SDN Controller Assignment and Load Balancing With Minimum Quota of Processing Capacity

Abderrahime Filali *, Abdellatif Kobbane *, Mouna Elmachkour *, Soumaya Cherkaoui †
* ENSIAS, Rabat IT Center, Mohammed V University of Rabat, Morocco
† University of Sherbrooke, Canada
abderrahime.filali@um5s.net.ma, abdellatif.kobbane@um5.ac.ma, mouna.elmachkour@gmail.com, soumaya.cherkaoui@usherbrooke.ca

***Abstract*—SDN technology has arrived to solve several limitations of standard networks such as flexibility, scalability and programmability. In a data center environment, adopting the SDN approach where switches are statically linked to controllers creates load balancing problems. This issue is due to the traffic variation between controllers and switches, which influences the response time of the controllers. In this paper, we propose a dynamic assignment of switches to controllers by formulating the problem as a one-to-many matching game with a minimum quota that each controller has to achieve. This quota represents the utilization of the processing capacity of the controller. In addition, an efficient algorithm is defined to ensure a stable matching between switches and controllers in order to maintain load balancing and reduce the latency of controllers. Numerical results confirm the performance of our proposed model compared to a static assignment of switches in terms of load balancing and minimization of the response time especially, when the network becomes too loaded.**

***Keywords*—Software Defined Networking (SDN), Matching game theory, Load balancing, Latency.**

I. Introduction

Currently, several approaches are emerging, which promise to increase the agility of the network and the management of its resources. Software Defined Networking (SDN) is part of these new technologies. The concept on which it is based consists of decoupling the control plane and the data plane, which makes possible to converge the network administration to a centralized and extensible orchestration platform capable to automate the supply and the configuration of the entire infrastructure. Thus, the control plane is placed in a controller (e.g Beacon [10], Floodlight [11], Opendaylight [12], Ryu [13]), which has visibility over the entire network, including the hosts that are connected to it. Furthermore, a global view of the network's topology. In Software Defined Networking pushing all the control functionalities to a centralized controller may pose scalability and resilience problems [1], because the size of the network grows so at some point the controller cannot handle all the received requests from the data plane. To address such problems and to avoid a single point of failure the control plane is implemented as a distributed system (e.g HyperFlow [14], Opendaylight [12], Onix [15], NVP [16]). In such architecture, the switch-controller mapping is static. Thus, it can present a suboptimal use of resources due to the frequently variation of traffic in a data center [23] networks, for example, having controllers that reach the maximum of their capacities while others are in idle state. For all these reasons, the assignment of switches to controllers must be dynamic [4], i.e. periodically re-assign switches to controllers for a better response time of controllers [5] and an efficient utilization of the processing resources through load balancing [24].

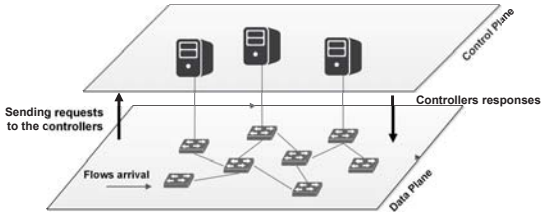

Fig. 1. In a SDN model all switches send their requests to the controllers who must respond within reasonable time

Several researches have been focused on improving the behavior of the control plane in order to adapted it to changes in the network's characteristics (e.g. Topology, links, load,... etc.). They aim providing more scalability by solving issues related to the connections between the two planes that influence the latency of the system.

The authors in [6] provide a solution that combines the dynamic switch-controller association and dynamic control devolution while maintaining a reasonable wait time in queues. This trade-off aims to reduce the communication cost between data and control planes and the computational cost spent by the switch to process its requests locally. For this purpose, they propose a Greedy algorithm based on a stochastic optimization problem which its used to make a scheduling decision that consists of either uploading requests to controllers or adding them to the local queue of the switch. Bari et al. [7] formulated the dynamic controller provisioning in a WAN as an Integer Linear Program in order to minimize flow setup time and communication overhead. They propose two algorithm (DCP-GK and DCP-SA) to re-assign switches to controllers taking into consideration the switch-controller exchange cost, inter-controller synchronization cost and switch re-assignment cost. The work in [8] responses to

