Computer Communications 173 (2021) 170–182

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Linking handover delay to load balancing in SDN-based heterogeneous networks

Modhawi Alotaibi [a,b,*], Amiya Nayak [a]

[a] School of Electrical Engineering and Computer Science, University of Ottawa, Canada
[b] College of Computer Science and Engineering, Taibah University, Saudi Arabia

ARTICLE INFO

Keywords:
SDN
Vertical mobility
Handover
Load balancing

ABSTRACT

Software-Defined Networking (SDN) paradigm provides the ability to handle mobility more efficiently due to its programmability and fine granularity. However, in this emerging setting, the handover procedure still suffers delay due to exchanging and processing handover signaling messages. In this paper, we study the relevancy between an SDN controller's load and handover delay. We show that an over-loading state can prolong handover delay, so as a countermeasure, reaching that state is mitigated by applying a load balancing mechanism. Our primary metric is the controller's response time, as it directly affects the completion of any mobility-related procedure. We propose a load balancing management framework that deploys two concepts: *network heterogeneity* and *context-aware vertical mobility*. Our proposal is composed of three main aspects. First, we identify candidate users based on their context information. Second, we reduce the frequency of load dissemination between multiple controllers, and hence, reducing processing and communication overhead. Third, after the candidate users are determined, we optimize the decision problem on the selection among heterogeneous candidate networks. Through simulation, our framework has shown as much drop as a 28% drop in response time compared to previous proposals.

1. Introduction

Nowadays, we witness the wide-spread availability of mobile handheld devices such as smartphones and tablets. The ease and convenience provided by these devices lead to extensive and explosive growth of mobile traffic. Indeed, the traffic for mobility management is massive, and the cost is enormous due to connection establishment/release, handover, and tracking area update events [1].

There are several key benefits of SDN that can be advantageous to mobility management tasks; for instance, automated management, low-cost forwarding devices, network technology heterogeneity support, virtualization and segmentation support [2,3]. Behind all these key benefits is the fine granularity forwarding imposed by SDN, which is considered one of its design strengths [3]. It provides end-user-specific flow forwarding and, in return, can enforce policies per user, device, session, application, etc.; therefore, the SDN paradigm can be highly beneficial to a critical procedure such as the handover [4].

In this emerging setting, however, the handover still suffers delay due to the processing delay of signaling messages. The processing delay can get amplified if the managing controllers are congested. The authors of [5] proved that the load of the requests plays a significant role in processing delay. This remark was backed up by another study by Tootoonchian et al. in [6], where they linked a controller load to its reactivity to requests. To tackle this issue, we target the decision before a handover is in effect to make sure that the upcoming/handed-over users will experience a satisfactory level of service and mitigate long delays or breakage. That cannot happen if the new network's controller is over-loaded; therefore, we need to ensure having an efficient load balancing mechanism.

The difference in characteristics among heterogeneous wireless technologies imposes the challenge of designing mechanisms to integrate different access technologies, protocols, and service demands [7,8]. For instance, small cells, such as Wi-Fi hotspots, provide better data rates and cost-effective connections; yet, the security and privacy aspects can be violated. On the contrary, macrocells offered by cellular service providers, ensure higher security and privacy levels, although their services are costly. An interesting discussion was carried on by the authors of [9]. They addressed the challenges in current heterogeneous networks, as well as SDN promising features to solve some of these issues. The main concept that this paper suggests is "openness" in the wireless world. This concept is currently constrained due to the proprietary and closeness nature of different technology providers. The authors promoted the combined open service irrespective of infrastructure in order to gain access to more wireless capacity. This idea can be beneficial to different applications, specifically, load balancing.

* Corresponding author at: School of Electrical Engineering and Computer Science, University of Ottawa, Canada.
E-mail addresses: msotaibi@taibahu.edu.sa (M. Alotaibi), nayak@uottawa.ca (A. Nayak).

https://doi.org/10.1016/j.comcom.2021.04.001
Received 31 January 2020; Received in revised form 18 January 2021; Accepted 2 April 2021
Available online 15 April 2021

In SDN-based heterogeneous networks, a distributed set of controllers are built on top of different wireless infrastructures, where each controller manages a domain/technology. SDN controllers have to handle control traffic to accomplish mobility procedures, and since these controllers have limited resources, handling a large amount of traffic/flows originating from switches may cause delays [10]. In practice, as a cellular network is expected to accommodate more users compared to other types of networks such as Wi-Fi, these networks' resources can be complementary in order to relieve cellular resources [11]. Vertical mobility is a solution to relieve a congested network, where it can be achieved by switching the mobile device interface to be connected to another network/technology [8].

A controller's load has a direct relationship with its response time; hence, the bigger the load, the longer the response time and the worse the users' perception of services. Therefore, we approach the over-loading controller problem, specifically to minimize the maximum response time. We mainly address the following scenario. Given a cellular network of a controller managing network devices and mobile users, and multiple nested other wireless networks where each has a managing controller, how do we relieve the cellular controller by exploiting users' preferences and the complementary resources of other cooperative controllers? Here, our main contribution is an approach that mainly aims at vertically handing over some edge users, considering some context information regarding the users and controllers. Note that the vertical mobility in our proposal is "network-initiated". As a result, a controller's response time to any mobility-related procedure decreases. All in all, we argue that the inter-networking between heterogeneous wireless technologies can be realized in practice if a confederation is established between different operators and the concept of SDN is present to orchestrate such confederation.

In this work, we propose a management framework that includes three main parts. We identify candidate users based on their context information; we adopted a decision-making tool to include the user's input into our framework. Then, we proposed a novel mechanism that reduces the frequency of load disseminating between multiple controllers. Once the candidate users are determined, we optimize the decision problem on the selection among several candidate networks.

The work in this paper is divided into seven sections. Section 2 presents some of the approaches that have addressed load balancing among multiple SDN controllers. In Section 3, we describe our system components. We then go through the modeling and formulation in Section 4. Then, our management framework is explained in Section 5. Our experiments and evaluation are discussed in Section 6. Finally, we conclude the paper with Section 7.

2. Related work

The scalability of the control plane is a challenge for the SDN control logic centralization [12]. The negative impact of such a challenge is partially mitigated when the control logic is distributed/delegated into either a one level of controllers (i.e., flat) or two levels of controllers (i.e., hierarchical) [13,14]. In our direction of research, we argue that the context of network heterogeneity fits right in the multi-controller modeling given that heterogeneous technologies belong to different operators/service providers.

In the multi-controller setting, load balancing is crucial. Incorporating load balancing techniques into SDN multi-controller design enforces the availability and scalability aspects of a network to provide a minimal response time [15]. Prior research has shown that the control plane load-balancing is correlated to several factors (i.e., design choices), including the type of mapping between switches and controllers, and the network state dissemination method.

The switch-controller deployment can either be fixed (i.e., static) or dynamic (i.e., switch migration). Each deployment method has its advantages, disadvantages, and applications. The researchers, in this matter, have been divided into three groups.

One group of researchers has adopted the static assignment of switches to the controllers and exploited other strategies to balance the load among controllers [16–18]. This approach has several advantages, such as simple control plane management, reduced infrastructure and complexity costs, and flexibility to be applied to connected or disconnected domains. One of the initial studies in SDN control plane load balancing was done by Hu et al. [17]. The authors of this paper proposed a hierarchical architecture for wide area networks (WAN). Their approach is based on a partitioning algorithm that reallocates flows to different controllers; it assigns the nearest controller to the switch. Considering a flat implementation for the control plane, Koponen et al. proposed Onix [18], which is intended to scale out a network and avoid congestion situations. This approach used the Network Information Base (NIB) to allow Onix instances to store, retrieve, and maintain states. Though, as the network size grows, this method may introduce overhead due to memory restrictions. Also, this method did not advise for an over-loading countermeasure. In another work, the authors of [16] proposed a rounding algorithm that showed improvement in the controllers' response times by using link balancing. Nonetheless, they did not provide a clear explanation of how the distributed controllers exchange statuses. In the static assignment, however, some controllers are susceptible to overloading and failure due to heavy and fluctuating loads [19,20]. Also, it suffers from inefficient resource utilization when the load shifts, compromising the network's ability to react efficiently to changes, such as failure and updates [21].

Another group of scholars has focused on the switch migration between controllers to achieve the load balancing [22–26]. This approach provides more adaptability to sudden network events than the static assignment. Also, it improves the network QoS [20], resource utilization [19], and security [27]. The topic of switch migration was initially triggered by [28] and was supported by the multi-controller feature added to OpenFlow v.1.2 and later versions [29]. The primary issue in the dynamic assignment is the selection of a switch-controller pair for migration. A considerable body of this group of studies has adopted the greedy approach by selecting the heaviest switch to be migrated from an overly utilized controller to another lightly utilized controller [22–24,30]. However, the dynamic assignment, in general, comes with some issues, such as complexity, high cost, and high latency due to migration and re-association [31,32]. Switch migration frequency has to be limited in number and time as frequent and longer changes in mappings can lead to network instability [19].

The third group of scholars has presented a so-called partial switch migration, which means the flow requests generated by a switch are distributed among multiple controllers [19,31,32]. The authors in [32] proposed a dynamic distributed control plane architecture, where they can dynamically manage the number of controllers, switches, and control flow while considering the topology and application demands. Another approach proposed by Wang et al. in [31] was mainly dependent on the idea of balancing the load of multiple controllers based on shifting aggregate flows from the management of one controller to another. The proposed approach is implemented in a hierarchical static architecture, where a set of distributed controllers report their load information to a root controller. Then, the root controller determines on aggregating and redirecting the flows to ensure fairness among all the controllers. This approach is pro-active and based on traffic prediction. The aggregation of flows relies on installing wildcard rules on some switches which can be hardware-wise, costly. In a recent work by Al-Tam and Correia [19], they proposed a heuristic approach to migrate fractional flow requests while minimizing the number of new mappings. Their approach showed more resilience and better load balancing in large-scale networks. All in all, the partial switch migration approach implies that a single switch is connected to several controllers, which still suffers from some of the dynamic migration issues.

