IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 15, 2018, accepted December 2, 2018, date of publication December 5, 2018,
date of current version December 31, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2885130

Adaptive Consistency Strategy of Multiple Controllers in SDN

BANG ZHANG[1], XINGWEI WANG[1], AND MIN HUANG[2]
[1]College of Computer Science and Engineering, Northeastern University, Shenyang 110169, China
[2]College of Information Science and Engineering, Northeastern University, Shenyang 110819, China

Corresponding author: Xingwei Wang (wangxw@mail.neu.edu.cn)

This work is supported by the Major International (Regional) Joint Research Project of the NSFC under Grant 71620107003; in part by the National Science Foundation for Distinguished Young Scholars of China under Grant 71325002; in part by the National Natural Science Foundation of China under Grant 61572123 and Grant 61502092; in part by the the Program for Liaoning Innovative Research Term in University under Grant LT2016007; and in part by the MoE and ChinaMobile Joint Research Fund under Grant MCM20160201.

ABSTRACT In large-scale software-defined networking (SDN), the logically centralized controller usually uses multiple distributed controllers to manage and operate the network in a cooperative manner from the perspective of global network view. One of the major challenges with respect to deploying multiple controllers in SDN is how to synchronize the state among controllers to maintain information consistency. Aiming at solving this problem, we propose a periodically adaptive synchronization strategy of controllers. At first, the consistency level of controllers in the network is quantified based on the characteristics of SDN. Then, an adaptive synchronization strategy is proposed. By the use of this strategy, controllers are divided into three kinds of roles, and the synchronization period is adjusted dynamically by an adaptive function according to the current network state to attain a certain consistency level. The simulation results show that the proposed strategy has merits of reducing communication overhead of controllers as well as improving network availability compared with the other non-adaptive strategies. However, it lacks consideration of load balance among controllers, which will be our future research direction.

INDEX TERMS Multiple controllers, consistency, periodic synchronization, communication overhead, availability.

I. INTRODUCTION

Software Defined Networking (SDN) is developed to address the limitations of traditional network in satisfying the currently complex requirements of network [1]. SDN decouples the control plane logic from the data plane by unloading networking control functions from forwarding devices to the logically centralized controller [2], and provides programmability for network services. In principle, SDN relies on the logically centralized controller using the global view which enabled by the OpenFlow protocol [3] to manage the entire network. However, one single controller in the network usually leads to poor scalability, low reliability and high response time [4], [5]. A simple scheme is to deploy multiple distributed controllers working cooperatively as one logically centralized controller in SDN.

In such case, SDN is divided into multiple domains with multiple controllers running simultaneously, and each controller manages a number of local switches within its own domain. The controllers deployed in SDN can obtain the current state of network by polling statistics from edge switches in its control domain. Controllers in different domains need to communicate with each other to exchange information across domains in order to synchronize their state and then achieve the logically centralized control [6]. There have been a surge of researches on different aspects of SDN, such as network architecture [7], distributed controller architecture [8], inter-domain communication platform [9] and application [10], and routing scheme [11].

In SDN, the network state is stored into the data structure of the controller, which is called Network Information Base (NIB) [12]. Here NIB is oriented to applications based on the global network state, and then the controller can make forwarding decisions for new flows according to NIB. The NIB of each controller records different contents. For instance, the NIB contents presented to a network load-balancing SDN control application would include at least link capacity and utilization state, and what we concern in this paper is the flow update rate of each controller. Furthermore, the NIB of each

controller is updated periodically and independently based on the statistics about the state collected from the physical network. One of the primary problems to be solved when deploying multiple controllers in SDN is the information consistency among controllers [13], because all controllers in the network must share information and communicate with each other to synchronize the network state in their NIB in order to realize the logically centralized control. If the state synchronization among controllers is not in time, the transient inconsistency among controllers may emerge and lead to some conflicting decisions, for example, the stale and inaccurate state of links may affect correctness [14] and quality of service of routing [15], [16]. However, the frequent synchronization among controllers may significantly degrade network performance, such as increasing network overhead and leading to load imbalance among controllers. Consequently, it is necessary to develop the optimized controller synchronization scheme in multi-domain SDN to improve network performance.

There are some proposed consistency models to improve the network performance. The so called strong consistency model works on a consistent network view [12] and ensures that the NIB information of all controllers is same, and never brings the inconsistent NIB to controllers, however, it may need more frequent state synchronization among controllers and thus lead to excessive overhead and delay in control plane. The so called eventual consistency model synchronizes the network state information periodically when it is enabled, and then the NIB of controller can be updated. Therefore, it has less communication latency and overhead than the strong consistency model, however, it may potentially lead to the temporarily inconsistent network view and then make the improper forwarding decisions.

In this paper, we study the controller consistency problem which is formulated as an optimization problem. We propose an adaptive controller state synchronization strategy to attain a certain level of consistency with communication overhead reduced and network availability improved. The major contributions of this paper are summarized as follows.

- We define the suitable metrics to measure consistency, performance and availability among multiple controllers based on the characteristics of SDN, and build their quantitative analysis model.
- The controllers in the network are divided into three kinds of roles, which are Leader, Acceptor and Learner. We propose an adaptive consistency strategy by tuning synchronization period dynamically according to the predefined metrics.
- We compare the performance of the proposed strategy with the other non-adaptive strategies under one proof-of-concept distributed SDN application, and then demonstrate that the proposed strategy can reduce communication overhead and improve network availability. It can make network be more resilient to network state changes than the non-adaptive strategies.

The rest of this paper is organized as follows. Section II reviews the related work. Section III formulates the problem to be solved and defines the performance metrics and models. Section IV expounds the adaptive consistency strategy for multiple controllers based on periodic synchronization. Section V simulates the proposed strategy and evaluates its performance. Finally, this paper is concluded in Section VI.