the dynamic assignment question by building an elastic distributed controller architecture. This solution increases and decreases the pool of controllers according to traffic conditions. They also propose an efficient protocol to migrate a switch from an overloaded controller to another less loaded. In [9], the dynamic assignment problem of controllers is studied in a datacenter environment. They identified the possibility of reducing response time of the controllers and balancing the load between them by formulated the problem as an optimization problem. Thus, for solving this problem and ensuring the performance of the mapping between switches and controllers they use game theory since it is a powerful technique for decision-making in communication networks as presented in [21], [25] and [22]. Therefore, they propose a two-phase algorithm that connects the many-to-one stable matching problem with coalition game in which switches participate to achieve a Nash stable solution [3]. However, applying the second phase of the algorithm (coalition game), which is essential for load balancing is very costly in terms of network's traffic load since the switch migration from one controller to another requires to exchange several messages. This work inspired us to exploit the first phase and abandon the coalition game. Accordingly, to achieve the load balancing we propose the utilization of the minimum quotas for each controller.

Our contribution consists in presenting a new method to associate switches to the SDN controllers. This approach aims to minimize the latency [2] between controllers and their switches by ensuring the balancing of their loads. First, the problem is formulated as an optimization problem to minimize the total response time of the control plane. Then, to solve this problem a one-to-many matching game with minimum resource utilization is defined in which each controller must reach its minimum quota in terms of requests transmitted by the switches in order to balance the network's load. Moreover, we propose an algorithm that allows the assignment of all switches to the controllers. The assignment is based on the number of requests that are going to be transmitted taking into account that the processing capacity of controllers will not be exceeded. To achieve a balanced load, controllers should approve switches' requests such that the minimum resource utilization and capacity constraints are met.

The remainder of this paper is structured as follows: Section II defines the structure of the network and formulates the optimization problem. Section III describes the matching game approach as a solution and establishes the proposed algorithm. Section IV presents the analysis of the results. Finally, section V end this paper with a conclusion.

II. System Model

A. Response Time Model

In a physically distributed architecture of the control plane within a datacenter we consider a set of controllers denoted by $C = \{c_1, c_2, ..., c_n\}, |C| = n$, where $|.|$ denotes the set cardinality. Each controller is characterized by a number of requests that it can handle in one time unit called the processing capacity $\alpha = \{\alpha_{c_1}, \alpha_{c_2}, ..., \alpha_{c_n}\}$. The data plane consists of a set S of switches, $S = \{s_1, s_2, ..., s_m\}, |S| = m$. The set of switches managed by the controller c_j is denoted by S_{c_j}, where $c_j \in C$ and $S_{c_j} \subseteq S$. In a data center, the switches are not in a same level compared to a controller so, we define d_{ij} as hop distance which refers to the number of intermediary switches that separate the switch s_i and controller c_j.

To calculate the response time of a controller we used the work that has been done in [9]. Let $l_i(t)$ be the load generated by the switch $s_i \in S$ in terms of the requests sent to the controller. Thus, the load of controller c_j at time t can be represented by:

$$L_{c_j}(t) = \sum_{i=1}^{|S_{c_j}|} l_i(t) \qquad (1)$$

Since the arrival times of the requests are distributed according to a Poisson process [9] and the processing times (service times) are independent of each other and independent of the arrival process, the controller can be modeled as an $M/M/1$ queue [17]. Therefore, the average sojourn time in the system (the j^{th} controller) by applying Little's law is : $\frac{1}{\alpha_{c_j} - L_{c_j}}$.