The applicability of the previous proposals can be seen in practice in datacenters or WANs, assuming the interconnectivity between different regions/domains. However, if these regions are autonomous and

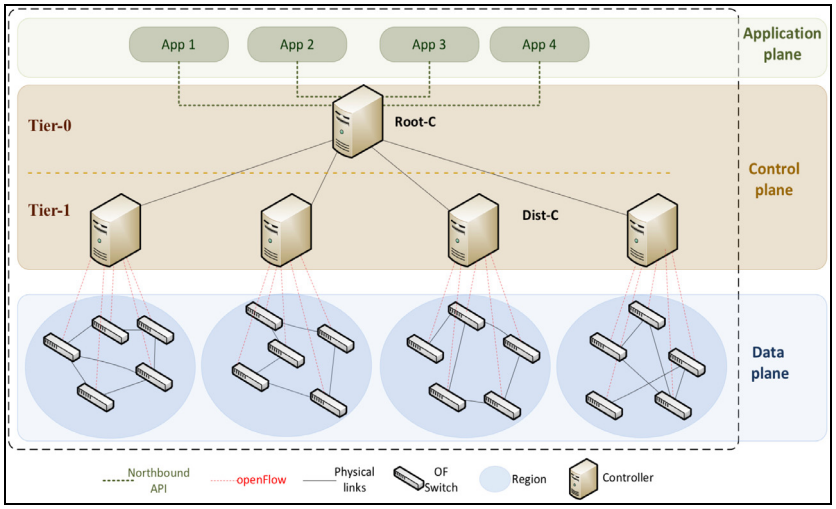


Fig. 1. SDN hierarchical architecture.

administratively disjointed, some of the approaches mentioned above may fail. With such a limitation, the static assignment is the sole option. Even though the static assignment is restrictive, its shortcomings can be alleviated in different ways, such as flow redirection, intelligent routing, controller placement, and exploiting users' mobility. Our approach is based on the static deployment, yet we exploit the dynamism of mobile users to shift loads from over-loaded controllers. However, as far as we know, no prior study has addressed disconnected domains, which can be a severe challenge for most load balancing strategies.

Besides the switch-controller deployment issue, one issue that contributes to degrading the performance of any multi-controller load-balancing mechanism is the controllers' status synchronization. A set of controllers need to inform each other of their load status to avoid forming choke points and, in some cases, make fair load distribution [30]. Consequently, the control overhead increases, which in turn, can result in a costly process, temporally and spatially [28]. In light of that, numerous studies have targeted minimizing the overhead of exchanging the status among controllers by decreasing the amount of load messages. For instance, the authors of [23] proposed Dynamic and Adaptive Load Balancing algorithm (DALB). Their mechanism requires each controller to collect its local load information periodically; when its load reaches a threshold, it collects other controllers' load information (i.e., load information aggregation). Their solution adjusts the triggering threshold to the average load of all controllers combined to minimize the frequent collection of statuses. In [30], Yu et al. proposed a mechanism based on reporting the load status periodically and then storing these controllers' information. When a controller becomes overloaded, there is no reactive collecting of information before making a decision locally, unlike [23]. Then, to reduce the frequency of load informing, they proposed an inhibition algorithm. Lan et al. in [22], make the frequency of status exchange dependent on a pre-defined threshold, the closer the load to the threshold, the more frequent the status exchange. They proposed the use of a distributed database to synchronize load information among the controllers. However, retrieving data from a database may cause unmanageable delays.

3. System description

We consider integrating the SDN paradigm into a network of heterogeneous technologies, such as LTE, Wi-Fi, and WiMAX. In our work, our system model is depicted in Fig. 1. We propose a hierarchy of controllers forming the control plane, where tier-1 controllers are different service providers (i.e., Dist-C) that manage different domains/technologies [33]. Each controller only manages a domain of a particular wireless network. We omit direct communication among them to ensure privacy, confidentiality, and guarantee that security standards are met. Additionally, a tier-0 controller represents the Root-C, where the connection and management between the different domains take place. Considering SDN-based heterogeneous networks, we argue that a hierarchical architecture provides better scalability over a centralized architecture and less delay, compared to a flat approach.

All the traffic that goes between the switches and controllers (i.e., through any southbound protocol) is control traffic. We adopt the OpenFlow southbound protocol as an enabler for our model.

As for the data plane, the Point of Attachment (PoA), whether they are base stations (BS) or access points (AP), are each connected to an OpenFlow-switch. MNs are hence, connected to PoAs. Within a domain, the set of MNs can be divided into edge-users (MN_e) that are within the coverage of other domain(s), and non-edge-users (MN_{ne}) that are within the coverage of only their current domain. Edge-users can be further divided into candidates and non-candidates, based on their mobility parameters, including the preference and position as we will explain in the next Sections.

4. Modeling and formulation

We consider a discrete-time model, where the length of each time slot matches the time scale at which the control requests can be recorded.

Our network is divided into M domains, where each domain is operated by a service provider, a single controller, and may represent a different wireless technology; an example is provided in Fig. 2. Thus, we have a set of M distributed controllers, C_i, where $i \in \{1, 2, \ldots, M\}$. Each C_i has a capacity in terms of the maximum number of control requests it can handle per a unit of time t, which is given as α_i. In order to handle traffic peaks, we advise including spare processing capacity, or a so-called decay factor β_i, for each C_i. The decay factor has a value of $\beta_i \in [0, 1]$; thus, the total capacity of C_i can be written as: $\alpha_i \beta_i$. Note that the decay factor can provide a network operator with the flexibility to adjust the capacity given different times of the day, congested places, or even handle synchronization overhead with the Root-C. On top of the set of controllers and connected to each of them is Root-C. The capacity of Root-C is not taken into consideration, as it does not handle control requests generated by the users; thus, the load on Root-C is beyond the scope of this work.

In our model, we define three adjacency matrices as in Fig. 3, as the following:

- The switch-controller adjacency matrix X is a binary $N \times M$ matrix that associates a group of switches to each controller. Each

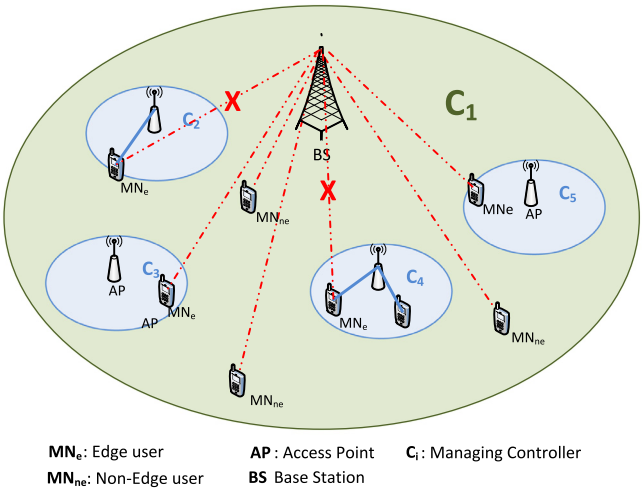


Fig. 2. The system model: consisting of a macrocell domain and multiple Wi-Fi domains.

area i consists of a set of inter-connected switches $s^i_j \in S$ where, $j \in \{1, 2, \dots, N\}$, and S represents the set of all switches. Note that each switch is connected to exactly one controller.

- The switch-user adjacency matrix Z_{s^i} is defined for each switch in region i to denote k associated users to n switches at time t, where $z_{bj} \in \{0, 1\}$, and $\sum_{j=1}^{n} z_{bj} \leq 1 \ \forall b$, $b = \{1, 2, \dots, k\}$. Knowing that the main focus of our work is on the hard handover, each user is connected to exactly one switch at any time t. Each switch is directly connected to a PoA, so we can refer to both s^i_j or its PoA as one entity. There are a limited number of channels provided by any PoA, where each channel can be assigned to one user. Let $l_{s^i_j}$ denote the number of channels provided by s^i_j; then, there are at most $l_{s^i_j}$ users connected to s^i_j, which means $\sum_{b=1}^{k} z_{bj} \leq l_{s_j}$.
- The user–controller adjacency matrix Y_t is defined as a binary $K \times M$ matrix to show the connectivity between the users and controllers. At any time t, each user has to be connected to only one controller; thus, the matrix reflects that: $\sum_{m=1}^{M} y_{xm} = 1$, where $x = \{1, 2, \dots, K\}$, and the number of users attached to each controller can be obtained from: $\sum_{x=1}^{K} y_{xm}$.

Flows (i.e., control traffic) that are reported to C_i by switch s^i_j, at time t is denoted as $\xi_{s^i_j}(t)$; therefore, the load on C_i, in terms of the total number of control requests, is given as (θ_i) and can be modeled as:

$$\theta_i = \sum_{s^i_j \in S^i} \xi_{s^i_j}(t) \tag{1}$$

Defining the load is essential to model the behavior of the controllers, since the load is closely related to the response time of the control traffic as the Little's theory showed [34]. In line with the results of [16], and inspired by their approach, we model Dist-C as $M/M/1$, and we assume that the flow requests follow the Poisson distribution. Therefore, the average response time of C_i, given its capacity and load, can be defined as:

$$R_{C_i}(t) = \frac{1}{\alpha_i \beta_i - \theta_i(t)} \tag{2}$$

In our model, $\bar{U}^i$ represents the set of all attached users to C_i. $\bar{U}^i$ can be further divided into edge users (MN^i_e) that reside in the overlap area with another domain, as in Fig. 2, and non-edge users (MN^i_{ne}) that represent all other users. Accordingly, the total attached users of a region i can be represented as $\bar{U}^i = MN^i_e \cup MN^i_{ne}$, where $|\bar{U}^i| = k$. These users are currently utilizing the C_i's resources.