II. RELATED WORK

There are some schemes of controller state synchronization based on Periodic Synchronization (PS), which conduct state synchronization among controllers within the specific period. In [12], two kinds of schemes, i.e., Link Balancer Controller (LBC) and Separate State Link Balancer Controller (SSLBC) are proposed. In LBC, each controller updates and synchronizes its NIB periodically, and directs the new flow to the currently least-loaded switch in the entire network. In SSLBC, the NIB of each controller is divided into two parts which are local NIB and remote NIB. The local NIB records the information of switches in local domain and does update in real time, while the remote NIB records the information in other domains except its local domain and does update periodically to achieve state synchronization. The local controllers in SSLBC detect the load variation of switches in real time and then distribute forwarding rules to the least loaded switch based on the current status of the local domain, thus SSLBC outperforms LBC in load balance by the same times of synchronization. Nevertheless, SSLBC may cause some false scenarios, such as forwarding loop and black hole [12], [17]–[20], due to the inconsistency in the interval between two consecutive synchronizations.

There are also some works regarding non-PS schemnes. A Load Variance-based Synchronization (LVS) scheme for controllers is proposed in [2] to improve load balancing performance in the multi-controller multi-domain SDN network. There are two specific schemes in LVS, which are the Least loaded Server Variation Synchronization (LSVS) and the Least loaded Domain Variation Synchronization (LDVS). LSVS is used to solve the forwarding loop problem, and LDVS can lower synchronization overhead. An elastic consistency protocol based on heuristic algorithm is proposed in [21]. It can coordinate the rate between consistency synchronization and network events update, in order to reduce communication overhead and improve availability of SDN.

In the traditional network, the forwarding loop can be relieved after the NIB of controllers become consistent again, because the flows are forwarded separately and the new NIB will be valid for new flows. However, the forwarding rules of flows are absolutely decided by controllers, and the flows are removed by time-out scheme or controllers in SDN. Therefore, the forwarding loop may last for a long time, which could lead to packet loss and communication failure. The consistency strategy based on configuration number is proposed in [13] to solve forwarding loop problem. There are two tables to separately store old NIB before updating network status and new NIB after synchronizing. Each table has a unique

specific configuration number, and each new flow is specified with the configuration number by the ingress switch. When a flow enters a domain, its configuration number is checked to select which table to use. It ensures that each flow is processed according to either the old NIB or the new NIB but not the mixture of these two, and then avoids forwarding loop. Several studies are based on LBC, such as consensus routing [14], [22] and consensus algorithms [23]. Their goal is similar to that of the controller consistency problem in this paper, and accordingly some effective strategies are developed to overcome communication failure such as disconnection or black hole during network information update. The consistent state of the distributed system with LBC is more available and securable.

A probabilistically bounded staleness model is proposed in [24] to predict the strength of eventual data consistency among distributed nodes, and it provides a reference of weighing consistency and network performance. However, it works at general distributed system but does not provide the specific optimization method by considering SDN characteristics. In [25], a hierarchical framework is proposed to realize the consistency among multiple controllers. Two layers of controllers at the top and bottom form tree structure, and controllers at the lower layer manage local switches while controllers at the upper layer coordinate synchronization in order to maintain the consistency of network view. Nevertheless, this strategy only changes the communication mode of multiple controllers, but does not relieve conflict between consistency and performance. A fault tolerant SDN controller platform called Ravana is introduced in [26], the replicated state machines can be extended with the lightweight switch-side mechanisms to guarantee correctness, and the complex consistency protocol does not involve switches. Reference [27] employs the following two online clustering techniques: sequential k-means and incremental k-means, to map a given application performance indicator into a feasible consistency level that can be used with the underlying tunable consistency model. In [28], a simple coordination layer is designed. It can turn a set of single-image controllers into a distributed SDN system with consistency achieved.

At present, the consistency of SDN controller is achieved mainly through the distributed data storage system. The selection of the consistency protocol, data interaction and synchronization method is limited, thus it is difficult to meet the diversified demands of the applications and their flows optimization. Some existing studies do not provide the specific optimization methods for the general distributed systems in combination with the characteristics of SDN. Some methods only change the communication mode among multiple controllers and do not alleviate the conflict between consistency and performance. Moreover, the research on controller consistency mainly focuses on design and implementation, mostly on the qualitative research on different levels of consistency, and often lacks further quantification and optimization. Therefore, how to achieve the minimum synchronous communication overhead and the highest availability under the condition of meeting the application consistency requirement is the research focus of this paper.

III. PROBLEM FORMULATION

The control plane of SDN is comprised of multiple controllers which are implemented in HyperFlow structure [4]. The set of OpenFlow switches is divided into multiple domains and deploys one controller per domain. Each controller is responsible for deciding routing paths for the switches within its domain. Connection of controllers and switches can be logically considered as the communication between control plane and data plane in a two-tier structure, as shown in Fig. 1. We represent the network as an undirected graph $G(V, E)$, where V is the set of switches and E is the set of edges in the network. $C = \{c_i | i = 1, 2, \cdots, n\}$ is the set of controllers, and there are n controllers working cooperatively in the network. Table 1 shows the notations used throughout this paper.

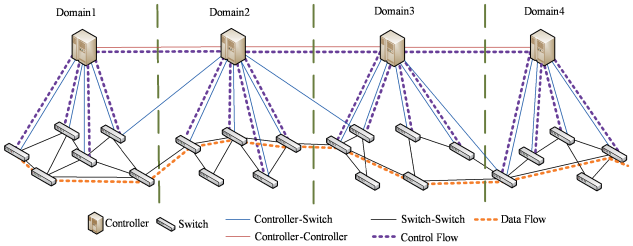

FIGURE 1. The multi-domain SDN control network.

TABLE 1. Notations used in this paper.