As mentioned in [2], the propagation delay within a datacenter depends on the physical properties of medium and on the various functions applied to the packet. In general, the processing delays of the controller dominate propagation delays. Accordingly, the response time of the controller is modeled as the processing time that can be considered as the sojourn time (waiting time + service time). We denote the response time of the c_j controller at time slot t as $T_{res_{c_j}}(t)$:

$$T_{res_{c_j}}(t) = \frac{1}{\alpha_{c_j} - L_{c_j}(t)} \qquad (2)$$

The processing capacity α_{c_j} differs from one controller to another and the number of requests received by the controller does not depend on the number of switches it manages. Thus, based on the load of the controller L_{c_j} in a time slot t, we can calculate the utilization rate of the processing capacity, which we have called the resource utilization and can be represented as follows:

$$U_{c_j}(t) = \frac{L_{c_j}(t)}{\alpha_{c_j}} \qquad (3)$$

B. Problem Formulation

Our primary objective is to minimize the response time of the controller while ensuring the load balancing among controllers. For this purpose, the assignment of switches to controllers must take into account the use of the controller's resource defined in equation (3). Thus, we formulate the following optimization problem:

$$min \sum_{c_j \in C} T_{res_{c_j}}(t) \quad (4)$$

$$\sum_{i=1}^{|S_{c_j}|} l_i(t) \leq \alpha_{c_j}, \forall c_j \in C \quad (5)$$

$$U_{c_j}(t) \geq U_{c_j}^{min}, \forall c_j \in C \quad (6)$$

$$S_{c_j} \cap S_{c_{j'}} = \emptyset, \forall j \neq j', S_{c_j} \subseteq S \, and \, S_{c_{j'}} \subseteq S \quad (7)$$

$$\bigcup_{c_j \in C} S_{c_j} = S \quad (8)$$

Where $U_{c_j}^{min}$ represents the minimum utilization of the controller capacity. This threshold represents a minimum quota in terms of the number of requests to be managed. The minimum quota is important to balance the network's load. The objective function (4) minimizes as much as possible the response time, while the constraints are used to ensure the following conditions:

- Constraint (5) guarantees that the capacity of any controller is not exceeded. The processing capacity α_{c_j} can be considered as the maximum quota of the controllers.
- Constraint (6) allows to maintain the load on each controller upper certain threshold, which avoids having an overloaded controller while others are not.
- Constraint (7) ensures that each switch maintains connectivity with a single master controller.
- Constraint (8) guarantees that all switches are assigned to the controllers.

In order to periodically balance the load between the controllers, the assignment of the switches to the controllers should be dynamic taking into consideration the current load within the controllers as well as the load that would be transmitted by the switches. As a solution, the matching game allows a stable association between the controllers and the switches.

III. Matching game solution with minimum resource utilization

A. Game formulation

To resolve the association problem between controllers and switches we present a solution based on one-to-many matching game. Such game is a two-sided assignment problem between two disjoint sets of players where each player can be adapted to several players of the opposite set according to preference relations. In our case, the matching problem have a minimum quota $U_{c_j}^{min}$, which must be taken into account.

Definition 1. Given two disjoint finite sets of players, C for controllers and S for switches, a mapping μ can be defined as a one-to-many matching relation, $\mu : C \rightarrow S$ that satisfies:
i-$\forall c_j \in C, \mu(c_j) \subseteq S$
ii-$\forall s_i \in S, \mu(s_i) \in C$
iii-$\mu(s_i) = c_j$, if and only if $s_i \in \mu(c_j)$.

Where, for each controller c_j, the request arrival rate of each assigned switches must not exceed the processing capacity α_{c_j} and, must achieve at least the minimum resource utilization (constraints 5-6). $\mu(s_i)$ denotes the current controller to which the switch s_i is associated.

Definition 2. A switch-controller pair (s_i, c_j) is said to be a blocking pair of the matching μ, if and only if:
1- $s_i \succ_{c_j} s_{i'}$ for some $s_{i'} \in S$.
2- $c_j \succ_{s_i} \mu(s_i)$.
The matching is stable if there is no incentive for any pair to deviate from μ, which means there is no blocking pair.

B. Preference Relations of the Switches and controllers

A matching game is characterized by the preference relations in which, each player has a preference profile over the players of the other set. In our problem each switch s_i has a preference list $\succ_{s_i}$ over controllers, while each controller c_j has a priority relation $\succ_{c_j}$ overs switches.
As mentioned in [9] the switches build their preferences based on the response time that the controller can provide. However, using only the response time will generate the same preference lists for all switches while they are not. For this, the preference list of the switches will be based on the response time defined in equation (2) multiplied by the hop distance d_{ij}. Therefor the preference relations of switches are:

$$c_j \succ_{s_i} c_{j'} \Leftrightarrow T_{res_{c_j}} * d_{ij} \leq T_{res_{c_{j'}}} * d_{ij} \quad (9)$$

Each switch s_i classifies its preferred controllers in an ascending order starting with the controller that provides the best product result of response time and hop distance. In addition, the processing capacity α_{c_j} of each classified controller must be equal at least to the number of requests $l_i(t)$ that the switch transmits in the time slot t.

