A Multi-Controller Load Balancing Strategy for Software Defined WiFi Networks

Sohaib Manzoor, Xiaojun Hei, and Wenqing Cheng

School of Electronic Information and Communications
Huazhong University of Science and Technology, Wuhan, China, 430074
{sohaibmanzoor,heixj,chengwq}@hust.edu.cn

Abstract. Software Defined WiFi networks (SD-WiFi) support scalable network control functions, flexible resource allocation and changes in traffic. But the load balancing in SD-WiFi is challenging due to involvement of numerous users in the network. In this paper, we propose an efficient algorithm approach to achieve load balancing in SD-WiFi architecture. The user generated traffic arrives at WiFi access points (APs), which is classified into high prioritized (HP) flows and low prioritized (LP) flows, based on flow size and delay constraint values using support vector machine (SVM). Controllers are organized as two-tier: global controller (GC) and local controllers (LC). Markov Chain Model (MCM) is employed with two transition states as overloaded and underloaded in GC to predict future load of LCs based on the current load. The optimal underloaded LC for flow migration is selected by using Type-2 Fuzzy based Particle Swarm Optimization (TFPSO) algorithm. We conducted extensive simulation experiments to evaluate the performance of the proposed scheme using OMNeT++ simulator. The proposed scheme outperforms flow stealer scheme by a 33% increase in throughput and 70% in workload performance. In comparison to MPSO-CO scheme the proposed scheme exhibits better latency results.

Keywords: WiFi · SDN · Load Balancing · Multi-Controllers

1 Introduction

Software Defined Network (SDN) is a new emerging network platform which provides flexibility to program a network through a centralized and decoupled architecture. SDN serves as a solution for managing the network with more flexibility and network visibility to achieve better resource utilization and high performance [1]. In SDN, load balancing is achieved by migrating switches from controller using greedy efficiency-aware switch migration algorithm [2]. Load balancing is an important subject in high density software defined WiFi networks [3]. Prioritizing flows and controller cluster management helps achieving dynamic two-tier load balancing in software defined WiFi networks [4]. Load balancing in WiFi network is performed by transferring traffic between WiFi and WiMAX network [5]. The handover process is performed by the following process: (i)

bandwidth reservation in Access Point (AP) and base station (BS) (ii) admission control at AP and BS and (iii) class aware load balancing. In SDN based wireless networks, the network devices such as accessing devices and forwarding devices are simplified and behave in accordance to the rules scheduled by centralized controller [6].

In SD-WiFi network load balancing is performed by incorporating Access Network Discovery and Selection Functions (ANDSF) to select proper APs for incoming packets [7]. An SDN based load balancing algorithm is proposed for load balancing in wireless networks [8]. The advantage of SDN controllers view is taken to select optimal neighboring BS in BSs list for handover. A wildcard method based on the SDN protocol is presented in [9] for server load balancing. The wildcard method is improved by incorporating SDN open flow rules to assign priorities for load balancing.

In this paper, we propose an efficient algorithm approach to achieve load balancing in SD-WiFi architecture. In our multi-controller SDN architecture the GC is responsible for taking decisions that require network-wide knowledge and pass instructions to LCs. The WiFi APs differentiate the traffic flows generated from user devices into two classes including HP flows and LP flows using support vector machine, a machine learning approach. HP flows are more delay sensitive, so they are forwarded to a switch earlier. Upon a new flow arrival at the switch, this switch inspects the first flow request and queries the GC for computing the corresponding flow policies. This global controller dispatches the requests to local controllers based on their load values for flow processing. The load of each LC is calculated. Markov Chain Model (MCM) decides whether a LC will remain in underloaded or overloaded state in the future. If the LC is forecasted as overloaded then the flow migration process is initiated by the GC. The LC to transfer the flows is decided by Type-2 Fuzzy based Particle Swarm Optimization (TFPSO) algorithm. Once the appropriate LC is chosen by the GC, the flows are forwarded to it for computation of the flow policies which are then deployed back to the switch to achieve high request processing at a reduced response time.