Notation	Meaning
e_{ij}	the type of network events involved in controller c_i
m	the number of network events
μ_j	the impact factor of e_{ij}
$\lvert\triangle e_{ij}\rvert$	the maximum number of network events that have updated but not synchronized yet in time
con_i	the consistency level of each controller
Con	the consistency level of the whole network for certain application
Con_t	the consistency level threshold of each controller
η	a constant which depends on the used consistency protocol in the network
v_{ij}	the generation rate of network event e_{ij} on c_i
cv_{ij}	the update rate of each controller
ca_i	the capacity of each controller
sy_i	the controller synchronization rate
Sy	the minimum rate of communication in the whole network
n_{ac}	the number of events that can be accepted
t_i	the recovery time of the failed controller
Av	the availability of the whole network

A. CONSISTENCY LEVEL

The network view among controllers may become inconsistent when controllers cannot share the updated information timely, for example, due to their local caching or communication delay. We introduce the diversity factor of network events to weigh the consistency among controllers. The greater the difference of network events among controllers is, the weaker

the consistency of the network will be. There are different types of network events on the controller, such as new host nodes joining in, link interrupt, traffic load update, etc. Moreover, the consistency depends on the specific application heavily. For instance, for path calculation application it is a weak consistency protocol, while for load balance application it could be a strong one. Meanwhile, different network events have different effects on the consistency of the same upper application. For example, for the consistency of path calculation application, the change of topology is more influential than the change of traffic load. Overall, the consistency analysis should aim at a specific upper application, and depends on the type and number of network events involved in this application. For a certain application deployed on multiple controllers, we define the consistency level of each controller c_i as follows.

$$con_i = \sum_{j=1}^{m} \mu_j |\triangle e_{ij}| \tag{1}$$

Here, e_{ij} $(1 \leq j \leq m)$ denotes the type of network events involved in controller c_i, and there are m network events; μ_j denotes the impact factor of e_{ij}; $|\triangle e_{ij}|$ denotes the maximum number of network events that have updated but not synchronized yet in time. Then, the smaller the con_i is, the stronger the consistency of this controller will be. We further define the consistency level of the whole network for the certain application as follows.

$$Con = \sum_{i=1}^{n} con_i = \sum_{i=1}^{n} \sum_{j=1}^{m} \mu_j |\triangle e_{ij}| \tag{2}$$

Specifically, when $con_i = 1, \forall j \neq i, con_j = 0$, it means that only one controller has event update in SDN control plane, which is corresponding to a redundancy backup controllers network for enhancing reliability. When $\forall i, con_i = 1$, it means that each controller has at most one event to be synchronized, corresponding to a strong consistency in distributed multi-controller network.

B. PERFORMANCE

The performance of consistency among controllers is mainly limited by two factors, which are synchronization load of one single controller and communication overhead among controllers. We analyze their relationship with consistency, in order to investigate the balance of performance and consistency.

The synchronization load of one single controller primarily refers to the synchronous requests sent locally or received remotely. It can be quantified as the number of synchronization in unit time. In order to reduce communication overhead, controller synchronization is activated according to the given consistency level threshold Con_t. Con_t is the upper bound of the consistency level of each controller c_i. The synchronization is performed when the consistency level of all controllers reaches the threshold Con_t, and then the minimum communication overhead can be obtained through the consistent synchronization among controller nodes. The controller synchronization rate can be calculated as follows to weigh the synchronization load of one controller.

$$cv_{ij} = \frac{\eta(n-1)\sum_{j=1}^{m} \mu_j v_{ij}}{Con_t} \tag{3}$$

Here, η is a constant which depends on the used consistency protocol in the network, and v_{ij} is the generation rate of network event e_{ij} on c_i, which is the amount of updates generated per unit time. The update rate of each controller in the network is cv_{ij}, which is the sum of all updated network event rates on c_i.

We use the number of communication per unit interval to weigh the communication overhead among controllers [29]. Therefore we can further compute the minimum rate of communication in the whole network as follows.

$$Sy = \sum_{i=1}^{n} cv_{ij} \tag{4}$$

That is, (3) and (4) jointly express the relationship between performance and consistency.

C. AVAILABILITY

Availability is another factor which affects the consistency. In a multi-controller network that requires strong consistency, the network view among controllers becomes different when some controllers break down. At the moment, the control plane cannot continue to work, and then become unavailable because it cannot achieve the strong consistent network view. If the controller reduces the consistency intensity and partial network events are updated in the local cache, then the network can tolerate the short time failure.

We define the availability as n_{ac}/n_{su} [29], where n_{ac} denotes the number of update events that can be accepted, and n_{su} denotes the number of events that have been submitted. n_{su} depends on the update rate of network events involved in the upper application, and is not related with the consistency parameters, thus we only concentrate on n_{ac} in this paper. The availability of each controller is considered both in the case of normal operation and in the case of failure. For a controller c_i, if it is working properly, when the consistency constraint is satisfied, the maximum value of n_{ac} is con_i. On the other hand, if the controller failed, for the recovery time t_i of the failed controller, the maximum value of n_{ac} is $t_i \sum_{j=1}^{m} \mu_j v_{ij}$. In view of improving the availability of the controller in the worst case, the availability of each controller can be defined as follows.

$$n_{ac} = min\{con_i, t_i \sum_{j=1}^{m} \mu_j v_{ij}\} \tag{5}$$

The availability of the whole network can be defined as follows.

$$Av = \sum_{i=1}^{n} n_{ac} \tag{6}$$

D. OPTIMIZATION OBJECTIVE

After the quantitative analysis of the consistency, performance and availability of multi-controllers in SDN, the balance among them is studied in this section to obtain the maximum benefit. The optimization objective is to ensure a certain level of consistency with network overhead minimized and network availability maximized. It is defined as follows. The stronger the consistency level of network, the lower the communication overhead, the higher the network availability, then the smaller the F value.

$$\begin{aligned} &minimize\ F = \lambda_1 \cdot \lg Con + \lambda_2 \cdot \lg Sy + \lambda_3 \cdot \lg \frac{1}{Av} \\ &s.t.\ i)\ 0 < \mu_j \leq 1 \\ &\quad ii)\ 0 \leq |\triangle e_{ij}| \leq m \\ &\quad iii)\ cv_{ij} = \sum_{j=1}^{m} \mu_j v_{ij} \\ &\quad iv)\ cv_{ij} \leq ca_i \end{aligned} \tag{7}$$

Here, λ_1, λ_2 and λ_3 are the weights to balance the above three factors, $0 \leq \lambda_1, \lambda_2, \lambda_3 \leq 1, \lambda_1 + \lambda_2 + \lambda_3 = 1$. λ_1, λ_2 and λ_3 can be set according to their relative importance in the viewpoint of the corresponding application by experience. For example, when we only highlight the consistency level of network, we can set $\lambda_1 = 1, \lambda_2 = 0$ and $\lambda_3 = 0$. Apparently, with Eq. (7), the multiobjective optimization problem is transformed into a single objective optimization one. Constraint i) and Constraint ii) are numerical constraints. Constraint iii) means that the update rate for each controller is equal to the sum of all updated network event rates on that controller. Constraint iv) means that the number of flows each controller process can not exceed its maximum capacity.