On the other side, the controllers define their preferences lists over switches with respect to the number of requests that must not exceed the maximum of the processing capacity as well as, the hop distance separating them. To this end, the preference relations of the controllers are defined as follow:

$$s_i \succ_{c_j} s_{i'} \Leftrightarrow l_i * d_{ij} \leq l_{i'} * d_{i'j} \, where,$$
$$L_{c_j}(t) + l_i(t) \leq \alpha_{c_j}, \quad (10)$$

Each controller c_j classifies its preferred switches in an ascending order starting with the switches which have the smallest load to be transmitted and which are closer.

To satisfy the minimum resource utilization (minimum quota) of the controllers in our matching problem a special mechanism is required. Thus, Daniel et al. [18] propose a Multi Stage Deferred Acceptance (MSDA) algorithm that

explicitly uses the minimum quota. This algorithm introduces the notion of precedence list that ranks all switches in an order that depends on the studied problem. The usefulness of this list is to define which switches may have to be assigned to a lower ranked controller. In our case we call the precedence list Load List (LL) $\succ_{LL}$ and, it contains switches ranked according to the load they generate in a descending order. In other words, we give a higher priority to a switch having a higher load so that it will be assigned to its preferred controller.

$$s_i \succ_{LL} s_{i'} \Leftrightarrow l_i(t) \geq l_{i'}(t), \forall s_i, s_{i'} \in S \qquad (11)$$

With the MSAD algorithm, the standard definition of the blocking pairs will have an additional requirement that the switch s_i envy toward $s_{i'}$ if it has a higher priority than switch $s_{i'}$. Accordingly, **Definition 2.** become: The pair (s_i, c_j) forms a PL-blocking pair of the matching if the following conditions are satisfied.
1- $s_i \succ_{c_j} s_{i'}$ and $s_i \succ_{LL} s_{i'}$ for some $s_{i'} \in S$.
2- $c_j \succ_{s_i} \mu(s_i)$.
Note that with this new definition we can say a matching μ is PL-fair if no switch-controller pair can form a PL-blocking pair.

C. Proposed algorithm

Our one-to-many matching algorithm inspired from [18] in which each switch has the right to choose one controller as his master according to its preference list (9), and each controller can admit a number of switches respecting the minimum resource utilization $U_{c_j}^{min}$ (3) and his processing capacity α_{c_j}. The matching with minimum quotas described in Algorithm 1, works by applying successively the following procedure.

The input is defined by the switches' and controllers' preference lists, the maximum and the minimum quotas of each controller ($q_{c_j}^{max}, q_{c_j}^{min}$). After initialization, the MSDA mechanism run until the Load List is not empty. In each stage k, we define in step 3 and 4 the set of temporarily reserved switches R^k having the lowest priority in the Load List such that the sum of their loads is less than or equal r^k. It can be seen that when the value of R tends to 0 the MSDA algorithm becomes approximately similar to the standard Deferred Acceptance DA algorithm in which all the switches will participate in the matching game during the first stage. In step 5 we check if there are still switches after reserving some of them to satisfy the minimum quotas. We apply the standard Deferred Acceptance Algorithm (DAA) on those switches. If not, it means that the minimum quotas is equal to the not yet assigned switches. Therefore, we run the DAA to ensure the assignment of switches to controllers who have not yet reached their minimum quota [19]. In steps 8 and 9 we define the new quotas for the controllers. The output of the algorithm is a set of the specific matchings at each stage defined by $\mu(c_j) = \cup_{k=1}^{K} \mu^k(c_j), \forall c_j \in C$.