The rest of this paper is organized as follows: In section 2 we describe the details of the proposed efficient algorithm approach to achieve load balancing in SD-WiFi. Then in section 3 we report the simulation results for evaluating the performance of the proposed scheme. In section 4 we summarize the related work on the load balancing issue in SD-WiFi. Finally we Conclude this paper in section 5.

2 Method

2.1 Overview

In this section, we discuss the proposed SD-WiFi architecture which consists of three modules, the first module performs flow classification at the data plane, the second module contains the multi-controller arrangement of local and global controllers at the control plane and the third module supports the load balancing algorithms at the application plane, as shown in Fig. 1. Flow requests

are generated from access networks (i.e., wireless local area networks (WLAN) and smart home networks), which are then classified with priority by the WiFi APs using SVM. The requests are then forwarded to the controller via OpenFlow switches for further processing. MCM predicts the future load of a LC and TFPSO determines the optimal underloaded LC for migration.

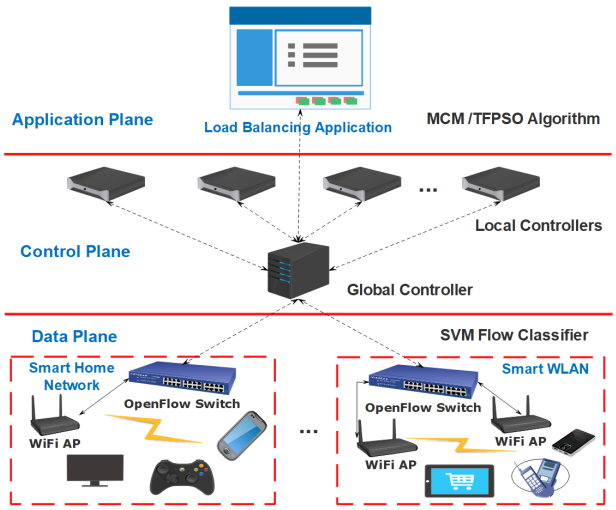

Fig. 1. The proposed software defined WiFi network architecture with multi-controller load balancing strategy.

SVM Flow Classifier To help achieve load balancing at the data plane, we classify the incoming flow requests into two types. SVM classifier which is an efficient binary classifier is used to find the type of incoming flows based on flow size and delay constraint.

SVM is a supervised approach which uses training data to categorize incoming flow requests by determining optimal hyperplane. In SVM classifier, flows are represented by support vectors in which each point can be laid on hyperplane. If we have n training observations $(\boldsymbol{x}_1, ..., \boldsymbol{x}_n) \in \mathbb{R}^p$,where $\mathbb{R}^p$ is the real valued $\mathcal{P}$ dimensional space, and n class labels $y_1, ..., y_n \in \{-1, 1\}$, support vector machine tends to build an optimal hyperplane boundary subject to the constraints given in equation 1,

$$y_i \left(\boldsymbol{b} \cdot \boldsymbol{x} + b_0\right) \geq M(1 - \epsilon_i), \quad \forall i = 1, ..., n \tag{1}$$

where y_i represent either HP i^{th} flow or LP i^{th} flow, $\boldsymbol{x}$ represent the flow size and delay constraint values, $(\boldsymbol{b} \cdot \boldsymbol{x} + b_0)$ defines an affine hyperplane, M represents

the maximum possible margin between two flows, ϵ_i are the slack variables used to handle the non separable cases.

The flows having a small size and low deadline are classified as HP flows and flows having a large size and high deadline are classified as LP flows. Based on the flow type, WiFi APs forward the flows to the GC through OpenFlow switches.

Load Balancing A Markov chain model is used in GC to predict the future load of LCs using the overload and underload states. If an LC is identified as overloaded, GC initiates the flow migration process. Fig.2 shows the transition states of LCs using MCM model.

