitable for heterogeneous WASDN that addresses concurrent control strategy inconsistencies in

multicontroller architecture. A single Floodlight instance of DISCO is assigned to an autonomous domain. The AMQP helps it via an EWi to transmit information to other controllers' instances using publish-and-subscribe mode. However, even though DISCO is suitable for a large network under different administrative control like the Internet, scalability and SPOF concerns still exist because of its one controller instance per network domain strategy. Katta et al. [48] proposed Ravana as an alternative to DISCO for fault tolerance at both DP and CP. Ravana is a tripod-phase replication procedure to preserve consistency of (1) DP switches run time, (2) control interface, and (3) controller instance runtime in a logically centralized CP architecture in a master-slave design. Instead, of merely keeping the controller state consistent in one phase, it considers the data plane by incorporating a mechanism to guarantee switches' state consistency. Ravana demands strong consistency guarantees when processing the failure events to preserve one exact semantic. Because the method is based on the Ryu controller platform, it is susceptible to the same security flaws as the Ryu system itself [49]. Another alternate fault-tolerant dCP with an additional transparency feature in controller instance capacity is proposed in [50] as IRIS-HiSA. The architecture of IRIS-HiSA comprises an assembled bunch of controller instances organized in a physically distributed manner, with each instance having access to network state information of the global topology. Controller instances are activated by its session management module when there is a failure or overload incident, and switches are assigned to it as per its residual capacity. All the controller instances shared their domain information in a publish-subscribe procedure with consistency in all the controller's network knowledge being pursued using the Hazelcast consensus algorithm.

Instead of using traditional topological partitioning for differential QoS provisioning, Hydra [51] is an alternativedistributed CP architecture that divides a computer network according to the functionality and role of network control applications. So that network applications are configured on different distinct controllers. Hydra uses the Paxos consensus algorithm for fault tolerance and consistency. However, due to the functional slicing, communication between various applications across other partitions may encounter high latency. Conversely, Elasticon [52] is built with a load measurement and an adaptation module to select load adjustment via switch migration across controller instances in the event of topology changes, making it ideal for adaptive load balancing in dCP.

2.3. Distributed Control Plane Challenges and Design Issues. Architecturally, the CP with multiple controllers is designed either as a logically centralized or a logically distributed CP. However, as summarised in Table 3, both designs face challenges such as consistency, interoperability, and the controller placement problem (CPP) (see Figure 6). Two of these challenges are about ensuring that the behaviour of the multiple controllers matches what is offered by a single controller. Indeed, this requires coordination to ensure consistency and interoperability among the controllers. However, ensuring this is a big challenge in SDN [43].

Meanwhile, consensus algorithms such as Paxos and RAFT are being relied upon by many dCP to achieve a consistent network state. However, the consensus algorithms relied upon are observed to be theoretically unsuitable and practically ineffective because they inhibit availability and incur extra latency, primarily when controllers are distributed across a WAN [43]. Furthermore, interoperability among the multiple controllers became even more difficult due to the controllers' heterogeneity. The other problems CP faces with multiple controllers is identifying the number of required controllers and their placement position in the network topology. This problem is called controller placement problem (CPP) [9].

2.3.1. Controllers Consistency Problem. Controller consistency in mCP refers to its ability to always have a stable, up-to-date global knowledge of all network states and policies. The update always aims to preserve one or more of the following aspects: i.e., (1) network state like connectivity and capacity, and (2) policy. Inconsistency in CP connectivity can cause traffic blackholes (i.e., dead-end paths), isolation or forwarding loops problems [54] (i.e., a situation where traffic keeps going back and forth without proceeding to destinations). The latter can deplete switch buffers to the extent of impairing availability in the network.

In contrast, inconsistency in capacity can cause transient congestion and latency problems during updates [55]. And lastly, consistency in policy ensures the requirements desired by the operator, like path selection such as in Isyaku et al. [56], ACL, and firewall are adhered to. Inconsistency in policy updates might have security implications and QoS violations. To avoid all these, the controllers must constantly share their state, policies, and version info. However, achieving a consistent global knowledge of an entire network by all controllers is one of the most challenging tasks in dmCP, unlike in csCP instances where global networks are easily acquired during network policy updates. The consistency in mCP covers three aspects of SDN devices operations: (1) controllers' uniform state consistency, (2) consistency in switches' flow tables rules, and (3) controllers' version update consistency [22]. Suppose the rules' update operation happens at the time of the version update. In that case, there might be an ongoing transfer of some packets of some flows, so at that time, they may be forwarded by a mix of old and new rules, leading to inconsistency in packet forwarding decisions because status updates might arrive late, which will cause jitter. Likewise, controller overhead may appear due to the high frequency of synchronization attempts. In contrast, state desynchronization at intervals between two syncs could bring connectivity consistency problems like forwarding loops and black holes [57].

2.3.2. Controllers Interoperability Problem. Similarly, it is essential to note that information synchronization for global view among the controllers is only made possible by an effective EWi. However, the lack of unified EWi makes interoperability between different controllers a prominent

Table 3: Distributed control plane summary table.

Ref	Design		Controllers instance	Consistency algorithm	Strength	Weakness	Challenges		
	LC	LD					Consistency	Heterogeneity	CPP
ONOS [45]		√	Floodlight	Raft and cassandra	SBi and NBi use TLS and HTTS. It includes IDS and library access authorization	—	Weak	Y	Y
ONIX [44]	√		ONIX	Paxos	Adaptable to network changes and good for high-availability networks	Insufficient protection of privacy and confidentiality	Strong	Y	Y
ODL [46]	√		OpenDayligh	Akka and raft	Arguably the most secured		Strong	Y	Y
DISCO [47]		√	Floodlight	Interdomain agents	Suitable for the heterogeneous network under different administrative control like the Internet	Scalability and reliability persist. It is also insecure	Strong	Y	Y
Ravana [48]	√		RYU	Two-phase replication protocol	Maintain DP and CP consistency	Vulnerable to spoofing, tempering, DoS, and repudiation attacks	Strong	Y	Y
IRIS-HiSA [50]			N/A	Hazelcast	—	—	NA	Y	Y
Hydra [51]		√	Floodlight	Paxos	Differential QoS provisioning	High comm latency between different applications	NA	Y	Y
Elasticon [52]			N/A	Hazelcast	Good load-balancing strategy	Switch migration overhead. It also lacks security measures	Strong	Y	Y
[53]			ODL	Fast paxos			Strong	Y	Y

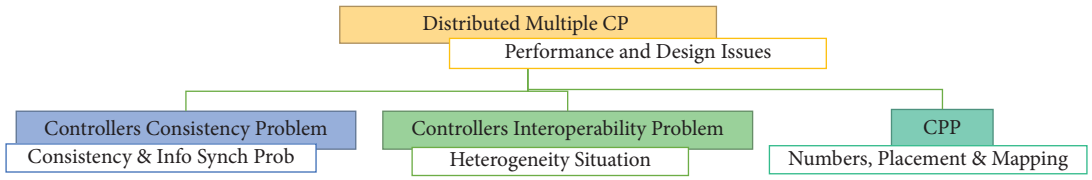

Figure 6: Distributed control plane design challenges.

problem in SDN. Hence, it is compounded even further when dealing with heterogeneous controllers. The motivation for designing CP with heterogeneous controllers can be seen from a security perspective to avoid the common mode fault of homogeneous controllers [10]. Thus, providing an abstraction that can support the integration and interoperation between CP with heterogeneous controllers from different vendors as did to DP elements by OpenFlow is a big challenge. Moreover, the variations in data models among other controllers hampered this collaboration [22].

2.3.3. Controllers Placement Problem. Another popular issue confronting CP with multiple controllers is a controller placement problem (CPP) [9]. To design a CP for any given network topology, the CPP is formulated to find answers to questions like how many controllers are needed for the network. What are their optimum location in the topology? And how can they be mapped with the DP devices to satisfy the QoS requirements of the network? The issues had received substantial research attention as knowing the number of controllers to use and where to put them is a prerequisite to meeting QoS metrics and fault tolerance. For instance, knowing answers to these questions is necessary to design efficient dmCP for SDWAN and DCN where latency and reliability are some of the most important performance requirements.

3. Consistency Problem in Distributed Control Plane

3.1. Problem Description. Network events such as the arrival of a new flow, link, or node failure are considered. When contacted, a controller will respond by designing a new rule and multicast it as a packet-out message to all switches involved to update their respective flow tables

asynchronously. However, there will be delays before all the switches affect the update because of the asynchronism. Thus, until the last switch affects the update, flows might be controlled by a combination of old and new rules. In this circumstance, different forms of invariant infringement, such as a forwarding loop, firewall bypass, isolation, or black hole, might be experienced in the network. This is called the "consistent update problem." The challenge, therefore, is to update the DP devices in a manner where no single invariant is infringed. The solution to the problem has been approached in three ways. The first method meticulously arranges the switch update sequence to guarantee that no invariant violations occur [58]. The second method involves a 2-phase commit, i.e., marking (tagging) any incoming packets at the ingress switch with a unique identifier before processing them according to whether they belong to a new or existing flow entry [11, 59, 60]. The third strategy has switches almost simultaneously switch over to the new flow entries, which is made possible by using switches with clocks that perfectly sync with one another [61]. In dmCP consistency, an issue arises because even though the multiple controllers are partitioned from each other, the DP devices they control remain connected [62].

In this situation, as explained in 2.3.1, the controllers synchronise their domain info with one another to update their states, versions, and rules with the help of consensus algorithms. The update always aimed to preserve one or more consistency properties, i.e., (1) connectivity, (2) capacity, or (3) policy. For instance, inconsistency in CP connectivity can cause traffic blackholes (i.e., dead-end paths) or forwarding loops problems (traffic keeps going back and forth without proceeding to destinations). The latter can deplete switch buffers to impair availability and connectivity.

Another instance is that inconsistency in capacity can cause transient congestion and latency problems during network state or policy updates. And lastly, in addition to connectivity and capacity, inconsistency in policy updates might have security implications or QoS violation. For example, some networks might have a policy that will enforce some flows to traverse through a firewall or some flows to be routed via certain subpath because of their QoS requirements. Therefore, such kind of policies must necessarily be updated consistently throughout the network.

Meanwhile, consensus algorithms such as Paxos and RAFT are being relied upon by many dCP to achieve a consistent network state. However, the algorithms are observed to be theoretically unsuitable and practically ineffective because they inhibit availability and incur extra latency, especially when controllers are distributed across a WAN [43]. For instance, Paxos [7] involves a four-delay state updating method: prepare-request, promise, accept-request, and accept-response. In large-scale SDN, flow requests can reach up to 11 million per second [63]. Every request made may require a state change and consensus run; this can hinder quick network reconfigurations and generate a bottleneck on CP and DP.

Similarly, this might not be suitable for some use cases, like in 5 G technology, where you have a connection setup requirement of <15–30 ms for low latency applications [63]. Although RAFT has optimised Paxos [9], the core notion remains. Besides, they are algorithmically complex and hard to implement as they are afflicted with errors. Additionally, RAFT [43] is susceptible to Byzantine failures [64].

Furthermore, each aspect's consistency can be either weak or strong. Strong consistency requires that all controllers' instances states can only be replicated and propagated through mutual consensus. After any state update, the leader facilitates the conflict-free distribution of state updates to all. In contrast, the eventual (weak) consistency model omits consensus and guarantees at least one delivery invariant. The selection of the consistency model utilized by the replication process impacts the incurred synchronization overhead in load, response times, availability, and the processing order of commits.

This study classifies the proposed solutions to preserve consistency among controllers in dmCP according to their targeted objectives. We considered two consistency properties that reside inside the controllers to do the classification: i.e., (1) the network state update to preserve "connectivity" and "capacity" properties and (2) the "policy" update, which is a concern with network operation procedure such as securities check and service differentiation.

3.2. Network State (Connectivity and Capacity). In response to network-changing events, controllers in dCP update their knowledge to preserve and prevent (1) connection disruption and (2) network capacity violations. An inconsistent connection can lead to issues such as black holes, isolation, and forwarding loops problems that may deplete switch buffers and limit network availability while congestion and update delay can be caused by inconsistent controllers' information on devices' residual capacity [47]. Therefore, the following section reviews the research efforts to update networks to maintain these consistency properties.

3.2.1. Controllers' Consistency for Connectivity Preservation. Mahajan and Wattenhofer [58] proposed update schedules based on combinatorial dependencies that do not require any packet tagging [60]. This will allow some updated connections to become available soon. The authors also provide an initial algorithm that, given the current state, swiftly updates routes in a loop-free way, with the controller greedily trying to update as many nodes as possible. However, the greedy operation of the algorithm might lock it up in local optimal. In another approach, Nguyen et al. [65] designed EZ-Segway, as a distributed method for updating network state consistently and swiftly while avoiding anomalies such as loops, blackholes, and congestion. In EZ-Segway, the controllers precompute any information necessary by the switches before the update. The data are sent to the switches to implement the change using a combination of direct message passing and partial knowledge. This removes communication and calculation bottlenecks at the CP. As such, it enhances the performance of update

processes. However, using partial data to affect the update may not guarantee loop and blackhole freedom in the system.

In the rule replacement approach, Forester et al. [66] suggest that a straightforward way to ensure blackhole freedom is for a mechanism to install new or default matching rules with higher priority and delete old ones to avoid blackhole problems. However, implementing this might induce a forwarding loop problem. Besides, the techniques are also restrained by TCAM limitation constraints. In another work, Canini et al. [67] proposed a solution based on principles of self-stabilization within a bounded communication delay. Known as "Renaissance," the method manages multiple controllers' connection and communication outages without compromising performance. This solution is an improvement over their previously proposed FixTag [68]. A wait-free technique that tackles rules updates consistency problem via a transactional interface. Zhou et al. [69] propose a consistency layer to actively and passively snapshot the cross-domain control states to reduce the complexities of service realizations to ensure a consistent link state for max throughput and min latency. The technique adopts both reactive and passive snapshots of a cross-domain layer in WAN to control a consistent state of the network controllers.

In a different approach, Mizrahi and Moses [70] propose a switches' clock synchronising technique to update the network in real time. The method preserves loop freedom and communication loss with perfect clock synchronisation and switches execution behaviour. Similarly, in [71], Mizrahi and Moses modify precision time protocol (PTP) to achieve microsecond update accuracy in SDN. This is because the normal network time protocol (NTP) lacks appropriate synchronisation behaviour for SDN. This leads to an increase in the number of messages necessary for time synchronisation across the whole network. However, when two unsplittable flows need to be swapped in the network with no alternative paths available, the synchronized updates are considered optimal, and the new flow paths can minimize the induced congestion. Furthermore, despite these benefits, clock synchronisation approaches do not prevent random fluctuations in command execution time on switches. This prompted the development of prediction-based scheduling techniques [72].

3.2.2. Controllers' Consistency for Capacity Preservation. Panda et al. [62] investigate the extent to which CAP theorem trade-offs apply to SDN with dCP. They examine network consistency properties that require tenant isolation and middlebox traversal for some traffic and prove that they cannot be all enforced without losing availability. The authors posit that linearizability is typically unneeded for ensuring effective enforcement of most network consistency properties since the monitored policies typically have simple correctness criteria. For this reason, in [43], Panda et al. designed a simple coordination layer SCL that avoids consensus algorithms like Paxos or Raft to achieve consistency in dCP. SCL broadcasts all CP communication to avoid the need for bootstrapping on the controllers. The approach has simplicity and eventual correctness, with a higher response time advantage. However, it might give the DP ovS conflicting instructions because of the response time. Hence, an implementation may consume higher bandwidth and replacement of the topology discovery module of the CP with the log provided in the controller proxy-SCL.

Also inspired by [62], Sakic et al. [63] and Bannour et al. [73] develop adaptive, eventual consistent and self-adaptive multilevel consistency models to solve blockage possibilities in a highly consistent dCP, respectively. These models are intended to facilitate developer implementation of numerous application-specific consistency models. In [63], Sakic et al. integrate eventual consistency models with a novel cost-based approach, where rigorous synchronization is used for crucial activities involving many network resources. At the same time, less critical changes are intermittently transmitted across cluster nodes. However, these techniques will suffer traffic separation overhead. Another study in [74] proposed a fast and generic system that imposes customizable network consistency during updates and information synchronization. The authors design a customizable consistency generator (CCG) to act as a shim layer between CP and DP, intercepting and scheduling real-time updates issued by the controllers. The authors use Mininet to emulate a fat-tree network with the shortest path routing and a load-balancing application in a NOX. However, CCG might require architecture modification and incur customization overhead. Luo et al. [75] also argue that during the update process, in-transit packets might misuse wrong versions of rules, and "hot" links could be burdened due to the unplanned update order. Even though earlier proposals like the 2-phase commit and CCG have provided generic and customizable solutions to address the problem of misusing rules, yet no flexible approach exists to avoid transient congestion on hot links with varied user requirements such as update deadlines, transient throughput, or loss. Motivated by this, they thus proposed a customizable update planner (CUP), to seek a solution to the problem. But just like CCG, CUP too might incur customization overhead Aslan and Matrawy published in [76] an adaptation technique that chooses feasible values for the consistency level indicators that satisfy a specific application indicator. The authors use K-means online clustering to determine an appropriate mapping between consistency level and application indicator. In a similar approach, Zhang et al. [77] also propose the current network state adaptive synchronization strategy of controller information in dCP. The authors formulate an optimization problem concerning overhead and availability constraints. The controller's roles are classified as leader, acceptor, and learner. However, the technique did not consider fair load balancing among the controllers, which if considered there is a possibility of reducing the synchronization overhead further down.