Algorithm 1 Proposed Controller Assignment algorithm

Inputs: $\succ_{LL}, \succ_{s_i}, \forall s_i \in S, \alpha_{c_j} \equiv q_{c_j}^{max}, q_{c_j}^{min}, \forall c_j \in C$.
Outputs: $\mu(c_j), \forall c_j \in C$

1: **Initialize:** $\mu(c_j) = \emptyset$, $R^0 = LL$, $q_{c_j}^{1,max} = q_{c_j}^{max}$, $q_{c_j}^{1,min} = q_{c_j}^{min}$, $\forall c_j \in C$
2: **while** $LL \neq \emptyset$ **do**
3: $r^k = \sum_{c_j}^{|C|} q_{c_j}^{k,min}$
4: $R^k = \{s_{i*}, ..., \}$ where, $\sum_{i^*=1}^{|R^k|} l_{i^*}(t) \leq r^k$
5: **if** $(R^{k-1} \setminus R^k \neq \emptyset)$
run the standard DA algorithm on the switches in $R^{k-1} \setminus R^k$ with maximum quotas for the controllers equal to $q_{c_j}^{k,max}$, $\forall c_j \in C$ Moreover, remove them from LL liste.
6: **else**
run the standard DA algorithm on the switches in R^k with maximum quotas for the controllers equal to $q_{c_j}^{k,min}$, $\forall c_j \in C$ Moreover, remove them from LL liste.
7: **end if**
8: $q_{c_j}^{k,max} = q_{c_j}^{k-1,max} - \sum_{i^*=1}^{|\mu^k(c_j)|} l_{i^*}(t)$
9: $q_{c_j}^{k,min} = max\{0, q_{c_j}^{k-1,min} - \sum_{i^*=1}^{|\mu^k(c_j)|} l_{i^*}(t)|\}$
10: **end while**
11: $\mu(c_j) = \cup_{k=1}^{K} \mu^k(c_j), \forall c_j \in C$
12: $\mu(s_i) = \mu^{k(s_i)}(s_i), \forall s_i \in S$, where $k(s_i)$ is the stage at which switch s_i participated.

IV. NUMERICAL RESULTS

To illustrate the proposed algorithm we adopted a data center architecture composed by $n = 5$ controllers and $m = 300$ switches which is equivalent to a commercial data center [20]. Each controller has a capacity equal to 12K flows/s, which is sufficient to support the maximum load of the network. In each time slot t we take into consideration that each controller can already be loaded at a certain threshold. According to [20], a switch in such a system can receive a number of new flow ranging between 1000 and 5000 and, the hop distance d_{ij} changes between 1 and 5, $d_{ij} = 1$ means that the switch is directly connected to the controller.

In order to assess the performances of our algorithm in terms of minimization of the response time as well as, the load balancing we compared it with the static matching. In the static matching we define a specific topology considering that each controller can handle the maximum load generated by the set of switches it manages. In the first two simulations we vary the load of the system and observe its behavior. In the third simulation, the level of utilization of the processing capacity of each controller is plotted in the two type of association.

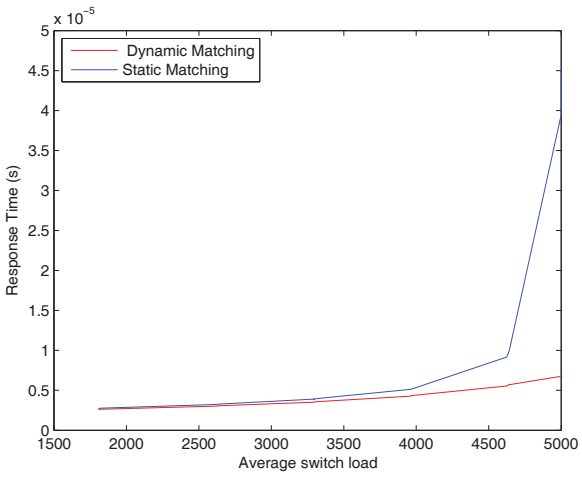

Fig. 2. Response time under different traffic load

Fig. 2 shows a scalability evaluation of the two matching models in terms of response time (equation 2) under a variable traffic load. In this experiment, we increased the arrival rate of the flows and the average response time of the various controllers is observed $\sum_{c_j \in C} \frac{T_{res_{c_j}}}{n}$. It is clear that the response time in dynamic association increases slightly when the load increases however, the controllers response time in the static assignment increases severely when the system become too loaded. A very big difference is observed when approaching to the 5000 flows.