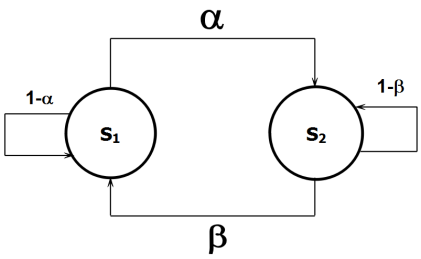

Fig. 2. Markov Chain Model

In figure 2, α represent the probability of a LC to become overloaded state (S_2) and β represents the probability of an LC to become underloaded state (S_1) at next state. The transition probability of LC to become S_2 state from S_1 is determined as,

$$P\left(S_1, S_2\right) = P\left(S_1/S_2\right) P\left(S_2\right) \tag{2}$$

Transmission matrix is constructed using transition probability and given as,

$$T = \begin{bmatrix} (1-\alpha) & \alpha \\ \beta & (1-\beta) \end{bmatrix}$$

Transition matrix is filled by the current load status values of the LC. These probability values help in determining the future load of an LC which is calculated as follow,

$$L_F\left(LC\right) = Current\ load\ on\ LC * Transition\ matrix \tag{3}$$

Where $L_F(LC)$ represent future load of LC. The current load of LC is computed using the number of flows in controller at time t and it is given as,

$$L = \frac{\sum_{i=1}^{n} S_i(T)}{T} \tag{4}$$

Where i represents the switch and n represents the number of switches, T denotes time instant and $S_i(T)$ represent amount of flows received from i^{th} switch by an LC.

If an LC is predicted as overloaded from 3, then GC migrates the flows from overloaded LC to underloaded LC. The destination LC for migration is selected using novel TFPSO algorithm. In TFPSO, initially the PSO algorithm is started. The algorithm takes every LC as a particle i with position X_i and these particles pass through search space to find the optima with velocity V_i. The position of the particle is updated after every cycle with a new position and velocity values represented as primes in equation 5 and 6.

$$X_i' = X_i + V_i' \tag{5}$$

$$V_i' = \omega V_i + c_1 \text{rand}\,()\,(Y_i - X_i) + c_2 \text{rand}\,()\,(Y^* - X_i) \tag{6}$$

Where c_1 and c_2 represent the learning factors that represent the weight of velocity of the particles flying towards global best position, $rand()$ is random function with uniform values between $[0-1]$, w is the inertia weight, Y_i represent the previous personal best and Y^* represent the best present solution among all Y_i.

Type-2 fuzzy logic helps in computing the fitness of each LCs. The fuzzy logic system consists of fuzzifier, rule base, inference machine and a defuzzifier giving a crisp output value. Load, available memory and CPU are taken as membership functions. Type 2 fuzzy set is fed into inference model which combine fuzzy set and rule base. It takes multiple membership functions as input and gives single output. The rule base consist of M rules and l^{th} rule is given as,

$$R^l : IF x_1 is \widetilde{F_1^l} and \ldots and x_p is \widetilde{F_p^l}\ THEN\ y\ is\ \widetilde{G^l} \tag{7}$$

Where $x = (x_1, x_2, x_3, \ldots, x_p)$ represent input membership functions and $\widetilde{F^l} = (\widetilde{F_1^l}, \widetilde{F_2^l}, \widetilde{F_3^l}, \ldots \widetilde{F_p^l})$ represent type 2 fuzzy set of x and $\widetilde{G^l}$ represent crisp output value for input membership function in l^{th} rule.

The output defuzzified values for each LC is updated as a fitness value in the PSO algorithm. The pbest and gbest values are updated at each iteration using fuzzy set and optimal LC is selected based on gbest value. As per our proposed work the defuzzified value is high for an LC which has a small amount of load, large memory and CPU.

Algorithm 1 explains the process of our proposed load balancing method. The optimal selection of an LC helps to prevent a specific LC from overloading.

3 Performance Evaluation

In this section we report the simulation setup and the parameters adjusted to evaluate the performance of the proposed scheme.