The set S^i is composed of two subsets of switches, edge switches S^i_e and every other switch S^i_{ne}. Intuitively, users connected to the edge switches are edge users. This means;

$$\forall u_x \in MN^i_e \quad \exists\, s^i_j \in S^i_e \text{ , where } z_{xj} = 1$$

Naturally, the expected change in user attachments is reported by a subset of edge switches, S^i_e. Note that not all the traffic generated by the switches are user-related; there is other management-related traffic, including statistics [29]. Out of each switch s^i_j, a set of control messages is generated by user u_x with variable rates c^i_x (modeled as Poisson [24,35]); then, having k attached users to the area of controller C_i, at time t, makes the following:

$$\theta_i - \sum_{s^i_j \in S^i} c_j = \sum_{u^i_x \in \bar{U}} c^i_x \tag{3}$$

where c_j is management-related traffic generated by the switches; however, they are relatively small compared to the user-related traffic [10] and can be neglected. Therefore, we define the remaining capacity of C_i as (r^i), as follows:

$$r^i(t) = \alpha_i \beta_i - \sum_{u^i_x \in \bar{U}^i} c^i_x(t) \tag{4}$$

Proposition 1. *One main influencing factor on a controller's load, and hence its response time, is the number of attached users.*

$$\theta_i(t) = (|MN^i_{ne} + MN^i_e|) c^i_x \tag{5}$$

As a result, we can minimize the controller's load by reducing the number of attached users to that controller.

The traffic going through southbound channels for connected users, whether *Packet-in*, *Port-status* or any control requests, is better processed by controllers providing shorter response times. Therefore, our objective function is to minimize the maximum response time of an over-loaded controller, as follows:

$$\textbf{min} \quad max\ R_{C_i}(t) \tag{6}$$

$$\textbf{subject to}: \quad \theta_i \leq \alpha_{th_i} \tag{7}$$

$$\sum_{m=1}^{M} x_{jm} = 1 \quad \forall s_j \in S \tag{8}$$

$$\sum_{j=1}^{n} z_{bj} = 1 \quad \forall u_b \tag{9}$$

$$\sum_{b=1}^{k} z_{bj} \leq l_{s_j} \quad \forall s_j \in S \tag{10}$$

The set of the above constraints impose the following. No controller is overloaded, meaning that the load of a controller is less than that of its pre-defined threshold, α_{th_i}, (Eq. (7)). Each switch is managed by exactly one controller (Eq. (8)). To enforce hard handover, each user has to be connected to exactly one switch at a time (Eq. (9)). Each switch/PoA has a limited number of channels that it can provide the users (Eq. (10)).

In this case, load-balancing among tier-1 controllers is required to relieve an over-loaded cellular network controller. Therefore, we propose a heuristic approach to balance the load among the controllers based on context-aware vertical mobility. We design a framework that tackles three main aspects. First, it relieves the over-loaded controller by exploiting vertical mobility whenever it is applicable. Second, it reduces the status synchronization overhead. Finally, it optimizes the network selection for handed-over users.

$$X = \begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_N \end{array} \begin{array}{c} \begin{array}{ccccc} C_1 & C_2 & C_3 & \dots & C_M \end{array} \\ \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1M} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2M} \\ x_{31} & x_{32} & x_{33} & \dots & x_{3M} \\ \dots & \dots & \dots & \dots & \dots \\ x_{N1} & x_{N2} & x_{N3} & \dots & x_{NM} \end{bmatrix}_{N\times M} \end{array} \quad Z_{s^i} = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_k \end{array} \begin{array}{c} \begin{array}{ccccc} s_1 & s_2 & s_3 & \dots & s_n \end{array} \\ \begin{bmatrix} z_{11} & z_{12} & z_{13} & \dots & z_{1n} \\ z_{21} & z_{22} & z_{23} & \dots & z_{2n} \\ z_{31} & z_{32} & z_{33} & \dots & z_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ z_{k1} & z_{k2} & z_{k3} & \dots & z_{kn} \end{bmatrix}_{k\times n} \end{array} \quad Y_t = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_K \end{array} \begin{array}{c} \begin{array}{ccccc} C_1 & C_2 & C_3 & \dots & C_M \end{array} \\ \begin{bmatrix} y_{11} & y_{12} & y_{13} & \dots & y_{1M} \\ y_{21} & y_{22} & y_{23} & \dots & y_{2M} \\ y_{31} & y_{32} & y_{33} & \dots & y_{3M} \\ \dots & \dots & \dots & \dots & \dots \\ y_{K1} & y_{K2} & y_{K3} & \dots & y_{KM} \end{bmatrix}_{K\times M} \end{array}$$

Fig. 3. Modeling adjacency matrices.

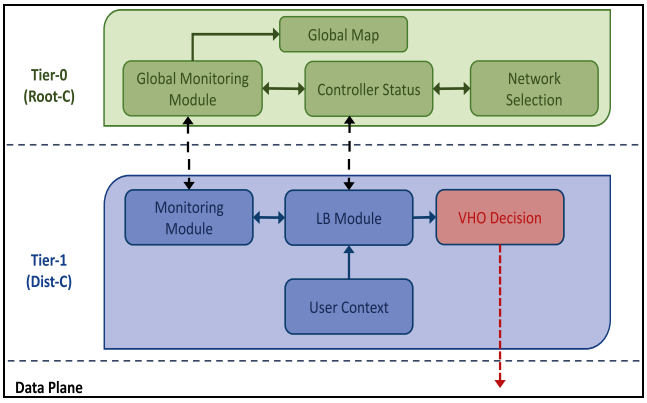


Fig. 4. A two-level load balancing management framework based on context-aware handover.

5. Management framework

Our management framework is constructed and distributed into a two-tier control plane. It is based on a collaboration between modules that reside on both Dist-Cs and Root-C. So, our proposed approach (proposed-LB) is not operated entirely by Root-C; we divide the functions into local ones done by Dist-Cs, and global ones done by Root-C, refer to Fig. 4. Consequently, the exchanging overhead between Dist-Cs and Root-C is minimized. The five main modules that shape our load balancing framework are:

- The monitoring module.
- The controller status module.
- The user context module.
- The load balancing decision module.
- The network selection module.

5.1. Monitoring module

This module has a global version placed on Root-C and a local version on Dist-C.

The local module gathers the statistics (i.e., stats) that reflect the switches' state regarding flow tables and port stats. For example, a controller can periodically collect port, table, and flow information within its domain through the use of particular stats OpenFlow messages, such as OFPST_PORT, OFPST_FLOW, and OFPST_TABLE [10].

The global module placed at Root-C periodically collects load state information from Dist-Cs. As maintaining a consistent view among the distributed controllers can result in significant overhead, we need to design a mechanism that reduces the frequency of load informing.

5.2. Controller status module

The authors of [30] proposed an inhibition algorithm to suppress an unchanged controller status from being reported to other controllers; otherwise, it is redundant information. Inspired by [30] to reduce the frequency of status exchange, we propose the following load informing strategy. We define two thresholds; maximum threshold and minimum threshold as $\overset{+}{\alpha}_{th}$, and $\overset{-}{\alpha}_{th}$, respectively. Using the two thresholds, any controller at time t belongs to only one of three sets: Under-Loaded (UL), Normally-Loaded (NL) and Over-Loaded (OL), based on the following:

- $\theta_i \leq \overset{-}{\alpha}_{th_i} \implies C_i \in UL.$
- $\overset{-}{\alpha}_{th_i} < \theta_i < \overset{+}{\alpha}_{th_i} \implies C_i \in NL.$
- $\theta_i \geq \overset{+}{\alpha}_{th_i} \implies C_i \in OL.$

Note that these thresholds are adjustable to each controller based on its network policy [19]. Root-C, in return, updates Dist-Cs with the others' statuses in terms of what load set they have. Meaning, the distributed controllers see only the load set that others belong to, without going further into other details. As long as the mapping between the Dist-Cs and their load sets do not change, the Root-C refrains from updating. Thus, we mitigate the case of reporting unchanged statuses. See Algorithm 1 that describes our load informing strategy.

Algorithm 1: Adaptive Load Informing

Data: $L_{current}$: Current load set at t
L_{former}: former load set at $t-1$
Two thresholds: $\overset{-}{\alpha}_{th_i}$, $\overset{+}{\alpha}_{th_i}$ $\forall i \leq M$
Result: report = True or False: Report load status to C_j $\forall j \leq M$, $j \neq i$.

```
begin
    report=False
    if L_current(C_i) ≠ L_former(C_i) then
        report=True
        if L_current == OL then
            Start load balancing
        end
    else
        report=False
    end
    L_former = L_current
    Return report
end
```

5.3. Candidate users selection module

A vertical handover decision is based on criteria that can be divided into the following [36]:

- Network context, such as link quality (RSSI, CIR, BER, SIR), coverage, bandwidth, latency, cost, and security level.
- Terminal context, such as velocity, battery, and location information.
- User context, such as user profile and preferences.
- Service context, such as service requirements and QoS.

A context-aware handover does not only consider the traditional ways, such as signal strength to trigger handover, but it also takes into consideration the knowledge of the mobile node and network's context information. Consequently, intelligent and improved handover decisions can be made [36]. However, combining a large number of criteria would considerably affect the decision delay due to the increased complexity of the decision algorithm [37].

In order to achieve our goal of relieving an overloaded controller, we exploit the fact that multiple coverage areas are overlapping, and

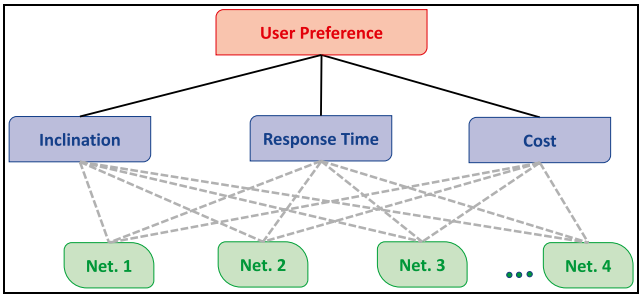


Fig. 5. A taxonomy of a user's perspective using AHP method.