IV. ADAPTIVE CONSISTENCY STRATEGY

The adaptive consistency strategy can achieve certain consistency level of network by changing the synchronization period based on the specific metrics and support the reliability and robustness of control plane under the frequent changes of network state. The adaptive consistency for distributed multiple controllers turns the problem of inconsistency into the automatic control problem. At first, the controllers are divided into three different roles, and then controllers synchronize the network information periodically to maintain the consistency.

A. CONTROLLER ROLE

The controllers in the network are divided into the following three kinds of roles, and each controller can take multiple roles simultaneously. (i) Leader: a controller is elected as the leader to gather network information from other controllers, and then send the current global network information to others periodically. (ii) Acceptor: each controller can be an accepter and send query to inquire the dataplane state and push network state change events to the Leader. When the Acceptor receives the current network information sent by the Leader, it updates its NIB to synchronize the information. (iii) Learner: each controller can be a learner, and learns the network state change from the notifications of the other controllers, and then make the corresponding decisions. Under the normal circumstances, at most one controller plays the Leader role, and other controllers act as the Acceptors, and all controllers act as the Learners. Note that all Acceptors communicate with the Leader but they do not communicate with each other directly, in order to reduce communication overhead and improve network reliability. The working process of a Leader is mainly divided into the following two phases. Learning phase is to learn from other controllers about the changed data. When a controller becomes a Leader, it should be aware of all the network events sent by other controllers, and start an active learning process immediately. The synchronization phase allows all controllers to keep network information consistency by forwarding the current network information to Acceptors.

The leader election algorithm counts the voting results and selects the recommended leader controller. An odd number of controllers are deployed in the network to elect the Leader. The election thread is held by the current controller that initiates the election. The election thread first initiates one query to all controllers (including itself), the inquired controller responds to it according to the controller capacity which is the number of flows that the controller can process simultaneously. The larger the capacity of the controller is, the more capable it is to act as a leader in network consistency coordination. After receiving the reply, the election thread obtains the leader information proposed by other controllers. The information is stored in the voting record table for the current election. Finally the controllers are queried, the statistical results are counted to figure out which controller wins after the last query (It may be itself or another controller depending on the result of the votes, but each controller votes for itself in the first round of voting). If one controller obtains not less than $n/2 + 1$ votes, the controller is set as the currently recommended leader. Set the controllers status based on the winning controller information. Each controller repeats the above process until the Leader is elected. The following Algorithm1 depicts the procedure of the Leader election.

The Leader election ensures that only one Leader is produced. Note that when Acceptors are unable to communicate with the Leader, they believe that the Leader has already failed and starts re-election. If the connections between the Leader and Acceptor are less than half of the number of Acceptors, the Leader is revoked and the Leader election is re-started.

B. PERIODIC SYNCHRONIZATION

The Leader pushes the global network state to other controllers periodically, and then Acceptors update their own NIB to realize the information consistency among controllers. We use the δ consistency model [30] in this paper under which all the controllers share network information every δ time units (e.g. synchronization period) and then maintain the same network state in each NIB.

Algorithm 1 Algorithm of Leader Election

Input: candidate Leader controllers $c_i \in C$
Output: Leader controller c_l
Begin
01: **while** there is no Leader or the connection between Leader and Acceptor does not satisfy the condition **do**
02: **for** $c_i \in C$ **do**
03: Initialize voting: Each controller votes for itself at the beginning and broadcast the votes to the controller cluster;
04: Collect the votes: Collect all the current votes of the controller nodes;
05: Count the votes: Count the votes for each controller node and generate a new vote for itself;
06: **if** Election success (More than half of the votes in a controller c_l) **do**
07: c_l is selected;
08: break;
09: **end if**
10: **end for**
11: **end while**
End

We propose an adaptation function $f(\varphi)$ to change synchronization period dynamically according to the current consistency level of the network. The adaptation function accepts the *Con* as the input and returns a value s as the new synchronization period from the set S. In this paper, the set of the allowed periods is $S = \{1, 2, 4, 8, 16, 32\}$. The value of s is selected to be powers of 2 (2^{α}), where $2^{\alpha_{min}} \leq s \leq 2^{\alpha_{max}}$. $\triangle u$ and $\triangle b$ denote the upper and lower margins of the network consistency level threshold $nCon_t$ respectively, and they are used to prevent too frequent synchronizations if the difference between *Con* and $nCon_t$ is very small.

$$f(\varphi)=\begin{cases} 2^{min(\log_2(s)+1,\alpha_{max})}, & 0 \leq Con < nCon_t - \triangle b \\ s, & nCon_t - \triangle b \leq Con \leq nCon_t + \triangle u \\ 2^{max(\log_2(s)-1,\alpha_{min})}, & nCon_t + \triangle u < Con \leq mn \end{cases} \tag{8}$$

The adaptation function divides the value of the current consistency level *Con* ($0 \leq Con \leq mn$) into three intervals. The first interval $0 \leq Con < nCon_t - \triangle b$ means that *Con* is below the network consistency level threshold with strong consistency. The synchronization period should be increased to decrease the number of unnecessary synchronization messages among controllers and further reduce the network overhead. The second interval $nCon_t - \triangle b \leq Con \leq nCon_t + \triangle u$ means that *Con* is within the reasonable range and is considered moderate, thus the synchronization period should remain the same. The third interval $nCon_t + \triangle u < Con \leq mn$ means that *Con* is above the network consistency level threshold with weak consistency, that is, the information inconsistency among controllers is severe, and the synchronization period should be decreased to exchange network information among controllers in time and further improve the network performance. In this paper, the adaptation function is triggered every 2 seconds.