The 2-phase commits [60] process used to ensure per-packet consistency can also be used to ensure per-flow consistency to avoid congestion. While this approach reduces congestion, it is insufficient for full bandwidth guarantees. Therefore, as demonstrated by Mizrahi and

Moses [70], strategies that go beyond simple flow switching are needed to ensure that the update does not impact the network's capacity constraints. Therefore, Hong et al. in [78] established the standard model for capacitated updates as part of their SWAN. It offers an LP formulation for splittable flows that, if satisfied, results in a migration plan with x updates. It provides the foundation for Liu et al.'s zUpdate [79], a technique for updating DCN with zero losses. The authors also demonstrate that consistent migration is doable with $\lceil 1/s \rceil - 1$ updates if all flow links have free capacity slacks. However, for smooth migration, it is best to eliminate some of the network's background traffic or temporarily limit its throughput if it contains noncritical traffic. Brandt et al. [80] additionally offered a method to distribute flows over the network. They provided an algorithm that attempts to produce slack capacity on all links, and they demonstrated that splittable flow migration can always be solved in polynomial time. The core concept is to repeatedly divide flows into new pathways until sufficient capacity is available to use the algorithm of [78]. However, these techniques will suffer from flow reassembling overhead at the destination.

In a different approach, Jin et al. [81] consider different update timings of network switches to avoid congestion. They construct a dependency graph of various updates and send them out in a greedy fashion once the requisite conditions are met. Flows are then slowed down when the greedy way of going through the dependency graph leads to a standstill. In a similar approach, Zheng et al. [82, 83] examine the use of consistent timed updates to reduce congestion in the context of flow migration and latency by employing switch clock synchronization. The authors presented chronus, a mechanism that allows for scheduling individual node updates in SDNs at precise periods. It is discovered that the strategy reduces transitory congestion and conserves flow table space by reducing rule size by 60 per cent. However, this reduction might be achieved at the cost of some policy inconsistency.

Amiri et al. [84] also explore node-ordering as an alternative to a 2-phase commit. With the flow version numbers removed from the packet header ("marking"), they offer a technique that uniquely identifies flows based on their source and destination. By omitting the 2-phase commit procedures, the process minimizes complexity and overhead. However, every node has an old and new forwarding rule for every flow. The difficulty is determining the proper sequence to apply these updates without causing congestion or forwarding loops.

Another option proposed by Botelho et al. [85, 86] is the use of distributed data storage such that applications on controllers could scarcely be aware of any inconsistency. This way, latency problems would not even arise. However, the high memory requirement for the data store might not allow it to work well with TCAM of openflow switches. In contrast, Levin et al. show in [55] that load-balancers and other distributed network functions can bypass weak (eventual) consistency while still providing adequate performance for production networks. They made their observation by experimenting with two-state distribution trade-offs between staleness and optimality and between app logic complexity and robustness to control state. They suggest similar investigation for other control applications such as routing and security. Thus, Guo et al. [57] build on Levin et al.'s work by reducing synchronization overhead using a load variance (LVS) technique. Unlike the periodic synchronization (PS) technique, LVS incurs less overhead because it gets activated only when the load exceeds a particular threshold. However, you cannot rule conflict in state update distribution because of weak consistency.

3.3. Controllers' Consistency for Policy Preservation. The 2-phase commit approach proposed in [11, 60] by Reitblatt et al. symbolizes the first foundation work for most rule updates' techniques for consistency preservation in SDN. It broadened the scope of consistency properties beyond just network/forwarding to include policy ones. The authors emphasise the per-packet consistency (PPC) criterion, which requires packets to be forwarded only on their old or new paths during an update (never on both). The concept revolves around labelling packets at the ingress switch so that either only all old or all new rules can be applied consistently all over the network but not both. This way, the problem of forwarding loops and inconsistent policy on packets is avoided. Fayazbakhsh et al.'s flawtags use the same principles [87] as Qazi et al. [88] to ensure network-wide policy compliance with middlebox usage in SDN. However, the technique is known to be memory hungry, as it requires additional free memory slots. This is its major drawback, considering how precious and expensive memory is to an OpenFlow switch. Meanwhile, Katta et al. in [59] designed a variant of the original technique to addresses intending to address the limitation using incremental 2-phase commit. The input update is proposed to be broken down into more minor updates that can be executed sequentially without putting much pressure on TCAM.

Another technique is that software transactional networking (STN) was introduced by Canini et al. [89] to resolve concurrency concerns that occur from the concurrent execution of control applications. Every policy update is either fully implemented or does not affect any packet because the STN relies on all-or-nothing principles. Consequently, this can lead to transitional delay. And to specify actual policy composition principles, the authors deemed it necessary to expand pyretic. In another effort, Canini et al. proposed a consistent policy composition (CPC) technique [68] for concurrent network policy updates. The method uses a replicated state machine, which offers a transactional interface to address the issue of conflicting policy updates. The methods consistently match desired network behaviour from the operator's perspective. However, this is only possible if switch ports allow atomic read,modify,and write permission. Undoubtedly, this prerequisite is the weak security spot of the technique. Schiff et al. [90] propose a synchronization framework for policy updates based on atomic transactions, implemented in-band, on the DP devices. The technique achieves consistency in the events of

controller failure via publish/subscribe model. In the event of any network-changing events, the controller will immediately publicize the events to its coresources to update their local information using standard OpenFlow protocol [91]. But the technique has not been integrated with any SDN controller.

Using a carefully calculated rule replacement sequence, Vissicchio et al. [92] investigate how network policies can be enacted on the OpenFlow switches based on per-packet consistency (PPC). They use a greedy algorithm GPIA that updates the flow table of switches in polynomial time to discover a sequence of rule replacements per switch that does not violate PPC. The memory overhead of this method is zero and can be implemented in a heterogeneous network because of tag-free packets as in a 2-phase commit. The authors, nevertheless, hybridize both the 2-phase commit and rule replacement approaches to propose FLIP in [93]. As expected, the 2-phase commit in FLIP tainted its smooth usage in hybrid SDN, and its complexity is not polynomial. But FLIP is powerful as it can handle per higher number of update incidence. In [94], Mcclurg et al. offer another method for preserving arbitrary network policies using rule replacement order. The paper models the update-consistency properties as linear temporal logical formulas, which can be used to generate updates that continue to uphold the original properties automatically.

3.4. Summarised Insights. A distributed control plane updates the states of its controllers to maintain a consistent network view whenever network-changing events, such as the arrival of a new flow, link, or node failure, occur. The update is always aimed at preserving one or more of the following aspects of the network state: (1) connectivity, (2) capacity, and (3) network policy. The consequences of violating either of them cause black holes, forwarding loops, or congestion problems. Different consistency strategies are put forward according to different situations. Table 4 summarises these strategies and the consistency property they optimised. These consistency properties are interdependent, i.e., every property must be preserved. Yet, none of the existing platforms attempts to consider all three properties at once; some only meet one or two.

Furthermore, based on these strategies, preserving connectivity to ensure loop freedom is better understood of the three because even straightforward greedy techniques fare relatively well, as demonstrated in [66]. We also suppose the problem can be minimised through adaptive and hybrid idle-hard timeout allocation and flow eviction mechanisms similar to [36] or with the approach as in [58]. However, the greedy approach might not be suitable to optimize the makespan metric because of the controller to switch interaction rounds might be higher. So getting a logarithmic time algorithm to optimize the makespan might require some deep learning techniques as applied by Poularakis et al. in [95–97] using some AI techniques like deep learning. On the other hand, capacity consistency for congestion freedom is stronger than connectivity consistency for loop freedom [66].

The 2-phase commits [60], dependency graph [81], and rule replacement are the most widely used method to address congestion. Though [80] posits that splittable flow can be solved in polynomial time, the authors of [84] show that omitting it can reduce complexity and overhead. Regarding network policy consistency, the two major techniques are packet tagging to avoid rule mismatch using a 2-phase commit and complete rule replacement. Each approach has its merit and demerit; while the former comes at the cost of switch memory consumption for algorithm simplicity, the latter comes at the expense of high algorithm complexity with the advantage of dealing with the problem heads on. So, considering the TCAM limitation of OvS, the former might not be a good approach [1].

Lastly, another insight is that most of the consistency methods used by these works do not provide any security, synchronization rate or when to synchronise. Thus, none of the techniques factored the network application into play to assess the impact of synchronization rate on their performance. For instance, it has been shown that load balancers can navigate weak consistency and still deliver good performance.

4. Controller Interoperability/ Heterogeneity Problem

4.1. Problem Description. Interoperability across distributed controllers in an SDN architecture requires an EWi, just like the NBi and SBi, for communicating with the AP and DP. In contrast to the broad adoption of standardized SBi with initiatives like OpenFlow, there is no standard EW communication interface between the controller in dCP [22]. Even at that, the gap has not received the requisite attention from the research community to provide the needed interoperability. But lately, there have been some fruitful study attempts [60, 93–105]. And the existing solutions show variation across a range of performance metrics. Some approaches are mainly concerned with achieving state synchronization in networks where the dCP comprises homogeneous controllers and can't coordinate the coexistence of heterogeneous controllers from different platforms [93, 94]. In other words, they have only solved the interoperability issue for a vendor's unique SDN where all controllers are of the same type. Because their EWi is private and the data model is different, they do not interoperate to share the same network. However, this interoperability is crucial for the survival of the modern-day network for the foreseeable future as SDN increasingly proves its advantages. Fortunately, some EWi for dCP SDN networks that use heterogeneous controllers have recently been proposed [60, 95–105]. However, studies on SDN's east-west interface are still preliminary, and there are no established industry standards yet.

4.1.1. Motivation for Heterogeneous Control Plane. Aside from the scalability and reliability arguments for constructing CP with numerous controllers, the differences in processing capacities of each controller provide additional

Table 4: Comparison of related works on dmCP consistency.

Ref	Consistency aspect/properties			QoS metrics optimized	Method/consensus model	Weakness
	Connectivity	Capacity	Policy			
[43]	X	√	√	Consensus/convergence time	Weak consistency with simple coordination layer	Higher conflicting tendency in the distribution of state updates to all nodes
[55]	X	√	X	Load balance	Weak consistency	
[57]	X	X	√	Overhead	Load variance (LVS)	
[58]	√	X	X	Forwarding loop	Greedy, combinatorial dependencies	Correctness issues
[59]	√	X	√	Memory, inconsistency	Incremental 2-phase commit	Increased packet header tagging overhead
[11, 60]	√		√	Violation, inconsistency	2-Phase commit	
[63]	X	√	X	Availability, scalability delay	Adaptive weak consistency using cost-based	Critical traffic separation overhead
[65]	√	X	X	Forwarding loops, congestion, & blackhole	Direct message passing and partial knowledge	Marking update decisions with partial information
[66]	√	X	X	Blackhole freedom	Rule replacement with default matching rule, greedy	TCAM constraint, a forwarding loop
[68]	√	X	√	Conflicting policy updates	2-Phase commit, replicated state machine	TCAM constraints, tagging overhead
[70–72]	√	√	X	Loop freedom, loss, congestion	Clock synchronization	Message overhead
[73]	X	X	X	Availability	Adaptive weak consistency	Application requirement separation overhead
[74]	X	√	X	Load balancing	Customizable consistency generator, shim layer	Architecture modification, customization overhead
[75]	X	√	X	Congestions, deadlines, loss	Customizable update planner	Customization overhead
[76]	X	√		Synergy AP and CP	Adaptive consistency, *K*-means	Incur customization overhead
[77]	√	√	X	Overhead, availability	Adaptive consistency	Adaptability overhead
[78]	X	√	X	Congestion, throughput	LP, splittable flow, flow migration splittable flow	Flow reassembling overhead, flow migration cost
[79]	X	√	X	Lost		
[80]	X	√	X	Congestion, convergence		
[81]	X	√	X	Congestion	Timed update, dependency graph, greedy	Policy consistency might be hurt
[82, 83]	√	√	X	Congestion, latency, memory	Timed update, dependency graph	
[84]	X	√	X	Congestion, complexity &overhead	Node-ordering	Prohibit *e* of 2-phase common thus, there is a high tendency of mismatch rule
[85, 86]	√	√	X	Latency	Distributed data store	High memory requirements
[87]	√	X	√	Security, inconsistency	2-PhaseCommit	TCAM constraint, tagging overhead
[88]	√	X	√			
[89]	X	X	√	Inconsistent update	All-or-nothing principles	Transitional delay
[90]	X	X	√	Failure	Publish/Subscribe model. Atomic transactions	Not integrated with any controller
[92, 93]	X	X	√	Per packet consistency (PPC)	GA: 2-phase commit and rule replacement (RR)	TCAM constraint, tagging overhead
[94]	X	X	√	Policies preservation	LTLF, RR	Transitional delay

justifications for designing it with heterogeneous controllers. For instance, a Ryu [49] could only process 6 K requests per second, a NOX [38] could process about 30 K requests per second, a Floodlight [98] could process about 250 K requests per second, and a Maestro [40] could process about 300 K requests per second. Therefore, controllers are chosen to be included in a CP for effective resource utilization and cost optimization based on their capacity relative to the DP devices they coordinate. In addition, modern networks are extremely complex, demanding the usage of specialist services such as advanced VPN, deep packet inspection, firewalling, and intrusion detection. As this list grows, the necessity for techniques to implement new network regulations increases. While many controller systems may support several services, this is not the case for all of them. It is also quite unlikely that a single controller vendor offers the performance gold standard for all services. Consequently, network operators may be forced to choose between not providing a service and making an expensive, disruptive, or unworkable migration to a different controller platform.

Each controller has unique performance characteristics, even though services can be simply transferred between them. Some controllers may have quick response times while others are better suited for large-scale applications. It may be advantageous to execute numerous services on multiple controllers to align controller performance characteristics with service requirements. Services that passively sample packets in the network, for example, may need to increase to keep up with the volume of network traffic, whereas services that reactively insert flow table entries to route flows require low response times.

Another motivation is security to avoid homogeneous controller common-mode faults. Threats to network security could occur on SDN composed of multiple homogeneous controllers. For it is homogeneous, it indicates that their designs have the same underlying functioning mechanism, meaning that any potential vulnerabilities in one controller would be reflected in all the other identical controllers. Consequently, if attackers could exploit one of these vulnerabilities, they would be able to execute malicious attacks, such as a message leaks or DDoS, on the remaining controllers to bring the whole network down. This might have devastating consequences for the entire network. This phenomenon is called the "homogeneous controller common-mode fault" (HC-CMF) [10]. Consider a scenario of an SDN for which all the multiple controllers are homogeneous (e.g., Floodlight), each of which manages the corresponding SDN subnet. By exploiting CVE-2014-2304, a known flaw in an OpenFlow protocol for the Floodlight 0.9 version, an attacker can easily crash the entire system after realizing it is using Floodlight controllers in all the other subnets. So, using the same vulnerability, the attackers can easily eliminate any number of backup floodlight controllers located on the control plane. However, the network might be insulated from this threat if the controllers deployed are heterogeneous because in practice, different controllers are susceptible to different vulnerabilities. This is partly because the heterogeneous controllers are independent of each other and have no dependencies during runtime. The same vulnerability hardly occurs among controllers written in different programming languages. For instance, NOX [38], Ryu [49], and Floodlight [98] controllers are programmed in C++, Python, and Java, respectively. These make the trigger mechanisms of their vulnerabilities different. Thus, heterogeneous dCP is considered one of the best defences against the "homogeneous controller common-mode fault" (HC-CMF). Because it is almost impossible for an adversary to make heterogeneous CP exhibit abnormal behaviour by exploiting the vulnerability of only one of the controllers' vulnerabilities, this is considered one of the motivations for designing dCP with heterogeneous controllers.