Algorithm 1 The Proposed Load Balancing Algorithm

```
Receive all flows from users
for allflows do
    Initialize flow size, delay constraint.
    Feed flows into SVM classifier
    Classify flows into HP and LP
    Send flows to GC through switches based on priority
end for
Assign flows to LCs
for allLCs do
    Compute current load on LCs using eqn 4
    Compute probability of state transition for each LC using eqn 2
    Construct transmission matrix
    Compute future load using eqn 3
end for
if LCisoverloaded then
    Migrate flows from that LC
    Initialize all particles with fitness value
    Initialize pbest & gbest values
    for allparticles do
        Compute fitness value for each particle using fuzzy logic
        Update pbest & gbest in each iteration
        If maximum epochs meet
        Stop.
    end for
else
    repeat the process;
end if
```

3.1 Simulation Setup

We have used OpenFlow to implement the simulation model of the software defined WiFi networks for the proposed load balancing algorithm. The proposed SD-WiFi architecture consists of mobile users, WiFi APs, OpenFlow switches, and controllers. POX is used as an SDN controller and the load balancing algorithm is designed in C++. The simulation setup is depicted in Fig. 3.

Finally, all our experiments are performed on a HP PC with windows 10 OS, a core i7 2.4 GHz CPU and 16 GB of RAM. Table 1 show the key simulation parameters used for the proposed SD-WiFi architecture.

3.2 Comparative Results

In this section we compare our proposed load balancing method based on a novel TFPSO algorithm, with existing research works [10], [11], on load balancing to show the efficiency of our proposed work.

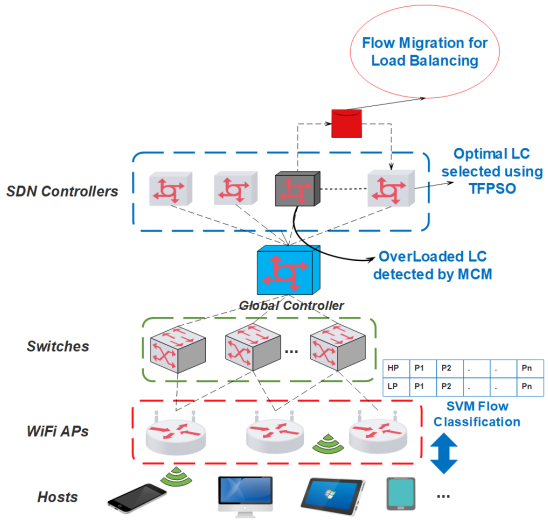

Fig. 3. Simulation Topology

Table 1. Simulation Parameters

Parameters	Value	
Number of Controllers	Global Controller	1
	Local Controllers	4
Number of OpenFlow Switches	3	
Number of WiFi Access Points	3	
Number of Users	40	
Number of CPU	4	
Memory	16 GB	
Hard disk	500 GB	
Connection Speed	300 Mbps	
Flow Size	100 Bytes	
Flow Request Interval	0.001 seconds	
Flow Table Size	1000 Entries	
Flow Timeout	0.2 seconds	
Transmission Range	1000*1000 meters	
Simulation Time	200 seconds	

Throughput Fig. 4 shows the improved throughput performance achieved by the TFPSO method. Our results are compared to the flow stealer method.

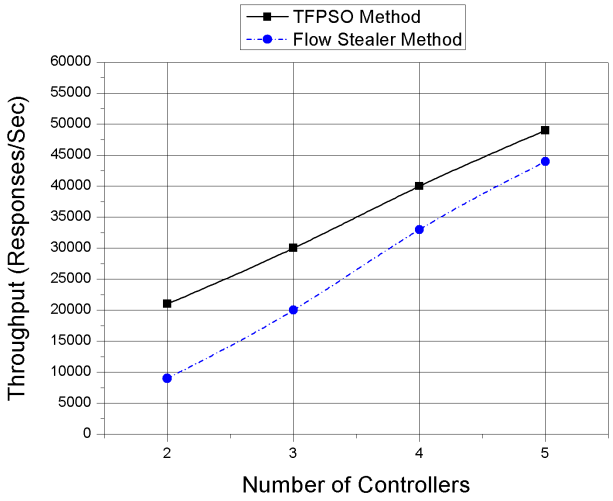

Fig. 4. The Throughput Performance.