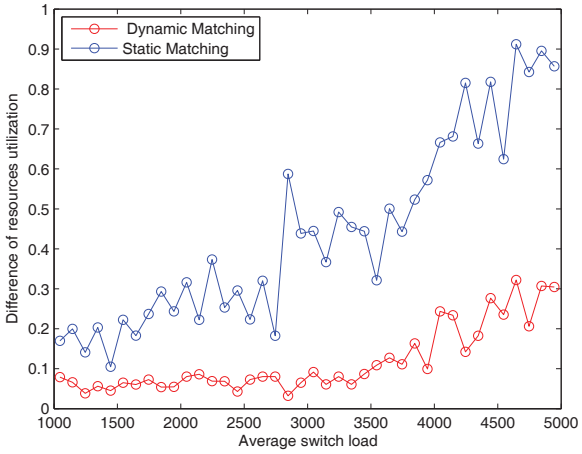

Fig. 3. Load balancing under different traffic load

From Fig. 3, we evaluate the load balancing among controllers by plotting the sum of the difference utilization of resource between controllers $\delta = \sum_{c_j, c_{j'} \in C} (U_{c_j} - U_{c_{j'}})$. It can be seen that even if the traffic load increases, the utilization of resources remains reasonable in dynamic matching, It varies between 0.1 and 0.2 in the interval of 1000 to 4000 flows. Which means the achievement of load balancing under different loads. In static matching, the difference δ is important mainly when the traffic load increases.

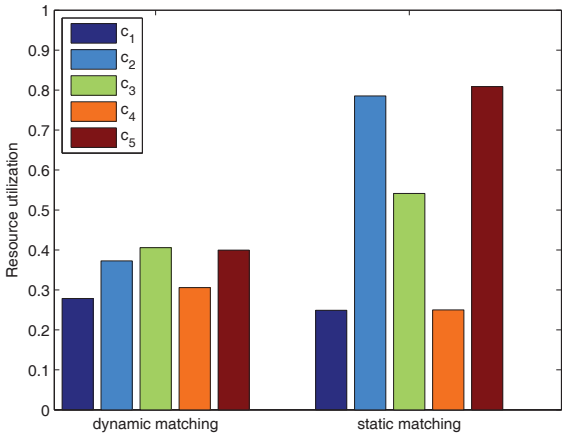

Fig. 4. Utilization of controllers' capacities

Fig. 4 illustrates the level of utilization of the capacity of each controller (equation 3) in the static and dynamic assignment. As depicted, our many-to-one matching game can ensure a balanced distribution of the load, which its not guaranteed in the static association, in which some controllers are over-exploited, e.g. the controller c2 and c5 are too loaded compared to the others. (U_{c_2} and $U_{c_5} \approx 0.8$). This could affect the subsequent arrival flows that will be processed by these controllers within a significant delay.

V. Conclusion

This paper studied the association problem between switches and controllers. We started with the formulation of the problem and proposed an optimization approach that aims to minimize the response time of the SDN controllers. Then, we solved this problem by resorting a one-to-many matching game with a minimum quota constraint to ensure the load balancing among controllers. Moreover, we designed a fair algorithm where the performance constraints that are crucial in the assignment operation, such as load balancing according the minimum quota and the respect of the processing capacity which must not be exceeded, were considered. The performance of the proposed solution is validated through a comparison with the static association of switches to the controllers. Numerical analysis proved that the dynamic relationship between the control plane and the data plane confirms the minimization of the response time. Therefore, the use of the minimum quota is appropriate to maintain a good distribution of the network's load.

REFERENCES

[1] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," in Proceedings of the IEEE, vol. 103, no. 1, Jan. 2015, pp. 14-76.

[2] C. Yu, et al., "Software-Defined Latency Monitoring in Data Center Networks," in Passive and Active Measurement, vol.8995, Mar 2015, pp. 360-372.

[3] E. A. Jorswieck, "Stable matchings for resource allocation in wireless networks," 2011 17th International Conference on Digital Signal Processing (DSP), Corfu, 2011, pp. 1-8.

