Multidisciplinary | Rapid Review | Open Access Journal

Digital Object Identifier 10.1109/ACCESS. 2017.Doi Number

Multi-controller Based Software-Defined Networking: A Survey

Tao Hu[1], Zehua Guo[2], Thar Baker[3], Julong Lan[1]

[1]National Digital Switching System Engineering R&D Center, Zhengzhou 450002, China
[2]Didi Chuxing, Beijing 100193, China
[3]Liverpool John Moores University, Liverpool 25175, UK

Corresponding author: Zehua Guo (e-mail: guolizihao@hotmail.com).

This work is supported by the Project of National Network Cyberspace Security (Grant No. 2017YFB0803204), the National High-Tech Research and Development Program of China (863 Program) (Grant No. 2015AA016102), Foundation for Innovative Research Group of the National Natural Science Foundation of China (Grant No.61521003).

ABSTRACT Software-Defined Networking (SDN) is a novel network paradigm that enables flexible management for networks. As the network size increases, the single centralized controller cannot meet the increasing demand for flow processing. Thus, the promising solution for SDN with large-scale networks is the multi-controller. In this paper, we present a compressive survey for multi-controller research in SDN. First, we introduce the overview of multi-controller, including the origin of multi-controller and its challenges. Then, we classify multi-controller research into four aspects (scalability, consistency, reliability, load balancing) depending on the process of implementing the multi-controller. Finally, we propose some relevant research issues to deal with in the future and conclude the multi-controller research.

INDEX TERMS Software-Defined Networking, multi-controller, scalability, consistency, reliability, load balancing.

I. INTRODUCTION

The Internet has been identified as an essential infrastructure that supports social development and technological progress in the past 30 years, and it has profoundly changed the people's working, studying and living styles [1] [2]. However, traditional network technology has inherent defects of rigid structure and complex configuration and cannot meet the requirement of network innovation [3]. Thus, it is deemed urgent to design and develop a new network architecture that can dynamically and flexibly manage the network [4].

Software-Defined Networking (SDN) [5-7] is proposed to overcome the aforementioned weaknesses of the traditional network. As a new network paradigm, the SDN revolutionizes network technology by breaking the fundamental idea of traditional networks. An SDN comprises three layers: data plane, control plane, and application plane. Data plane comprises of network devices (e.g., a router, and switch) and forwards packets according to a decision made by the control plane. Control plane acts as a mediator for the data plane and the application plane and handles the traffic flow in the network. Application plane is on the top of the control plane and achieves customized application logic (e.g., intrusion detection systems [8], big data analyses [6]).

The preliminary design of the control plane only uses one controller for a network. Though the advantages of centralized control in SDN network, SDN faces some problems challenging its nature (i.e., centralized control) due to day-to-day increasing network demands. Further, network operators try their best to strengthen the performance of the network controller, but it is still hard to meet the high demands (e.g., flow request sent by switches and network statistics) due to the limited capacity of the single controller. For instance, Ryu [9], as the early controller, can server only 6000 Packet-in requests per second with an average latency less than (6ms). Particularly, this deficiency presents more obviously in the large-scale network. Moreover, the single point of failure is also the crucial factor in the one controller SDN network. The controller failure will cause disconnections between the controller and the switches. Since the controller software runs on a server and it may suffer from the hardware or software failure, characterization of a server failure in a production network or cluster gives us the description of the controller failure [10]. Therefore, the controller failure is

common in the network because of hardware or software breakdown [11]. In a word, the above problems triggered by the single controller will hinder the deployment of SDN in actual production networks. To overcome those issues, several works propose using multi-controller working together to achieve the function of the logically centralized controller [12-14]. There are some surveys of SDN, but they have different concentrations. For example, [15] and [16] introduce a comprehensive literature survey on SDN, including the motivation, architecture and an overview of three layers; [17-19] survey SDN network update, testbeds, and security architecture, respectively. The authors in [17] focus on the control plane scalability of multi-controller; and the work is deemed closer to our work.

In this paper, we focus on the survey of multi-controller research in SDN. We discuss the multi-controller overview in SDN and present the SDN issues of multi-controller: *scalability*, *consistency*, *reliability,* and *load balancing*. Following the design logic, we first present the scalability research of multi-controller to cope with single controller problem (single point of failure, limited control resources, etc.). Moreover, we present consistency, reliability, and load balancing research caused by multi-controller. Further, we propose some promising research directions as future work. Finally, we summarize this paper in the conclusion. To the best of our knowledge, our work is the first comprehensive survey for multi-controller research in SDN from the perspective of design logic. The main contributions of this paper are summarized below:

- We present the controller evolution that discusses the origin of multi-controller; and we introduce the two basic multi-controller architectures.
- We summarize the four challenges (i.e., scalability, consistency, reliability and load balancing) in the multi-controller research, and present existing solutions.
- We introduce major research problems that need to be considered to implement multi-controller in real scenarios.

The rest of this paper is organized as follows. Section 2 presents the overview of multi-controller. From Section 3 to Section 6, we discuss the research challenges of multi-controller scalability, consistency, reliability and load balancing, respectively. In Section 7, we discuss the promising research directions and issues to deal with in the future. In Section 8, we conclude this survey.

II. MULTI-CONTROLLER OVERVIEW

A. CONTROLLER EVOLUTION

In this subsection, we will firstly introduce the origin of multi-controller by using two examples, then illustrate the two basic multi-controller architectures that are flat design and hierarchical design.

1) FROM SINGLE CONTROLLER TO MULTI-CONTROLLER

In the initial stage of SDN design, a single controller manages the entire network. In Fig. 1, the controller (c1) manages four switches in the network. When the source host sends a new packet to switch (s1), the switch cannot achieve the forwarding function due to the lack of routing information of the new packet. Then, the switch (s1) sends (Packet-in) messages to the controller (c1) to get the routing for the new packet. After getting the response message from the controller, the switch forwards the packet to the next device. Finally, the packet reaches the destination host successfully. The controller plays a major role in the process of traffic transmission. Unfortunately, as the network traffic increases fast, one single controller cannot deal with a great number of flow requests send from switches because of the limited controller capacity. Meanwhile, once the single controller fails, the switches cannot plan the routing for the newly arrived packet, which affects the communication and applications of the network. Consequently, it is necessary to propose a modern controller design.

Benefiting from the development of OpenFlow (e.g., OpenFlow 1.2 [20] has proposed the concept of the master, slave, and equal controller, and one switch could connect one master controller and several slaves or equal controllers), multi-controller becomes a new SDN design scheme, which could solve the problem caused by the single controller. In Fig. 2, there are two controllers in the network topology, and each of them manages the part of the network. In this scenario, (c1) and (c2) are sharing the same logic in a logically centralized manner such that when new packets arrive at (s1), both (c1) and (c2) can directly install forwarding paths in all corresponding switches. By this means, it can effectively alleviate the flow processing pressure of a single controller. Meanwhile, these two controllers are backup each other, which could resolve the single point of failure for the controller.

Based on the above analysis, we can discover that compared with a single controller, a multi-controller design can effectively improve the performance of SDN network. Therefore, multi-controller gradually becomes a popular research in the recent years.

2) TWO BASIC MULTI-CONTROLLER ARCHITECTURES

When placing multi-controller, the key point is how to design the multi-controller architecture. After surveying the literature, we conclude that the basic multi-controller architecture can be divided into flat design and hierarchical design. In flat design, a network is structured into several domains, where each domain is controlled by a controller situated within its own local network view. Controllers communicate with others through their east-westbound interfaces to get the global view of the network. Fig. 3 shows the flat design of multi-controller. Typical examples are HyperFlow [21] and Onix [22].

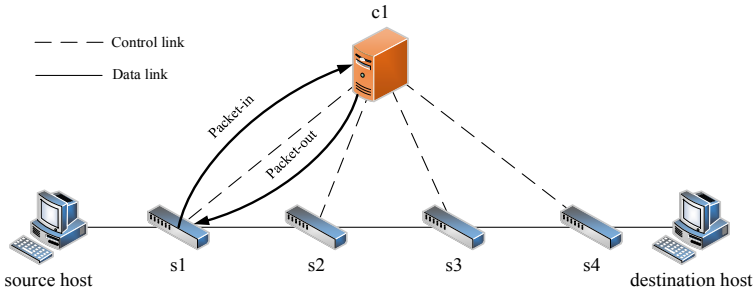

FIGURE 1. An example of single controller works in routing packets

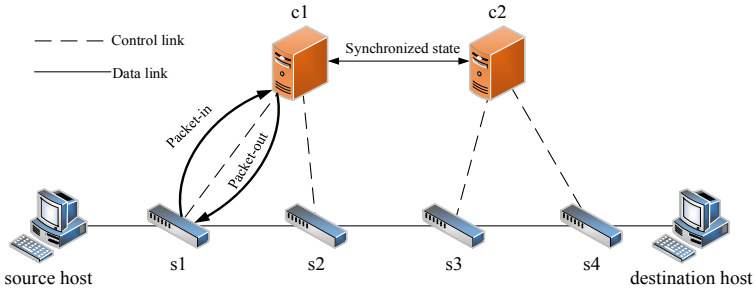

FIGURE 2. An example of multi-controller works in routing packets

HyperFlow is designed on Network Operating System (NOX) for the distributed file system WheelFs [23]. In HyperFlow, each controller only processes flow requests sent from the switches in its local domain. Network events (e.g., flow information, routing information) are transmitted based on specific "publish/subscribe" mode [24] among controllers.