4.2. EWi for Homogeneous Control Plane. In 2012, SDNi [12] was introduced by Huawei as a multidomain SDNS message exchange protocol. The connection between controllers is a mechanism for controllers to synchronise their data. ODL employs SDNi to promote cross-domain communication among its numerous controllers. Benamrane et al. [99] proposed a communication interface for distributed control plane (CIDC) to synchronise the controllers' information via the EWi. The technique makes provision for network managers to customize the controller's function by selecting from the following three communication modes: (1) notification on (events or mode), (2) service (e.g., load balancing, security etc.), and (3) full (both mode and event). The effectiveness of CIDC is tested in simulated network configurations that include the firewall and load balancer services. But CIDC is only used for interoperability between dCP that use Floodlights and ODLs. Adedokun and Adekale [100] assert that they improved the original CIDC system to create mCIDC. Nonetheless, mCIDC's architecture lacks a clear indication of where the modification section should be located.

4.3. EWi for Heterogeneous Control Plane. One of the earliest efforts to facilitate sharing network views between different domains in a multidomain network was initiated by Lin et al. [101, 102]. The work proposed a high-performance mechanism known as east-west bridge (EWBridge) with support for controllers from different platforms enabled by JSON. The method specified which pieces of information should be shared and how. In addition, EWBridge provided a method for protecting network privacy by hiding the underlying hardware behind a virtual network. EWBridge has been utilized in CERNET, Internet2, and CSTN (CSTNET). Another effort came a year later by Dixit et al., in FlowBricks [103]. FlowBricks is also a framework for composing heterogeneous controllers on the same CP. The framework integrates the services implemented on the heterogeneous controllers into the same network traffic. This is motivated by the fact that no single controller has or can provide the best-in-class implementation of all desired network services such as VPN and firewall. FlowBricks is implemented as a module in Floodlight. Yu et al. [104] also introduced a Zebra architecture for consideration. Zebra is divided into the HCM module for managing heterogeneous controllers and the DRM module for managing domain relationships. Zebra had recorded a remarkable improvement in request completion time and scalability-related metrics like throughput and latency. The proposal's authors were optimistic that it would stimulate the interest of relevant stakeholders, leading to further, fruitful research.

Meanwhile, Qi and Li in [105] point out that dealing with heterogeneous controllers in a large network is very challenging to coordinate due to ununified APIs. Thus, we propose a controller management system that generates a global network view with unified APIs for both AP and DP in a way that shields the heterogeneity of the controllers. The method comprises four (4) modules: heterogeneous controller management (HCM), domain relationships management (DRM), a database, and front-end modules. They used a fat-tree-based DCN topology split into three domains: deployed Ryu, Floodlight, and Pox controllers, respectively.

Another recent effort is Yu's et al. [106] proposal of WECAN. WECAN is designed to act as an EWi for controllers in SDN dCP. It consists of three parts: (1) a controller

selection management (CSM) module, (2) a cross-domain management (CDM) module, and (3) a network information collection (NIC) module. The NIC provides a domain-wide network state by aggregating data from controllers, while CDM offers a global network state by aggregating domain data. The CSM module determines the most efficient controller for each network flow depending on domain-wide and network states. As a result, WECAN enhanced network latency and throughput compared to a single-controller network like Floodlight or Maestro.

Hence, Almadani et al. [107] propose DSF, an adaptive framework for the EWi design for heterogeneous dmCP to synchronise topologies using a standardized communication protocol. DSF has been tried out with both the ONOS and Floodlight controllers. However, although the authors claim that DSF can run on different platforms, the current version of DSF only works on Java-based control platforms.

The authors of MNOS [108] and Mcad-SA [109] explore a heterogeneous SDN control plane that can deal with security challenges. They developed a dCP to counteract hijacking and modification attempts. Particularly, a mimic cyberspace defence (CMD) is included as the core concept underpinning MNOS. By incorporating CMD into SDN controllers' design, they could produce an N-variant controller framework with dynamic, heterogeneous, and redundant properties. The CMD protects the controllers against any backdoor, or modification attacks, but not against attacks such as DDoS or OpenFlow known vulnerabilities. And MNOS is only limited to the variety of controllers used. Recently, Yi et al. [10] were inspired by these works to formulate a secure aware heterogeneous CP (SQHCP) to optimize delay, resource utilization, and failure rate in SDN with homogeneous controllers due to common-mode fault (CMF). The techniques involve two steps: step one deals with determining the number of heterogeneous controllers using a knapsack approach based on dynamic planning to ensure tight control plane security. Step two deals with network partitioning using the K-means clustering algorithm. The inclusion of heterogeneity in [10] is security motivated. Because of the CMF of CP homogeneity, attackers familiar with one controller's vulnerabilities can bring down the entire network. They got rid of the threat by using heterogeneous controllers such as NOX, Ryu, Floodlight, and ONOS in the network.

Similarly, Hoang et al. [110] have recently proposed (SINA), an EWi, to guarantee the interoperability of a decentralised and heterogeneous SDN system. A unique consistency algorithm for an adaptive quorum-based replication mechanism is also provided. The former shows that SINA's consistency strategy, active replication, based on broadcasting, is valid. The latter evaluates SINA's Q-learning-based quorum-based replication strategy. SINA achieves better reading and writing latency and overhead than other well-known interfaces. Despite its excellent consistency, active replication overuses system resources (bandwidth, processing capacity, etc.), which is one of its downsides. In similar work, Moeyersons et al. [111] proposed "Domino" a pluggable framework for managing heterogeneous SDN. Domino incorporates a microservice architecture allowing users to integrate multiple SDN controllers.

4.4. Summarised Insight. All the techniques cited in Table5 offer an EWi solutions of communication in dCP. Among all the proposed techniques, only [10, 108, 109] addressed security issues related to hijacking, flow rule modification attacks and a CMF. Furthermore, none of them provides protections against such as DDoS or any known OpenFlow protocol vulnerabilities. SDNi is an IETF initiative and is already in use in production networks. However, as there is no generally accepted specification for the east-west interface, the growth of SDN is stymied in extremely large-scale network settings.

5. Controller Placement Problem

5.1. Problem Description. Managing a large-scale network with a single controller was inefficient due to its lack of scalability and reliability; hence, CPP has emerged as a fundamental research subject in SDN. The idea was first conceived by Heller et al. [9]. For any given network, the problem is determining how many controllers must be employed in the network and (ii) where they should be positioned on the network so that the impact of average and maximum latency between controller and switch is reduced. The problem has been modelled over the years to consider multiple factors, including reliability, load balancing, cost, and security. As a result, we highlight the CPP's mathematical modelling and problem formulation, solution approaches, and design objectives in Sections 5.2–5.4.

5.2. Mathematical Modelling and Problem Formulation Approaches. CPP is an NP-hard problem similar to a well-known facility location problem (FLP) [129]. With FLP, the controllers are treated as facilities, while switches are as demand points. However, other mathematical techniques such as Knapsack, Vertex Cover (VC), and Set Cover Problem [130] are also reducible to CPP [131]. Furthermore, techniques such as field matching problem [132] are used in [133, 134] and dominating set problem [135] in [136].

To formulate CPP, the network being considered is modelled as graph $G = (V, E, S)$ with V, S & E representing the controllers, switches, and links, respectively. Suppose you have $n = |V| \forall\, v \in V$ as the number of controllers and $k = |S| \,\forall\, s \in S$ as the number of switches, the solution of the CPP is to find the number n and the mapping of $v \in V \longrightarrow S$ in the network. Every CPP solution is aimed to improve network efficiency as measured by QoS KPIs such as latency, throughput, overhead, loss, response time. Accordingly, the goal of any given CPP method is to optimize one or a combination of these metrics subject to several constraints. Table 6 summarises most of the widely used mathematical symbols in CPP formulation.

5.3. Solution Approaches to Controller Placement Problem. After the problem is formulated, many different solution-seeking approaches can be applied to solve the problem. As shown in Figure 7, for optimal solution, optimization approaches using linear programming (LP), integer linear programming (ILP), binary integer programming (BIP),

Table 5: Comparison of controller interoperability.

Reference	Design		Motivation/objectives metrics test				Controllers used	Method	Weakness
	HM	HT	Security	Consistency	Interoperability	Capacity/ delay			
SQHCP [10]	X	√	√	√	NA	√	NOX, Ryu, Floodlight, ONOS	Knapsack, *K*-means, GA	It has not been exposed to practical network scenario
SDNi [12]	√	X	√	NA	NA	√	NA	IETF protocol	Did not describe how messages are stored and exchange
CIDC [99]	√	X	NA	NA	√	√	NA		
mCIDC [100]	√						NOX, Floodlight	NA	No detail available
EWBridge [101, 102]	X	√	NA	√	NA	√			
FlowBricks [103]	X	√	NA	√	NA	√	Differ versions of Floodlight	FlowBricks	Not tested across different platforms
Zebra [104]	X	√	NA	NA	√	√			
HCM [105]	X	√	NA	NA	NA	√	Floodlight, Ryu, POX	Unified API	
WECAN [106]	X	√	NA	NA	NA	√	Floodlight, Maestro, Pox	NA	It has not been tested with popular DP like OvS but PICA8
DSF [107]	X	√	NA	√	NA	√	Floodlight, ONOS	Data-centric RTPS Markov chain	Lack of QoS profiles for C-C polling
MNOS [108]	X	√	√	NA	NA	NA	ODL, ONOS, NOX, POX, and Floodlight	Mimic layer	Cannot protect against DDoS, OpenFlow protocol known vulnerabilities
Mcad-SA [109]	X	√	√	NA	NA	NA	NOX, POX, Floodlight	Theory analysis	
SINA [110]	X	√	NA	√	√	√	Faucet, ONOS, ODL	*Q*-leaning	Excessive use of resources due to the active replication for consistency
Domino [111]	X	X	NA	NA	√	√	Ryu, Floodlight, ONOS	Plugging	Require intensive hardware modification

Table 6: Most commonly used CPP mathematical symbols.

Symbols	Description
$\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{S})$	Network graph
$\mathbf{V}$	The set of multiple controllers
$\mathbf{S}$	The data plane switches
$\mathbf{E}$	Control path
$\mathbf{n}$	Total number of controllers
$\mathbf{k}$	Total number of switches
$\mathbf{d}(\mathbf{v}, \mathbf{s})$	The shortest path between $v \in V$ and $s \in S$
$\mathbf{l}(\mathbf{c})$	Number of switches under the controller v_i
$\mathbf{p}$	Chances of failure
$\boldsymbol{\rho}$	Controller's capacity
$\boldsymbol{\Delta}$	S_C latency threshold
$\mathbf{x_i} = \begin{cases} 1 & \text{if} \mathbf{c_i} \text{exist} \\ 0 & \text{otherwise} \end{cases}$	Binary variable to indicate if a switch s_i is associated with a controller c_i
$\mathbf{y_{ij}} = \begin{cases} 1 & \text{if} \mathbf{v_i} \text{mapped} \mathbf{c_j} \text{exist} \\ 0 & \text{otherwise} \end{cases}$	Binary variable to indicate if a switch s_i is associated with a controller c_j

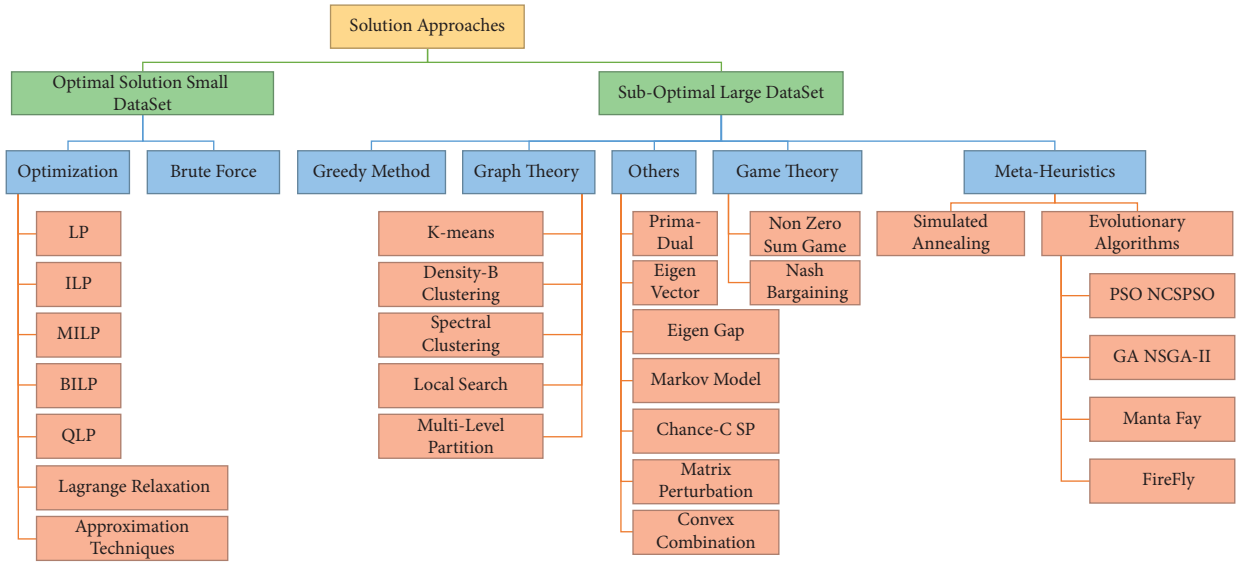

Figure 7: Taxonomy of CPP solution approaches.

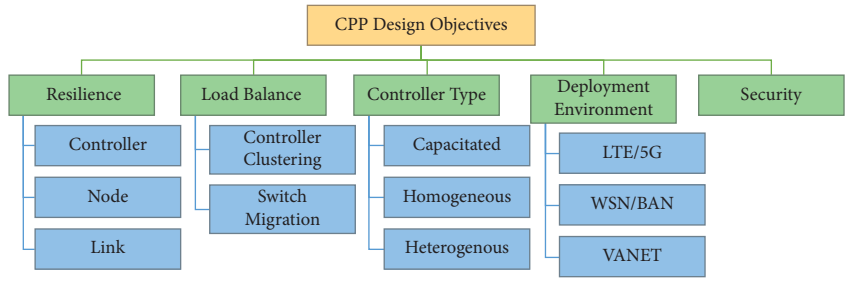

Figure 8: Taxonomy of CPP design objectives.

mixed integer linear programming (MILP), or quadratic programming (QP) are used. Brute force methods can also use when dealing with small data set on optimizer such as CPLEX [137]. But for suboptimal solutions, heuristics, and meta-heuristics algorithms such as simulated annealing (SA) and evolutionary algorithm (EA) such as particle swam optimization (PSO) and its variant NCPSO, genetic algorithm (GA) and its variant NSGA-ii, KnEA, Firefly, Manta Foraging Ray algorithms are usually used on expanded data set. Similarly, random, greedy methods, game theory such as Nash, nonzero sum and machine learning techniques are also applied [17].

5.4. Controller Placement Problem Design Objectives. Figure 8 illustrates the performance metrics for the CPP. One of these metrics is controller-to-switch latency. Latency has several causes such as propagation, packet transmission, switch processing, controller processing, and controller queuing latency [138]. Various CPP methods have attempted to minimize the effects of one or more of these delays. Reliability is another important metric that has been studied extensively. In an SDN, failure might originate from either the DP or the CP [139]. In the former, failure comes from the forwarding devices or links; in the latter, failure arises due to software or hardware. Existing works that proposed resilient dmCP designed their techniques to guarantee control path redundancy via multiple control message paths. Others elect to shorten the length of the control path. In contrast, others implement numerous controllers, as shown in Figure 9. These proposals are reviewed in Section 5.4.1 and summarised in Table 7. Next is load balance, measured by synchronization, statistics collecting, and flow setup overhead. Controllers' maximum and average loads are also measured. CPP with load-balancing efforts is summarised in Section 5.4.2 and Table 8. The capacity and model of the controllers used impact CPP design. This study classified the CPP as capacitated, uncapacitated, homogeneous, or heterogeneous [58]. Section 5.4.3 provides the review of CPP when deployment and application environments are considered.

In addition to DCN and WAN, SDN is widely used in emerging technologies such as WSN, VANET, and IoTs. Section (5.4.4) reviews these proposals, and Table 9 compares them. Similarly, numerous CPP solutions have incorporated a cost component. According to [13], capital expenditures (CAPEX) and operating expenses (OPEX) are the two main ways in which cost is viewed in CPP. CAPEX refers to the funding allotment for hardware components. It is usually related to their sheer quantity and robust features such as processing power and number of ports while OPX addresses energy costs.