A light weight load balancing method for multiple SDN controllers named as flow stealer was proposed for load balancing [11]. In this method the load was balanced by idle controllers which shared workloads temporarily from overloaded controllers. This method failed to balance the load, if the idle node became busy as opposed to our proposed scheme where an optimal underloaded LC is selected prior to a specific controller getting overloaded.

The TFPSO method achieves throughput about 5000 responses/second which is significantly greater than the flow stealer method.

Work Load Fig. 5 depicts the work handled by the controllers in a given period of time.

In [11] the work load for controllers was measured under burst traffic for different time durations. Best work load performance was achieved when the time duration of burst traffic was short. After the burst traffic a certain controller gets overloaded and when the flow steal event is initiated the workload decreases linearly. In comparison to this method, the TFPSO method shows constant behavior in the work load pattern as the flows are shifted prior to the controller getting overloaded.

At 12 seconds the flows handled by the controller in the flow stealer method are 4500 in comparison to under 1000 flows handled by the controller under the TFPSO method. This shows a better workload performance by our proposed load balancing algorithm.

Latency Fig.6 shows the comparative analysis of latency of our proposed scheme to the MPSO-CO method. Latency is defined as the average time between the

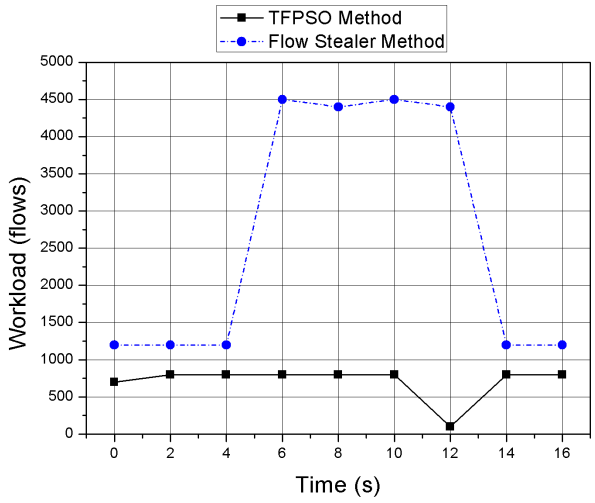

Fig. 5. The Workload Performance.

start of the flow transmission from one end-user to the time it reaches the other end user

In [10] a Modified Constrained Optimization Particle Swarm Optimization algorithm (MPSO-CO) is used to achieve load balancing in software defined cloud/fog networking. In MPSO-CO method, cloud, SDN controllers and fog are integrated to form software defined cloud/fog networking which is then applied to IoV. The cloud server computation system is enhanced when the load is high or a controller gets overloaded.

The latency curve for TFPSO shows better performance in comparison to the MPSO-CO method as the controllers are prevented from going into the overloaded state, which implies fast flow processing and response time. Our proposed work maintains latency within 0.5 seconds upto a flow size of about 1 GB.

4 Related Work

Load balancing was performed using a Genetic Algorithm in an SDN controller [12]. GA performs selection, mutation and cross over processes for load balancing in SDN. In this method all switches were ranked according to their load and a switch with the largest load was assigned to a controller with the least load. The switches were assigned based only on the load and other parameters were not considered which increased the execution time. The initialization of GA is complex due to many parameters such as mutation rate, cross over rate etc. which need to be initialized. Our proposed scheme takes into account other parameters apart from load such as available memory and CPU. Finite

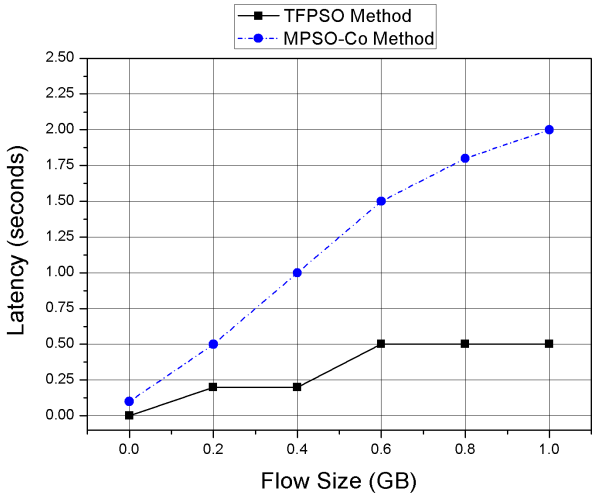

Fig. 6. Comparative Analysis of Latency.