Onix adopts the distributed architecture to offer the programmatic interface for the upper control logic and uses Network Information Base (NIB) to maintain the global network state. Onix gets network status from physical infrastructure and conducts operation from the control logic via the connectivity infrastructure.

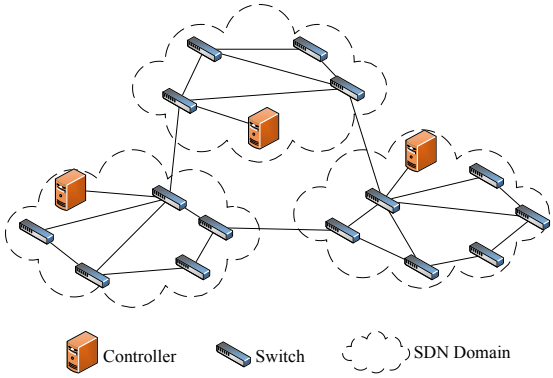

FIGURE 3. Flat design of multi-controller

The flat design extends the capability of the control plane, but it also requires complicated controller management and extra control overheads. For example, the controllers must frequently communicate with each other to guarantee the consistent network view. The hierarchical design is proposed to solve the problems.

Hierarchical design usually uses two-layer controllers: domain controller, which manages switches in its local domains and runs local control applications, and root controller, which manages domain controllers and maintains a global network view. Kandoo [25] is a typical hierarchical controller structure. In the Kandoo, the root controller communicates with domain controllers to get the domain information, but the domain controllers do not contact with each other. Fig. 4 shows the basic architecture of hierarchical design.

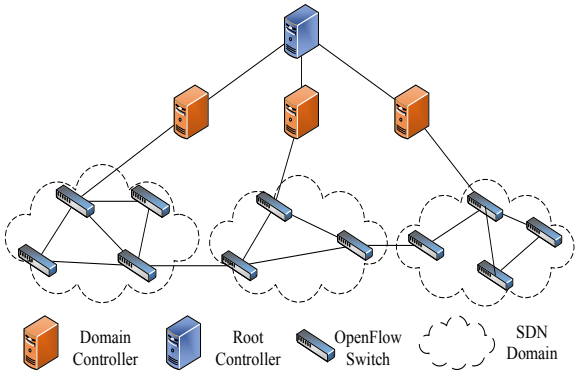

FIGURE 4. Hierarchical design of multi-controller

B. RESEARCH CHALLENGES

The design of the multi-controller has solved the problems encountered by a single controller, but it also presents a set of overlooked challenges. Fig. 5 summarizes the research

roadmap of multi-controller challenges. One of the most critical challenges in multi-controller is the way to cope with its scalability problem. Therefore, the researchers introduce to place multi-controller in the SDN network. However, how to place those controllers to solve scalability is still an outstanding challenge, which includes two layers of meaning: one refers to finding controller locations, the other allocates the switches for different controllers. Further, once there is a multi-controller in the network, the consequent results bring about the challenges of consistency, reliability and load balancing. Though different controllers manage the respective SDN domains, they must maintain consistent network views. It is necessary to guarantee the consistency of the multi-controller. Meanwhile, different types and locations of controllers may suffer from the indeterminate failure and indeterminate attack, which influence the reliability of the control plane. Besides, unbalanced distribution of controller loads will degrade the network performance, and how to balance multi-controllers' loads is also a key point of multi-controller research.

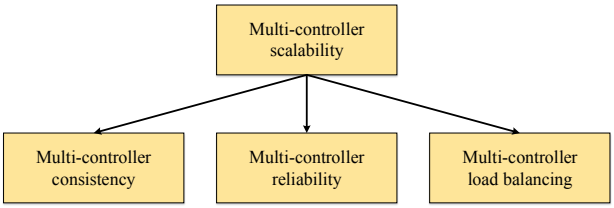

FIGURE 5. The research of multi-controller challenges

III. MULTI-CONTROLLER SCALABILITY

Based on the two basic multi-controller architectures, the proposed for multi-controller is to overcome the shortages of the single controller, such as single controller failure and limited controller capacity. However, multi-controller also raises the challenges of scalability: how to select the controller locations and how to allocate the switches for multi-controller in the network. In fact, the multi-controller's scalability depends on the number of controllers and the deployment mode. If controllers are irrationally deployed, it could assign unbalanced processing load on controllers and lower the control plane's capacity. The coarse-grained domain partition could also make it difficult to guarantee the agreeable effect of scalability. After analysis, we categorize existing solutions in two aspects: (1) controller placement; (2) domain partition, as shown in Fig. 6. Controller placement focuses on selecting appropriate locations to improve the scalability while domain partition emphasizes on partitioning the entire network into several SDN domains.

A. CONTROLLER PLACEMENT

Placing multi-controller is an effective method to cope with the challenge of scalability, and existing multi-controller placements in [26-30] consider some network parameters (e.g., delay, traffic, distance) to identify the number and locations of controllers in the network.

Controller Placement Problem (CPP): Heller et al [26] firstly improves the scalability of multi-controller for solving CPP. The CPP focuses on two questions: how many controllers are required, and where should they go? The authors conduct experiments on the Internet 2 [31] production deployments and 100 publicly available WAN topologies to examine control plane propagation latency. The results indicate the latency is topology dependent and one controller location is often sufficient to meet existing reaction-time requirements (though certainly not fault tolerance requirements). Unfortunately, the authors have no algorithm design and theoretical demonstration.

Optimal Controller Placement: In [27], the authors present a non-zero-sum game [32] based distributed technique to optimally deploy the multi-controller. With the non-zero-sum game, each controller has an optimization engine, which computes a payoff function and compares its own payoff value to save costs and improve Quality of Service (QoS) through optimizing the locations of controllers.

Bargaining Game: Similarly, in [28], the authors also introduce a game model to study the placement of multi-controller. This model considers multiple metrics: the communication delay between controllers and switches, the communication overhead among controllers, processing loads on controllers. Based on the metrics, the paper formulates an optimization problem with two contradictory objectives: minimizing communication delay and minimizing communication overhead. The authors use a bargaining game to find the optimal placement of controllers to achieve a trade-off between the two objectives.

Mathematical Model: In [29], the authors propose a mathematical model, which simultaneously determines the optimal number, location, and type of controllers in SDN. The model seeks to minimize the controller placement cost of the network while considering different constraints (e.g., controller capacity, path latency). The simulation results demonstrate that the model can be used to plan small-scale SDN. Meanwhile, this model can also be applied to various enterprises and cloud-based networks to start integrating SDN or plan a new SDN. However, this model needs a long-time computation time and huge memory when used in large topologies of the controller placement.

Hybrid Hierarchical Control Plane: For large-scale SDN networks, in [30], the authors introduce a hierarchical hybrid control plane, named Orion to effectively reduce the computational complexity of an SDN control plane by several orders of magnitude. Orion uses two control layers: (1) area controller layer is responsible for handling the physical switches and collecting link information; (2) domain controller layer includes several controllers that supervise the area controllers as devices. Differing from Kandoo [25], the authors design an abstracted hierarchical routing method between area controller layer and domain controller layer to

solve the path stretch problem and achieve fast rerouting in the hierarchical hybrid control plane.