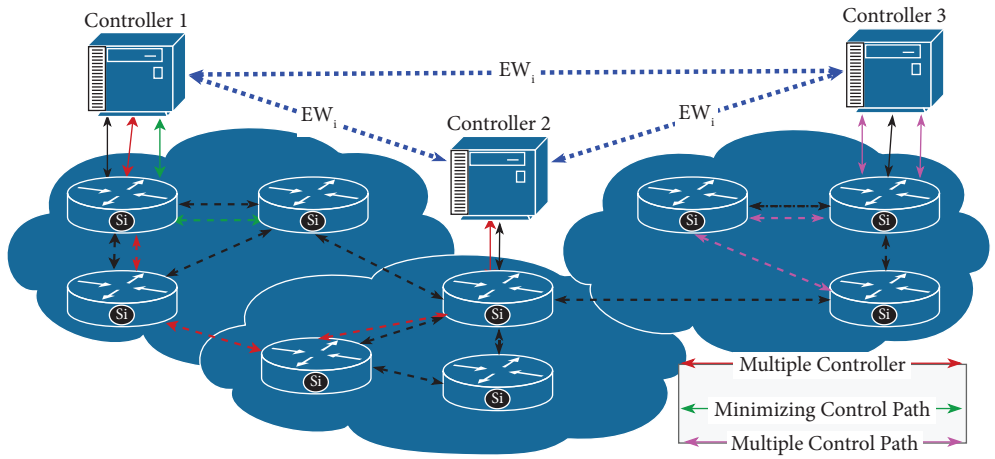

Figure 9: Resilience CPP design options.

5.4.1. Resilience Controller Placement. Control messages are exchanged between CP and DP during the SDN update operation. The end-to-end transmission line for this exchange consists of one or more switches at the DP, a controller, and a link. A loss of packets in the exchange may occur due to congestion, a faulty component (controller, switch, or link), or a security incident. In either case, any loss in the control packets will devastate the network's behaviour. Several research efforts are made to mitigate this problem by proposing a resilient dmCP [140–143]. As shown in Figure 9, some of these efforts proposed resiliency in dmCP by guaranteeing redundancy in the control path via multiple paths. Similarly, there have been some efforts to shorten the control path by using fewer nodes along the path in order to reduce probable failure points while others seek resillience in the event of failure, by connecting switches to multiple controllers [14]. Table 7 also provides a summary comparison of these techniques reviewed.

Hu et al. [140] initiated a novel reliability metric to formulate a fault-tolerant CPP (FTCPP) as ILP. The objective function aims to maximize the expected percentage of valid control paths, i.e., the logical links between DP to controllers. The authors proposed three (3) variant techniques: random placement, brute force, and l-w-greedy algorithm to solve the problem. They used networks from the ITZ repository to validate the methods. However, the proposed technique is not benchmarked against any other previous study, perhaps because it is one of the pioneer's works on CPP with reliability.

For this reason, its performance cannot be substantiated at the time, but it can be done now that different approaches are available. Guo and Bhattacharya [142] adopt a partition approach to achieve the triplet of reliability, scalability, and security in SDN. By considering the interdependence networks cascading failure, they proposed a network partition technique for controller placement. Simulation results show that the expected network failure tendency is inversely proportional to the average path length of the network. However, the technique has not been evaluated on real topology rather, and they used a synthetic method using the igraph library to generate three different networks with a ring, binary tree, and Erdos-Renyi random network topology for the evaluation. The approach in [143] used MIP to design a capacity-aware CP technique. The authors use two strategies to prevent network outages with smooth failover plans in the event of one. In the first strategy, an assumption that a node is joined to the controller via two separate control paths was made, while in the second strategy, they assume that a node is linked to multiple controller replicas via two separate paths. The effect of different topology sizes and the number of controllers on the average path length is investigated using networks from the SNDlib database.

The work in [144] proposed two optimization models to improve reliability in CP. They formulate a CPP under a comprehensive network states (CPCNS) in the first model. While in the second model, they develop the problem for a single link failure" (CPSLF). Then, we proposed an optimal CP algorithm and greedy algorithm to solve the two problems. Their motivation for adopting this approach was to solve the 70% network failure experienced by a single link. Another study [145] used a greedy algorithm to design a technique for finding multiple control paths for exchanging control messages in SDN. They used a clustering-based global optimization for finding the shortest path among them. For average reliability and minimum computational complexity, a reliability factor is defined. The authors in [146] adopt the strategy of network partitioning to design a distributed and reliable CP. Depending on the subnet size, a formula for calculating the reliability of each network domain is proposed. Similarly, based on each subnet load, controllers are distributed accordingly. Hence, the authors considered the packet loss rate, and the node degree for the assignment. They also designated a coordinator to detect any nonactive node, so that an appropriate controller can be relocated to take charge of the failed subnet. The coordinator considers the calculated reliability of the failed subnet and its distance relative to the new controller before the reallocation. However, all these techniques may have succeeded in minimizing the length of the control path to minimize the

Table 7: Comparison of resilience aware controller placement.

Ref	Reliability aspect			Metrics: path loss, failure rate, response time, latency, controller, reliability, loss, load	Problem formulation	Solution approach
	MCL	MC	MCP			
[131]	—	—	√	Controller, reliability	LP	Min-cover
[131]	—	—	√	Controller, reliability	LP	Min-cover
[136]	—	—	√	Latency, reliability	ILP	GA
[140]	√	—	—	Path loss	ILP	Random brute and greedy
[141]	—	—	√	Response time	ILP	—
[142]	√	—	—		Analytical	Selection
[143]	√	—	—	Failure rate, path loss	MIL	—
[144]	√	—	—	Response time, failure rate	LP	Greedy
[145]	√	—	—		NA	GA
[146]	√	—	—	Failure rate	NA	Bully
[147]	—	√	—	Failure rate, latency, controller	NA	—
[148]	—	√	—	Latency	NA	—
[150]	—	√	—	Failure rate, latency	Analytical	Mathematical
[152]	—	√	—	Failure rate, latency	MILP	SA
[112]	—	√	—	Latency, controller	ILP	—
[153]	—	√	—	Failure rate, controller and load	NA	Graph theory
[154]	—	√	—	Latency	NA	MOO, KnEA
[155]	—	√	—	Response time	NA	*K*-median
[156]	—	√	—	Failure rate, load balance	NA	PAM-B, NSGAII
[157]	—	—	√	Latency, loss	Analytical	*K*-critical
[151]	√	—	—	Failure rate, latency	Analytical	Mathematical
[158]	—	√	—	Latency	Fuzzy integral	AHP&, B-PSO

Minimizing control links, multiple controller: MC, multiple control path: MCP.

Table 8: Comparison of controller placement with load balance awareness.

Ref	Approach		Load balance optimization objectives	Problem formulation	Solution approach	Weakness
	CCA	SMA				
[52]	X	√	Response time, throughput, migration time	NA	4-phase migration protocol	Not support HetCP, CMF, controller location, scalability
[112]	√	X	Latency, no of controllers	ILP	CPlex solver,	High overhead at master controller, CMF
[113]	√	X	Setup time	None	Flow request partition	CMF, additional overhead at CP
[114]	√	X	Flow setup time, comm overhead	ILP	Heuristic	Master controller overhead
[115]	√	X	Overhead	QIP	Greedy	No support HetCP, CMF
[116]	√	X	Setup time	None	Greedy	CMF, imprecise controllers' load collection. Vulnerable
[117]	√	X	Latency, availability	Graph theory's centrality stress	Heuristics, lattice graph	Not support HetCP, CMF
[118]	√	X	Response time	NA	Cluster vector	High overhead at master, not support HetCP, CMF
[119]	X	√	Flow setup time, utilization	Markov chain	Distributed hopping algorithm	Not support HetCP, CMF
[120]	X	√	Setup time	Bin-packing	Greedy algorithm	Not support HetCP, CMF
[121]	X	√	Throughput, load oscillation	Load informing strategy	Inhibition algorithm	Heterogeneous interoperability problem, CMF
[122]	X	√	Latency, saturation attack	3-D Earth mover model	Heuristics	Controller location, not support HetCP, CMF
[123]	X	√	Effective controller load	LP, graph theory	Heuristic	Not support HetCP, CMF
[124]	X	√	Setup time, overhead, latency	ILP	TOPSIS, heuristic	Not support HetCP, CMF
[125]	X	√	Response time, throughput	Game theory	Game theoretic	Not tested on a production network, does not support HetCP, CMF
[126]	X	√	Overhead, response time	NA	Heuristic	Not support HetCP,CMF
[127]	X	√	Failover, throughput, loss	Mathematical	Heuristic	Prolong end-to-end RTT, not support HetCP, CMF
[128]	X	√	Response time, throughput	Mathematical	Heuristics	Not support HetCP, CMF

Table 9: Comparison of deployment environment aware controller placement.

Reference	Environment	Objective metrics							Traffic	Problem formulation	Solution approach
		Failure rate	Throughput	Latency	No of controllers	Load	Loss	RT			
[164–166]	LTE/5 G	—	—	—	√	—	—	—	Voice, videos	Bin packing MIQCP	Greedy heuristics
[167]		—	—	—	—	√	—	—	Dynamic	ILP, random Waypoint	Nash game
[124, 168, 169]	LTE/5 G Wi-fi	—	—	—	√	—	—	√		MILP	2-stage stochastic
[170, 171]	Wi-Fi	—	√	—	—	—	—	—		LP	Heuristic
[172]	WSN/BAN	—	—	—	—	—	—	—		LP	Analytical
[173]	VANET	—	—	√	—	—	—	—		Mathematical	Game theory, GA
[174]	LTE/5 G	—		√	—	—	—	—			Use cases
[175]	WSN/BAN	—	—	—	—	—	—	—	Static	Experiments	Proof of concept
[176, 177]		—	√	√	—	—	—	—	Dynamic	MAP	Experiment
[178, 179]	Wi-Fi	√	√	√	—	—	—	—		MOOP, LP	SAGA heuristics
[180, 181]	WSN/BAN	—	—	√	—	—	√	—		Mathematical	Sector divide routing
[25]	WSN/BAN	—-	—	√	—	—	—	—		Mathematical	Greedy
[182]	LTE/5 G Wi-Fi	—	—	√	—	—	—	—		—	PoC
[183]	LTE/5 G	—	√	√	—	—	—	—		Comparative	PoC
[184]	LTE/5 G Wi-Fi	√	√	—	—	—	—	—		Stochastic K-Median	SA, ray-shooting

PoC: proof of concept, SA: simulated annealing.

possible failure points, but the shortest path selected may not be the best path in terms of other QoS metrics like bandwidth. Although control messages may not have a high bandwidth demand, if there are many requests, the shortest path with limited bandwidth may be overloaded, causing a high response time and subsequent failure.

In contrast to minimizing the control path length to ensure CP reliability, other techniques assign switches to multiple controllers for redundancy [112, 147–154]. For instance, Tanha et al. in [147] proposed a technique that maps each switch on DP to one primary and multiple levels of backup controllers. Sridharan et al. in [148] designed an algorithm for mapping switches to multiple controllers in a distributed controller architecture. The algorithms distribute flow setup requests among the multiple controllers to minimize the controller's response time and satisfy the elastic constraints. The outcome reveals that increasing the network budget increases the network's resilience level. The work in [149] switches is assigned to a primary and backup controller(s) to prevent controller failures. Killi and Rao [150] propose a technique for controller placement that minimises worst-case latency in the event of controller failures. The authors assumed that switches have a failure-foresight ability. This means that the switches are aware of the current state of the controllers. The authors employ LP to mathematically formulate a reliable and capacitated aware CPP that can withstand the failure of up to $(k-1)$ controllers. The technique is evaluated using networks from ITZ. However, the authors did not consider the average latency between the switch to controller and controller to controller. Therefore, in [151], Killi and Rao proposed another mathematical model that investigates the worst-case latency in the event of single-link failure. They formulate a reliability-aware CPP as an ILP to determine the expected percentage of control path loss. The authors develop a greedy algorithm to solve the reliability problem. The aim is to find answers to the questions: How many of them will be enough to maximize reliability if controllers are carefully placed? However, the greedy approach cannot guarantee an optimal solution to the problem. Thus, a different problem variant is formulated in [152] as capacitated next controller placement (CNCP) strategy. CNCP's failure-foresight assumption of switches made in [150] is relaxed. They also assign multiple controllers to each switch to ensure redundancy in the event of controller failure. The problem is expressed as a MILP, and a simulated annealing algorithm is designed to solve it heuristically. The technique is also tested on ITZ networks. Perrot and Reynaud [112] propose another resilient controller placement strategy in which switches are assigned to 1+ controllers for redundancy against possible controller failures. The problem is formulated as an ILP with the authors assuming that the probability of the controller's failure is the same. Reference [153] proposed a delay-guaranteed, capacity-aware CP for SD-WAN. The authors consider a controller-to-controller node failure to design a resilient and capacitated aware CP (RCCP) using a master-slave (M/S) model that can restore network operation in the event of failure. They deployed the technique on networks at ITZ for validation. However, the technique is lacking in inclusiveness and flexibility concerning adaptability to delay-sensitive applications and their service level agreements (SLAs). Conversely, the

work in [155] proposed a method for dealing with CP failure recovery using a backup controller selection algorithm that minimizes average failure recovery time free of load oscillation caused by switch migration. To design the proposed technique, they consider QoS requirements as well as controller activation costs. The authors assume that flow requests have the same processing time. However, this assumption cannot be true because of flow variability and different controller capacities. In another technique, Hu et al. [154] proposed a solution, arguing that the inappropriate selection of slave controllers to handle the additional load of switches whose master controller is down can lead to what they described as "controller chain failure." To this end, they designed an adaptive slave controller assignment (ASCA) technique to avoid it. ASCA has three modules, namely, the selecting slave module, the assignment slave module, and the adaptive adjustment module. The problem is formulated as an ILP with the objective to minimize load variance difference (LVD). This is done such that a lower value of LVD can be selected to get better fault tolerance and load balancing performance. They used KnEA to design heuristics to solve the problem. And the solution has been shown to be effective in avoiding the chain failure of controllers. However, the migration of the affected switches to the slave controller is not done with respect to the type of flow coming out of these switches, and this can have QoS violation consequences. It lacks a clear explanation of how the load distribution affects the queuing delay. It also did not show the differences in controllers' utilization due to load differences. Lastly, Bannour et al. [156] proposed two-stage novel context-based techniques that cover load imbalance and latency metrics to optimize response time in the event of failure. They created an information-gathering mechanism in stage I to collect and transmit topology information to the placement algorithm in stage II. The information is used by the cluster leader election scheme to select a leader. The followers send their cluster's neighbourhood latency status to their respective leaders to be used for synchronization and global topology building. Among the leaders, one is designated as the "hyper leader," responsible for building global logical topology and running the Dijkstra algorithm. In stage II, a partition around medoids (PAM-B, aka the "k-Medoid method") is used to partition switches into K clusters of controllers, whose quality is gauged based on the average distinction of all nodes to their nearest medoid. Then, NSGAII is used to optimize the solution.

One drawback of these methods is maintenance costs. Deployment costs include the purchase price and ongoing operating costs of network devices such as controllers and switches. This consists of the cost of purchasing and installing the controllers into the network and the cost of connecting the controller to the switch. In addition, energy costs are an operational expense.

Therefore, in a different approach to deploying multiple controllers, some techniques choose to deploy various control path instead. The multiple paths guarantee at least two disjoint paths connecting DP and CP, which protect the control path against single link and node failures by switching to an alternate path. One example of this approach is proposed by Muller et al. [141] where a proactive capacity-aware CPP solution called survivor enhances CP resilience to failure and recovery. Survivors use the path diversity approach to ensure the redundancy of transmission paths between the CP and DP. This way, an auxiliary connection between the two planes is usually assumed to be available. The redundancy reduces connection loss by 66%. Survivor also adds a mechanism that periodically checks the controller's load concerning its capacity to avoid overload.

The work in [157] proposes a novel, reliable controller deployment mechanism using a K-critical technique. They aim to construct a robust control layer that considers network characteristics such as interference while selecting appropriate controllers. The paper earlier proves how and why choosing only the shortest control path is an inefficient way of enhancing control layer load and robustness. Similarly, Zhong et al. [131] defined two metrics for checking control path reliability by looking at how many switches may lose connection with their controllers when a single-link failure occurs. Then, we formulate a problem that can find a controller's neighbourhood minimum coverage area in the network. Furthermore, it keeps a list of backup links if a link fails due to unforeseen circumstances. The goal is to increase dependability and simultaneously reduce the number of controllers required. In addition, the study proposes a heuristic based on particle swarm optimization that begins with all switches as controllers and then generates a nearly optimal solution that is practicable. However, both [131, 141, 157] do not account for the switch-to-controller delay, intercontroller latency, or controller load (s).

(1) Summarised Insight. All the techniques that minimize the length of the control path may have succeeded in reducing the possible failure points, but the shortest path selected may not be the best path in terms of other QoS metrics like bandwidth. Although control messages may not have a high bandwidth demand, if there are many requests, the shortest path with limited bandwidth may be overloaded, causing a high response time and subsequent failure. Therefore, it is important to minimize the number of potential failures while optimizing the QoS parameters for optimal performance. As a result, taking into account other QoS metrics in relation to the total number of flow requests is necessary in order to reduce the number of potential failure points. As for the multiple controllers or multiple control path approaches, one drawback is the significant initial investment and ongoing maintenance costs. Deployment costs include the purchase price and ongoing operating costs of network devices such as controllers and switches. This includes the cost of purchasing and installing the controllers into the network and connecting the controller to the switch. Energy costs are operational expenses.