State Machine (FSM) and policy was introduced for load balancing based on SDN [13]. In FSM state of each controller was related to a policy which decides the packet forwarding in network. In this method the load of each controller is not determined for load balancing. A Collaborative platform based on restless framework was used to balance the load among controllers in a distributed SDN network [14]. The platform uses the data collecting algorithm to collect the controllers load information. Load was balanced by migration of the control permission of the switch among distributed controllers. In migration process, load was considered only in source controller, as apposed to our proposed method where the load on destination controller was also considered. In dense Wireless Local Area Network (WLAN), load balancing among APs was performed by a semi-matching based load balancing scheme [15] which runs in centralized controller. The controller determined the load distribution among APs according to collected Channel Busy Time Ratio (CBTR) information of entire network. The problem was modeled as a weighted bipartite graph matching problem and semi-matched APs for overloaded APs were found that maximize the total weight of weighted bipartite graph. The computational complexity is increased due to several model calculations. In our proposed scheme PSO is used which has a limited number of parameters to tune.

In [16] SDN based WiFi network controllers and APs were organized into a two-tier architecture to allow the controller to evaluate the degree of load balancing among APs. The load balancing was performed at AP side and controller side by using openflow extension messages. The Controller receives reports from an AP such as capacity, load and association table. The associated client stations are de associated using Last In First Out (LIFO) which increases the waiting

time for first requested user since user priorities are not considered. The TFPSO method process flows on the base of their nature, HP flows sensitive to delay are processed first. AP load balancing scheme based handover in SDN based WiFi networks was proposed in [17] which involves two processes including a load balancing and hand over process. Load balancing was achieved by incorporating three modules in the network as follows: (i) load control module (ii) Metric monitoring module and (iii) Decision conversion module. If an AP was overloaded then the station in that AP was transferred to a neighboring AP by an SDN controller. But this method does not consider load of a neighboring AP, so there are chances of that AP becoming overloaded as opposed to our proposed scheme where the load of the destination controller is also measured. SDN based load balancing for LTE network was proposed in [18]. In this method when a new flow was received at the switch it informs the controller by sending PACKET_IN message. The new flow rules were installed by the controller whenever a flow was received at the switch. The load of the gateway was calculated using static information about traffic per volume which was determined by OpenFlow switches. In this method one controller is employed which is insufficient for numerous packets from users. Our proposed scheme takes into account the scalability issues and hence has introduced multi controllers.

5 Conclusion

In this paper, we have proposed a novel multi-controller load balancing algorithm to balance load efficiently in SD-WiFi. We have designed an SDN architecture with multiple controllers and balance the load among those controllers. Initially we have classified the incoming flow requests into HP and LP flows at WiFi APs using an SVM approach. The GC is responsible for monitoring all LCs and predicting future load of LCs using an MCM model. The flows in an overloaded LC are migrated to another LC which is selected by a TFPSO algorithm to balance load. TFPSO ensures load balancing in a migrated LC by selecting optimal LC with consideration of load, memory and CPU. Our simulation results have shown that the proposed scheme has higher throughput by 33% and higher workload performance by 70 % than the flow stealer approach. Our scheme outperforms the MPSO-CO method by achieving better latency results. Our proposed scheme can be applied to real time applications running on cloud-based software defined WiFi network frameworks with heavy loading to achieve improved QoS.

Acknowledgment This work was supported in part by the National Natural Science Foundation of China (No. 61370231).