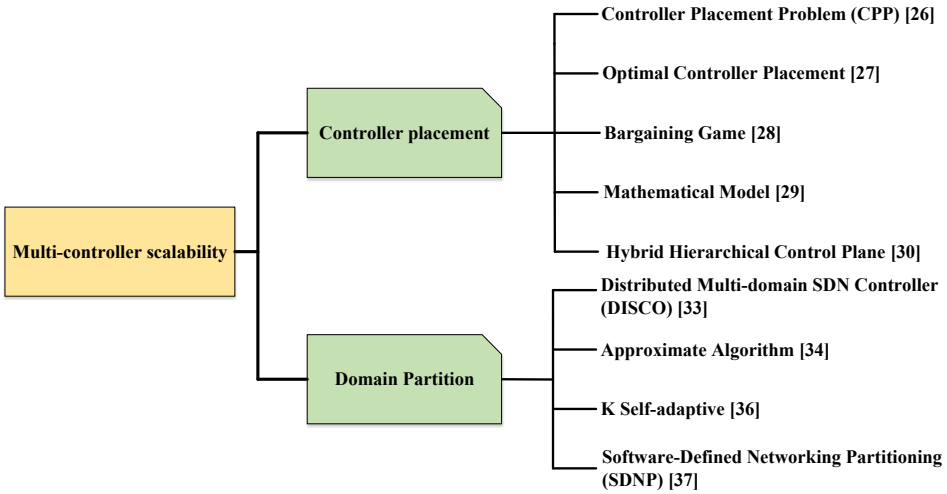

FIGURE 6. The multi-controller scalability solutions

We investigate and analyze the controller placement techniques for multi-controller scalability in Table 1. We compare different techniques from the aspects of authors, mode, objective, complexity, real time, simulation/evaluation and application scenarios including enterprise, Data Centers (DC), Cloud and Wide Area Network (WAN). The √ represents feasible and × represents not feasible. The rest tables in the paper follow the same notation.

B. DOMAIN PARTITION

Deploying multi-controller in one domain restricts the large-scale implementation and scalability of SDN. Therefore, the literature proposes to divide a network into multiple domains to improve the scalability of multi-controller [33] [34] [36] [37].

Distributed Multi-domain SDN Controller (DISCO): DISCO [33], implemented on top of Floodlight, is introduced to partition wide area network (WAN) with constrained overlay networks. A DISCO controller manages its own domain and communicates with other controllers via a lightweight and manageable control channel to provide end-to-end network services. In particular, DISCO adopts the innovative technology (e.g., link discovery agent, path computation agent) to well discriminate heterogeneous inter-domain links (e.g., high-capacity MPLS tunnels) and improve the utilization of link bandwidth.

Approximate Algorithm: In [34], the authors efficiently configure controllers in a multi-domain SDN to find the least number of controllers for the network. They formulate the multi-domain partition as a NP-hard problem and transfer the problem to the Greedy Sub-Graph Cover Problem (GSGCP) by abstracting domain as nodes. The authors then solve the GSGCP with a modified approximate optimal solution, and the simulation results demonstrate the solution achieves the equivalent multi-domain partition and has an acceptable computation complexity for any given network topology.

K Self-adaptive: Based on the spectral clustering [35], the authors introduce a self-adaptive partition and placement algorithm for controllers in wide area networks [36]. This algorithm uses matrix perturbation theory to determine the topology of domains and the optimal number of domains automatically. The authors also present a Beacon-based test framework and verify the algorithm's validity in Internet2 OS3E topology.

SDN Partitioning (SDNP): In [37], the authors propose new SDN-IP hybrid network architecture, named SDNP, for multi-domain partition in large-scale SDN network. The SDNP builds centralized control over a distributed routing protocol by dividing the network into sub-domains with SDN-enabled border nodes. SDNP can evenly partition the topology and dynamically modify the size of domains. Therefore, SDNP achieves high network control capabilities with a few SDN-enabled routers.

We investigate and analyze the domain partition techniques for multi-controller scalability. The results are presented in Table 2.

IV. MULTI-CONTROLLER CONSISTENCY

Multi-controller design can divide the entire network into several domains, and each controller manages its own SDN domain. To make sure that packets are transmitted correctly in the network, the controllers must interact the domain information with each other to keep the consistent view. Therefore, controller consistency also becomes an important issue for the multi-controller SDN.

The multi-controller must make a decision based on the consistent and coherent network information. However, during the data transmission, the out-sync between

controllers and concurrent strategic conflicts of controllers may lead to the inconsistency of the controller state.

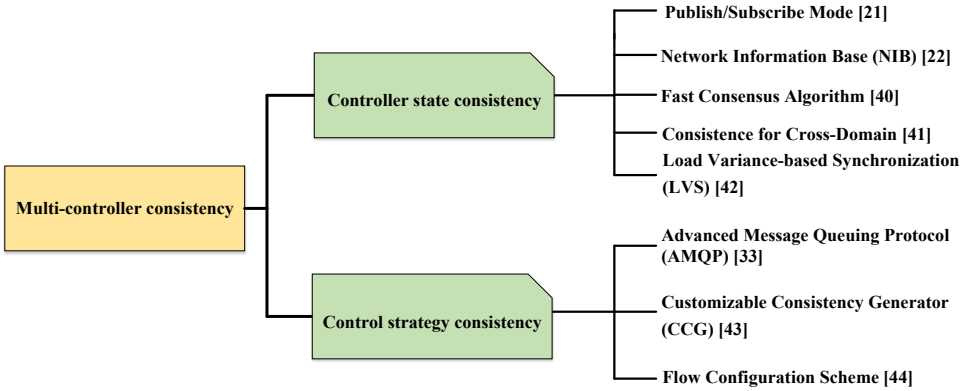

FIGURE 7. The multi-controller consistency solutions

Meanwhile, due to propagation delay and flow table order, control strategies of multi-controller are easy to be inconsistent, which would produce the packet loss and service interruption. Both DIFANE [38] and DevoFlow [39] improve the consistency of controller through adding the partial control functions into SDN switch. However, this action is contrary to the original design of SDN.

Based on the aforementioned analysis, in Fig. 7, we classify the existing research results of multi-controller consistency into two aspects: (1) consistency of controller state; (2) consistency of control strategy. The controller state consistency emphasizes on keeping the consistent local domain view once the network state changes. The control strategy consistency devotes to avoid the conflict of flow tables pushed by controllers. Both two ways can effectively guarantee the consistency of multi-controller.

A. CONTROLLER STATE CONSISTECNY

When the network state changes, the controllers must have the consistent view for the global network to make the correct decision for networks, which require the controllers with the consistent state [21] [22] [40-42].

Publish/Subscribe Mode: Based on the "publish/ subscribe" mode, HyperFlow [21] achieves consistent state among controller via WheelFs distributed file system. This is obviously due to WheelFs facilitates rapid prototyping and is resilient against network partitioning. The "publish/ subscribe" mode has a network-wide scope and three channel types (control channel: controllers advertise themselves there; data channel: events of general interest published here; individual controllers' channels: send commands and replies to a specific controller). If an event (e.g., OpenFlow messages) that changes the network happens, the controller that identifies the event will publish the event to switches. Other controllers receive the published event and update their network state to achieve status synchronization.

Network Information Base (NIB): Onix [22] stores the network information in the NIB and writes and reads the contents of NIB to synchronize the state of each controller. As the control platform, Onix is responsible for giving the control logic programmatic access to the network (reading and writing network state). In order to scale to very large networks (millions of ports) and to provide the requisite resilience for production deployments, Onix instance is also responsible for disseminating network state to other instances within the cluster. When one controller node has been changed in Onix, this information will be distributed among NIBs.

Fast Consensus Algorithm: As a new fast consensus algorithm, Fast Paxos-based Consensus (FPC) is proposed based on a strong consistency model [40]. FPC creatively defines three roles for controllers: listener, proposer, and chairman. Through applying the voting mechanism, the proposer can handle the request from the switch if receiving acceptance votes from a majority of the controllers. Moreover, each controller has a definite priority, and an aging mechanism is applied to avoid the starvation for the low priority. These settings could promise that FPC has stable consensus control logic.

Consistency for Cross-Domain: In [41], the authors consider the consistency of controller states in WAN. They propose a consistent layer that actively and passively snapshots the cross-domain control states to reduce the complexities of service realization. The consistent layer is applied and evaluated in the PlanetLab testbed by putting OpenFlow switch implementation on the overlay networks for evaluating performance in an enlarged WAN environment. The results show this method has four properties: (1) the scalability of the snapshot on large-scale domains, (2) the reliability for dealing with the physical network instabilities, (3) the responsiveness for reacting on a few state changes of domains, (4) the security of cross-domain control.

Load Variance-based Synchronization (LVS): In [42], the authors propose a new type of controller state synchronization scheme, Load Variance-based

Synchronization (LVS), to improve the load-balancing performance in the multi-controller multi-domain SDN network. Compared with PS (Periodic Synchronization)-based schemes, LVS-based schemes conduct effective state synchronizations among controllers only when the load of a specific server or domain exceeds a certain threshold, which significantly reduces the synchronization overhead of controllers. The results of simulation show that LVS achieves loop-free forwarding and good load-balancing performance with much less synchronization overhead, as compared with existing schemes.

We investigate and analyze the controller state consistency techniques for multi-controller consistency in Table 3.

B. CONTROL STRATEGY CONSISTECNY

The concurrent control strategy will bring about inconsistency issue, which can be resolved by strategy rules formed in the control layer. In order to avoid the involvement of physical devices, the controller could combine the strategies and use the fine-grained locking to ensure there are no conflicts between different control strategies [33] [43] [44].