5.4.2. Load Balance Aware CPP. The task of flow request handling at CP had made it a source of performance bottleneck [159]. As a result, various works proposed CPP solutions to balance load across the CP. There are two approaches to achieving this objective. A controller clustering approach (CCA) and switch migration approach

(SMA) are discussed in Sections 5.4.2.1 and 5.4.2.1, respectively. Table 8 shows the summary comparison of each technique proposed under this category.

(1) Controller Clustering Approach (CCA). The controller clustering approach (CCA) can be considered a proactive approach to dCP load balancing in SDN. In CCA, one controller is designated as super while others as subordinate. The super coordinates the functions of the subordinate and balances the load across them centrally.

BalanceFlow [113] is a typical CP load balancing using CCA. It is based on a hierarchical deployment of controllers with one of them adopting a super role, which reallocates flow setup requests to others in the event of traffic changes. It uses the multicontroller feature of OpenFlow 1.2 to regulate the process. The controllers maintain and synchronise their load information periodically via an EWi. The method has the advantage of flexible tuning of flow requests by each controller without introducing extra latencies. However, it introduces additional overhead on the CP. An approach [114] formulated CPP as ILP to regulate controllers' number and location and their assignment to switches in dynamic traffic conditions. The objective function of the ILP seeks to optimize the weighted sum of statistics collection, synchronization, flow setup, and reassignment costs. The authors design a scheme comprising three monitoring, reassignment, and provisioning modules. They proposed two algorithms using a GA and SA based on the knapsack problem and meta-heuristic approaches. Although the simulated annealing method takes more time, it was proven more effective than the greedy strategy. In the paper [115], the authors propose an innovative framework known as MDCP with an objective to minimize overhead. The authors formulate the problem as measurement-aware CPP, which considers synchronization and communication costs. MDCP is designed to be application-agnostic, cost-effective, and lightweight. To avoid computational complexity, a discrete approximation algorithm and a connectivity ranking algorithm are developed to obtain the desired placements. Experiments were carried out on 240 network topologies to validate the technique. The results reveal a 40% reduction in CP overhead. Furthermore, Selvi et al. in [116] propose a cooperative load balancing scheme for hierarchical controller deployment (COLBAS) similar to [113]. COLBAS is a low-cost greedy algorithm in which controllers release their load regularly and coordinate with one another to achieve LB. However, this COLBAS strategy is only centred on implementing LB for distributed controllers without any security considerations for the distributed controller architecture. In addition, the algorithms incorporated into these designs may not be precise enough to collect the controllers' load. Periodic collection of controllers' loads may also result in resource waste. In static approaches [117], we attempted to minimize the burden on switches using the concept of stress centrality to specify the weight of each node based on the number of edge-disjoint paths. This way, the authors proposed a controller placement algorithm to alleviate the burden. The algorithms run in polynomial time to compute the appropriate placement position of the controller. Network topologies from ITZ are used for validation on the simulator running FloydWashall algorithm. However, one major limitation of the suggested technique is that it can only be used for intracluster controller placement or single-controller networks. Similar to BalanceFlow and COLBAS, Sufiev and Haddad [118] proposed a cluster vector (CV) approach to achieve load balancing in CP. The authors simplify the load balancing operation by defining a self-label CV, which contains addresses of controllers in the same cluster. It breaks the dependency of slave controllers on the super as in BalanceFlow [113] by designing high-level operations and low-level operations in the controllers. The CV is built into every controller so that a regular controller can discover the address of another regular controller in an inherently reliable way. In this method, standard controllers can poll other standard controllers to gather load data.

(2) Switch Migration Approach (SMA). The switch migration approach is a reactive way to restore load balance across controllers in dCP as it only comes into play when the load imbalance occurs. SMA posits that whenever a controller is down, or the load of one controller exceeds its capacity, all or a portion of the load of that controller will be transferred to another controller. This way, the controller's memory and CPU availability and resilience can be enhanced to amplify its swift response to any request switch might make [13].

Elastic distributed controller (ElastiCon) proposed by Dixit et al. [52] is the pioneer load balancing approach that uses SMA. ElastiCon is a dmCP designed to grow dynamically or shrink concerning traffic changes. It has three modules: load measurement, load adaptation decision, and action modules. The LMM keeps track of the load of each controller. It collects and sends the load information to the LADM which then determines load allocation among controllers. Based on the load adaptation module, the AM carries out a load balance sequence, like identifying the immigrant switch. The target controller then migrates the switch to achieve the required load balance.

However, the proposed scheme could not provide clear information on the controller location that can reduce CP latency and the key challenge of resilient architecture. Also, the authors of [119] explore SM with network utility optimization for scalable CP within the constraints of limited resources. The authors formulate the problem as NUM and develop a distributed hopping algorithm (DHA) to address it. However, the scheme suffers from high migration costs and frequent load shifting. For this reason, a trade-off between migration costs and load balance rate is considered in [120] to design an SM decision-making (SMDM) scheme. They formulated the SM procedure as a bin-packing problem and applied a greedy algorithm to find an optimal solution.

Similarly, in Wang et al. [121], an SMA based on load informing strategy (LILB) was developed. Each controller actively and periodically synchronises its load information with the other controllers to detect load imbalance. The algorithm then traverses load measurement, load informing,

decision, and SM modules to restore load balance at CP with maximum throughput and minimum load oscillation. Nevertheless, the algorithm may be susceptible to CMF [10] because it does not accommodate heterogeneous controllers. Thus, the authors in [122] combined an SMA with security features to protect important controllers against DDoS and eavesdropping via load relief by adjusting load differences among the controllers. This is a flexible SMA designed using a 3D-EMD algorithm. Nonetheless, the specific CMF [10] vulnerabilities are still unaddressed in the proposed strategies as it doesn't support heterogeneous controllers in the system. Also in [123], Tarai and Shailendra tackled challenges of poor resource utilization and wastage owing to load imbalance and security in deploying IoT devices for smart cities using SMA. As such, they formulate the problem as LP to obtain the initial device placement. Subsequently, they develop an SMA to minimize migration costs in the event of load imbalance. The limitation of this scheme is that the immigrant switches and destination controller selection did not adhere to traffic engineering principles. In contrast, the solution proposed by Sahoo et al. [124] uses a multicriteria decision-making procedure called "the technique for order preference by similarity to an ideal solution" (TOPSIS) to facilitate the selection of the target underutilised controller that immigrant switches will be reassigned in their proposed framework for load balancing in SDN control plane while a zero-sum game theory is used in [125] to help choose an immigrant switch(es) from an overloaded controller(s) and a recipient controller for the migration. To do this, the recipient controllers take the role of players in the game, while the immigrant switches serve as commodities. Numerical results reveal that the technique can relieve controllers from heavy loads beyond their capacities. However, despite its apparent speed, game theory is probably not well suited for usage in a wide-area SDN. Similarly, solving the SMP requires a lengthy time to reach the optimal result (which may not be acceptable under a dynamic traffic distribution) or generates heuristic solutions that are inadequate in migration performance. In the event of multiple controllers overload, instead of relieving their load one-by-one via independent SM execution, the authors of [126] proposed an SMCLBRT technique that executes the operation of all the affected controllers at a time to minimize time. To improve the selection of outmigration controllers, they consider response time delay in addition to the current load. However, because of the strategy's emphasis on switch migration by several controllers simultaneously, it is resource-intensive and might lead to congestion. And in a closely similar idea to [126], Mahjoubi et al. [127], in their proposed LBFT technique, grouped all the immigrant switches after their identification and migrated them all at once to restore load balance. The scheme is effective in terms of failover recovery time and packet loss but at the expense of RTT. However, like the other proposals, traffic flow classification is not considered in terms of its uniqueness.

One common shortcoming of these reviewed CPP solutions with load balancing mechanisms using a switch migration approach is their failure to accommodate heterogeneous controllers in their proposed design. In other words, the solutions only support homogeneous controllers. However, as explained in the section, this might have security vulnerabilities such as "homogeneous controller common-mode fault." (HC-CMF) [10]. Any potential vulnerabilities in one controller would be reflected in all identical controllers. If attackers could exploit one of these vulnerabilities, they would be able to bring the whole network down. This might have devastating consequences for the entire network. However, the network might be insulated from this threat if the controllers deployed are heterogeneous.

Apart from these works, many other methods like [128, 160–163] have presented various load balancing solutions at dCP employing a switch migration approach without necessarily resolving a CPP to get the initial CP to DP mapping.

(3) Summarised Insight. Based on this review, we may deduce some possible deficiencies in CCA: memory, CPU, and bandwidth limitation may reduce the performance of the centralized "super controller." First, the super controller collects load information periodically and routinely exchanges a large number of messages with other controllers, resulting in a decrease in system performance. Second, there is the possibility of SPOF. Thus, if the super controller fails, the entire technique for load balancing fails. This compromises the availability of distributed controllers. Third, each load balancing operation necessitates two network transmissions: one for collecting load and one for giving commands. In such a scenario, the aggregated load data may be outdated, and the command may lag behind the actual load status. Similarly, none of the methods considers that network entities would have different processing needs based on characteristics such as flow table size, queue size, and flows request variability.

Furthermore, the inability to accommodate heterogeneous controllers in their proposed designs is one crucial shortcoming with security threat implications shared by all these CPP solutions with load balancing mechanisms either using the CCA or switch migration approach. This means that the solutions can only be used with homogeneous controllers. However, security flaws and vulnerabilities like the "homogeneous controller common-mode fault" (HC-CMF) are possible, as detailed in Section 4.1.1. HC-CMF posits that any security flaws in one controller would be present in every other controller of the same model. A successful exploit of even a single vulnerability might compromise the entire network. This could have far-reaching effects on the network as a whole. However, if the installed controllers are heterogeneous, the network may be protected from this threat. Moreover, the designs presented in these publications achieve load balancing across controllers by switching the controller's function over the switch. However, they do not consider the possibility that each controller may have a unique routing strategy; moving the switch directly could disrupt ongoing operations. Finally, flow classification is ignored when switches are reallocated, which can compromise the quality of service for some traffic types.

5.4.3. Deployment and Application Environment Aware CPP. As summarised and compared in Table 9, several approaches have been proposed to address the CPP while considering deployment or application environment.

For e.g., the works in [164–166] proposed a framework for flow processing-aware CP that considered data flow processing and control applications. The framework aims to support SDN architecture deployment in DenseNets. A flexible flow processing-aware controller placement framework (FlexFCPF) places and flexibly reassigns controller devices to manage the future wireless network efficiently. They formulate the problem as a mixed integer quadratic constrained program (MIQCP) for which they design a heuristic using a greedy approach. The study in [185] advocates the integration of the SDN concept in managing the emerging 5 G technology because in 5 G, you will be confronted with huge carrier aggregation and dynamic bandwidth provisioning. Thus, optical integration with the wireless at both the front haul and back haul is eminent. Therefore, elastic optical networking is necessary at the core to optimize resource allocations and utilization centrally. Similarly, massive MIMO and digital beamforming mechanisms require a lot of computing power in the 5 G technology. In this vein, SDN will come handy in providing agile and flexible distributed management of 5 G from a centralized controller. As such, [167] joined the vision of integrating the novel 5 G technology with SDN. For this to happen, it will involve the separation of control and user DP functions of the evolved packet core (EPC) of the 5 G technology. This will give birth to serving packet gateway controller S/PGW-C and packet data gateway-user S/PGW-U. The authors focused on the placement problem of the serving gateway controller (SGW-C) in a 5 G network. As such, they made a trade-off between minimizing the SGW relocation rate and traffic load balance among the underlying SGW-C virtual network functions (VNFs) in the problem formulation. They formulate the problem as an ILP optimization model and apply game theory using Nash bargaining game and the threat point techniques to obtain a fair solution aka (Pareto optimal). Another work [174] proposed a hybrid hierarchical architecture of multiple SDN controllers to manage 5 G networks. The architecture is designed as a federating unit of multiple subnetwork controllers, with each focusing on a single subsection of the network but centrally coordinated by a hierarchically superior controller. The architecture is made up of a global controller module, area controller module, and user equipment (UE) with publisher/subscriber, routing, and topology modules. They integrate a data distribution service (DDS) on the publisher and subscriber module on the ODL controller at each control level. They experiment with three (3) different use case scenarios to check the functionality of the architecture performance. Similarly, the authors of [183] also harvest the potential of SDN to enhance the management of 5 G technology. They integrate the logically centralized-physically distributed LC-PD SDN CP architecture in the management of a 5 G network. They conduct experiments on Mininet to demonstrate how LC-PD architecture can optimize the overall output of 5 G network quality of services (QoS). They provide proof of concept experimental results and recommendations to adopt SDN LC-PD CP architecture in 5 G technology.

Other technique in [168] introduced two approaches for CPP formulation and assignment to switches in a wireless and wired SDN environments. In the first approach, they investigate the controller's average response time when the connection between the controllers and switches is wired, while in the second approach, they consider per-link response time constraint using chance-constrained stochastic programming (CCSP) when the transmission links between the controllers and the switches are wireless. Other approaches [168] study a CPP in cellular networks, factoring the uncertainties of user mobility to propose a C^3P^2 and CPPA, respectively. Is a static and dynamic joint stochastic CP and evolved node B (eNB) controller assignment method to minimize the number of controllers required to manage the eNB in the cellular network concerning response time, probability, request rate, and user mobility. Other works [170, 171] simulate a 6-Queue system with Bernoulli arrival processes of different rates to investigate the optimality of their controller placement techniques in wireless networks concerning throughput under delayed channel state information (CSI). They model the problem in static and dynamic controller positions to track how the delay in CSI affects the network's throughput. With this, they were able to characterize the variability in the throughput in different regions to allow them to define network policies that best stabilize the system for all traffic dynamics.

In wire sensor network (WSN) environment, an approach in [172] applied SDN principles for energy-efficient resource allocation. First, they formulate an LP optimization problem to minimize the energy consumption of sensor nodes concerning quality-of-service constraints. Afterwards, they propose a software-defined centralized adaptive bandwidth and power allocation scheme (CABPA). Numerical analysis suggests a positive result of the scheme. While in a VANET environment, the techniques proposed in [173] use SDN for load balancing. First, they investigated an optimal rebating strategy to balance latency and cost in VDVNs. They formulated a mathematical model of the problem. Then, they proposed a two-stage game (IGA) to optimize the rebating strategy to balance the latency and the cost based on metaheuristics genetic algorithm to solve it.

Through simulation, the number of packets transmitted through cellular lines positively correlates with the rebate ratio and the other parameters. Another approach [175] also demonstrates how SDN can be deployed in the management of IoT and WBAN applications. The authors emulate SDN functionalities on Mininet with a personal digital assistant PDA acting as the OpenFlow switches under the central control of the programmable SDN controller. They measure deployment complexity and network overhead. The author concluded that the approach is simple, reliable, and cost-effective. As such, the authors of [176, 177] accept the recommendation of [175] to propose an architecture to support the application-specific requirements of WBAN,

named SDWBAN. Unlike classical SDN, a novel HUBsFlow is designed to replace OpenFlow as the SBi protocol in the architecture. In another effort, the authors of [180, 181] proposed an SDWBAN framework that allows centralized administrative controls of incoming data traffic to give flexibility for traffic differentiation of sensitive data with deadline constraints from normal data that require only the best effort in WBAN applications. In a similar health-related application, the authors of [25] applied SDN to optimize the routing of medical emergency packets in the WBAN application. Using mesh topology, the SDN controller is placed on defining the best forwarding node while considering propagation delay and intrabandwidth.

Recently, the authors of [182] conducted a proof-of-concept experiment for using a multicontroller for heterogeneous wireless networks. Therefore, to that end, the authors of [184] proposed a novel technique to tackle the WCPP in a heterogeneous wireless network environment of Wi-Fi and 4GLTE-U. The method aims to improve the throughput, link failure rate, and transparency of the SBi in cases where hybrid providers coexist to choose from as link-layer technologies. The problem of determining the placement of LTE-U and Wi-Fi-based controllers is modelled as an optimization problem, and two heuristic algorithms are proposed to find its solutions.