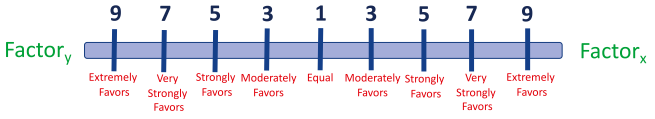


Fig. 6. A preference scale in pair-wise comparison.

mobile users can switch to other networks based on a combination of criteria. We think that the involvement of users' preferences is essential and beneficial to both the user and network. In line with other studies, for example, [38], we consider having a supporting module in the users' devices, allowing them to specify their preferences initially and then dynamically changing them as needed.

In the literature, several strategies are applied to include a user's preferences, such as Simple Additive Weighting (SAW), Multiplicative Exponent Weighting (MEW), Technique for Order Preference by Similarity to Ideal Solution (TOPSIS), Analytic Hierarchy Process (AHP) and Gray Relational Analysis (GRA) [36,39]. Every strategy has its attributes and compliant situations.

The AHP tool is considered a powerful decision-making tool originally proposed by Saaty [40]. Typically, the AHP tool defines a hierarchy of at least three levels, where the top level is the goal, the bottom level is the solution alternatives, and the middle level is the input to this tool (i.e., decision factors). These factors can be obtained from measurements such as weight, cost, etc., or from subjective opinions such as satisfaction and preference, which can be conflicting in some cases.

In our case, we looked for a strategy that would help us evaluate different preferences inputted by a user and compare them to each other. On top of that, we want to get a rank to help prioritize those preferences, thus making a decision. Since the judgments are made by humans, they can be inconsistent. AHP provides means to check the consistency of the user's input. Therefore, we think this tool is in alliance with our direction of measuring users' perspectives.

AHP has the following four stages: decomposition, pair-wise comparison, weight calculation, and weight synthesis [40,41]. By going through an example, we explain in detail the four stages.

- *Stage-1: Decomposition.* In this stage, a thorough understanding of the problem and then dividing it into subproblems are crucial. As a result, a hierarchy of the problem is obtained. In our problem, for the user's perspective, we can show the hierarchy as depicted in Fig. 5. We see that the goal is to include the user's preferences. Different factors affect the user's decision; for example, a user's inclination, response time, and cost.
- *Stage-2: Pair-Wise Comparison.* In this stage, each user declares their subjective opinions regarding each pair of decision factors. For example, let us model the hierarchy shown in Fig. 5 in a matrix form (Δ), and consider three decision factors: inclination as I, response time as R, and cost as C. Now, we draw a pair-wise comparison, to show how each factor is preferred over the other.

$$\Delta = \begin{array}{c} \\ I \\ R \\ C \end{array} \begin{array}{c} \begin{array}{ccc} I & R & C \end{array} \\ \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \end{array} \tag{11}$$

As shown by Eq. (11), we have a 3×3 matrix. The elements of this matrix are determined by selecting values that show how the user is in favor of one factor over the other. Note that the number of comparisons is a function of the number of decision factors (dn); thus, the total number of comparisons is $\frac{dn(dn-1)}{2}$. There are different attempts to quantify such scale, yet the most famous one is the linear scale from 1 to 9, where each number describes the factor importance degree as in Fig. 6.

Based on the user's input, the pair-wise matrix in Eq. (11) is filled with either a value (1–9) or a reciprocal value, while the diagonal is always one. If the importance value is picked from the left side of 1, then we fill the matrix element with that value, otherwise, the matrix element is the reciprocal of that value. At this point, we end up filling the upper triangular matrix, to get the values of the lower triangular, we use the relation $a_{ij} = \frac{1}{a_{ji}}$. Carrying on with our example in Fig. 5, we show a sample of a user's choices as the following matrix:

$$\Delta = \begin{array}{c} \\ I \\ R \\ C \end{array} \begin{array}{c} \begin{array}{ccc} I & R & C \end{array} \\ \begin{bmatrix} 1 & 1/7 & 1/5 \\ 7 & 1 & 3 \\ 5 & 1/3 & 1 \end{bmatrix} \end{array} \tag{12}$$

Apparently, to that user, the response time to their control traffic is the most critical factor; however, we show this result mathematically as we proceed further to the following stages.

- *Stage-3: Weight Calculation.* In this stage, we prioritize the factors by computing the priority vector. In order to do that, we find the normalized Eigenvector, we use the formula:

$$a_{i1} = \frac{a_{i1}}{\sum_{i=1}^{3} a_{i1}} \tag{13}$$

Back to our example, the resulting matrix is:

$$\underline{\Delta} = \begin{array}{c} \\ I \\ R \\ C \end{array} \begin{array}{c} \begin{array}{ccc} I & R & C \end{array} \\ \begin{bmatrix} 1/13 & 3/31 & 1/21 \\ 7/13 & 21/31 & 15/21 \\ 5/13 & 7/31 & 5/21 \end{bmatrix} \end{array} \tag{14}$$

Next, we obtain the normalized Eigenvector by taking the average of each row according to:

$$v_{rj} = \frac{\sum_{j=1}^{3} a_{rj}}{3} \quad \text{where } r = 1, 2, 3. \tag{15}$$

Thus, we get the following Eigenvector V:

$$V = \frac{1}{3} \begin{bmatrix} 1/13 + 3/31 + 1/21 \\ 7/13 + 21/31 + 15/21 \\ 5/13 + 7/31 + 5/21 \end{bmatrix} = \begin{bmatrix} 0.0738 \\ 0.6434 \\ 0.2828 \end{bmatrix} = \begin{bmatrix} v_I \\ v_R \\ v_C \end{bmatrix} \tag{16}$$

Here, we obtained our priority vector, $V^T = [0.0738, 0.6434, 0.2828]$, where the sum of elements of our priority vector is 1.

As indicated by the priority vector of our example above, the response time factor has the highest priority with 64.34%, while the cost factor is the second in ranking with 20.28%, and finally, the least important factor is the inclination factor with only 7.38%. At this point, we know the ranking of each decision factor. To find the relative weight, we simply obtain the ratio among every pair; for example, the cost factor is 3.83 (i.e., 28.28/7.38) times more important than the inclination factor. Thus, the relative weight matrix can be obtained as the following:

$$W = \begin{array}{c} \\ I \\ R \\ C \end{array} \begin{array}{c} \begin{array}{ccc} I & R & C \end{array} \\ \begin{bmatrix} v_I/v_I & v_I/v_R & v_I/v_C \\ v_R/v_I & v_R/v_R & v_R/v_C \\ v_C/v_I & v_C/v_R & v_C/v_C \end{bmatrix} \end{array} \tag{17}$$

Table 1
The random consistency index.

dn	1	2	3	4	5	6	7	8	9	10
RC	0	0	0.58	0.9	1.12	1.24	1.32	1.41	1.45	1.49

- *Stage-4: Weight Synthesis.* Since the judgments are made by humans, they can be inconsistent. In this stage, we follow the steps proposed by Saaty in [40] to check the consistency of the user's input. Firstly, we find the maximum Eigenvalue as the following:

$$\upsilon_{max} = 13(0.0738) + \frac{31}{21}(0.6434) + \frac{21}{5}(0.2828) = 3.097 \approx dn \tag{18}$$

Then, Saaty in [40] proposed the use of a Consistency Index (CI) to measure the deviation of consistency using the following formula:

$$CI = \frac{\upsilon_{max} - dn}{dn - 1} \tag{19}$$

In our example $CI = 0.0485$. Then, we compare it to the Random Consistency Index ($RC(dn)$), whose values are given in Table 1. Finding the Consistency Ratio (CR) of CI and RC estimates whether the inconsistency is acceptable or not.

$$\begin{cases} CR = \frac{CI}{RC} \leq 10\% & \text{Acceptable inconsistency.} \\ \text{Otherwise} & \text{Unacceptable inconsistency.} \end{cases}$$

Back to our example, $CR = 0.0836 < 10\%$, so the user's input is acceptable, and to be considered. Otherwise, the process is repeated with a new pair-wise comparison matrix. When the inconsistency is unacceptable, the subjective judgment of a user needs to be revised.

We then consider the users that provide higher weight for response time, along with their locations, retrieved from the Global Positioning System (GPS), as the candidate users.

5.4. Load balancing module

This module is local at Dist-C, where the information is gathered from the monitoring modules, user context module, and other networks' statuses. Upon the event of C_i being overloaded, this module is triggered in collaboration with Root-C. This module triggers the procedure by first identifying the candidate edge users. Those users who preferred "response time (R)" as their main metric, are candidates to be vertically handed over to other controllers with better response times.

We know that there are four handover execution strategies: network-controlled handover, mobile-controlled handover, network-assisted handover, and mobile-assisted handover [37]. For our approach to work, the vertical handover is a network-controller that is initiated and mainly controlled by the controller as a resolution method for load balancing. The flow chart in Fig. 7 summarizes the work-flow of our proposed framework from C_i's perspective.

Let C_i control a cellular domain, with the coexistence of multiple Wi-Fi controllers within its domain. C_i vertically hands over edge users $\in MN_e^i$ to candidate Wi-Fi networks that have their load set $\in NL \cup UL$. Given that Dist-Cs with load belonging to UL are preferred over those in NL. The selection method is presented in the following part.