Algorithm 2 Algorithm of Controller Synchronization and Adaptive Consistency Level

Begin
Input: initial consistency level *Con*, Con_t, $\triangle b$, $\triangle u$
Output: s, adaptive consistency level *Con*, *Sy*, *Av*
01: Calculate current *Con* according to Eq. (2);
02: **if** $0 \leq Con < nCon_t - \triangle b$ **then**
03: increase s according to the first interval of $f(\varphi)$;
04: **else if** $nCon_t + \triangle u < Con \leq mn$ **then**
05: decrease s according to the third interval of $f(\varphi)$;
06: **else**
07: s remains the same;
08: **end if**
09: Calculate *Con*, *Sy* and *Av* respectively to evaluate the network performance under current s;
End

The following Algorithm2 depicts the state synchronization procedure and adaptation of consistency level. The initial synchronization period of controllers is selected from S randomly.

According to the above statements, we present a theorem about the time complexity of the proposed adaptive consistency strategy as follows.

Theorem: The time complexity of the proposed strategy is $O(n^2)$, where n is the number of the deployed controllers.

Proof: The strategy consists of Algorithm 1 and Algorithm 2. The computation time of Algorithm 1 mainly depends on the election of the Leader controller, and Algorithm 2 mainly depends on the change of synchronization period according to the consistency level in the network.

1) ELECTION OF THE LEADER CONTROLLER

The computation of Leader election is divided into three parts, i.e., vote initialization e_i, collection e_{co} and count e_c. In the worst case, each controller holds the election thread until the Leader is selected, that is, the election algorithm is executed n loops. The computation time of Leader election is denoted by N_e as follows, where the time complexity of e_i, e_{co} and e_c are all equal to $O(n)$.

$$N_e = O(n(O(e_i) + O(e_{co}) + O(e_c))) \sim O(n^2) \tag{9}$$

2) PERIODIC SYNCHRONIZATION

The computation of periodic synchronization depends on the adaptation function $f(\varphi)$. It is divided into three parts, i.e., synchronization period increasing s_i, decreasing s_d and invariable $s_i n$ as follows. There execution time is constant and equal to $O(1)$. Thus, the computation time of periodic synchronization, N_f, is as follows.

$$N_f = O(O(s_i) + O(s_d) + O(s_{in})) \sim O(1) \tag{10}$$

Since two algorithms are performed in the sequential order, the time complexity of the proposed strategy depends on Eq. (9) and Eq. (10), that is

$$N_e + N_f \sim O(n^2) + O(1) \sim O(n^2) \tag{11}$$

V. EVALUATION

A. SIMULATION SETUP

The proposed strategy has been implemented on the Intel(R) Core(TM) i5-4590, 3.30GHz CPU, 4G RAM over Ubuntu 14.04 LTS system. We implement a Mininet [31] simulator to simulate the consistency interaction among multiple controllers based on Internet2 OS3E [32]. The controller used in this implementation is Ryu [33], and the software switch used is Open vSwitch [34]. As shown in Fig.2, Internet2 OS3E has 34 nodes and 42 links. Each node denotes an independent university or organization and usually needs to deploy an SDN controller. We suppose there is a controller deployed on each node, each controller maintains a communication channel with every other controller. A controller sends its local information to other controllers through these communication channels during synchronizations.

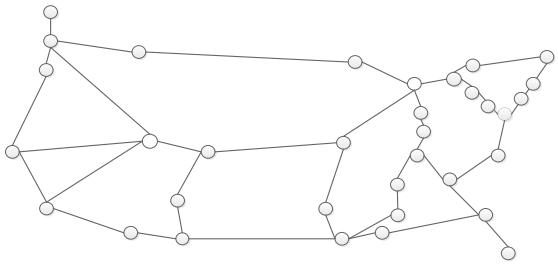

FIGURE 2. Internet2 OS3E topology.

In order to evaluate the validity and effectiveness of the proposed strategy, we implement a distributed load balancing application running on the top of controllers. We use the statistical data of Stanford University Network [35] and ignore the type of the specific event. The load distribution of each controller acts as the variation of update rate of each node over time. The average update rate of network events per node in Internet2 OS3E is randomly selected within [3500, 7000].

When the controller node c_i synchronizes a network event e_{ij}, the data grouping generated is usually small, averaging about 1KB with a maximum of 4KB in the experiment [36]. For the reasonable parameters of the proposed adaptive strategy, we set the global constraint of consistency level $nCon_t = 1500$, upper and lower margins of the network consistency level threshold $\triangle u = 200$ and $\triangle b = 100$ by experience in the simulation.

We compare the results of the adaptive synchronization strategy proposed in this paper (which is called AS for short) with the non-adaptive strategies. We set the initial synchronization period of the adaptive strategy as 8 seconds. We use PS plus synchronization period to denote the non-adaptive synchronization strategy with different synchronization periods for convenience. For example, PS-4s is short for the non-adaptive strategy with a fixed synchronization period of 4 seconds. We simulate AS, PS-4s, PS-2s and PS-1s strategies to evaluate the network performance next.

B. RELATIVE NETWORK PERFORMANCE

In this paper, the network performance is measured by the consistency level, communication overhead and availability jointly. The shorter the synchronization period, the stronger network consistency level can be guaranteed, while the availability cannot be ensured once some controllers failed. The longer the synchronization period, the weaker the network consistency level, thereby the communication overhead is reduced and network availability is improved. The evaluation criterion is the proposed objective function value F defined in Eq. (7). We define the relative performance value of the algorithms as follows.

$$RP = F/optimal\ value \tag{12}$$

We use the brute force algorithm to get the optimal result and evaluate all the above algorithms. It generates all possible failure scenarios of controllers, measures the performance and returns the best one. $RP = 1.0$ implies that the algorithm finds an optimal solution, and the value which much closer to 1.0 means that the corresponding synchronization period has much better impact on network performance. Fig.3 depicts the impact of the synchronization period of controllers over time on the network performance.