References

1. Rawat, D.B., Reddy, S.R.: Software defined networking architecture, security and energy efficiency: A survey. IEEE Communications Surveys & Tutorials **19**(1), 325–346 (Firstquarter 2017). https://doi.org/10.1109/COMST.2016.2618874

2. Wang, C., Hu, B., Chen, S., Li, D., Liu, B.: A switch migration-based decision-making scheme for balancing load in sdn. IEEE Access **5**, 4537–4544 (2017)
3. Chen, Z., Manzoor, S., Gao, Y., Hei, X.: Achieving load balancing in high-density software defined WiFi networks. In: International Conference on Frontiers of Information Technology (FIT) (Dec 2017)
4. Manzoor, S., Hei, X., Cheng, W.: Towards dynamic two-tier load balancing for software defined WiFi networks. In: International Conference on Communication and Information Systems (ICCIS) (Nov 2017)
5. Sarma, A., Chakraborty, S., Nandi, S.: Deciding handover points based on context-aware load balancing in a wifi-wimax heterogeneous network environment. IEEE Transactions on Vehicular Technology **65**(1), 348–357 (2016)
6. Yang, M., Li, Y., Jin, D., Zeng, L., Wu, X., Vasilakos, A.V.: Software-defined and virtualized future mobile and wireless networks: A survey. Mobile Networks and Applications **20**(1), 4–18 (2015)
7. Yang, S.N., Ke, C.H., Lin, Y.B., Gan, C.H.: Mobility management through access network discovery and selection function for load balancing and power saving in software-defined networking environment. EURASIP Journal on Wireless Communications and Networking **2016**(1), 204 (2016)
8. Duan, X., Akhtar, A.M., Wang, X.: Software-defined networking-based resource management: data offloading with load balancing in 5g hetnet. EURASIP Journal on Wireless Communications and Networking **2015**(1), 1–13 (2015)
9. Lin, T.L., Kuo, C.H., Chang, H.Y., Chang, W.K., Lin, Y.Y.: A parameterized wildcard method based on sdn for server load balancing. In: Networking and Network Applications (NaNA), 2016 International Conference on. pp. 383–386. IEEE (2016)
10. He, X., Ren, Z., Shi, C., Fang, J.: A novel load balancing strategy of software-defined cloud/fog networking in the internet of vehicles. China Communications **13**(2), 140–149 (2016)
11. Song, P., Liu, Y., Liu, T., Qian, D.: Flow stealer: lightweight load balancing by stealing flows in distributed SDN controllers. Science China Information Sciences **60**(3) (2017)
12. Kang, S.B., Kwon, G.I.: Load balancing of software-defined network controller using genetic algorithm (2016)
13. Zhou, Y., Ruan, L., Xiao, L., Liu, R.: A method for load balancing based on software defined network. Advanced Science and Technology Letters **45**, 43–48 (2014)
14. Zhong, H., Sheng, J., Cui, J., Xu, Y.: Scplbs: a smart cooperative platform for load balancing and security on sdn distributed controllers. In: Cybernetics (CYBCONF), 2017 3rd IEEE International Conference on. pp. 1–6. IEEE (2017)
15. Lei, T., Wen, X., Lu, Z., Li, Y.: A semi-matching based load balancing scheme for dense ieee 802.11 wlans. IEEE Access **5**, 15332–15339 (2017)
16. Lin, Y.D., Wang, C.C., Lu, Y.J., Lai, Y.C., Yang, H.C.: Two-tier dynamic load balancing in sdn-enabled wi-fi networks. Wireless Networks pp. 1–13 (2017)
17. Kiran, N., Changchuan, Y., Akram, Z.: Ap load balance based handover in software defined wifi systems. In: Network Infrastructure and Digital Content (IC-NIDC), 2016 IEEE International Conference on. pp. 6–11. IEEE (2016)
18. Adalian, N., Ajaeiya, G., Dawy, Z., Elhajj, I.H., Kayssi, A., Chehab, A.: Load balancing in lte core networks using sdn. In: Multidisciplinary Conference on Engineering Technology (IMCET), IEEE International. pp. 213–217. IEEE (2016)