5.5. Network selection module

This module is done by Root-C. Based on the system modeling presented in Section 4, now we formulate the network selection problem as a 0–1 integer optimization problem. Let $C_1 \in C$ be considered as overloaded at time t, then, C_1 triggers the load-balancing module. First, it identifies edge users with R as their main criterion as U_h^1, where $U_h^1 \subseteq MN_e^1$ and the number of those candidates is $|U_h^1| = H$. Second, for each candidate user, there is a set of candidate networks that currently cover that user. Note that they can be obtained from the user–controller adjacency matrix Y_t as the corresponding vector to each row (user). For a user u_h, let $\overrightarrow{P_h}$ denote the possible candidate networks vector, where $\overrightarrow{P_h} = \{p_h^2, p_h^3, \ldots, p_h^M\}$, and the elements of vector $\overrightarrow{P_h}$ are either 0 or 1, with each element corresponds to a controller. Third, the corresponding remaining capacity vector of candidate controllers is obtained from Eq. (4) as $\overrightarrow{R_h^m} = \{r_h^2, r_h^3, \ldots, r_h^M\}$. Now, based on the remaining capacities of candidate networks, Root-C selects the new network/controller for each edge user, u_h currently attached to the overloaded controller, C_1.

With the main objective of matching mobile users with better alternatives, taking into consideration their remaining capacities, we formulate our problem as a 0–1 integer programming problem, as follows:

$$\textbf{max} \quad \sum_{h=1}^{H} \sum_{m=1}^{M} (\, p_h^m \cdot r_h^m \,) \tag{20}$$

$$\textbf{subject to}: \quad p_h^m \in \{0, 1\} \tag{21}$$

$$\theta_i < \alpha_{th_i}^{+} \tag{22}$$

$$\sum_{m=1}^{M} y_{xm} = 1 \quad \forall u_x \tag{23}$$

$$\sum_{b=1}^{k} z_{bj} \leq l_{s_j} \quad \forall s_j \in S \tag{24}$$

Our objective is vertically handing over as many users as possible, while maintaining their requirements in terms of response time. The value of p_h^m is either 1 or 0 to indicate whether the corresponding network is a candidate or not, respectively (Eq. (21)). The load on the candidate networks have to be below their thresholds in order to accommodate more users and provide them with satisfied experiences (Eq. (22)); meaning, only under-loaded and normally-loaded controllers can be candidates. Each user is connected to only one network (Eq. (23)). When a user, u_h is switching to a new network, the new PoA has to have an available channel to be assigned to that u_h, otherwise, the vertical handover fails (Eq. (24)). To solve this, a greedy search would give us a solution in a linear time with respect to the number of available networks and candidate users. In the worst-case scenario, the search would consume $O(HM)$, where H is the number of candidate edge users and M is the number of available networks. We argue that the computation overhead introduced by this step of our proposed-LB is negligible, as the number of candidate users and controllers are limited in practice.

6. Experiment and evaluation

In this section, we describe our simulation environment, including the parameters and conditions. We also divide our simulations into four experiments, where we analyze the results and draw conclusions.

6.1. Setup

In our experiments, we chose Ryu [42] as our SDN controllers to communicate with the data plane and to test our management framework. We implemented our data plane on Mininet v.2.1.0 [43]. The data plane is composed of OVS-switches that support OpenFlow v1.3. We ran Ryu and Mininet on a virtual machine with intel core i7 processor and a 4 GB of RAM. The virtual machine is running Ubuntu 16.04 LTS.

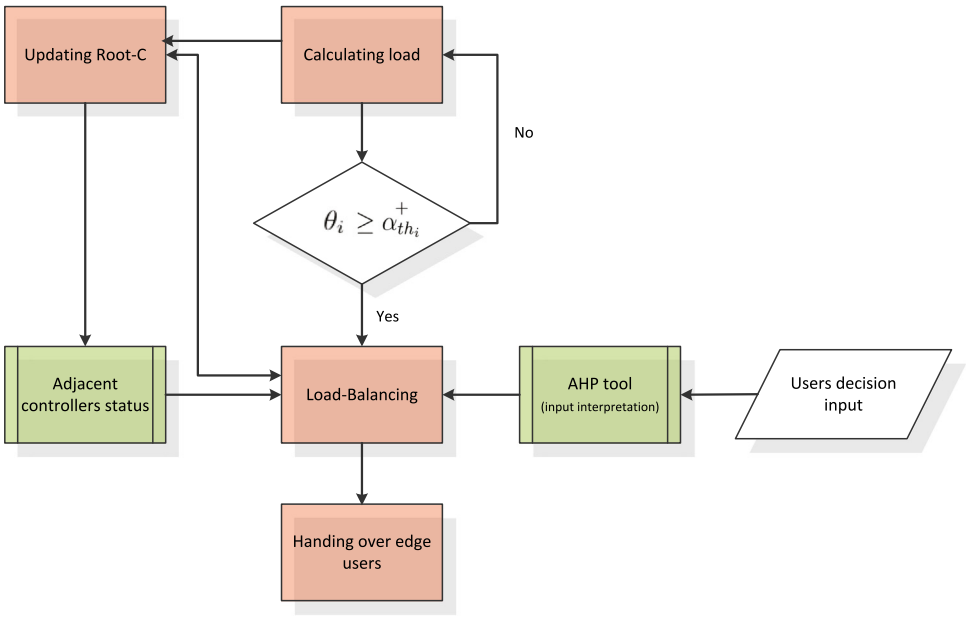


Fig. 7. A flow chart of an over-loaded controller load balancing method.

Table 2
Simulation parameters.

Parameter	Setting
Tool	Mininet 2.1.0
Links delay between switches	2 ms
Links delay between switches and hosts	5–10 ms
OpenFlow message	v.1.3.0
Controllers	Ryu
Capacity α	6000 flows/s
Decay factor β	0.2,1
Threshold α_{th}	900,5780 flows/s
Test tools	iperf, ping, Tcpdump
UDP datagram	1470 Byte
UDP buffer size	208 kB
TCP window size	15 kB

In our setup, we implemented a two-tier hierarchical control plane; we had two controllers as tier-1 Dist-Cs (C_1, C_2), and a third controller connected to each of them as the Root-C. We used the TCP socket as our messaging system to get messages moving between the Ryu controllers. C_1 represented the controller of a cellular network with six OVS-switches and 20 users, while C_2 represented a Wi-Fi network with a smaller capacity and connected to two OVS-switches and four attached users. The users were placed and moved randomly. Meanwhile, they randomly generated streams of UDP and TCP traffic that started at different times for random periods of time. The list of simulation parameters is shown in Table 2.

Unfortunately, due to the restrictions imposed by Mininet, we could not implement AHP at the user's end. Mininet supports only OpenFlow protocol, so it does not allow messages other than OpenFlow messages to be exchanged between the data plane and the control plane. However, to overcome such a limitation, we chose random percentages of users to be our candidates to be handed over. The percentages were between 10% to 14% of users, since a higher percentage can be unrealistic.

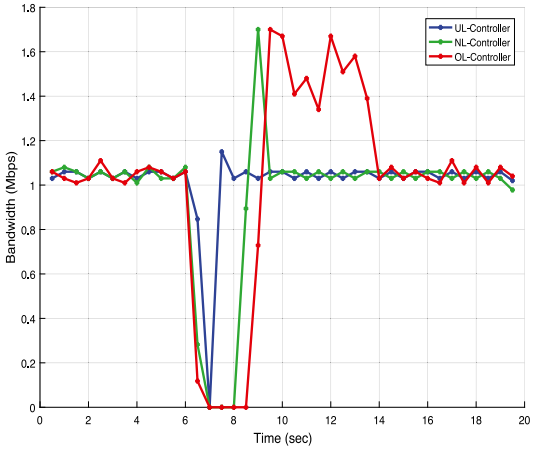
6.2. Experiment results

Handover, along with other procedures associated with mobile users, can exhaust the SDN controllers' resources. Thus, load balancing can be achieved earlier, as a part of the handover preparation stage, to alleviate longer delays due to saturated buffers that are unable to process handover signaling messages efficiently. Different controller loads have different impacts on the completion of a critical procedure such as a handover. To demonstrate this, we conducted an experiment by running a UDP traffic between two hosts, where one is mobile. Fig. 8 shows the effect on two metrics, UDP throughput and jitter. As it is depicted, we started UDP iperf traffic between two nodes for 20 s. As one of them moved from its PoA to another, a hard handover happened at the seventh second; we recorded the measured throughput every 0.5 s. Fig. 8a reflects the time a controller took to handle a handover under different loads, when the controller was under-loaded, normally-loaded, and over-loaded. It was seen that the handover delay was higher when the controller handled more OpenFlow packets. That being said, as the number of packets increased, the controller response time increased as well, and thus, a handover's completion took longer; the handover delay has almost tripled as the controller's load changed from under-loaded to being over-loaded. This worsens the user experience during a hard handover. Regarding jitter, a high variation in delay degrades the user experience while using sensitive applications such as VoIP. As Fig. 8b shows, as the controller load increased, the variation in delay prolonged, thus disturbing or even disconnecting the service.

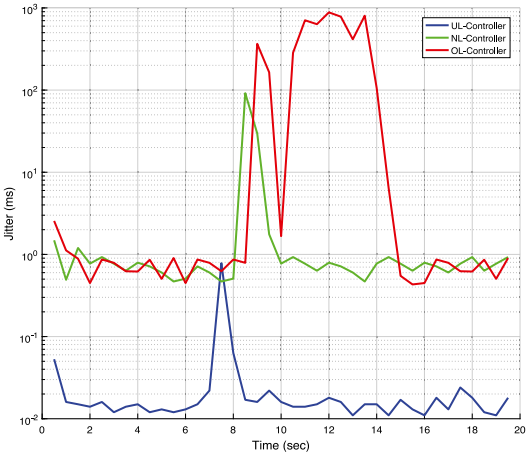
The results of the experiment found clear support for the importance of load balancing in minimizing handover delay since the controller response time is actually a main contributor in handling handover signaling messages.