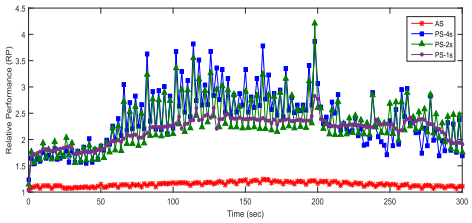

FIGURE 3. Relative network performance over time.

The proposed AS scheme in this paper is the best approach to find the optimal solution. It has the best influence on the network performance comprehensively and superior to the other non-adaptive strategies significantly. The relative performance value of AS is about 52.01% better than PS-4s, 50.24% better than PS-2s and 47.53% better than PS-1s on average respectively. The performance of AS scheme over time is relatively stable.

We then evaluate the consistency level, synchronization period, communication overhead and network availability of the proposed adaptive strategy and the other non-adaptive strategies within 300 seconds in order to make simulation effectively.

C. CONSISTENCY LEVEL

The consistency level of these strategies over time are shown in Fig.4. The lower the consistency level is, the better the

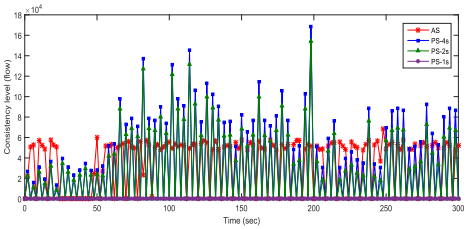

FIGURE 4. Consistency level over time.

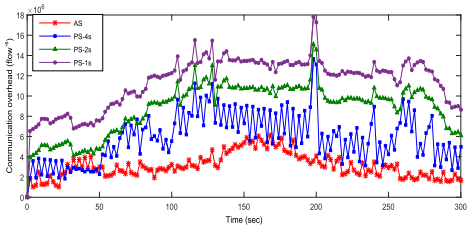

FIGURE 6. Communication overhead over time.

network performance will be. The value of consistency level is equal to 0 at the time of synchronization. As shown in Fig.4, the consistency level of AS is worse than the non-adaptive ones at the beginning, because the initial synchronization period of AS is set to be longer than the other non-adaptive strategies. The consistency level is affected greatly by the load of network condition variation in the case of non-adaptive strategies in general. While in the case of the adaptive strategy, the consistency level is much more stable under the drastic change of the network load. The reason is that the synchronization period of the proposed strategy can be adjusted adaptively according to the network changes to attain a certain level of network consistency, while the synchronization periods of non-adaptive strategies are fixed.

PS-1s can be considered as the strong real-time synchronization consistency strategy, thus the consistency level of AS is about 10.03% less than PS-4s and 34.27% more than PS-2s on average respectively except PS-1s. Therefore, AS outperforms the non-adaptive ones in general. For the non-adaptive strategies, the consistency level is better with the shorter synchronization period, thus the performance of PS-1s is best.

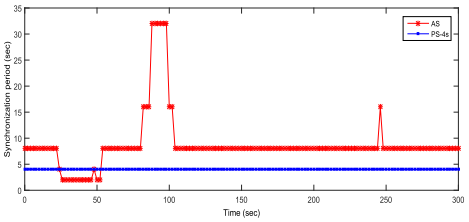

FIGURE 5. Synchronization period over time.

D. SYNCHRONIZATION PERIOD

Fig.5 shows the change of synchronization period of the adaptive strategy over time. We outline the constant period of PS-4s to show the comparison more clearly. It shows that the synchronization period is altered smoothly with the consistency level according to the predefined threshold interval. As shown in Fig.5, the synchronization period of AS changes with the network load variation. If the consistency level is low, the synchronization period is decreased, while if the consistency level is high, the synchronization period is increased.

E. COMMUNICATION OVERHEAD

As shown in Fig.6, the communication overhead of PS-4s is lower than AS only at very few time points, because the suitable synchronization period of AS is smaller than PS-4s at that time interval, and frequent synchronization leads to the relative high communication overhead. Compared with PS-2s and PS-1s, the communication overhead of AS is much lower. AS has no extreme value of communication overhead due to its elastic synchronization period based on the current consistency level. AS divides controllers into three different kinds of roles. A controller only communicates with the Leader to achieve synchronization, and then the frequent communication among controllers is avoided with synchronization overhead reduced significantly. We can see that AS has lower communication overhead than the non-adaptive strategies in general, and the communication overhead of all strategies vary with the network state.

The communication overhead of AS is about 47.35% less than PS-4s, 64.83% less than PS-2s and 72.88% less than PS-1s on average respectively. Therefore, AS can provide the better network performance. The communication overhead of the adaptive strategy changes more smoothly, which can adapt to the consistency constraint under different situations. For non-adaptive strategies, the communication overhead is lower with the longer synchronization period, thus the performance of PS-4s is best.

F. NETWORK AVAILABILITY

When doing simulation, we suppose that the failure can recover in the network in a short time, that is, $t_i = 0.01s$. The non-adaptive strategies PS-1s and PS-2s can be considered as the strong consistency ones, because the network events of each controller can be shared globally without delay (the synchronization period is small). As shown in Fig.7, we can see that the availability of PS-2s and PS-1s is constant and much lower than the adaptive one. The trend of network availability of PS-4s is about the same with AS. The network availability of PS-4s is about 39.98% less than AS, PS-2s is about 80.98% less than AS and PS-1s is about 94.40% less than AS on average respectively. In order to guarantee the network availability in the worst case, we consider the normal operation and failure of the controller, and set the fault recovery time of the controller. Moreover, AS is elastic and its consistency configuration among controllers can be coordinated with the network update rate. At the same time,

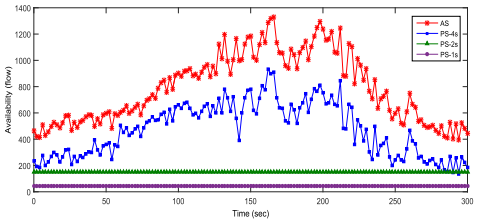

FIGURE 7. Availability over time.

the availability of the network improves with the adjustment of the consistency constraint. Therefore, AS is superior to the non-adaptive ones. For the non-adaptive strategies, the network availability is better with the shorter synchronization period, thus the performance of PS-1s is best.