[4] A. Krishnamurthy, S. P. Chandrabose, and A. Gember-Jacobson, "Pratyaastha: An efficient elastic distributed SDN control plane", in Proc. 3rd Workshop Hot Topics Softw. Defined Netw., Aug 2014, pp. 133-138.

[5] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks", in Proc. 2nd USENIX Conf. Hot Topics Manage. Internet Cloud Enterprise Netw. Services, 2012, p. 10.

[6] Xi Huang, Simeng Bian, Ziyu Shao, and Hong Xu, "Dynamic Switch-Controller Association and Control Devolution for SDN Systems", in IEEE ICC Networking and Internet Architecture, 2017, p. 14.

[7] M. F. Bari et al., "Dynamic Controller Provisioning in Software Defined Networks," Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013), Zurich, 2013, pp. 18-25

[8] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, "Towards an elastic distributed SDN controller", in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. , Aug 2013 , pp. 7-12.

[9] T. Wang, F. Liu, J. Guo and H. Xu, "Dynamic SDN controller assignment in data center networks: Stable matching with transfers," IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, 2016, pp. 1-9.

[10] D. Erickson, "The Beacon OpenFlow controller", in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. , 2013, pp. 13-18.

[11] Project Floodlight, "Floodlight", 2012. [Online], Available: http://www.projectfloodlight.org/.

[12] OpenDaylight, A Linux Foundation Collaborative Project, 2013. [Online], Available: http://www.opendaylight.org.

[13] Nippon Telegraph and Telephone Corporation, "RYU network operating system", 2012. [Online]. Available: https://osrg.github.io/ryu/.

[14] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow", in Proc. Internet Netw. Manage. Conf. Res. Enterprise Netw. , 2010, p. 3.

[15] T. Koponen et al., "Onix: A distributed control platform for large-scale production networks", in Proc. 9th USENIX Conf. Oper. Syst. Design Implement., 2010, pp. 1-6.

[16] T. Koponen et al., "Network virtualization in multi-tenant datacenters", in Proc. 11th USENIX Symp. Netw. Syst. Design Implement., Apr. 2014, pp. 203-216.

[17] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems", Synthesis Lectures on Communication Networks, vol. 3, no. 1, 2010, pp. 1-211.

[18] D. E. Fragiadakis, A. Iwasaki, P. Troyan, S. Ueda, and M. Yokoo, "Strategyproof matching with minimum quotas", in ACM Transactions on Economics and Computation, vol 4, December 2015 .

[19] O. Semiari, W. Saad and M. Bennis, "Downlink Cell Association and Load Balancing for Joint Millimeter Wave-Microwave Cellular Networks," 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, 2016, pp. 1-6.

[20] T. Benson, A. Akella, and D. A. Maltz, "Network Traffic Characteristics of Data Centers in the Wild". In Proc. ACM IMC, 2010, pp. 267-280.

[21] Z. Han, D. Niyato, W. Saad, T. Basar, and A. Hjrungnes., "Game Theory in Wireless and Communication Networks - Theory, Models and Applications". Cambridge University Press, New York, USA, 2012.

[22] O. A. Oualhaj, E. Sabir, A. Kobbane, J. Ben-Othman and M. E. Koutbi, "A college admissions game for content caching in heterogeneous delay tolerant networks," 2016 23rd International Conference on Telecommunications (ICT), Thessaloniki, 2016, pp. 1-5.

[23] R. Cziva, S. Jout, D. Stapleton, F. P. Tso and D. P. Pezaros, "SDN-Based Virtual Machine Management for Cloud Data Centers," in IEEE Transactions on Network and Service Management, vol. 13, no. 2, June 2016, pp. 212-225.

[24] A. Craig, B. Nandy, I. Lambadaris and P. Ashwood-Smith, "Load balancing for multicast traffic in SDN using real-time link cost modification," 2015 IEEE International Conference on Communications (ICC), London, 2015, pp. 5789-5795.

[25] M. Amine, A. Walid, A. Kobbane and S. Cherkaoui, "A many-to-many matching game in ultra-dense LTE HetNets," 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), Valencia, 2017, pp. 1245-1250.