Advanced Message Queuing Protocol (AMQP): in [33], the authors propose DISCO, an extensible DIstributed SDN COntrol plane able to deal with the distributed and heterogeneous nature of modern overlay networks. DISCO sets a messenger module and four agents, including monitoring, reachability, connectivity, and reservation. The messenger module is based on the AMQP [86] and its function is to identify neighboring controllers and establish a distributed publish/subscribe channel. Different agents use this channel to share network information with other controllers. Each agent publishes messages according to controller status and publishes the synchronous messages. Finally, the results demonstrate that DISCO can adapt to heterogeneous network topologies while being resilient enough to maintain the consistency of control strategy.

Customizable Consistency Generator (CCG): In [43], the authors propose CCG, a fast and generic framework to support customizable consistent policies during network updates. CCG adopts the hierarchical strategy, which divides the concurrent strategies into an organized tree. In this tree, each node can achieve the independent forwarding principle. They put in place the self-defined conflict processing for each node, so the entire processing will be turned into a reverse search tree. Mininet and physical testbed evaluations prove strategy's capability to achieve various types of consistency, such as path and bandwidth properties, with zero switch memory overheads.

Flow Configuration Scheme: similarly, in [44], the authors research control strategy from the perspective of flow configuration, and they combine the flow allocation cost to minimize the number of control strategies.

We investigate and analyze the control strategy consistency techniques for multi-controller consistency in Table 4.

V. MULTI-CONTROLLER RELIABILITY

Using multi-controller resolves the single point of failure problem for the controller, but it cannot guarantee the high reliability of the control plane. The connection links among switches and controllers have limited capacity. If these links experience congestion, interruption or failure, controllers and switches cannot normally communicate with each other, leading to the isolation among controllers and switches. Additionally, controllers could be failed or overwhelmed by malicious attacks (e.g., excessive packet-in requests). Thus, the multi-controller reliability is also important for actual deployment of multi-controller. In Fig. 8, we classify the existing research results of multi-controller reliability into two aspects: (1) control path reliability; (2) controller node reliability. Control path reliability considers multi-controller design from the perspective of reliable network links. On the contrary, the controller node reliability faces on the multi-controller design from the perspective of reliable and dependable network nodes.

A. CONTROL PATH RELIABILITY

Control actions (e.g., Packet-in sending, flow entry distribution) must be transmitted through the control paths. Therefore, optimizing control path is an efficient method to achieve the reliability of controllers [45-48].

Reliability-Optimized Scheme: In [45], the authors define a new metric, named "*expected percentage of control path loss*", to characterize the reliability of SDN. First, they analyze the reliability framework of the control plane and the control path. Then, the reliability-aware control placement is proved as an NP-hard problem. Moreover, several placement algorithms and their advantages are examined based on the actual topology. The authors demonstrate that reliability-aware controller can effectively improve the reliability of the control plane without introducing unacceptable latencies.

Fast Failover Design: In [46], the authors achieve fast failover for control traffic when controllers fail. The authors propose a protection metric for the connections between controllers and switches, and take into account both distance and resiliency factors: the algorithm builds a routing tree that results in a short distance and high resiliency in the connection between the switches and the controller. The solution suggests pre-configuring some backup outgoing links for switches and re-connecting switches to controllers if a link failure is detected. Therefore, this optimization scheme can be used to select the best controller location for maximizing the number of protected switches.

Survivor: Survivor is an enhanced controller placement strategy that reduces connectivity loss and enables smart recovery to improve the SDN survivability [47]. It enhances connectivity by employing path diversity, adds the capacity

awareness for controllers and builds the failover mechanism through the methodology for composing the backups. Survivor also has the strong topological adaptability and can be run on any given network topology.

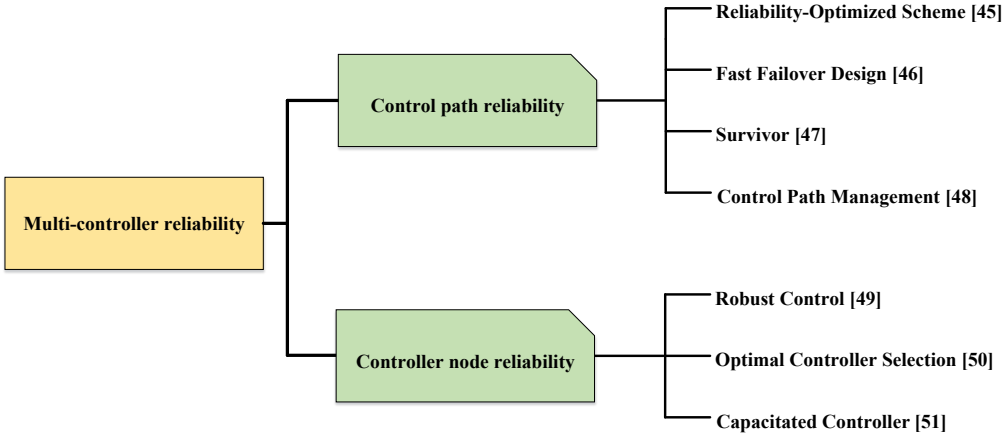

FIGURE 8. The multi-controller reliability solutions

Control Path Management: Control Path Management framework [48] addresses the problem of reliability from the perspective of the control path. The framework designs two strategies: (1) Reliable Controller Placement-Disjoint Control Path (RCP-DCP), which protects the control plane against single link and node failures by connecting switches to a controller over two disjoint control paths, and (2) Reliable Controller Placement-Different Controller Replicas (RCP-DCR), which provides seamless failover by connecting each switch to two different controller replicas over two disjoint paths. By combining the controller placement problem with resilient routing principle, both two strategies minimize the latency of the control plane and simplify the management of the control path.

We investigate and analyze the control path reliability techniques for multi-controller reliability in Table 5.

B. CONTROLLER NODE RELIABILITY

If a node fails, it can be quickly mapped or migrated to another node, or flows are rerouted on new paths disjoint with the node. However, different from traditional network nodes, a controller is responsible for traffic management in a network or domain and cannot be migrated and remapped. If a controller fails or crashes, the operation of the network controlled by the controller would be severely interrupted. Therefore, researching the controller node reliability has an important effect on multi-controller reliability [49-51].

Robust Control: in [49], the authors design an algorithm called K-Critical that places controllers to achieve a robust control. K-Critical discovers the minimum number of controllers and their locations to create a robust control topology that deals robustly with failure and balances the load among the selected controllers. This solution finds the best controller location as the network scale dynamically increases or decreases. However, it neglects several network performance metrics (e.g., controller throughput, link bandwidth, processing delay).

Optimal Controller Selection: in [50], the authors combine Greedy method and simulated annealing to optimize the selection of controller nodes to achieve the high reliability of the control plane. In the proposed optimization problem, the aim is to minimize the transmission paths between switches and controllers, and the constraints involve linking distance and latency. The results show that proposed solution Greedy-SA improves the reliability of the control plane and manages more switches with few controllers. However, this heuristic algorithm is only practically feasible for small and medium-size networks and cannot satisfy the time and resource demand for large-scale networks.

Capacitated Controller: in [51], the authors formulate a mathematical model for the capacitated controller placement that aims to reduce the worst-case latency between switches and controllers to deploy a limited number of controllers. Meanwhile, the authors also introduce a variant of the proposed model that minimizes the worst-case latencies with and without failure together. The results show that this controller placement that plans ahead for the failure result in much lower latency compared with the placement without planning ahead. However, they do not provide detailed algorithm design for implementing the strategy.

We investigate and analyze the controller node reliability techniques for multi-controller reliability in Table 6.

VI. MULITI-CONTROLLER LOAD BALANCING

The introduction of multi-controller partitions the network into several SDN domains, while the controllers monitor the local switches in the domain, respectively. However, due to the network traffic variation and the static mapping between switches and controllers, it is likely to produce overloaded controller and underloaded controller in the network. Further,

imbalanced load distribution among controllers will seriously degrade the network performance (e.g., high packet loss rate, high response time of controller and low controller throughput). Therefore, for a given multi-controller SDN

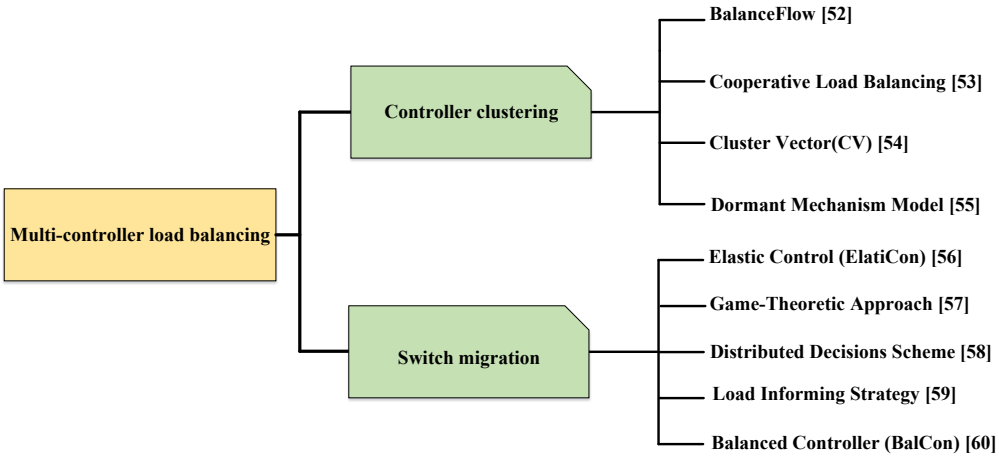

FIGURE 9. The multi-controller load balancing solutions

network, it is essential to ensure the nice load balancing performance of multi-controller. By investigating the literature, we conclude that the existing research solves the problem in two ways: (1) controller clustering; (2) switch migration, as is illustrated in Fig. 9. As a comparison, controller clustering pays greater attention to architecture design by constructing the dynamic controller resource pool, while the switch migration concentrates on adjusting the distribution of controller loads to keep load balancing.