5.4.4. Security Aware CPP. Driven by the future of the Internet, the solution in [186] proposed a decentralised SDN framework that supports both the physical and logical distribution of CP. D-SDN incorporated a security requirement of identity based cryptography (IBC) which requires a trusted third party (TTP) for secret key generation to the definition of the hierarchy of controllers. The feature is compatible with Internet organizational and administrative structures. It supports administrative decentralization and autonomy to enhance the integrated security feature. For proof of concept, the authors experiment with two use cases of network capacity sharing and public safety service. The mechanism in [187] presents a secure and reliable design of SDN using a cloud-based multiple CP. It is dynamic and uses isolated instance mapping of controller resources in a cloud using a Byzantine mechanism. But the architecture aims to minimize the number of controllers required to be mapped to satisfy the security requirement of each switch. They model the problem as controller assignment in fault-tolerant SDN (CAFTS). This is done concerning the controller capacity and control of message latency. The Byzantine mechanism offers architectural security features. Leveraging on the visualization of controllers' replicas via Byzantine fault-tolerance protocol, the authors proposed a cost-effective requirement first assignment algorithm to solve the CAFTS. It significantly reduces CAPEX of the CP as revealed by the experiments. Zhou et al., in [122], combined the SM technique with security features to protect significant controllers against DDoS by reliving overloaded controllers' load and eavesdropping by adjusting the load differences among the controllers. They consider two types of security breach tactics of adversaries. The first is a reconnaissance attack in which the hacker tracks controller traffic, and the second is a saturation attack through IP spoofing. To mitigate these attacks, the authors proposed a flexible SM model designed using a 3D-EMD algorithm. But the scheme incurred high computation time and lacked a flow classification module to help give differential treatment to flow with QoS requirements. The approach in [123] tackled the CPP with security in a heterogeneous WAN. They addressed issues of delay and resource wastage due to load imbalance, fault tolerance, and insecurity in deploying IoT devices for the smart city via SM. They formulate the problem as LP to get the initial controller placement. Later, we designed an optimization algorithm to minimize migration costs in the event of load imbalance when the network evolved. A consensus protocol is integrated to address malicious security issues. The limitation of this scheme is that the immigrant switches and destination controller selection did not adhere to traffic engineering principles. They did not classify traffic flow according to their characteristics concerning types, variability, QoS requirement etc. Defending against a spectrum sensing data falsification (SSDF), Byzantine attack on an SDN controller is incredibly difficult. If successful, the adversaries will acquire full control of all network devices and behave arbitrarily to disrupt the network. Protection against such a threat requires a $3f+1$ mapping of a switch to a controller, which has the consequence of overload. In [149, 188], the authors propose a novel primary-backup controller mapping and remapping approach in which a switch is mapped to only $f+1$ primary and f failover controllers in the event of a simultaneous Byzantine attack. They formulate two separate minimization problems of primary and backup controller mapping (PBCM) and remapping (PBCR) as ILP, respectively. They then design two heuristics MINCON and MINRUS to solve the problem for large-size networks. The performance study shows that the optimal mapping requires up to 50% fewer controllers compared to an existing scheme and the heuristics perform within 8% of the optimum, see Table 10 for a summarised comparison of these techniques.

6. Open Issues and Future Study

The research presents several EWI communication, consistency, and CPP solutions for the various use cases in SDN. Despite this, there is a pressing need for more inquiry into the issues, as many concerns remain unresolved. This section pointed out some of the problems left unsolved and offered suggestions for new lines of inquiry.

6.1. Resources Utilization Related Issues in dmCP Controller Placement. Adaptive and fair resource allocation-based controller placement is desirable to cope with today's network's dynamic nature and traffic variability. Many of the existing CP approaches with load balancing bias used network partition and controller clustering. Therefore, they are proactive and static in their resource mapping with no provision to adjust to any possible traffic changes. The few works in corporate traffic dynamics use SM or controller reallocation (CR) mechanism to restore load distribution

Table 10: Comparison of related works on dmCP security.

Reference	Attack type				Metrics						Problem formulation	Solution approach
	Sniffing	Byzantine	DDoS	Other	Failure rate	Throughput	Latency	No of CL	Compromised	Loss		
[186]	√		—	√	—	√	—	—	—	—	Cryptography	AKA
[187]	—	√	—	—	—	—	√	√	√	—	Byzantine, bin packing	VMs based IaaS
[149]	—	√	—	—	√	—	√	√	—	√	ILP	Heuristics
[123]	—	√	—	—	—	—	—	—	—	—	LP, RR consensus	Byzantine principles
[122]	—	—	√	—	—	—	—	—	—	—	SM	3D-EMD
[188]	—	√	—	—	√	—	√	√	—	√	ILP	Heuristics
[10]	—	—	—	√	√	—	√	—	—	—	Knapsack *K*-means, GA	HCP

fairness. They track controllers' overload using a threshold value. Therefore, the approaches can be described as reactive, as they only occur when load imbalance occurs. The SM trigger in these solutions is a speculative fixed threshold parameter that lacks any experimental reference [126]. Depending on the threshold size, it may lead to premature or delayed detection, thus leading to network instability. Also, the reactive nature of the approach can cause a delay in the load balance restoration process. Furthermore, there is a migration cost to consider. For example, load oscillation problems may surface if an inappropriate switch or controller is selected. In addition, controller chain failure might occur if switches are not properly mapped to the appropriate controller. Therefore, it will be interesting to explore TE prediction principles to avoid the highlighted issues while addressing fair load distribution in a dynamic environment.

6.2. CAP Theory Issues in dmCP. It is difficult to collectively achieve the three aspects of consistency, availability, and partition tolerance, i.e., the CAP theorem, in the dmCP of SDN. Designers of SDN with partition requirements must deal with performance trade-offs in choosing a consistency level in their designs. You will be confronted with the choice between having weak (eventual) consistency for high availability or strong consistency at the expense of availability. A weak consistency can assure you of the availability of network resources, but it will lead to state staleness in the network, causing abnormal application behaviour. However, with strong consistency and correct adherence to all network policies, you will pay a steep price for network unavailability. Most current CP works in large-scale networks that sacrifice one for the other. Therefore, it will be interesting to consider adopting a hybrid approach by merging these conflicting levels of consistency to strike a balance and look for the optimum trade-off between consistency and availability.

6.3. Heterogeneity of Controllers in dmCP Controller Placement. Another design challenge in SDN dmCP is in the placement and interoperability of heterogeneous controllers from different vendors. Here, you will be faced with both CPP and knowledge-sharing problems, where you will deal with controllers' instances' consistency and compatibility. Overcoming these challenges might require a generic and all-inclusive standardization of the SBi, NBi, and EWBi. The motivation for these can be seen from many perspectives. First, a homogeneous CP poses a possible security vulnerability from a security perspective, considering the controllers possess a common-mode fault, aka a common vulnerability point. If adversaries are familiar with the vulnerabilities of one controller, they can easily bring down the entire network under the common vulnerability of the controllers. Second, interoperability between different controller platforms and traditional IP networks can significantly encourage and simplify the universal adoption of SDN commercially. So far, very few studies have looked in this direction. As a result, conducting additional research in this area will be a worthwhile contribution.

6.4. Security Aware Controller Placement. Given that the control of a network in SDN is centralized via the controller, all transactions that involve the CP ought to be considered critical, as any disruption owing to a successful attack can be catastrophic to the business. Hence, the CP is susceptible to several security threats like the man-in-the-middle attack (MITM), DoS and DDoS attacks, saturation attacks, control packet sniffing and tempering, CP isolation, and IP spoofing. This can be attributed to vulnerabilities such as weak authentication, incomplete encryption, and information disclosure. Therefore, it will be a significant contribution to designing a CPP scheme that incorporates security measures such as role-based authentication, DDoS blocking application (DBA), secure socket layer (SSL), locator ID separation protocol (LISP), the strong message authentication code (MAC) algorithm, or content-oriented networking architecture (CONA) in their solution. Unfortunately, to the best of our knowledge, we have not come across any solution with these features.

6.5. CP with Mobility Tolerance in Wireless Environment. The architecture of the current mobile networks suffers from the same complexities as the traditional network. Fortunately, the concept of "software-defined mobile networking"

(SDMN) is expected to be instrumental in modifying the architecture of the current LTE, the emerging 5 G, and the IoT framework for applications such as WBAN and VANET, respectively. However, despite the SDMN architecture's potential in resource and mobility management, it cannot be fully utilized until the fundamental issue of CP design that it has raised is resolved. In addition to determining the number and placement of controllers in the network, CPP in SDMN must determine the best controller position compatible with a dynamically changing topology. Likewise, it must work in harmony with "on the fly" ubiquitous and heterogeneous networks and handoff support to different radio access networks. Second, in SDMN, controllers will heavily rely on statistics that may include user metadata received from APs, BST, and even UEs for mobility and location tracking. These have security and privacy concerns, as the information can be exploited for mischievous purposes. In addition, these factors will reflect on aspects such as problem formulation and solution methodologies. Thus, the privacy issues and spatiotemporal changes in system parameters required in the algorithms will pose additional intrinsic challenges that further complicate the problem model. Therefore, research in this direction is still open to contributions.

7. Conclusion

SDN architecture is structured in a way that makes the controller the most important component for its smooth and efficient operations, as every decision is made by it. For this reason, the design and operation of the CP are confronted with some challenges that need significant attention to facilitate the adoption of SDN. These challenges might be specific to an application scenario. Several solutions have been proposed over the years to address these challenges. This paper critically reviews the profound issues associated with interoperability, consistency, and CPP in control plane design with multiple controllers. The discussion touches on a wide range of issues, covering the origin of the problem, alternate solutions found at the DP, its evolution, and the state-of-the-art solutions proposed to address them. The objective is to provide an updated evolution of the problem concerning the solutions proposed to guide future research directions. To accomplish this, we begin with a brief overview of SDN fundamentals and then narrow it down to the subject matter, where we discuss the source of the problem. The proposed solutions were then reviewed based on their application environment, controller type, and optimization objectives. The findings of the critical review reveal that a substantial number of solutions were proposed with different degrees of strengths and weaknesses concerning their optimization objectives. Thus, based on that, certain future research directions were pointed out and briefly discussed.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was funded by the Tertiary Education Trust Fund (TetFund) Nigeria, under University Staff Training and Development.

References

[1] B. Isyaku, M. S. Mohd Zahid, M. Bte Kamat, K. Abu Bakar, and F. A. Ghaleb, "Software defined networking flow table management of OpenFlow switches performance and security challenges: a survey," *Future Internet*, vol. 12, no. 9, 2020.

[2] P. Newman, "The Internet of things 2020," 2020, https://store.businessinsider.com/products/the-internet-of-things-report?IR=T&itm_source=businessinsider&itm_medium=content_marketing&itm_campaign=report_teaser&itm_content=full_report_text&itm_term=store_text_link-internet-of-things-report&vertical=conne.

[3] I. Z. Bholebawa, R. K. Jha, and U. D. Dalal, "Performance analysis of proposed network architecture: OpenFlow vs. Traditional network," *Wireless Personal Communications*, vol. 86, no. 2, pp. 943–958, 2016.

[4] A. K. Singh and S. Srivastava, "A survey and classification of controller placement problem in SDN," *International Journal of Network Management*, vol. 28, Article ID e2018, 3 pages, 2018.

[5] K. Qin, B. Fu, P. Chen, and J. Huang, "MCRA: multicost rerouting algorithm in SDN," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 24, no. 6, pp. 728–737, 2020.

[6] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[7] M. Jeon, N. Kim, Y. Jang, and B. D. Lee, "An efficient network resource management in SDN for cloud services," *Symmetry*, vol. 12, no. 9, 2020.

[8] F. Benamrane, M. B. Mamoun, and R. Benaini, "New method for controller-to-controller communication in distributed SDN architecture," *International Journal of Communication Networks and Distributed Systems*, vol. 19, no. 3, pp. 357–367, 2017.

[9] B. Heller, R. Sherwood, and N. Mckeown, "The controller placement problem," *ACM SIGCOMM - Computer Communication Review*, vol. 42, no. 4, pp. 473–478, 2012.

[10] P. Yi, T. Hu, Y. Hu, J. Lan, Z. Zhang, and Z. Li, "SQHCP: secure-aware and QoS-guaranteed heterogeneous controller placement for software-defined networking," *Computer Networks*, vol. 185, Article ID 107740, 2020.

[11] M. Reitblatt, N. Foster, J. Rexford, and D. Walker, "Consistent updates for software-defined networks: change you can believe in," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks - HotNets '1110*, Massachusetts, MA, USA, November, 2011.

[12] H. Yin, H. Xie, T. Tsou, P. Aranda, D. Lopez, and R. Sidi, "SDNi: a message exchange protocol for software defined networks (SDNS) across," 2012, http://datatracker.ietf.org/drafts/current/.

[13] T. Hu, Z. Guo, P. Yi, T. Baker, and J. Lan, "Multi-controller based software-defined networking: a survey," *IEEE Access*, vol. 6, Article ID 15980, 2018.

[14] J. Lu, Z. Zhang, T. Hu, P. Yi, and J. Lan, "A survey of controller placement problem in software-defined networking," *IEEE Access*, vol. 7, Article ID 24290, 2019.

[15] A. Jalili, H. Nazari, S. Namvarasl, and M. Keshtgari, "A comprehensive analysis on control plane deployment in SDN: in-band versus out-of-band solutions," in *Proceedings of the 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, pp. 1025–1031, Tehran, Iran, December 2017.

[16] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, "A survey on software defined networking with multiple controllers," *Journal of Network and Computer Applications*, vol. 103, pp. 101–118, 2018.

[17] T. Das, V. Sridharan, and M. Gurusamy, "A survey on controller placement in SDN," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 1, pp. 472–503, 2020.

[18] G. Wang, Y. Zhao, J. Huang, and W. Wang, "The controller placement problem in software defined networking: a survey," *IEEE Network*, vol. 31, no. 5, pp. 21–27, 2017.

[19] B. P. R. Killi and S. V. Rao, "Controller placement in software defined networks: a Comprehensive survey," *Computer Networks*, vol. 163, Article ID 106883, 2019.

[20] B. Isong, R. R. S. Molose, A. M. Abu-Mahfouz, and N. Dladlu, "Comprehensive review of SDN controller placement strategies," *IEEE Access*, vol. 8, Article ID 170070, 2020.

[21] A. Shirmarz and A. Ghaffari, "Taxonomy of controller placement problem (CPP) optimization in Software Defined Network (SDN): a survey," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 12, Article ID 10473, 2021.

[22] S. Ahmad and A. H. Mir, "Scalability, consistency, reliability and security in SDN controllers: a survey of diverse SDN controllers," *Journal of Network and Systems Management*, vol. 29, no. 1, pp. 9–59, 2021.

[23] K. T. Foerster, S. Schmid, and S. Vissicchio, "Survey of consistent software-defined network updates," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 2, pp. 1435–1461, 2019.

[24] S. Burikova, J. Lee, R. Hussain et al., "A trust management framework for software defined networks-based internet of things," in *Proceedings of the IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 325–331, Vancouver, BC, Canada, October 2019.

[25] B. Manickavasagam, B. Amutha, and S. Priyanka, "Optimal packet routing for wireless body area network using software defined network to handle medical emergency," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 1, pp. 427–437, 2020.

[26] A. K. Al Mhdawi and H. S. Al-Raweshidy, "SDQ-6WI: software defined quadcopter-six wheeled IoT sensor architecture for future wind turbine placement," *IEEE Access*, vol. 6, Article ID 53426, 2018.

[27] B. Isyaku, K. B. A. Bakar, F. A. Ghaleb, and A. Al-Nahari, "Dynamic routing and failure recovery approaches for efficient resource utilization in OpenFlow-SDN: a survey," *IEEE Access*, vol. 10, Article ID 121791, 2022.

[28] B. Yan, Q. Liu, J. Shen, D. Liang, B. Zhao, and L. Ouyang, "A survey of low-latency transmission strategies in software defined networking," *Computer Science Review*, vol. 40, Article ID 100386, 2021.

[29] G. Li, X. Wang, and Z. Zhang, "SDN-based load balancing scheme for multi-controller deployment," *IEEE Access*, vol. 7, Article ID 39612, 2019.

[30] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: scaling flow management for high-performance networks," *ACM SIGCOMM - Computer Communication Review*, vol. 41, no. 4, pp. 254–265, 2011.

[31] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with DIFANE," *ACM SIGCOMM - Computer Communication Review*, vol. 40, no. 4, pp. 351–362, 2010.

[32] P. Phaal, "Traffic Monitoring using sFlow," vol. 5, 2003, http://www.sflow.org/sFlowOverview.pdf.

[33] A. Favaro and E. P. Ribeiro, "Reducing SDN/OpenFlow control plane overhead with blackhole mechanism," in *Proceedings of the 2015 Global Information Infrastructure and Networking Symposium (GIIS)*, pp. 22–25, Guadalajara, Mexico, October 2015.

[34] D. Kotani and Y. Okabe, "A Packet-In message filtering mechanism for protection of control plane in OpenFlow switches," *IEICE - Transactions on Info and Systems*, vol. E99D, no. 3, pp. 695–707, 2016.

[35] A. A. Pranata, T. S. Jun, and D. S. Kim, "Overhead reduction scheme for SDN-based data center networks," *Computer Standards and Interfaces*, vol. 63, pp. 1–15, 2019.