On the whole, the adaptive strategy performs much better than the non-adaptive ones. Under the realistic distribution workload, AS achieves the better consistency level with the lower communication overhead and higher availability. For non-adaptive strategies, the consistency level and network availability improve with synchronization period shorten, meanwhile lead to high communication overhead due to frequent synchronization.

TABLE 2. Wilcoxon test results over Internet2 OS3E within 300 sec with respect to relative network performance.

Comparison	R^+	R^-	p
AS versus PS-4s	151	0	0
AS versus PS-2s	151	0	0
AS versus PS-1s	150	1	0

TABLE 3. Wilcoxon test results over Internet2 OS3E within 300 sec with respect to consistency level.

Comparison	R^+	R^-	R^0	p
AS versus PS-4s	79	39	33	0.009
AS versus PS-2s	70	47	34	0
AS versus PS-1s	0	109	42	0

TABLE 4. Wilcoxon test results over Internet2 OS3E within 300 sec with respect to synchronization period.

Comparison	R^+	R^-	R^0	p
AS versus PS-4s	136	13	2	0
AS versus PS-2s	138	0	13	0
AS versus PS-1s	151	0	0	0

G. WILCOXON-BASED STATISTICAL ANALYSIS

In order to make the experimental results more convincing, the Wilcoxon-based [37] statistical testing based on SPSS [38] is done. In this paper, the level of significance is set to be 0.01 and the confidence interval is set to be 99%. 5 tables are produced to report the Wilcoxon test results. We show 5 groups of Wilcoxon test results over Internet2 OS3E within 300 seconds with respect to the five metrics in Tables 2-6 respectively. It can be seen that all p value is close to zero and less than the set confidence level of 0.01, which indicates that AS has significant advantage.

TABLE 5. Wilcoxon test results over Internet2 OS3E within 300 sec with respect to communication overhead.

Comparison	R^+	R^-	R^0	p
AS versus PS-4s	137	13	1	0
AS versus PS-2s	150	0	1	0
AS versus PS-1s	150	0	1	0

TABLE 6. Wilcoxon test results over Internet2 OS3E within 300 sec with respect to availability.

Comparison	R^+	R^-	p
AS versus PS-4s	151	0	0
AS versus PS-2s	151	0	0
AS versus PS-1s	126	25	0

In summary, we can observe that AS has the optimal relative network performance, synchronization period, communication overhead and network availability under certain consistency level as well as the suboptimal consistency level over time generally. Thus, AS has better performance than other non-adaptive strategies comprehensively.

VI. CONCLUSIONS

In this paper, we studied the controller state synchronization issue in SDN. We proposed an adaptive synchronization strategy by adjusting synchronization period according to the current network state. We introduced and quantified the network consistency level firstly, and then defined communication overhead during synchronization and network availability as the evaluation metrics. We defined an adaptation function to change synchronization period among controllers dynamically according to the current consistency level. Simulation results showed that our proposed strategy can achieve lower communication overhead and higher network availability effectively compared with the non-adaptive synchronization strategies in general. The communication overhead of AS is about 47.35% less than PS-4s, 64.83% less than PS-2s and 72.88% less than PS-1s on average respectively. The network availability of PS-4s is about 94.40% less than AS, PS-2s is about 80.98% less than AS and PS-1s is about 35.38% less than AS on average respectively. The next work is to apply this method to the OpenDayLight controller, and use it as the east-west interface to realize the synchronization among controllers and test consistency effect in more real SDN environments. We also plan to extend the adaptive strategy to further reduce synchronization overhead of controllers while maintain reasonable load-balancing performance among controllers.

REFERENCES

[1] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (SDN)," *Comput. Netw.*, vol. 112, pp. 279–293, Jan. 2017.
[2] Z. Guo *et al.*, "Improving the performance of load balancing in software-defined networks through load variance-based synchronization," *Comput. Netw.*, vol. 68, pp. 95–109, Aug. 2014.