A. CONTROLLER CLUSTERING

The state-of-the-art works propose controller clustering [52-55] to achieve load balancing. Generally, a network contains one super controller and multiple regular controllers, which construct the controller resources pool. The super controller is exclusively used in managing all controller loads and periodically collects the number of flows in each domain from the regular controllers. A regular controller manages its domain and uploading load information with a cross-controller interaction system periodically. When the traffic load surges, the super controller executes the load balancing algorithm and maps each switch to a specific controller. By controller clustering, the load information can be centralized collected, and the super controller makes the balanced load management without producing the other superfluous overheads between regular controllers.

BalanceFlow: BalanceFlow [52] is a typical controller clustering solution based on hierarchical deployment. The main advantage of this method is flexible tuning the flow requests handled by each controller without introducing extra propagation latencies. It follows the multi-controller feature in the OpenFlow 1.2. All controllers in the BalanceFlow maintain their own load information and publish this information periodically to each other through a cross-controller communication system. Upon traffic condition changes, one of the BalanceFlow controllers is selected as the super controller, which partitions the traffic and reallocates different flow setups to appropriate controllers. BalanceFlow also proposes a reasonable extension action for switches: CONTROLLER X action. By using this extension action, the overloaded controller will reduce the process of flow request, and those requests will be allocated to the controller with light load dynamically.

Cooperative Load Balancing: similar to BalanceFlow, [53] and [54] also define a super controller to manage controllers' loads. Differently, [53] introduces Cooperative Load Balancing Scheme for hierarchical controller deployment (COLBAS) relying on controller cooperation via cross-controller communication. The main thought of this scheme is similar to BalanceFlow, but the authors adopt a Greedy algorithm to reassign the controllers' loads. In particular, COLBAS can keep the system performance high and the load reassigning cost low.

Cluster Vector (CV): in [54], the authors simplify the load balancing operation with a self-defined label CV, which is a vector that contains addresses of the controllers in the same cluster. Meanwhile, it also breaks the dependency between the super controller and regular controllers. The proposed design consists of two levels: high-level operations in a super controller and low-level operations in a regular controller. Each controller has its own CV, and a regular controller finds the address of other regular controllers from its CV and uses the address to query other regular controllers about their load.

Dormant Mechanism Model: in [55], the authors design a dormant mechanism model based on flat deployment for multi-controller to save network resource, reduce energy consumption and improve the utilization of controller. The key idea is to let some idle controllers enter the dormant state to be inactive or power off when the network's load is light.

The authors propose a genetic algorithm to locate the optimal values of various parameters (e.g., latency, traffic, distance) to minimize system cost for the deployment decision-making and use queuing model to analyze the scheme's performance.

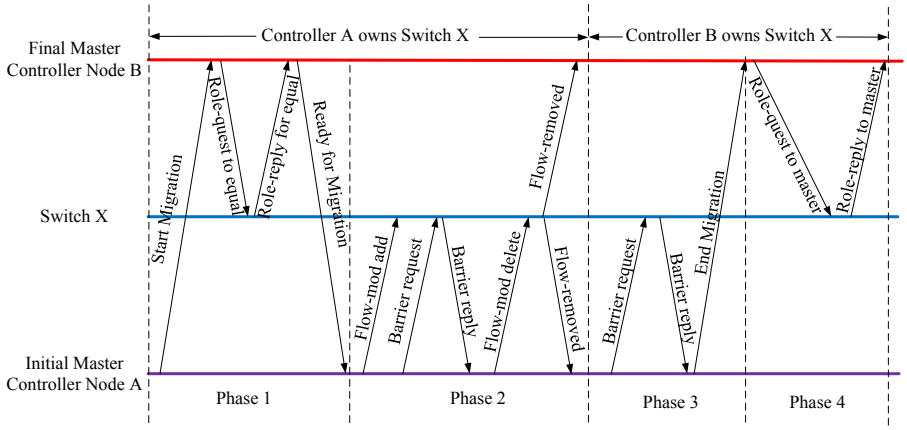

FIGURE 10. Switch migration process

We investigate and analyze the controller clustering techniques for multi-controller load balancing in Table 7.

B. SWITCH MIGRATION

Benefit from three roles of controllers (OpenFlow 1.2), researchers propose balancing multi-controller loads through switch migration [56-60] based on dynamic multi-controller architecture. In a domain, when the controller overloads or the flow requests of switches increase sharply, some switches would be reassigned to the controller of the other domain with a light load. The core idea of switch migration is to dynamically change the relationships between switches and controllers by migrating switches from the overloaded controller to the underloaded controller.

Fig. 10 shows a complete description of the switch migration procedure, which consists of four phases. In phase 1, it achieves changing the role of the target to equal. The initial master (A) sends a start migration message to B through controller-to-controller channel. Then, (B) sends Role-quests to the switch that needs to be migrated. After (B) receives Role-Reply from the switch, it notifies (A) that the role changing has accomplished. After (B) changes its role to equal, it receives asynchronous messages from the switch, but does not provide a response. In phase 2, it inserts and removes a dummy flow. (A) firstly sends Flow-mod to (X) to add a new flow entry, which does not match any packet. Then, it sends another Flow-mod to delete the entry. In return, the switch can send a Flow-removed message to controllers because of (B) is an equal controller right now. The Flow-removed offers a transfer of ownership for the switch (X) from (A) to (B). Besides, a barrier message is requested after the insertion of the dummy flow. In phase 3, it flushes pending requests for a barrier. (A) transmits a Barrier-request and waits for the Barrier-reply, only after which it sends "end migration" to the final master (B). In phase 4, it makes the target controller final master. The final master (B) sets its role to master for the switch by sending a Role-request message to the switch. Finally, it updates the distributed data store.

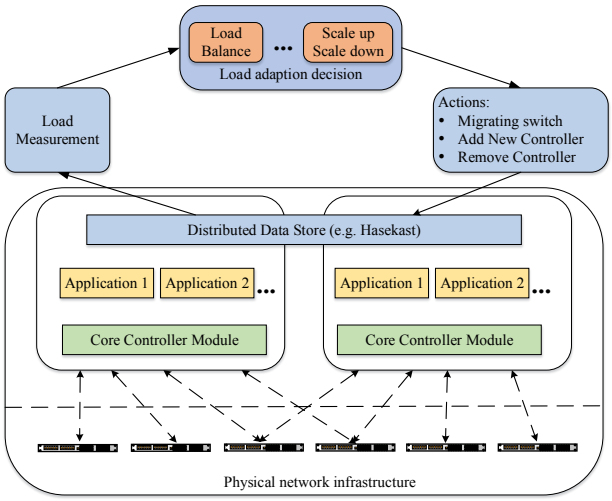

FIGURE 11. Elastic framework

Elastic Control (ElatiCon): ElastiCon [56] is the first switch migration framework based on dynamic multi-controller architecture. Fig. 11 sets the complete framework of ElastiCon, which contains three modules: load measurement modules, load adaptation decision modules, and action modules. The load measurement module collects the load of each controller and sends the load information to load adaptation decision module, which decides load allocation among controllers. The action module conducts control actions (e.g., migrating switch, adding and removing controllers) to achieve the dynamic control of controllers and switches. ElastiCon periodically monitors the load on each controller, detects imbalances, and automatically balances the load across controllers by migrating switches from the overloaded controller to a lightly loaded one. Meanwhile, in order to harmonize the migration, a novel switch migration

protocol is designed for enabling such load shifting, which conforms to the OpenFlow standard. Finally, a prototype of ElastiCon is built and its performance is evaluated based on Mininet. Therefore, ElastiCon ensures predictable controller performance even under highly dynamic workloads.