[36] B. Isyaku, K. A. Bakar, M. S. M. Zahid, and M. Nura Yusuf, "Adaptive and hybrid idle–hard timeout allocation and flow eviction mechanism considering traffic characteristics," *Electronics*, vol. 9, no. 11, 2020.

[37] M. Casado, M. Freedman, J. Pettit et al., "Rethinking enterprise network control," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1270–1283, 2009.

[38] N. Gude, T. Koponen, J. Pettit et al., "Nox: towards an operating system for networks," *ACM SIGCOMM - Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[39] A. Tootoonchian, M. Casado, and R. Sherwood, "Controller-perf-SDN," 2012, https://www.usenix.org/system/files/conference/hot-ice12/hotice12-final33_0.pdf.

[40] Z. Cai, A. Cox, and E. T. S. Ng, "Maestro: a system for scalable OpenFlow control," vol. 10, 2011.

[41] B. Zhao, J. Zhao, X. Wang, and T. Wolf, "RuleTailor: optimizing flow table updates in OpenFlow switches with rule transformations," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1581–1594, 2019.

[42] C. Wang and H. Y. Youn, "Entry aggregation and early match using hidden Markov model of flow table in SDN," *Sensors*, vol. 19, no. 10, 2019.

[43] A. Panda, W. Zheng, S. Shenker, X. Hu, and A. Krishnamurthy, "Simplifying distributed {SDN} control planes," in *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pp. 329–345, Boston, MA, USA, March 2017.

[44] T. Koponen, M. Casado, N. Gude et al., "Onix A distributed control platform for Large.pdf," in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, vol. 10, Vancouver, BC, Canada, October 2010.

[45] P. Berde, M. Gerola, J. Hart et al., "Software Defined Networking (SDN)," in *Proceedings of the third workshop on Hot*

topics in software defined networking, Illinois, IL, USA, August, 2014.

[46] J. Medved, R. Varga, A. Tkacik, and K. Gray, "OpenDaylight: towards a model-driven SDN controller architecture," in *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, Sydney, NSW, Australia, June 2014.

[47] K. Phemius, M. Bouet, and J. Leguay, "DISCO: distributed multi-domain SDN controllers," in *Proceedings of the 2014 IEEE Network Operations and Management Symposium (NOMS)*, pp. 1–4, Krakow, Poland, May 2014.

[48] N. Katta, H. Zhang, M. Freedman, and J. Rexford, *Ravana: Controller Fault-Tolerance in Software-Defined Networking*, Princeton University, Princeton, NJ, USA, 2015.

[49] Github ryu project team, *Ryu SDN Framework*, RYU project team, 2014, https://osrg.github.io/ryu-book/en/Ryubook.pdf.

[50] J. Shin, T. Kim, B. Lee, and S. Yang, "IRIS-HiSA: highly scalable and available carrier-grade SDN controller cluster," *Mobile Networks and Applications*, vol. 22, no. 5, pp. 894–905, 2017.

[51] Y. Chang, A. Rezaei, B. Vamanan, J. Hasan, S. Rao, and T. N. Vijaykumar, "Hydra: leveraging functional slicing for efficient distributed SDN controllers," in *Proceedings of the 2017 9th International Conference on Communication Systems and Networks, COMSNETS 2017*, pp. 251–258, Bengaluru, India, January 2017.

[52] A. A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. R. Kompella, "ElastiCon: an elastic distributed SDN controller," in *Proceedings of the tenth ACM/IEEE symposium on Architectures for networking and communications systems - ANCS '14*, pp. 17–27, Marina del Rey, CA, USA, October 2014.

[53] C. C. Ho, K. Wang, and Y. H. Hsu, "A fast consensus algorithm for multiple controllers in software-defined networks," in *Proceedings of the 2016 18th International Conference on Advanced Communication Technology (ICACT)*, pp. 112–116, PyeongChang, Korea, January 2016.

[54] M. Aslan and A. Matrawy, "Adaptive consistency for distributed SDN controllers," in *Proceedings of the 2016 17th International Telecommunications Network Strategy and Planning Symposium (Networks)*, no. 2, pp. 150–157, Montreal, QC, Canada, September 2016.

[55] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically centralized? State distribution trade-offs in software defined networks," in *Proceedings of the first workshop on Hot topics in software defined networks - HotSDN '12*, pp. 1–6, Helsinki, Finland, August 2012.

[56] B. Isyaku, K. b. Abu Bakar, F. A. Ghaleb, and M. S. M. Zahid, "Path selection with critical switch-aware for software defined networking," in *Proceedings of the 2021 IEEE Symposium on Wireless Technology and Applications (ISWTA)*, pp. 22–26, Shah Alam, Malaysia, August 2021.

[57] Z. Guo, M. Su, Y. Xu et al., "Improving the performance of load balancing in software-defined networks through load variance-based synchronization," *Computer Networks*, vol. 68, pp. 95–109, 2014.

[58] R. Mahajan and R. Wattenhofer, "On consistent updates in software defined networks," in *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks 2013*, College Park, Maryland, November 2013.

[59] N. P. Katta, J. Rexford, and D. Walker, "Incremental consistent updates," in *Proceedings of the 47th Des. Autom. Conf. (DAC)*, pp. 737–742, Hong Kong, China, August 2013.

[60] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, "Abstractions for network update," *ACM SIGCOMM - Computer Communication Review*, vol. 42, no. 4, pp. 323–334, 2012.

[61] T. Mizrahi, E. Saat, and Y. Moses, "Timed consistent network updates," in *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, Santa Clara, California, June 2015.

[62] A. Panda, C. Scott, A. Ghodsi, T. Koponen, and S. Shenker, "CAP for networks," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pp. 91–96, Hong Kong, China, August 2013.

[63] E. Sakic, F. Sardis, J. W. Guck, and W. Kellerer, "Towards adaptive state consistency in distributed SDN control plane," in *Proceedings of the 2017 IEEE International Conference on Communications (ICC)*, Paris, France, May 2017.

[64] E. Sakic, N. Deric, and W. Kellerer, "MORPH: an adaptive framework for efficient and byzantine Fault-Tolerant SDN control plane," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2158–2174, 2018.

[65] T. D. Nguyen, M. Chiesa, and M. Canini, "Towards decentralized fast consistent updates," in *Proceedings of the 2016 workshop on Applied Networking Research Workshop - ANRW 16*, pp. 19–25, Berlin, Germany, July 2016.

[66] K. T. Forster, R. Mahajan, and R. Wattenhofer, "Consistent updates in software defined networks: on dependencies, loop freedom, and blackholes," in *Proceedings of the 2016 IFIP Networking Conference (IFIP Networking) and Workshops*, pp. 1–9, Vienna, Austria, June 2016.

[67] M. Canini, I. Salem, L. Schiff, E. M. Schiller, and S. Schmid, "Renaissance: a self-stabilizing distributed SDN control plane," in *Proceedings of the 2018 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pp. 233–243, Vienna, Austria, July 2018.

[68] M. Canini, P. Kuznetsov, D. Levin, and S. Schmid, "A distributed and robust SDN control plane for transactional network updates," in *Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM)*, vol. 26, pp. 190–198, Hong Kong, China, April 2015.

[69] B. Zhou, C. Wu, W. Gao, X. Hong, M. Jiang, and S. Chen, "Achieving consistence for cross-domain WAN control in Software-Defined Networks," *China Communications*, vol. 12, no. 10, pp. 136–146, 2015.

[70] T. Mizrahi and Y. Moses, "On the necessity of time-based updates in SDN," 2014, http://tx.technion.ac.il/.

[71] T. Mizrahi and Y. Moses, "ReversePTP: a clock synchronization scheme for software-defined networks," *International Journal of Network Management*, vol. 26, no. 5, pp. 355–372, Sep 2016.

[72] T. Mizrahi and Y. Moses, "Time capability in NETCONF," 2016, http://www.rfc-editor.org/info/rfc7758.

[73] F. Bannour, S. Souihi, and A. Mellouk, "Software-defined networking: a self-adaptive consistency model for distributed SDN controllers," 2017, https://hal.inria.fr/hal-01558074.

[74] W. Zhou, D. Jin, J. Croft, M. Caesar, P. B. Godfrey, and I. Nsdi, "Enforcing Customizable Consistency Properties in Software-Defined Networks," in *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, Oakland, CA, USA, May 2015.

[75] S. Luo, H. Yu, L. Luo, and L. Li, "Customizable network update planning in SDN," *Journal of Network and Computer Applications*, vol. 141, pp. 104–115, 2019.

[76] M. Aslan and A. Matrawy, "A clustering-based consistency adaptation strategy for distributed SDN controllers," in

Proceedings of the 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), pp. 257–261, Montreal, QC, Canada, June 2018.

[77] B. Zhang, X. Wang, and M. Huang, "Adaptive consistency strategy of multiple controllers in SDN," *IEEE Access*, vol. 6, Article ID 78640, 2018.

[78] C.-Y. Hong, S. Kandula, R. Mahajan, and M. Zhang, "Achieving high utilization with software-driven WAN," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, Hong Kong, China, August 2013.

[79] H. Liu, Xi. Wu, and M. Zhang, "zUpdate: updating data center networks with zero loss," *ACM SIGCOMM Computer Communication Review*, vol. 43, 2013.

[80] S. Brandt, K. T. Förster, and R. Wattenhofer, "On consistent migration of flows in SDNs," in *Proceedings of the IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications 2016*, San Francisco, CA, USA, April 2016.

[81] X. Jin, H. H. Liu, R. Gandhi et al., "Dynamic scheduling of network updates," *ACM SIGCOMM - Computer Communication Review*, vol. 44, no. 4, pp. 539–550, 2015.

[82] J. Zheng, G. Chen, S. Schmid, H. Dai, and J. Wu, "Chronus: consistent data plane updates in timed SDNs," in *Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 319–327, Atlanta, GA, USA, June 2017.

[83] J. Zheng, G. Chen, S. Schmid, H. Dai, J. Wu, and Q. Ni, "Scheduling congestion-and loop-free network update in timed SDNs," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2542–2552, 2017.

[84] S. A. Amiri, S. Dudycz, S. Schmid, and S. Wiederrecht, "Congestion-free rerouting of flows on DAGs," 2016, http://arxiv.org/abs/1611.09296.

[85] F. A. Botelho, F. M. V. Ramos, D. Kreutz, and A. N. Bessani, "On the feasibility of a consistent and fault-tolerant data store for SDNs," in *Proceedings of the 2013 Second European Workshop on Software Defined Networks*, pp. 38–43, Berlin, Germany, October 2013.

[86] F. Botelho, T. A. Ribeiro, P. Ferreira, F. M. V. Ramos, and A. Bessani, "Design and implementation of a consistent data store for a distributed SDN control plane," in *Proceedings of the 2016 12th Eur. Dependable Comput. Conf. EDCC 2016*, pp. 169–180, Gothenburg, Sweden, September 2016.

[87] S. K. Fayazbakhsh, V. Sekar, M. Yu, and J. Mogul, "FlowTags: enforcing network-wide policies in the presence of dynamic middlebox actions," in *Proceedings of the 2013 ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, Hong Kong, China, August 2013.

[88] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-Fying middlebox policy enforcement using SDN," in *Proceedings of the ACM SIGCOMM 2013 Conference Applications, Technologies, Architectures, and Protocols for Computer Communication*, Hong Kong, China, August 2013.

[89] M. Canini, D. De Cicco, P. Kuznetsov, D. Levin, S. Schmid, and S. Vissicchio, "STN: a robust and distributed SDN control plane," 2014, https://www.usenix.org/sites/default/files/ons2014-poster-canini.pdf.

[90] L. Schiff, S. Schmid, P. Kuznetsov, K. Argyraki, S. Schmid, and P. Kuznetsov, "In-band synchronization for distributed SDN control planes," *ACM SIGCOMM - Computer Communication Review*, vol. 46, no. 1, pp. 37–43, 2016.

[91] N. McKeown, T. Anderson, H. Balakrishnan et al., "OpenFlow," *ACM SIGCOMM - Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[92] S. Vissicchio, L. Vanbever, L. Cittadini, G. G. Xie, and O. Bonaventure, "Safe update of hybrid SDN networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1649–1662, 2017.

[93] S. Vissicchio and L. Cittadini, "Safe, efficient, and robust SDN updates by combining rule replacements and additions," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 3102–3115, 2017.

[94] J. Mcclurg, H. Hojjat, P. Cerny, and N. Foster, "Efficient synthesis of network updates," in *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, Portland, OR, USA, June 2015.

[95] K. Poularakis, Q. Qin, L. Ma, S. Kompella, K. K. Leung, and L. Tassiulas, "Learning the optimal synchronization rates in distributed SDN control architectures," in *Proceedings of the IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 1099–1107, Paris, France, April 2019.

[96] Z. Zhang, L. Ma, K. Poularakis, K. K. Leung, J. Tucker, and A. Swami, "MACS: deep reinforcement learning based SDN controller synchronization policy design," in *Proceedings of the 2019 IEEE 27th International Conference on Network Protocols (ICNP)*, Chicago, IL, USA, October 2019.

[97] Z. Zhang, L. Ma, K. Poularakis, K. K. Leung, L. Wu, and D. Q. Scheduler, "DQ scheduler: deep reinforcement learning based controller synchronization in distributed SDN," in *Proceedings of the 2019 IEEE International Conference on Communications (ICC)*, Shanghai, China, May 2019.

[98] L. V. Morales, A. F. Murillo, and S. J. Rueda, "Extending the floodlight controller," in *Proceedings of the 2015 IEEE 14th International Symposium on Network Computing and Applications*, pp. 126–133, Cambridge, MA, USA, September 2015.

[99] F. Benamrane, M. Ben mamoun, and R. Benaini, "An East-West interface for distributed SDN control plane: implementation and evaluation," *Computers and Electrical Engineering*, vol. 57, pp. 162–175, 2017.

[100] E. A. Adedokun and A. Adekale, "Development of a modified East-West interface for distributed control plane network," *Arid Zone Journal of Engineering, Technology and Environment*, vol. 15, pp. 242–254, 2019.

[101] P. Lin, J. Bi, and Y. Wang, "East-West bridge for SDN network peering," *Frontiers in Internet Technologies*, vol. 401, pp. 170–181, 2013.

[102] P. Lin, J. Bi, S. Wolff et al., "A west-east bridge based SDN inter-domain testbed," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 190–197, 2015.

[103] A. Dixit, K. Kogan, and P. Eugster, "Composing heterogeneous SDN controllers with flowbricks," in *Proceedings of the 2014 IEEE 22nd International Conference on Network Protocols*, pp. 287–292, Raleigh, NC, USA, October 2014.

[104] H. Yu, K. Li, H. Qi, W. Li, and X. Tao, "Zebra: an east-west control framework for SDN controllers," in *Proceedings of the 2015 44th International Conference on Parallel Processing*, pp. 610–618, Beijing, China, September 2015.

[105] H. Qi and K. Li, "Management system of heterogeneous software-defined networking controllers," *Software Defined Networking Applications in Distributed Datacenters. SpringerBriefs in Electrical and Computer Engineering*, Springer, Manhattan, NY, USA, 2016.

[106] H. Yu, H. Qi, and K. Li, "WECAN: an efficient west-east control associated network for large-scale SDN systems," *Mobile Networks and Applications*, vol. 25, no. 1, pp. 114–124, 2020.

[107] B. Almadani, A. Beg, and A. Mahmoud, "DSF: a distributed SDN control plane framework for the east/West Interface," *IEEE Access*, vol. 9, Article ID 26735, 2021.

[108] H. Hu, Z. Wang, G. Cheng, and J. Wu, "MNOS: a mimic network operating system for software defined networks," *IET Information Security*, vol. 11, no. 6, pp. 345–355, 2017.

[109] C. Qi, J. Wu, H. Hu et al., "An intensive security architecture with multi-controller for SDN," in *Proceedings of the 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, vol. 14, no. 9, pp. 401-402, San Francisco, CA, USA, April 2016.

[110] N. T. Hoang, H. N. Nguyen, H. A. Tran, and S. Souihi, "A novel adaptive east–West Interface for a heterogeneous and distributed SDN network," *Electronics*, vol. 11, no. 7, 2022.

[111] J. Moeyersons, P. J. Maenhaut, F. Turck, and B. Volckaert, "Pluggable SDN framework for managing heterogeneous SDN networks," *International Journal of Network Management*, vol. 30, no. 2, pp. 1–20, 2020.

[112] N. Perrot and T. Reynaud, "Optimal placement of controllers in a resilient SDN architecture," in *Proceedings of the 2016 12th International Conference on the Design of Reliable Communication Networks (DRCN)*, pp. 145–151, Paris, France, March 2016.

[113] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "BalanceFlow: controller load balancing for OpenFlow networks," in *Proceedings of the 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, vol. 2, pp. 780–785, Hangzhou, China, October 2013.