In a different set of experiments, we run an experiment ten times and we recorded the average perceived response time by the connected users every two seconds, as the load varied on their managing controllers C_1 and C_2, refer to Fig. 9. As the load on C_1 reached its threshold at around the 30th second, our proposed-LB started by handing over candidate users to C_2. Afterward, the handed-over users' traffic was handled by C_2. We noticed a drop in perceived response time by 20%, as the load decreased by 14% of the connected users and their on-going traffic. Meanwhile, users connected to C_2 experienced an increase in C_2's response time, which was still acceptable since C_2 was under-utilized and it resolved that increase rapidly.

In the previous experiment, we recorded the impact of our proposed-LB on users within their domains. Now, we show the impact on a candidate user, let us assume it is u_x, in two scenarios. In one scenario, we recorded the impact of our proposed-LB in terms of the perceived response time, every two seconds, including the effect of the vertical handover. In the second scenario, we studied what would



(a) UDP throughput during a handover.



(b) UDP jitter during a handover.

Fig. 8. Impact of different loads on handover during a udp traffic.

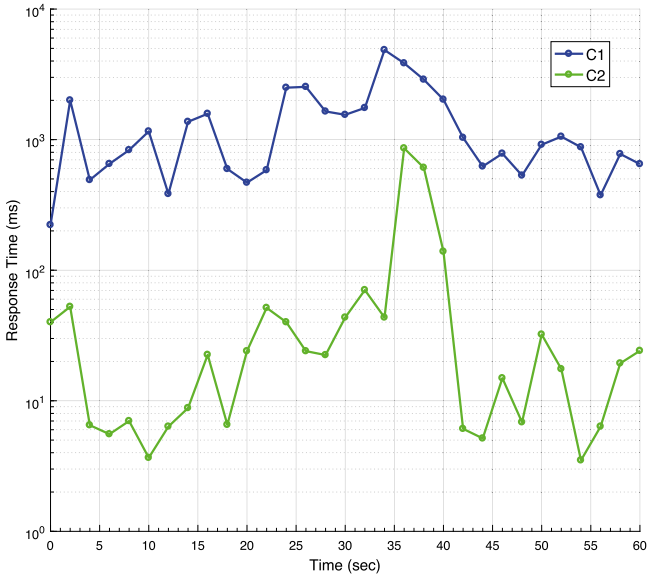


Fig. 9. Recorded users' perceived response time.

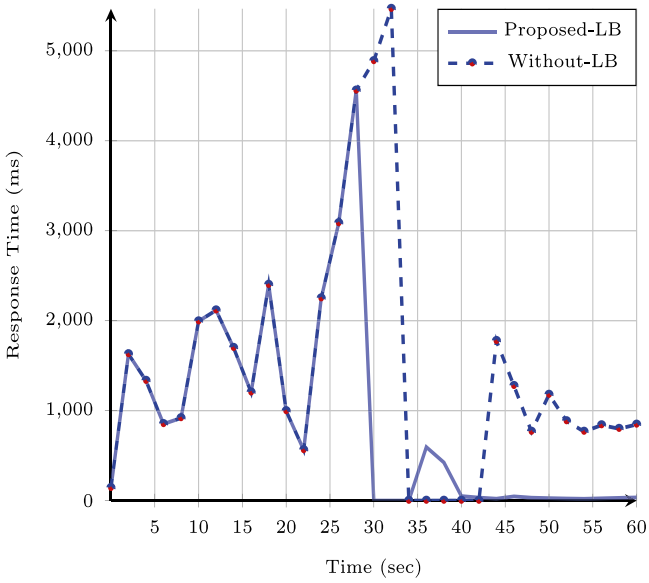


Fig. 10. Perceived response time in two scenarios.

happen if that user stayed connected to an over-loaded controller, without a countermeasure. As Fig. 10 shows, as the managing controller approached its threshold at around 28th second, it took longer to respond to u_x. Then, since u_x was a candidate user, it got vertically handed over to another available under-utilized controller. Thus, u_x experienced around four seconds of disconnection, and then restored connection with very low response time from the new controller, which is around an 80% decrease in the perceived response time. For a user with response time as their main concern, that shift is favorable. However, if u_x does not get handed over, the response time would reach unacceptable levels (the dashed line), and then the managing controller would start dropping requests causing disruption for longer periods until the over-loading state is resolved.

In another experiment, we compared our proposed-LB against other two static approaches, [18] and [31]. We chose these two approaches because they share a common ground with our approach, the static assignment of controller and switches. The authors of Onix [18] proposed a flat static structure to scale out their topology. We used their approach as a benchmark of the static mapping, so we refer to it as SM. The other approach proposed by Wang et al. in [31] is mainly dependent on the idea of balancing the load of multiple controllers based on shifting *macroflows* from the management of one controller to another. A *macroflow* is a set of flows originating from a particular switch and destined to the same switch. Unlike *microflows*, where the controller installs rules in the most granular level for each connection. Even though *a* macroflow can relieve the controller from handling each flow, it requires the use of *wildcards* that rely on TCAM, which is expensive and restrictive. The proposed approach by [31] is implemented in a hierarchical architecture, where a set of distributed controllers report their load information and flow entries stats to a root controller. Then, the root controller determines the *macroflow(s)* to be redirected to ensure fairness among all the controllers.

As shown in Fig. 11, we compared the controller load between three approaches: SM [18], GFRD [31], and our proposed-LB. Expectedly, a controller's load in SM increases linearly as the requests rate per second increases. As depicted, GFRD can minimize the load by 50%, and that is due to the pre-installation of wildcard rules in adjacent edge switches. Our approach, however, increases linearly as in SM except that once the requests reach a pre-defined threshold (i.e., $\alpha^{+}_{th} = 0.75{\times}\alpha$), the balancing module is triggered to hand over candidate users. Hence, the controller

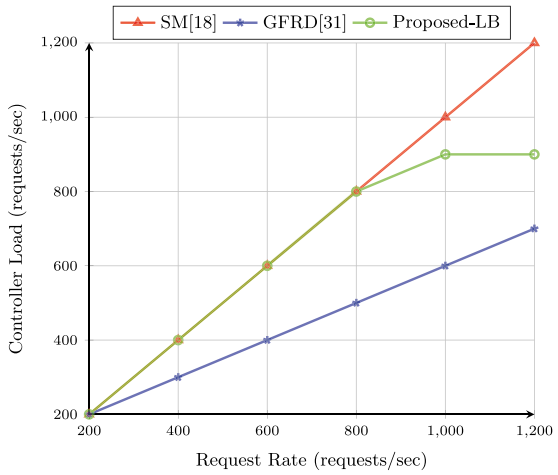


Fig. 11. Controller load as request rate increases in three approaches.

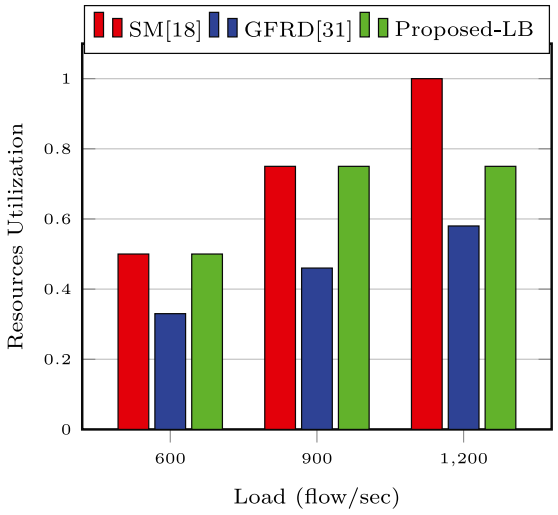


Fig. 12. Resource utilization at different loads.

is relieved as candidate users' traffic is shifted to other controllers. In this experiment, we chose $\beta = 0.2$ to make our results comparable to the results reported by [31]. Based on the results shown in Fig. 11, we estimated the resource utilization under different loads for the three approaches in Fig. 12. Knowing that resource utilization is computed as: $util = \frac{\theta}{\alpha}$. Apparently, [31] reports 50% lower controller load compared to SM and that is because of the pre-installation of *macroflows* wildcard rule for traffic redirection. This rule sends out those flows as chunks from one controller to another through switches on the path, the thing that shows a reduction in the load from the beginning even before load balancing is triggered. In their approach, their target is to ensure fairness between controllers as well as reducing the response time of the whole system; unlike our target, which aims to relieve the over-loaded cellular controller.

We simulated the three aforementioned approaches on the topologies that are shown in Fig. 13 to measure their reactiveness and effectiveness. We triggered traffic randomly from users for 120 s and recorded the response times that the users perceived of their controllers. The simulations were repeated ten times and the average response times were taken. We plotted the results of our approach against the other two approaches in Fig. 14.

From Fig. 14, we observe the following. There were two periods of over-loading, one started around the 28th second, while the other started at the 92nd second; the three approaches handled these periods differently. GFRD did not show drastic changes as this approach based on a proactive strategy that ensures no over-utilization may happen. Regarding SM, there is no proactive mechanism to over-loading states; therefore, the controller load reached its cap and started dropping flows as it could not handle more traffic. As for our proposed-LB, once the controller load reached a pre-defined threshold (i.e., $\alpha_{th} = 5780$ flows/s), 10% of the connected users and their on-going traffic got handed over to an under-utilized controller. Those users helped in relieving the over-loaded controller resources enhancing the response time for other connected users, and gaining better response time from their new controller. Yet, they experienced some data loss due to the vertical hand over that is around 8%, while in the SM, the data loss is at least 15% until overloading state is resolved.

In this part, we refer to an essential metric called "balance time" [26].