Game-Theoretic Approach: In [57], the authors solve the switch migration algorithm with game theory. By taking light controllers as the game players and switches as the commodities, a zero-sum game model is exploited to emulate the competitions for migrating switches among overloaded controllers. The controller selects the optimal elements to implement the transaction by increasing or decreasing the commodity value of the switch. The game model is fast and efficient to achieve switch migration but is not suitable for large-scale network due to the high complexity of algorithm design.

Distributed Decisions Scheme: In [58], the authors define the Switch Migration Problem (SMP) and a Network Utility Maximization (NUM) problem with the objective of maximizing the number of serving requests under the available control resource. Distributed Hopping Algorithm (DHA) is designed to achieve optimal switch migration via Log-Sum-Exp function. The DHA procedure is a time-reversible markov chain process. The simulation results show DHA outperforms existing schemes by reducing flow setup time and improving the average utilization ratio of controller.

Load Informing Strategy: In [59], the authors present a load balancing mechanism based on a load informing strategy for controllers. Emphatically, it builds distributed decision architecture, including four components that were load measurement, load informing, and balancing decision and switch migration. In this strategy, each controller can periodically actively report its load information to other controllers, and it also handles and stores the load information from others. While the periodical active load informing can decrease the decision delay, it also causes additional processing and communication overhead in the control plane. Especially, when the current load value does not change much compared to the last value, reporting it to other controllers is a redundant operation.

Balanced Controller (BalCon): BalCon is a heuristic solution proposed in [60]. It is based on two key observation: (1) an effective switch migration should consider the communication patterns of the SDN switches, (2) the switch migration should be processed at the granularity of clusters: switches with strong connections, which has the shorter distance to controller, should always be assigned to the same controller. BalCon is achieved by a realistic prototype based on Ryu, and the results show BalCon significantly reduces the number of migrating switches.

We investigate and analyze the controller clustering techniques for multi-controller load balancing in Table 8.

VII. FUTURE WORK

The existing research focuses on solving challenges on multi-controller scalability, reliability, consistency and load balancing. However, there are still several problems that deserve deep research. We briefly discuss the research emphasis and development direction of multi-controller in the future.

(1) The development of control software

Control software is an important application in the control plane, and its main form is the controller. Therefore, implementing multi-controller architecture is greatly related to the development of control software. Based on the existing controller versions, simplifying the deployment way and improving compatibility are the most important tasks for the exploitation of control software that supports multi-controller architecture.

(2) Controller safety

The controller plays a critical role in monitoring and dispatching the network traffic, but the existing multi-controller architecture is lack of safety mechanism and anomaly detection. The hostile attack is not difficult to break the protection measures of controllers. Therefore, Enhancing the anti-attack performance of multi-controller architecture is another important research topic.

(3) Multi-controller architecture

In the initial phase, application scenarios of SDN mostly focus on colleges, enterprises or data centers, and SDN is lack of deployment experience in the large-scale network due to the constraint of scalability. The introduction of multi-controller provides the possibility for the widespread deployment of SDN. Unfortunately, the actual deployment of multiple controllers still lacks relevant technical guidance. There is still a long way to go before the multi-controller is promoted.

(4) Heterogeneous multi-controller

The existing researches about the heterogeneous controller focus mainly on security and convergence area. However, in analogy with the homogeneous multi-controller, the exploration of the heterogeneous multi-controller must be applied into more research fields, such as availability and consistency. Meanwhile, the performance interruption between different types of controllers also should get more attention.

VIII. CONCLUSION

The design and performance of the control plane are the critical part of SDN. In order to achieve the large-scale application of SDN, the control plane has evolved from the single centralized controller to multiple controllers. In this paper, based on the existing literature, we first provide an overview of multi-controller, including the origin of multi-controller and its challenges. Then, we summarize the main research challenges of multi-controller: scalability, consistency, reliability, and load balancing. Meanwhile, we also consider the corresponding solution for these challenges.

Further, we give some promising research problems of multi-controller in the future.

REFERENCES

[1] H. c. Wang and H. s. Doong, "Validation in Internet Survey Research: Reviews and Future Suggestions," System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on, Waikoloa, HI, 2007, pp. 243-243.

[2] A. M. Ahmed, T. Qiu, F. Xia, B. Jedari and S. Abolfazli, "Event-Based Mobile Social Networks: Services, Technologies, and Applications," in IEEE Access, 2014, vol. 2, pp. 500-513.

[3] T. Benson, A. Akella, and D. Maltz, "Unraveling the complexity of network management," in Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, ser. NSDI'09, Berkeley, CA, USA, 2009, pp. 335–348.

[4] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, "Intelligent design enables architectural evolution," in Proceedings of the 10th ACM Workshop on Hot Topics in Networks, ser. HotNets-X. New York, NY, USA: ACM, 2011, pp. 3:1–3:6.

[5] B. Raghavan, M. Casado, T. Koponen, S. Ratnasamy, A. Ghodsi, and S. Shenker, "Software-defined internet architecture: Decoupling architecture from infrastructure," in Proceedings of the 11th ACM Workshop on Hot Topics in Networks, ser. HotNets-XI. New York, NY, USA: ACM, 2012, pp. 43–48.

[6] FEAMSTER N, REXFORD J, ZEGURA E. "The road to SDN: an intellectual history of programmable networks," ACM Sigcomm Computer Communication Review, vol. 44, no. 2, pp. 87-98.

[7] NADEAU T D, GRAY K. "SDN: Software Defined Networks," O'Reilly Media, Inc, 2013.

[8] V. López, O. González de Dios, B. Fuentes, M. Yannuzzi, J. P. Fernández-Palacios and D. López, "Towards a network operating system," OFC 2014, San Francisco, CA, 2014, pp. 1-3.

[9] Ryu. [Online]. Available: http://osrg.github.com/ryu/

[10] Zehua Guo, Ruoyan Liu, Yang Xu, Andrey Gushchin, Anwar Walid and H. Jonathan Chao, "STAR: Preventing flow-table overflow in software-defined networks," Computer Networks, 2017.

[11] L. Sidki, Y. Ben-Shimol and A. Sadovski, "Fault tolerant mechanisms for SDN controllers," 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Palo Alto, CA, 2016, pp. 173-178.

[12] Oktian Y E, Lee S G, Lee H J, et al. "Distributed SDN controller system: A survey on design choice," Computer Networks, 2017, vol. 121, pp. 100-111.

[13] Y. Jia, N. Hua, Y. Yu, Y. Li and X. Zheng, "Experimenting with multi-controller collaboration for large-scale intra-data center networks," 2017 Optical Fiber Communications Conference and Exhibition (OFC), Los Angeles, CA, 2017, pp. 1-3.

[14] Karakus M, Durresi A. "A survey: Control plane scalability issues and approaches in Software-Defined Networking (SDN)," Computer Networks, 2017, vol. 112, pp. 279-293.

[15] Xia W, Wen Y, Foh C H, et al. "A Survey on Software-Defined Networking," IEEE Communications Surveys & Tutorials, 2015, vol. 17, no. 1, pp. 27-51.

[16] Kreutz D, Ramos F M V, Esteves Verissimo P, et al. "Software-Defined Networking: A Comprehensive Survey," Proceedings of the IEEE, 2014, vol. 103, no. 1. pp. 10-13.

[17] Songtao, WANG, Konglin, et al. "A survey of network update in SDN," Frontiers of Computer Science, 2017, vol. 11, no. 1, pp. 4-12.

[18] Huang T, Yu F R, Zhang C, et al. "A Survey on Large-scale Software Defined Networking (SDN) Testbeds: Approaches and Challenges," IEEE Communications Surveys & Tutorials, 2016, pp. 1-1.

[19] Rawat D B, Reddy S R. "Software Defined Networking Architecture, Security and Energy Efficiency: A Survey," IEEE Communications Surveys & Tutorials, 2017, vol. 19, no. 1, pp. 325-346.

[20] https://www.scribd.com/document/117471434/OpenFlow-1-2.

[21] D. Dotan and R. Y. Pinter, "HyperFlow: an integrated visual query and dataflow language for end-user information analysis," 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05), 2005, pp. 27-34.

[22] R. Y. Shtykh and T. Suzuki, "Distributed Data Stream Processing with Onix," 2014 IEEE Fourth International Conference on Big Data and Cloud Computing, Sydney, NSW, 2014, pp. 267-268.

[23] A. M. Al-Sadi, A. Al-Sherbaz, J. Xue and S. Turner, "Routing algorithm optimization for software defined network WAN," 2016 Al-Sadeq International Conference on Multidisciplinary in IT and Communication Science and Applications (AIC-MITCSA), Baghdad, 2016, pp. 1-6.

[24] M. Hungyo and M. Pandey, "SDN based implementation of publish/subscribe paradigm using OpenFlow multicast," IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Bangalore, 2016, pp. 1-6.