[114] M. F. Bari, R. A. Raton, C. S. Rahman et al., "Dynamic controller provisioning in software defined networks," in *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, pp. 18–25, Zurich, Switzerland, October 2013.

[115] Z. Su and M. Hamdi, "MDCP: measurement-aware distributed controller placement for software defined networks," in *Proceedings of the 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 380–387, Melbourne, VIC, Australia, December 2015.

[116] H. Selvi, G. Gur, and F. Alagoz, "Cooperative load balancing for hierarchical SDN controllers," in *Proceedings of the 2016 IEEE 17th International Conference on High Performance Switching and Routing (HPSR)*, pp. 100–105, Yokohama, Japan, June 2016.

[117] G. Ishigaki and N. Shinomiya, "Controller placement algorithm to alleviate burdens on communication nodes," in *Proceedings of the 2016 International Conference on Computing, Networking and Communications (ICNC)*, Kauai, HI, USA, February 2016.

[118] H. Sufiev and Y. Haddad, "A dynamic load balancing architecture for SDN," in *Proceedings of the 2016 IEEE International Conference on the Science of Electrical Engineering (ICSEE)*, pp. 1–3, Eilat, Israel, November 2017.

[119] G. Cheng, H. Chen, Z. Wang, and S. Chen, "DHA: distributed decisions on the switch migration toward a scalable SDN control plane," in *Proceedings of the 2015 IFIP Networking Conference (IFIP Networking)*, pp. 1–9, Toulouse, France, May 2015.

[120] C. Wang, B. Hu, S. Chen, D. Li, and B. Liu, "A switch migration-based decision-making scheme for balancing load in SDN," *IEEE Access*, vol. 5, pp. 4537–4544, 2017.

[121] Y. Wang, J. Yu, K. Pei, and X. Qiu, "A load informing based load balancing mechanism for multiple controllers in SDN," *Dianzi Yu Xinxi Xuebao/Journal of Electronics and Information Technology*, vol. 39, no. 11, pp. 2733–2740, 2017.

[122] Y. Zhou, K. Zheng, W. Ni, and R. P. Liu, "Elastic switch migration for control plane load balancing in SDN," *IEEE Access*, vol. 6, pp. 3909–3919, 2018.

[123] S. K. Tarai and S. Shailendra, "Optimal and secure controller placement in SDN based smart city network," in *Proceedings of the 2019 International Conference on Information Networking (ICOIN)*, pp. 254–261, Kuala Lumpur, Malaysia, January 2019.

[124] K. S. Sahoo, D. Puthal, M. Tiwary et al., "ESMLB: efficient switch migration-based load balancing for multicontroller SDN in IoT," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5852–5860, 2020.

[125] H. Chen, G. Cheng, and Z. Wang, "A game-theoretic approach to elastic control in software-defined networking," *China Commun*, vol. 13, no. 5, pp. 103–109, 2016.

[126] J. Cui, Q. Lu, H. Zhong, M. Tian, and L. Liu, "A load-balancing mechanism for distributed SDN control plane using response time," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1197–1206, 2018.

[127] A. Mahjoubi, O. Zeynalpour, B. Eslami, and N. Yazdani, "LBFT: load balancing and fault tolerance in distributed controllers," in *Proceedings of the 2019 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–6, Istanbul, Turkey, June 2019.

[128] W. H. F. Aly, "Generic controller adaptive load balancing (GCALB) for SDN networks," *Journal of Computer Networks and Communications*, vol. 2019, Article ID 6808693, 9 pages, 2019.

[129] M. T. Melo, S. Nickel, and F. Saldanha-da-Gama, "Facility location and supply chain management - a review," *European Journal of Operational Research*, vol. 196, no. 2, pp. 401–412, 2009.

[130] B. M. M. Alom, S. Das, and M. A. Rouf, "Performance evaluation of vertex cover and set cover problem using optimal algorithm," Dhaka University of Engineering and Technology (DUET), Gazipur, Bangladesh, Dept. of CSE, 2011.

[131] Q. Zhong, Y. Wang, W. Li, and X. Qiu, "A min-cover based controller placement approach to build reliable control network in SDN," in *Proceedings of the NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pp. 481–487, Istanbul, Turkey, April 2016.

[132] A. E. Monge and C. P. Elkan, "The field matching problem: algorithms and applications," in *Proceedings of the Second Int. Conf. Knowl. Discov. Data Min*, pp. 267–270, Portland, Oregon, August 1996.

[133] T. Yuan, X. Huang, M. Ma, and J. Yuan, "Balance-based SDN controller placement and assignment with minimum weight matching," in *Proceedings of the 2018 IEEE International Conference on Communications (ICC)*, Kansas City, MO, USA, May 2018.

[134] Y. Liu, H. Gu, F. Yan, X. Yu, and K. Wang, "Multi-controller placement based on two-sided matching in inter-datacenter elastic optical networks," in *Proceedings of the ICC 2020 IEEE International Conference on Communications (ICC)*, Dublin, Ireland, June 2020.

[135] D. G. Corneil and L. K. Stewart, "Dominating sets in perfect graphs," *Topics on Domination*, vol. 48, pp. 145–164, 1991.

[136] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "On reliability-optimized controller placement for Software-Defined Networks," *China Communications*, vol. 11, no. 2, pp. 38–54, 2014.

[137] IBM, *Getting Started with CPLEX for MATLAB*, IBM, Armonk, NY, USA, 2013.

[138] G. Wang, Y. Zhao, J. Huang, and Y. Wu, "An effective approach to controller placement in software defined wide area networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 344–355, 2018.

[139] B. Isyaku, K. Bin Abu Bakar, M. N. Yusuf, and M. S. Mohd Zahid, "Software defined networking failure recovery with flow table aware and flows classification," in *Proceedings of the 2021 IEEE 11th IEEE Symposium on Computer Applications and Industrial Electronics (ISCAIE)*, pp. 337–342, Penang, Malaysia, April 2021.

[140] Y. N. Hu, W. D. Wang, X. Y. Gong, X. R. Que, and S. D. Cheng, "On the placement of controllers in software-defined networks," *The Journal of China Universities of Posts and Telecommunications*, vol. 19, pp. 92–171, 2012.

[141] L. F. Muller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, and M. P. Barcellos, "Survivor: an enhanced controller placement strategy for improving SDN survivability," in *Proceedings of the 2014 IEEE Global Communications Conference*, pp. 1909–1915, Austin, TX, USA, December 2014.

[142] M. Guo and P. Bhattacharya, "Controller placement for improving resilience of software-defined networks," in *Proceedings of the 2013 Fourth International Conference on Networking and Distributed Computing*, pp. 23–27, Los Angeles, CA, USA, December 2013.

[143] P. Vizarreta, C. M. MacHuca, and W. Kellerer, "Controller placement strategies for a resilient SDN control plane," in *Proceedings of the 2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pp. 253–259, Halmstad, Sweden, September 2016.

[144] S. Guo, S. Yang, Q. Li, and Y. Jiang, "Towards controller placement for robust software-defined networks," in *Proceedings of the 2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*, Nanjing, China, December 2015.

[145] J. Liu, J. Liu, and R. Xie, "Reliability-based controller placement algorithm in software defined networking," *Computer Science and Information Systems*, vol. 13, no. 2, pp. 547–560, 2016.

[146] S. Moazzeni, M. R. Khayyambashi, N. Movahhedinia, and F. Callegati, "On reliability improvement of Software-Defined Networks," *Computer Networks*, vol. 133, pp. 195–211, 2018.

[147] M. Tanha, D. Sajjadi, and J. Pan, "Enduring node failures through resilient controller placement for software defined networks," in *Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, Washington, DC, USA, December 2016.

[148] V. Sridharan, M. Gurusamy, and T. Truong-Huu, "On multiple controller mapping in software defined networks with resilience constraints," *IEEE Communications Letters*, vol. 21, no. 8, pp. 1763–1766, 2017.

[149] P. M. Mohan, T. Truong-Huu, and M. Gurusamy, "Primary-backup controller mapping for Byzantine Fault tolerance in software defined networks," in *Proceedings of the GLOBECOM 2017 IEEE Global Communications Conference*, pp. 1–7, Singapore, December 2017.

[150] B. P. R. Killi and S. V. Rao, "Optimal model for failure foresight capacitated controller placement in software-defined networks," *IEEE Communications Letters*, vol. 20, no. 6, pp. 1108–1111, 2016.

[151] B. P. R. Killi and S. V. Rao, "Link failure aware capacitated controller placement in software defined networks," in *Proceedings of the 2018 International Conference on Information Networking (ICOIN)*, pp. 292–297, Chiang Mai, Thailand, January 2018.

[152] B. P. R. Killi and S. V. Rao, "Capacitated next controller placement in software defined networks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 514–527, 2017.

[153] M. Tanha, D. Sajjadi, R. Ruby, and J. Pan, "Capacity-aware and delay-guaranteed resilient controller placement for software-defined WANs," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 991–1005, 2018.

[154] T. Hu, Z. Guo, J. Zhang, J. Lan, and A. C. C. Failure, "Adaptive slave controller assignment for Fault-tolerant control plane in software-defined networking," in *Proceedings of the 2018 IEEE International Conference on Communications (ICC)*, Kansas City, MO, USA, May 2018.

[155] G. Wu, L. Wang, Z. Xu, L. Yao, M. S. Obaidat, and A. M. Sdn, "A QoS and Cost Aware Fault Tolerant Scheme Insult-Controller SDNs," in *Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, UAE, December 2018.

[156] F. Bannour, S. Souihi, and A. Mellouk, "Scalability and reliability aware SDN controller placement strategies," in *Proceedings of the 2017 13th International Conference on Network and Service Management (CNSM)*, pp. 1–4, Tokyo, Japan, November 2017.

[157] Y. Jiménez, C. Cervelló-Pastor, and A. J. García, "On the controller placement for designing a distributed SDN control layer," in *Proceedings of the 2014 IFIP Networking Conference*, Trondheim, Norway, June 2014.

[158] W. Ren, Y. Sun, H. Luo, and M. Guizani, "A novel control plane optimization strategy for important nodes in SDN-IoT networks," *IEEE Internet of Things Journal*, vol. 6, 2018.

[159] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks," *Computer Networks*, vol. 71, pp. 1–30, 2014.

[160] Y. Xu, M. Cello, I. C. Wang et al., "Dynamic switch migration in distributed software-defined networks to achieve controller load balance," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 515–529, 2019.

[161] Z. Li, Y. Hu, T. Hu, and P. Wei, "Dynamic SDN controller association mechanism based on flow characteristics," *IEEE Access*, vol. 7, Article ID 92661, 2019.

[162] K. S. Sahoo and B. Sahoo, "CAMD: a switch migration based load balancing framework for software defined networks," *IET Networks*, vol. 8, no. 4, pp. 264–271, 2019.

[163] M. Priyadarsini, J. C. Mukherjee, P. Bera, S. Kumar, A. H. M. Jakaria, and M. A. Rahman, "An adaptive load balancing scheme for software-defined network controllers," *Computer Networks*, vol. 164, Article ID 106918, 2019.

[164] S. Auroux and H. Karl, "Flow processing-aware controller placement in wireless DenseNets," in *Proceedings of the 2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, pp. 1294–1299, Washington, DC, USA, September 2014.

[165] S. Auroux and H. Karl, "Efficient flow processing-aware controller placement in future wireless networks," in *Proceedings of the 2015 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1787–1792, New Orleans, LA, USA, March 2015.

[166] S. Auroux and H. Karl, "Flexible reassignment of flow processing-aware controllers in future wireless networks," in *Proceedings of the 2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio*

Communications (PIMRC), pp. 1850–1855, Hong Kong, China, August 2015.

[167] A. Ksentini, M. Bagaa, and T. Taleb, "On using SDN in 5G: the controller placement problem," in *Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM)*, Washington, DC, USA, December 2016.

[168] M. J. Abdel-Rahman, E. A. Mazied, A. MacKenzie, S. Midkiff, M. R. Rizk, and M. El-Nainay, "On stochastic controller placement in software-defined wireless networks," in *Proceedings of the 2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, San Francisco, CA, USA, March 2017.

[169] M. J. Abdel-Rahman, E. A. Mazied, K. Teague, A. B. MacKenzie, and S. F. Midkiff, "Robust controller placement and assignment in software-defined cellular networks," in *Proceedings of the 2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–9, Vancouver, BC, Canada, July 2017.

[170] M. Johnston and E. Modiano, "Controller placement for maximum throughput under delayed CSI," in *Proceedings of the 2015 13th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pp. 521–528, Cambridge, MA, USA, 2015.

[171] M. Johnston and E. Modiano, "Controller placement in wireless networks with delayed CSI," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1775–1788, 2017.

[172] Y. Zhang, Y. Zhu, F. Yan, W. Xia, and L. Shen, "Energy-efficient radio resource allocation in software-defined wireless sensor networks," *IET Communications*, vol. 12, no. 3, pp. 349–358, 2018.

[173] C. C. Lin, H. H. Chin, and W. B. Chen, "Balancing latency and cost in software-defined vehicular networks using genetic algorithm," *Journal of Network and Computer Applications*, vol. 116, pp. 35–41, 2018.

[174] A. Llorens-Carrodeguas, C. Cervello-Pastor, I. Leyva-Pupo, J. M. Lopez-Soler, J. Navarro-Ortiz, and J. A. Exposito-Arenas, "An architecture for the 5G control plane based on SDN and data distribution service," in *Proceedings of the 2018 5th International Conference on Software Defined Systems, SDS 2018*, pp. 105–111, Barcelona, Spain, April 2018.

[175] F. Sallabi, A. Ain, and A. Ain, "Managing IoT-based smart healthcare systems traffic with software defined networks," in *Proceedings of the 2018 International Symposium on Networks, Computers and Communications (ISNCC)*, vol. 31, pp. 1–6, Rome, Italy, June, 2018.

[176] M. Cicioglu and A. Calhan, "SDN-enabled wireless body area networks," in *Proceedings of the 2018 6th International Conference on Control Engineering and Information Technology CEIT 2018*, pp. 25–27, Istanbul, Turkey, October 2018.

[177] M. Cicioğlu and A. Çalhan, "HUBsFLOW: a novel interface protocol for SDN-enabled WBANs," *Computer Networks*, vol. 160, pp. 105–117, 2019.

[178] A. Dvir, Y. Haddad, and A. Zilberman, "Wireless controller placement problem," in *Proceedings of the 2018 15th IEEE Annual Consumer Communications and Networking Conference (CCNC)*, pp. 1–4, Las Vegas, NV, USA, January 2018.

[179] A. Dvir, Y. Haddad, and A. Zilberman, "The controller placement problem for wireless SDN," *Wireless Networks*, vol. 25, no. 8, pp. 4963–4978, 2019.

[180] K. Hasan, X. W. Wu, K. Biswas, and K. Ahmed, "A novel framework for software defined wireless body area network," in *Proceedings of the 2018 8th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, pp. 114–119, Kuala Lumpur, Malaysia, May 2018.

[181] K. Hasan, K. Ahmed, K. Biswas, M. Saiful Islam, and O. Ameri Sianaki, "Software-defined application-specific traffic management for wireless body area networks," *Future Generation Computer Systems*, vol. 107, pp. 274–285, 2020.

[182] J. Park and W. Yoon, "SDN-based heterogeneous network architecture with Multi-Controllers," in *Proceedings of the 2020 22nd International Conference on Advanced Communication Technology (ICACT)*, pp. 559–561, Phoenix Park, Korea, February 2020.

[183] C. N. Tadros, M. R. M. Rizk, and B. M. Mokhtar, "Software defined network-based management for enhanced 5G network services," *IEEE Access*, vol. 8, Article ID 53997, 2020.

[184] A. Zilberman, Y. Haddad, S. Erlich, Y. J. Peretz, and A. Dvir, "Heterogeneous SDN controller placement problem—the Wi-Fi and 4G LTE-U case," *Computer Networks*, vol. 198, Article ID 108376, 2021.

[185] S. K. Routray and K. P. Sharmila, "Software defined networking for 5G," in *Proceedings of the 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pp. 4–8, Coimbatore, India, January 2017.

[186] M. A. S. Santos, B. A. A. Nunes, K. Obraczka, and T. Turletti, "Decentralizing SDN 's control plane," in *Proceedings of the 39th Annual IEEE Conference on Local Computer Networks*, pp. 402–405, Edmonton, AB, Canada, September 2014.

[187] H. Li, P. Li, S. Guo, and A. Nayak, "Byzantine-resilient secure software-defined networks with multiple controllers in cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 4, pp. 436–447, 2014.

[188] P. M. Mohan, T. Truong-Huu, and M. Gurusamy, "Byzantine-resilient controller mapping and remapping in software defined networks," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2714–2729, 2020.