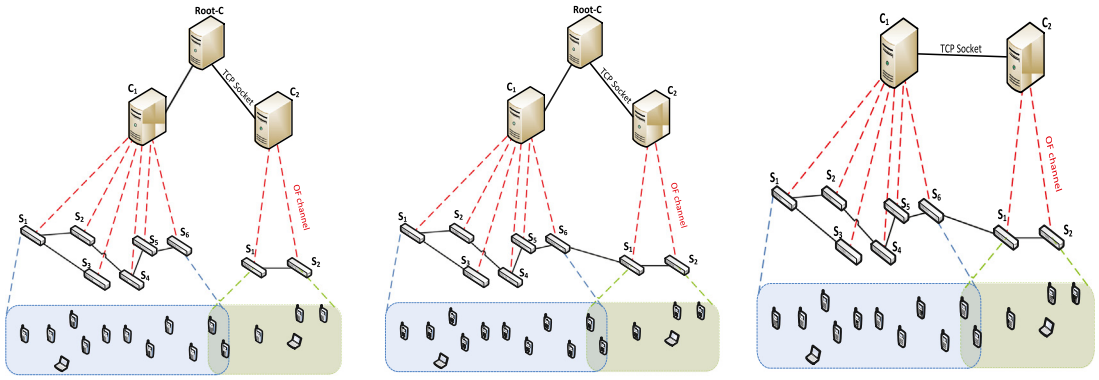
Definition 1 (*Balance Time*)**.** The period of time starting from when the unbalance state is detected until that state is resolved.

For instance, the balance time in our approach starts from reaching a threshold in term of load, then identifying the candidate users, selecting more suitable network(s), and then vertically handing over the users. Intuitively, the lower the balance time, the more efficient the load balancing strategy. In our simulation, the balance time was the period between [28–36] seconds.

To further analyze and measure the effectiveness of our approach, we need to address the impact on different types of users, including those that remain connected to the over-loaded controller and the ones that are connected to the under-loaded controller before, during and after the balance time. Therefore, we recorded the response time of four categories of users. Since both periods of over-loading showed the same trend, we focus only on the first 60 s that contained the first over-loading period as in Fig. 15. Here, u_y is a non-edge user connected to C_1, u_z is a non-edge user connected to C_2, u_w is a user in GFRD, and u_x is an edge user to be vertically handed over when the load balancing mechanism is triggered. Note that C_1 got over-loaded, while C_2 was under-loaded at the time of network selection. The user connected to C_1, $u_y \in MN^1_{ne}$, experienced a response time around 4.5 s at the 28th second, then about 10% of users got handed over resulting in 36% drop in the perceived response time from C_1. Meanwhile, as the load on C_2 was incremented by the load of the handed-over users, C_2's response time increased at the 36th second for a short period, which is still acceptable given that C_2 was able to accommodate more users and provided less delay. Regarding the user's perceived response time in GFRD, we can see that they experienced a nearly-steady pattern of response time due to the pre-configuration of wildcard rules that maintain the whole system in a fair state among controllers. Meanwhile, the candidate edge users connected to C_1, $u_x \in MN^1_e$, after the balance time, as they got connected to another controller C_2, they experienced in average 28% drop in the response time compared to GFRD, during the period between [36–60] seconds.

Based on multiple criteria, we evaluate our approach and the other two, GFRD and SM, in Table 3. Note that we use (—) to indicate that this property has not been investigated, and the (✖) to indicate unsatisfied property by that method. An important property that we want to shed light on is the fact that the methods followed by [18] and [31] require inter-connection between domains for their algorithms to work, meaning that the concept of completely isolated domains is not taken into consideration. Unlike our approach, we provide a solution for both connected and disconnected domains since our proposal is irrespective to infrastructure. Still, the SPoF issue persists; however, not as severe as placing the control logic of a network entirely on a single controller.

On another note, in our approach, we aim at reducing the number of status synchronization messages between tier-1 controllers in order not to introduce intercommunication overhead. Note that the previously discussed two approaches, [18] and [31], did not provide algorithms



(a) Proposed-LB topology. **(b)** [31] topology. **(c)** [18] topology.

Fig. 13. Simulation topologies.

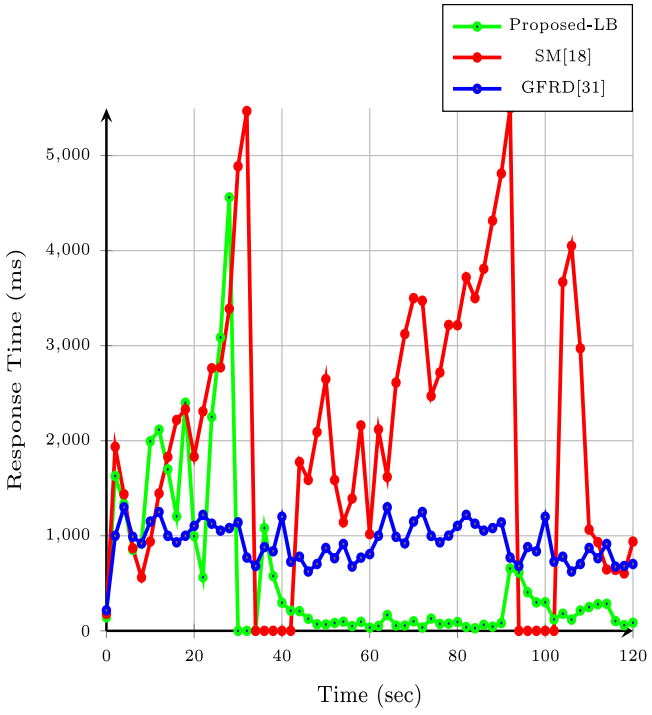


Fig. 14. The perceived response time by a mobile users in three scenarios.

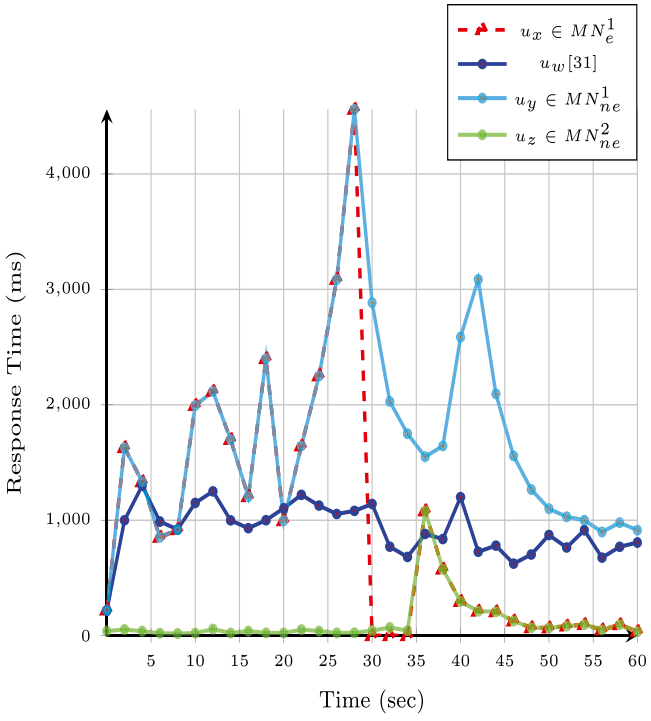


Fig. 15. The perceived response time by different users in two approaches.

Table 3
Evaluation of the three approaches.

Property	SM	GFRD	Proposed
Scalability	✔	✔	✔
Static switch-controller assignment	✔	✔	✔
Minimized synchronization	✖	–	✔
Functionality distribution	✔	✖	✔
Mitigating bottlenecks	✔	✖	✖
Isolated domains	✖	✖	✔
Low response time	✖	✔	✔
Efficient resource utilization	✔	✔	✔
Hardware restrictions	–	✔	✔
Data loss percentage	15%	–	8%
Mobility consideration	✖	✖	✔
User preference inclusion	✖	✖	✔
Fairness	✖	✔	✖
Considering non-uniform capacities	✖	✖	✔

to reduce the inter-controller overhead. Therefore, we consider other studies that have proposed mechanisms to minimize the number of messages between controllers. In this part, we compare the complexity in terms of the number of messages, in the worst-case scenario, between [26], DALB [23], and our strategy. As reflected in Fig. 16, the approach in [26] has the highest synchronization messages among the three. As for DALB, it performs well when there are fewer controllers interconnected; however, as the number of controllers grows, the overhead increases. Our strategy performs well regardless of the number of interconnected controllers. As the number grows, our proposed strategy outperforms the other two approaches, since Root-C limits the frequency of updating when a controller does not change its load set.

7. Conclusion

Indeed, the timely delivery of a huge number of control messages in SDN is critical. Since the control/signaling messages have to be handled by a controller, that controller's response time has a vital impact on the

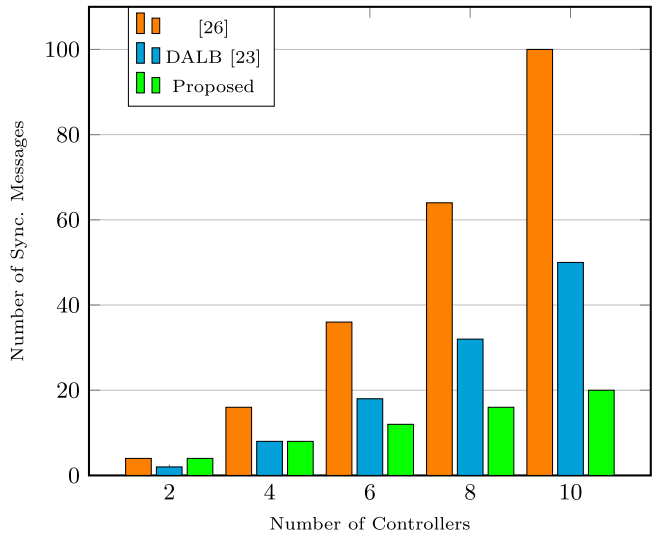


Fig. 16. Synchronization complexity in term of number of messages.

completion of mobility management procedures, including handover. Therefore, we approach the over-loading controller problem, specifically to minimize the maximum response time by exploiting users' preferences and the complementary resources of other cooperative controllers. We have proposed a management framework that requires three main functions. First, we should identify candidate users based on their context information; we have used AHP, a decision-making tool to include the user's input into our framework. Second, we need a mechanism that reduces the frequency of load disseminating between multiple controllers, and hence, reducing overhead. Third, we have to optimize the decision problem on the selection among several candidate networks. We have compared our mechanism against two other static ones, and in our approach, the vertically handed-over users experienced lower response time compared to the other approaches.