[25] S. H. Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," 1st Workshop HotSDN, 2012, pp. 19-24.

[26] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," 1st Workshop HotSDN, 2012, pp. 7–12.

[27] H. K. Rath, V. Revoori, S. M. Nadaf and A. Simha, "Optimal controller placement in Software Defined Networks (SDN) using a non-zero-sum game," IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, Sydney, NSW, 2014, pp. 1-6.

[28] A. Ksentini, M. Bagaa, T. Taleb and I. Balasingham, "On using bargaining game for Optimal Placement of SDN controllers," 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, 2016, pp. 1-6

[29] A. Sallahi and M. St-Hilaire, "Optimal Model for the Controller Placement Problem in Software Defined Networks," in IEEE Communications Letters, 2015, vol. 19, no. 1, pp. 30-33.

[30] Y. Fu et al., "A Hybrid Hierarchical Control Plane for Flow-Based Large-Scale Software-Defined Networks," in IEEE Transactions on Network and Service Management, 2015, vol. 12, no. 2, pp. 117-131.

[31] F. Yeung, "Internet 2: scaling up the backbone for R&D," IEEE Internet Computing, 1997, vol. 1, no. 2, pp. 36-37.

[32] B. Soper and J. Musacchio, "A non-zero-sum, sequential detection game," 2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, 2015, pp. 361-371.

[33] K. Phemius, M. Bouet and J. Leguay, "DISCO: Distributed multi-domain SDN controllers," 2014 IEEE Network Operations and Management Symposium (NOMS), Krakow, 2014, pp. 1-4.

[34] G. Wang, Z. Zhao, J. Peng, R. Li, "An approximate algorithm of controller configuration in multi-domain SDN architecture," 9th International Conference on Communications and Networking in China, Maoming, 2014, pp. 601-605.

[35] P. Xiao et al., "A Traffic Classification Method with Spectral Clustering in SDN," 2016 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), Guangzhou, China, 2016, pp. 391-394.

[36] Peng, XIAO, Zhi-yang, et al. "A K self-adaptive SDN controller placement for wide area networks," Frontiers of Information Technology & Electronic Engineering, 2016, vol. 17, no 7, pp. 620-633.

[37] M. Caria, A. Jukan and M. Hoffmann, "SDN Partitioning: A Centralized Control Plane for Distributed Routing Protocols," in IEEE Transactions on Network and Service Management, 2016, vol. 13, no. 3, pp. 381-393.

[38] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with difane," SIGCOMM Comput. Commun. 2014, Rev., vol. 41, no. 4, pp. 1-6.

[39] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: scaling flow management for highperformance networks," Comput. Commun. Rev., 2011, vol. 41, no. 4, pp. 254-265.

[40] Ho, Chia Chen, K. Wang, and Y. H. Hsu. "A fast consensus algorithm for multiple controllers in software-defined networks." International Conference on Advanced Communication Technology IEEE, 2016, pp. 1-1.

[41] ZHOUBoyang, WUChunming, GAOWen, et al. “Achieving Consistence for Cross -Domain WAN Control in Software-Defined Networks,” China Communication, 2015, vol. 12, no. 10, pp. 136-146.
[42] Guo, Zehua, et al. "Improving the performance of load balancing in software-defined networks through load variance-based synchronization." Computer Networks 68 (2014): 95-109.
[43] X. Xiong and J. Fu, “Active Status Certificate Publish and Subscribe Based on AMQP,” 2011 International Conference on Computational and Information Sciences, Chengdu, China, 2011, pp. 725-728.
[44] Zhou W, Jin D, Croft J, et al. “Enforcing customizable consistency properties in software-defined networks,” Usenix Conference on Networked Systems Design and Implementation. USENIX Association, 2015, pp. 73-85.
[45] Y. Hu, W. Wang, X. Gong, X. Que and S. Cheng, “On reliability-optimized controller placement for Software-Defined Networks,” in China Communications, 2014, vol. 11, no. 2, pp. 38-54.
[46] N. Beheshti and Y. Zhang, “Fast failover for control traffic in Software-defined Networks,” 2012 IEEE Global Communications Conference (GLOBECOM), Anaheim, CA, 2012, pp. 2665-2670.
[47] L. F. Müller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, “Survivor: An enhanced controller placement strategy for improving SDN survivability,” 2014 IEEE Global Communications Conference, Austin, TX, 2014, pp. 1909-1915.
[48] S. Song, H. Park, B. Y. Choi, T. Choi and H. Zhu, “Control Path Management Framework for Enhancing Software-Defined Network (SDN) Reliability,” in IEEE Transactions on Network and Service Management, 2017, vol. 14, no. 2, pp. 302-316.
[49] Y. Jiménez, C. Cervelló-Pastor and A. J. García, “On the controller placement for designing a distributed SDN control layer,” 2014 IFIP Networking Conference, Trondheim, 2014, pp. 1-9.
[50] Sahoo, Kshira Sagar, et al. "Optimal controller selection in Software Defined Network using a Greedy-SA algorithm." IEEE Conference Indiacom IEEE, 2016.
[51] B. P. R. Killi and S. V. Rao, “Optimal Model for Failure Foresight Capacitated Controller Placement in Software-Defined Networks,” in IEEE Communications Letters, 2016, vol. 20, no. 6, pp. 1108-1111.
[52] Hu Y, Wang W, Gong X, et al. “BalanceFlow: Controller load balancing for OpenFlow networks,” IEEE International Conference on Cloud Computing and Intelligent Systems, 2013, pp. 780-785.
[53] H. Selvi, G. Gür and F. Alagöz, “Cooperative load balancing for hierarchical SDN controllers,” 2016 IEEE 17th International Conference on High Performance Switching and Routing (HPSR), Yokohama, 2016, pp. 100-105
[54] H. Sufiev and Y. Haddad, “A dynamic load balancing architecture for SDN,” 2016 IEEE International Conference on the Science of Electrical Engineering (ICSEE), Eilat, 2016, pp. 1-3.
[55] F. Yonghong, B. Jun, W. Jianping, C. Ze, W. Ke and L. Min, “A dormant multi-controller model for software defined networking,” China Communications, 2014, vol. 11, no. 3, pp. 45-55.
[56] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman and R. R. Kompella, “ElastiCon; an elastic distributed SDN controller,” 2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), Marina del Rey, CA, 2014, pp. 17-27.
[57] Hongchang Chen, Guozhen Cheng, Zhiming Wang. “A Game-Theoretic Approach to Elastic Control in Software-Defined Networking,” China Communication, 2016, vol. 13, no. 5, pp. 103-109.
[58] Cheng G, Chen H, Wang Z, et al. “DHA: Distributed decisions on the switch migration toward a scalable SDN control plane,” Ifip NETWORKING Conference. IFIP, 2015, pp. 473-477.
[59] Yu J, Wang Y, Pei K, et al. “A load balancing mechanism for multiple SDN controllers based on load informing strategy,” Network Operations and Management Symposium, 2016, pp. 1-6.
[60] Cello M, Xu Y, Walid A, et al. “BalCon: A Distributed Elastic SDN Control via Efficient Switch Migration,” IEEE International Conference on Cloud Engineering, 2017, pp. 40-50.

TABLE 1. An overview of current controller placement techniques for multi-controller scalability

Authors	Mode	Objective	Method	Complexity	Simulation/Evaluation	Application scenario			
						Enterprise	DC	Cloud	WAN
Heller et al. [26]	Flat	Studying the impact of propagation latency on multi-controller placement.	K-center	Low	• The analysis shows that deploying one controller is sufficient to meet the latency constraint for the most topologies. • Adding controllers in the network could effectively reduce both average and worst-case latency.	√	√	√	√
Rath et al. [27]	Flat	Minimizing the packet drops and delay, and saving the cost of deployment and operation.	Non-Zero-Sum game	Low	• Packet drops can be avoided by the addition of new controllers and/or offloading.	√	√	×	×
Ksentini et al. [28]	Flat	Minimizing the propagation latency and communication overhead.	Bargaining game	High	• The proposed solutions show the better performance in reducing communication overhead of switch-controller and controller-controller. • The solutions can also achieve the Pareto-Optimal efficiency in the network.	√	√	×	×
Sallahi et al. [29]	Flat	Minimizing the network cost that includes (installing controllers, linking switches to controllers and linking the controllers).	Linear programming	Low	• The results show that the proposed model can be used to design the new SDN network, or migrate the traditional network into SDN.	×	√	√	×
Fu et al. [30]	Hierarchical	Reducing the computational complexity of the control plane by several orders of magnitude and planning the fast reroute for the flow.	Heuristic approach and graph theory	Low	• The flow set-up rate of the proposed solutions is better than Floodlight. • If there are 3 hops from the source host to the destination host, the proposed solutions cost 8.6 ms to reroute the data flow.	×	√	√	×