[3] "OpenFlow switch specification (version 1.3.0)," Open Netw. Found., White Paper, 2012.
[4] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Proc. INM/WREN*. Berkeley, CA, USA: USENIX, 2010, pp. 1–6.
[5] P. Lin, J. Bi, and H. Hu, "ASIC: An architecture for scalable intra-domain control in OpenFlow," in *Proc. ACM CFI*, Seoul, South Korea, Sep. 2012, pp. 21–26.
[6] F. X. A. Wibowo, M. A. Gregory, K. Ahmed, and K. M. Gomez, "Multi-domain software defined networking: Research status and challenges," *J. Netw. Comput. Appl.*, vol. 87, pp. 32–45, Jun. 2017.
[7] P. Helebrandt and I. Kotuliak, "Novel SDN multi-domain architecture," in *Proc. 12th IEEE Int. Conf. Emerg. Learn. Technol. Appl. (ICETA)*, Dec. 2014, pp. 4–5.
[8] K. Phemius, M. Bouet, and J. Leguay, "DISCO: Distributed multi-domain SDN controllers," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, May 2014, pp. 1–4.
[9] P. Lin *et al.*, "A west-east bridge based SDN inter-domain testbed," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 190–197, Feb. 2015, doi: 10.1109/MCOM.2015.7045408.
[10] R. Jahan, S. Shaik, K. Kotaru, and D. C. Kuppili. (Dec. 2016). *ODL-SDNi*. [Online]. Available: https://wiki.opendaylight.org/view/ODL-SDNi_App:Main
[11] V. Kotronis, A. Gämperli, and X. Dimitropoulos, "Routing centralization across domains via SDN: A model and emulation framework for BGP evolution," *Comput. Netw.*, vol. 92, no. 2, pp. 227–239, 2015, doi: 10.1016/j.comnet.2015.07.015.
[12] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically centralized?: State distribution trade-offs in software defined networks," in *Proc. HotSDN*, 2012, pp. 1–6.
[13] M. Reitblatt, N. Foster, J. Rexford, and D. Walker, "Consistent updates for software-defined networks: Change you can believe in!" in *Proc. 10th ACM Workshop Hot Topics Netw.*, New York, NY, USA, Nov. 2011, pp. 14–15.
[14] J. P. John, E. Katz-Bassett, A. Krishnamurthy, T. Anderson, and A. Venkataramani, "Consensus routing: The Internet as a distributed system," in *Proc. 5th USENIX Symp. Networked Syst. Des. Implement. (NSDI)*, San Francisco, CA, USA, Apr. 2008, pp. 351–364.
[15] M. Nikkhah, R. Guérin, and M. Nikkhah, "Migrating the Internet to IPv6: An exploration of the when and why," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2291–2304, 2016.
[16] G. Li, L. Boukhatem, and J. Wu, "Adaptive quality-of-service-based routing for vehicular ad hoc networks with ant colony optimization," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3249–3264, Apr. 2017.
[17] J. Xie, D. Guo, Z. Hu, T. Qu, and P. Lv, "Control plane of software defined networks: A survey," *Comput. Commun.*, vol. 67, pp. 1–10, Aug. 2015.
[18] A. Khurshid, X. Zou, M. Caesar, and P. B. Godfrey, "VeriFlow: Verifying network-wide invariants in real time," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, 2012, pp. 49–54.
[19] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, "Abstractions for network update," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, 2012, pp. 323–334.
[20] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling flow management for high-performance networks," in *Proc. ACM SIGCOMM Conf.*, 2011, pp. 254–265.
[21] J.-F. Li, J.-L. Lan, Y.-X. Hu, and J.-X. Wu, "Quantitative approach of multi-controller's consensus in SDN," *J. Commun.*, vol. 37, no. 6, pp. 86–93, 2016.
[22] D. Katabi, N. Kushman, and J. Wrocklawski, "A consistency management layer for inter-domain routing," Massachusetts Inst. Technol. Comput. Sci. Artif. Intell. Lab., Cambridge, MA, USA, Tech. Rep. MIT-CSAIL-TR-2006-006, Jan. 2006.
[23] A. Mostéfaoui and M. Raynal, "Intrusion-tolerant broadcast and agreement abstractions in the presence of Byzantine processes," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 1085–1098, Apr. 2016.
[24] P. Bailis, S. Venkataraman, M. J. Franklin, J. M. Hellerstein, and I. Stoica, "Probabilistically bounded staleness for practical partial quorums," *Proc. VLDB Endowment*, vol. 5, no. 8, pp. 776–787, 2012.
[25] S. H. Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, 2012, pp. 19–24.
[26] N. Katta, H. Zhang, M. Freedman, and J. Rexford, "Ravana: Controller fault-tolerance in software-defined networking," in *Proc. SOSR*, 2015, Art. no. 4.
[27] M. Aslan and A. Matrawy. (2017). "A Clustering-based consistency adaptation strategy for distributed SDN controllers." [Online]. Available: https://arxiv.org/abs/1705.09050
[28] A. Panda, W. Zheng, X. Hu, A. Krishnamurthy, and S. Shenker, "SCL: Simplifying distributed SDN control planes," in *Proc. NSDI*, 2017, pp. 329–345.
[29] R. Govindan, I. Minei, M. Kallahalla, B. Koley, and A. Vahdat, "Evolve or die: High-availability design principles drawn from Googles network infrastructure," in *Proc. ACM SIGCOMM Conf.*, 2016, pp. 58–72.
[30] Y. Huang, J. Cao, B. Jin, X. Tao, J. Lu, and Y. Feng, "Flexible cache consistency maintenance over wireless ad hoc networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 8, pp. 1150–1161, Aug. 2010.
[31] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proc. HotNets*, 2010, pp. 1–6.
[32] *Internet2 Open Science, Scholarship and Services Exchange*. [Online]. Available: http://www.internet2.edu/network/ose
[33] [Online]. Available: https://osrg.github.io/ryu-book/en/html/
[34] Open vSwitch. [Online]. Available: http://openvswitch.org/
[35] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, "Advanced study of SDN/OpenFlow controllers," in *Proc. 9th Central Eastern Eur. Softw. Eng. Conf. Russia*, 2013, Art. no. 1.
[36] 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 *Proc. 2nd Eur. Workshop Softw. Defined Netw. (EWSDN)*, Oct. 2013, pp. 38–43.
[37] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, 2011.
[38] Weaver Bill, "Scientific computing," vol. 23, no. 11, p. 46, Oct. 2006. [Online]. Available: https://www.ibm.com/analytics/spss-statistics-software

BANG ZHANG received the B.S. degree in software engineering from the Changchun University of Science and Technology, Jilin, China, in 2013, and the M.S. degree in computer science from Northeastern University, Shenyang, China, in 2015, where she is currently pursuing the Ph.D. degree. She has published more than 10 journal and conference papers. Her research interests include software-defined networking resources management.

XINGWEI WANG received the B.S., M.S., and Ph.D. degrees in computer science from Northeastern University, Shenyang, China, in 1989, 1992, and 1998, respectively. He is currently a Professor with the College of Computer Science and Engineering, Northeastern University. He has published more than 100 journal articles, books and book chapters, and refereed conference papers. His research interests include cloud computing and future Internet. He has received several best paper awards.

MIN HUANG received the B.S. degree in automatic instrument, the M.S. degree in systems engineering, and the Ph.D. degree in control theory from Northeastern University, Shenyang, China, in 1990, 1993, and 1999, respectively. She is currently a Professor with the College of Information Science and Engineering, Northeastern University. She has published more than 100 journal articles, books, and refereed conference papers. Her research interests include modeling and optimization for logistics and supply chain system.