So far, in our proposed-LB, we assume the vertical handover happens to users and their on-going traffic with a guarantee of an available channel only, yet, there was no guarantee of minimum channel bandwidth with respect to their service requirements. An interesting research direction would be coupling the control plane response time with the service requirement at the data plane level in our network selection approach. In this case, we would extend our vertical handover decision process to include the service type to ensure the channel capacity of the new attachment guarantees the bandwidth requirement of the handed-over user QoS.

CRediT authorship contribution statement

Modhawi Alotaibi: Conceptualization, Methodology, Software, Formal analysis, Writing - original draft. **Amiya Nayak:** Resources, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was partially funded by a research grant from the Natural Sciences and Engineering Research Council of Canada.

References

[1] D. Nowoswiat, G. Milliken, Alcatel-Lucent: Managing the Signaling Traffic in Packet Core, White Paper, 2013.
[2] V. Nguyen, T. Do, Y. Kim, SDN and virtualization-based LTE mobile network architectures: A comprehensive survey, Wirel. Pers. Commun. 86 (3) (2016) 1401–1438.
[3] S. Kukliński, Y. Li, K. Dinh, Handover management in SDN-based mobile networks, in: Proceedings of IEEE Global Communications Conference Workshops, 2014.
[4] N. Jagadeesan, B. Krishnamachari, Software-defined networking paradigms in wireless networks: A survey, ACM Comput. Surv. 47 (2) (2015) 27.
[5] C. Marquezan, X. An, Z. Despotovic, R. Khalili, A. Hecker, Identifying latency factors in SDN-based mobile core networks, in: Proceedings of IEEE Symposium on Computers and Communication, 2016.
[6] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, R. Sherwood, On controller performance in software-defined networks, in: Proceedings of the 2nd USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, 2012.
[7] I. Akyildiz, J. Xie, S. Mohanty, A survey of mobility management in next-generation all-IP-based wireless systems, IEEE Wirel. Commun. 11 (4) (2004) 16–28.
[8] S. Lee, K. Sriram, K. Kim, Y. Kim, N. Golmie, Vertical handoff decision algorithms for providing optimized performance in heterogeneous wireless networks, IEEE Trans. Veh. Technol. 58 (2) (2009) 865–881.
[9] K. Yap, R. Sherwood, M. Kobayashi, T. Huang, M. Chan, N. Handigol, N. McKeown, G. Parulkar, Blueprint for introducing innovation into wireless mobile networks, in: Proceedings of the 2nd ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures, 2010.
[10] M. Bari, A. Roy, S. Chowdhury, Q. Zhang, M. Zhani, R. Ahmed, R. Boutaba, Dynamic controller provisioning in software defined networks, in: Proceedings of the 9th International Conference on Network and Service Management, 2013.
[11] X. Duan, X. Wang, A. Akhtar, Partial mobile data offloading with load balancing in heterogeneous cellular networks using software-defined networking, in: Proceedings of the 25th IEEE Annual International Symposium on Personal, Indoor, and Mobile Radio Communication, 2014.
[12] J. Palicot, C. Moy, B. Résimont, R. Bonnefoi, Application of hierarchical and distributed cognitive architecture management for the smart grid, Ad Hoc Netw. 41 (2016) 86–98.
[13] F. Wibowo, M. Gregory, K. Ahmed, K. Gomez, Multi-domain software defined networking: Research status and challenges, J. Netw. Comput. Appl. 87 (2017) 32–45.
[14] H. Ali-Ahmad, C. Cicconetti, A. De la Oliva, V. Mancuso, M. Sama, P. Seite, S. Shanmugalingam, An SDN-based network architecture for extremely dense wireless networks, in: Proceedings of the IEEE SDN for Future Networks and Services, 2013.
[15] P. Kaur, J. Chahal, A. Bhandari, Load balancing in software defined networking: A review, Asian J. Comput. Sci. Technol. 7 (2) (2018) 1–5.
[16] H. Wang, H. Xu, L. Huang, J. Wang, X. Yang, Load-balancing routing in software defined networks with multiple controllers, Comput. Netw. 141 (2018) 82–91.
[17] Y. Hu, W. Wang, X. Gong, X. Que, S. Cheng, Balanceflow: Controller load balancing for OpenFlow networks, in: Proceedings of the 2nd IEEE International Conference on Cloud Computing and Intelligence Systems, 2012.
[18] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, S. Shenker, Onix: A distributed control platform for large-scale production networks, in: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, 2010.
[19] F. Tam, N. Correia, Fractional switch migration in multi-controller software-defined networking, Comput. Netw. 157 (2019) 1–10.
[20] A. Krishnamurthy, S. Chandrabose, A. Gember-Jacobson, Pratyaastha: An efficient elastic distributed SDN control plane, in: Proceedings of the 3rd ACM Workshop on Hot Topics in Software Defined Networking, 2014.
[21] H. Liu, S. Kandula, R. Mahajan, M. Zhang, D. Gelernter, Traffic engineering with forward fault correction, in: Proceedings of the ACM SIGCOMM Computer Communication Review, 2014.
[22] W. Lan, F. Li, X. Liu, Y. Qiu, A dynamic load balancing mechanism for distributed controllers in software-defined networking, in: Proceedings of the 10th International Conference on Measuring Technology and Mechatronics Automation, 2018.
[23] Y. Zhou, M. Zhu, L. Xiao, L. Ruan, W. Duan, D. Li, R. Liu, M. Zhu, A Load Balancing strategy of SDN controller based on distributed decision, in: Proceedings of the 13th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 2014.
[24] H. Selvi, G. Gür, F. Alagöz, Cooperative load balancing for hierarchical SDN controllers, in: Proceedings of the 17th IEEE International Conference on High Performance Switching and Routing, 2016.
[25] G. Cheng, H. Chen, Z. Wang, S. Chen, DHA: Distributed decisions on the switch migration toward a scalable SDN control plane, in: Proceedings of the 14th IEEE IFIP Networking Conference, 2015.

[26] J. Cui, Q. Lu, H. Zhong, M. Tian, L. Liu, A load-balancing mechanism for distributed SDN control plane using response time, IEEE Trans. Netw. Serv. Manag. 15 (4) (2018) 1197–1206.
[27] P. Wu, L. Yao, C. Lin, G. Wu, M. Obaidat, FMD: A DoS mitigation scheme based on flow migration in software-defined networking, Int. J. Commun. Syst. 31 (9) (2018) e3543.
[28] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, R. Kompella, Towards an elastic distributed SDN controller, ACM SIGCOMM Comput. Commun. Rev. 43 (4) (2013) 7–12.
[29] Open Networking Foundation, OpenFlow Switch Specification Version 1.3.1, Tech. rep., 2012.
[30] J. Yu, Y. Wang, K. Pei, S. Zhang, J. Li, A load Balancing mechanism for multiple SDN controllers based on load informing strategy, in: Proceedings of the 18th Asia-Pacific Network Operations and Management Symposium, 2016.
[31] P. Wang, H. Xu, L. Huang, C. Qian, S. Wang, Y. Sun, Minimizing controller response time through flow redirecting in SDNs, IEEE/ACM Trans. Netw. 26 (1) (2018) 562–575.
[32] B. Görkemli, S. Tatlıcıoğlu, A. Tekalp, S. Civanlar, E. Lokman, Dynamic control plane for SDN at scale, IEEE J. Sel. Areas Commun. 36 (12) (2018) 2688–2701.
[33] M. Alotaibi, H. Lu, A. Nayak, A hierarchical approach to handle inter-domain mobility in SDN-based networks using MobileIP, in: Proceedings of the 25th IEEE International Conference on Telecommunications, 2018.
[34] J. Little, A proof of the queueing formula: L=λW, Oper. Res. 9 (3) (1961) 383–387.
[35] T. Wang, F. Liu, J. Guo, H. Xu, Dynamic SDN controller assignment in data center networks: Stable matching with transfers, in: Proceedings of the 35th IEEE International Conference on Computer Communications, 2016.
[36] M. Kassar, B. Kervella, G. Pujolle, An overview of vertical handover decision strategies in heterogeneous wireless networks, Comput. Commun. 31 (10) (2008) 2607–2620.
[37] M. Zekri, B. Jouaber, D. Zeghlache, A review on mobility management and vertical handover solutions over heterogeneous wireless networks, Comput. Commun. 35 (17) (2012) 2055–2068.
[38] Q. Wei, K. Farkas, C. Prehofer, P. Mendes, B. Plattner, Context-aware handover using active network technology, Comput. Netw. 50 (15) (2006) 2855–2872.
[39] E. Navarro, V. Wong, Comparison between vertical handoff decision algorithms for heterogeneous wireless networks, in: Proceedings of the 63rd IEEE Vehicular Technology Conference, 2006.
[40] T. Saaty, How to make a decision: The analytic hierarchy process, European J. Oper. Res. 48 (1) (1990) 9–26.
[41] Q. Song, A. Jamalipour, A network selection mechanism for next generation networks, in: Proceedings of the IEEE International Conference on Communications, 2005.
[42] RYU SDN Framework. https://osrg.github.io/ryu-book/en/Ryubook.pdf.
[43] Mininet: An Instant Virtual Network on your Laptop (or other PC). http://mininet.org.