TABLE 2. An overview of current domain partition techniques for solving multi-controller scalability

Authors	Mode	Objective	Method	Complexity	Simulation/Evaluation	Application scenario			
						Enterprise	DC	Cloud	WAN
Phemius et al. [33]	Flat	Studying the resilient, scalable and easily extensible SDN control plane by designing domain organization.	Service agent	High	• The proposed solutions could achieve the dynamic network partition for the most topologies.	×	×	×	√
Wang et al. [34]	Flat	Minimizing the number of controllers needed in the network.	Interdependence graph	Low	• The number of controllers needed to be deployed in the network is reduced by 35%. • The compute complexity of the proposed algorithm is less than $O(n^2)$.	×	×	×	√
Peng et al. [36]	Flat	Partitioning a large network into several small SDN domains for improving the scalability.	Matrix theory	Low	• Results show the controller latency becomes more balanced in the network. • The throughput has been improved about 10%, compared with the average-latency placement under the realistic traffic.	×	×	√	√
Caria et al. [37]	Flat	Balancing the network topology partition and designing the network management schemes.	Heuristic approach	High	• The minimum capacity requirements of controllers have been reduced about 25.3%. • The link utilization of the proposed schemes is close to 60%.	√	×	×	×

TABLE 3. An overview of current controller state consistency techniques for solving multi-controller consistency

Authors	Mode	Objective	Method	Complexity	Simulation/Evaluation	Application scenario			
						Enterprise	DC	Cloud	WAN
Dotan et al. [21]	Flat	Achieving the consistent state among controller via WheelFs distributed file system.	Publish/subscribe mode	High	• The proposed method guarantees the bounded window of inconsistency, if network changes occur at a rate < 1000 event/sec.	√	√	√	√
Shtykh et al. [22]	Flat	Providing a general API for control plane, allowing controllers to make trade-offs for consistency.	Structure optimization	High	• The controller with the proposed API can handle more than 24,000 Packet-in messages per second. • The solutions take 120 ms at most to repair the tunnel once the failure has been detected.	×	√	×	√
Ho et al. [40]	Flat	Reaching a consistent network state among SDN controllers to provide strong consistency.	Consensus Algorithm	High	• The proposed solution has lower average consensus time (35.3% lower) than Raft protocol.	√	√	×	×
ZHOU et al. [41]	Flat	Reducing the complexity of the cross-domain control.	Consistence layer	High	• Results show that the active and passive snapshots are executed with the mean times of 1.873s and 105ms in 135 controllers.	×	×	×	√
Guo et al. [42]	Flat	Reducing the synchronization overhead of controllers.	Variance synchronization	High	• The proposed solutions can provide the loop-free forwarding and keep the synchronization overhead in the low level.	×	√	√	√

TABLE 4. An overview of current control strategy consistency techniques for solving multi-controller consistency

Authors	Mode	Objective	Method	Complexity	Simulation/Evaluation	Application scenario			
						Enterprise	DC	Cloud	WAN
Phemius et al. [33]	Flat	Coping with the distributed and heterogeneous nature of modern overlay networks.	Service agent	High	• The proposed scheme can provide a consistent network view by inter-domain agent, which has low latency and high stability.	×	×	×	√
Xiong et al. [43]	Hierarchical	Supporting customizable consistent policies during network updates.	Policy tree	High	• When the network updates, it can provide the update time that is less than 100ms. • The consistent state can stay longer even if the network has node or link failures.	√	√	×	√
Zhou et al. [44]	Flat	Reducing the task of synthesizing an update plan under the constraint of a given consistency policy.	Uncertainty-aware model	High	• The proposed solutions can complete the different types of consistency (path, bandwidth), which has zero switch memory overhead. • The update complete time is close to the optimal solutions (100 ms).	×	×	×	√

TABLE 5. An overview of current control path reliability techniques for solving multi-controller reliability

Authors	Mode	Objective	Method	Complexity	Simulation/Evaluation	Application scenario			
						Enterprise	DC	Cloud	WAN
Hu et al. [45]	Flat	Maximizing the reliability of SDN control networks.	Heuristic approach	High	• When placing only one controller, it can produce the optimal reliability. • When placing three to four controllers, the average and worst-case latencies has been reduced by 13.7% and 13.8%, respectively.	√	×	×	√
Beheshti et al. [46]	Flat	Maximizing the possibility of fast failover once the connection between switches and controller breaks.	Greedy approach	Low	• The reliability improvement is between 51% and 100%, where all nodes are protected. • The average path length of the proposed solutions is less than Shortest Path Tree's.	√	√	√	×
Müller et al. [47]	Flat	Designing the SDN control plane, considering the path diversity, capacity, and failover mechanisms.	Linear programming	Low	• The results show the probability of connectivity loss is still around 80% when the chance of failure is 60%.	×	×	×	√
Song et al. [48]	Flat	Minimizing the length of control path to enhance SDN reliability.	Cluster approach	High	• When using the registration facility, the unsynchronized and redundant control messages can be filtered. • The recovery time of controller failure has been reduced 10ms at least.	√	×	√	×

TABLE 6. An overview of current controller node reliability techniques for solving multi-controller reliability

Authors	Mode	Objective	Method	Complexity	Simulation/Evaluation	Application scenario			
						Enterprise	DC	Cloud	WAN
Jiménez et al. [49]	Hierarchical	Studying the minimum number of controllers and their location to create a robust control topology that deals robustly with failures.	K-center	High	• In terms of the sparse networks, the proposed solutions show that five controllers are the optimal choice. • In terms of the dense network, one controller is an optimal choice, considered with controller latency.	×	×	×	√
Sahoo et al. [50]	Flat	Deciding where to place the controllers with a limited amount of resources within the network.	Greedy approach	High	• Result shows that the average delay gained by the proposed solution is always relatively stable compared to simulated annealing.	×	×	√	√
Killi et al. [51]	Flat	Minimizing the worst-case latency between the switches and their Kth reference controllers.	Linear programming	Low	• The worst latency has been reduced by 15.3% and 17.8% in failure and no-failure scene, respectively. • As the number of controllers increases, the proposed solution can minimize the probability of network congestion.	×	×	×	√

TABLE 7. An overview of current controller clustering techniques for solving multi-controller load balancing

Authors	Mode	Objective	Method	Complexity	Simulation/Evaluation	Application scenario			
						Enterprise	DC	Cloud	WAN
Hu et al. [52]	Flat	Partitioning control traffic load among different controller instances in a more flexible way.	Heuristic approach	Low	• The results show that the mean value of the average delay is about 8.7% larger than the optimal value. • The recovery time of overload controller has been reduced to 60s.	√	√	×	√
Selvi et al. [53]	Hierarchical	Studying a load-balancing scheme for hierarchical controller configurations.	Greedy approach	Low	• The time to load balancing has been reduced by 16% compared to ElastiCon. • The controller throughput of the proposed method is higher than Random controller configuration.	×	√	√	√
Sufiev et al. [54]	Hierarchical	Enabling dynamic load balancing among multi-controller.	Cluster approach	High	• The solution breaks the dependency during the periodical load balancing. • The run time of periodic controller operation has reduced by 50%.	×	√	√	√
Fu et al. [55]	Flat	Constituting logically centralized control plane to provide load balancing and fail over.	Queuing analysis	High	• The results show the solutions have saved the 30% energy consumption by setting dormant controllers. • The probability of controller overload has been reduced 28%.	×	×	√	√

TABLE 8. An overview of current switch migration techniques for solving multi-controller load balancing

Authors	Mode	Objective	Method	Complexity	Simulation/Evaluation	Application scenario			
						Enterprise	DC	Cloud	WAN
Dixit et al. [56]	Flat	Achieving the dynamic mapping between switches and controllers.	Linear programming	Low	• The controller response time has been reduced to 5ms averagely.	×	√	√	×
Chen et al. [57]	Flat	Studying how to improve the load balancing performance of controllers in SDN.	Game theory	High	• Only 1.25% switches have been migrated when a half of controllers need master reelection operation.	×	×	√	√
Cheng et al. [58]	Flat	Studying which switch should be migrated and where it will be moved.	Heuristic approach	High	• Only 10% of switches in such a network have been migrated to load rebalance when there are 40~50% heavy controllers.	√	×	×	×
Yu J et al. [59]	Flat	Reducing the load balancing decision time as rapidly as possible.	Linear programming	Low	• Results show that the load balancing is completed within 5s. • The proposed method has the higher throughput, compared with the static mapping between switch and controller.	√	√	×	×
Cello et al. [60]	Flat	Reducing the number of the migrated switches to keep efficient switch migration.	Heuristic approach	High	• The proposed solutions reduce the load imbalance among SDN controllers by 40% by migrating only a small number of switches. • The computational time is 11.51s in the proposed method.	×	×	√	√