DSFSM: Delay-Sensitive Fractional Switch Migration For Multi-Controller Software-Defined Networks

Upendra Prajapati, Bijoy Chand Chatterjee, and Amit Banerjee
South Asian University, New Delhi, India

***Abstract*—Multi-controller software-defined networks (SDNs) are an emerging technology that transforms the networking industries by enhancing the system performance of applications with massive traffic. Delays due to frequent fractional switch migrations among controllers are a significant obstacle in the multi-controller SDN, degrading overall performance. To resolve the above issue, this paper proposes DSFSM, a delay-sensitive fractional switch migration approach in multi-controller SDN. DSFSM aims to minimize delay while performing fractional switch migrations among controllers. This delay is considered three-fold: first, the total time taken for a flow request travel from switches to the controller; second, the time taken by the controller to the request (i.e., nearest switch selection and fraction ratio calculation); and third, the time required to respond from the controller to the switch. We formulate DSFSM as an optimization problem using a mixed-integer linear program (MILP). When the network becomes large, or MILP is not tractable, we present a heuristic approach based on MILP. Numerical results demonstrate that the performance of the MILP and the heuristic approach outperforms the conventional scheme that follows the fractional switch migration approach in terms of delay and synchronization costs.**

***Index Terms*—Software-defined networks, delay, multi-controller, fractional switch migration.**

I. Introduction

Software-defined networks (SDNs) have enormous potential for enabling programmability and simplified network management for delay-sensitive applications, such as industrial automation, face recognition, smart devices, and big data streaming [1]. Typically, a multi-controller SDN is considered a well-studied architecture to resolve the challenges of centralized controllers. SDN requires an efficient management approach to enhance performance, and researchers are continuously investigating solutions to address the same [2]. Delay while performing fractional switch migration is an intriguing problem due to the frequent switch migrations among the controllers. Such frequent switch migrations can make the controllers busy processing the migration request, and other requests have to wait to receive a response [3], [4]. More specifically, in Fig. 1, if a controller c_3 is overloaded and has to migrate its load to the underloaded controllers, say c_1 and c_2, a suitable switch (a switch with less hop count) selected from controller c_3 and perform the fraction of switch's flow that gives minimum delay. Otherwise, it leads to longer delays and degrades performance by calling the re-migration process.

Several researchers [5]–[9] have considered switch migration an effective approach for redistributing load from one controller to another, which can be classified into two types, i.e., single mapping (SM) and multiple mapping (MM). In the SM approach, a switch is mapped to a single controller, and all the requests a switch generates are sent to that controller. In [5], Zafar *et al.* introduced a dynamic switch migration-based load balancing approach (DSMLB) to efficiently distribute traffic load by considering the heterogeneous Internet of Things (IoT). The candidate switches are selected using migration efficiency constraints and migrated from the overload controller to the ideal underloaded controller having maximum residual resource utilization. The work in [6] introduced a greedy-based load-balancing approach to minimize the number of switch migrations. The switches are selected for migration based on the condition that the controller's load always falls within the specified range from the average load of the controllers.

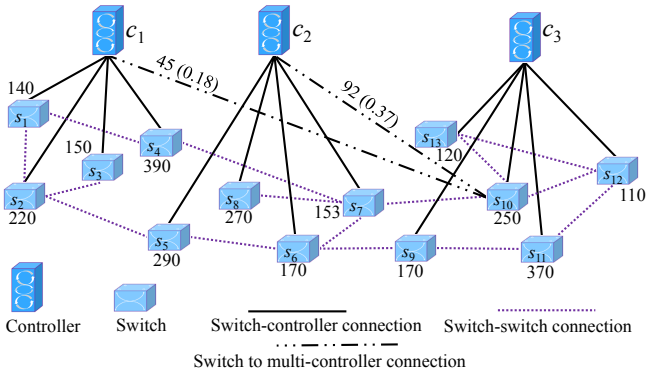

Fig. 1: NSF-NET topology with multi-controllers.

In comparison with SM, the MM approach allows a switch can be mapped to multiple controllers simultaneously, enabling more than one controller to access the switch's flow table. The work in [7] introduced a dynamic switch-controller mapping model for achieving high load balancing by minimizing the number of switch-controller reassignments. This model formulated a convex quadratic programming where various properties and feasibility are analyzed to reduce the control plane overhead. In [8], the authors presented a fractional switch migration-based controller load balancing (FractionalLB). FractionalLB is used to reduce the difference between the controllers' load and their corresponding threshold, and that threshold is used to categorize the overloaded and underloaded controllers. The authors performed a fraction of the switch's flow without considering the hop count to select the nearest suitable switch for migration that directly impacts the delay. In [9], the authors introduced a time-sharing switch migration (TSSM) method where a switch's flow traffic is shared among multiple controllers simultaneously. Two controllers sequentially control a switch's flow using time-sharing by splitting the switch flow table to keep the controllers' load within the given threshold.

Although SM approaches provide simplified network management while balancing the load among the controllers, MM generally provides superior performance and resource utilization for delay-sensitive devices. To enhance the perfor-

This work is supported in part by the Core Research grant (Grant Number: CRG/2020/002663), Govt. of India.

mance and resource management, a fractional switch migration approach is studied in the context of MM, where a switch's flow fractionally migrates to multiple controllers simultaneously [7]. However, inappropriate fractional ratios with a high switch-controller reassignment and a switch with a higher hop may induce heavy loads and make the controller busy updating the load information [10]. Therefore, selecting a switch with an appropriate fractional ratio and lower hop count among multiple connected controllers is essential to minimize delays during fractional switch migration. A question may arise: *How to minimize the delay while performing fractional switch migration among controllers?*

To answer the above question, this paper, for the first time, proposes DSFSM, a delay-sensitive fractional switch migration approach for multi-controller software-defined networks. DSFSM aims to minimize the delay while performing fractional switch migration among controllers. To achieve this, we redistribute the switch's load fractionally from the highly loaded controller to other underloaded controllers while assuring that the processing capacity and hop count restrictions are satisfied. We model DSFSM as an optimization problem using a mixed-integer linear program (MILP). A heuristic approach can be considered when the MILP problem is not tractable for large SDN. We observe that the minimum delay is achieved through MILP compared to the heuristics approach and conventional method.

II. DSFSM: Proposed Approach

The proposed approach, DSFSM, aims to attain the minimum delay while performing the fractional switch migration among the controllers. This can be achieved by properly making the fraction of the switch's flows and migrating it to the multiple underloaded controllers from an overloaded controller. This is performed using a fractional migration approach.

A. Assumptions

We have made the following assumption for DSFSM. (i) We consider $packet\_in$ messages and rule installation as the load-creating factor for each controller, and all the controllers have equal capacity. (ii) The network operator can choose a suitable threshold to determine overloaded and underloaded controllers. (iii) All controllers can communicate with each other to share their load information. (iv) The migration process is atomic, meaning that once it begins, it can not be interrupted, halted, or reversed until the process is finished. (v) All controllers can not be overloaded simultaneously. Finally, (vi) each controller in the network can perform the fractional switch migration.

B. Problem formulation

In a multi-controller SDN, switches may experience delays due to inappropriate fractions of a switch's flow and distant controller selections. The existing works in the literature typically execute fractional migration using heuristic and optimization techniques, as discussed in the introduction, without considering the minimization of the delay while performing the fraction switch migration. The formal problem definition is as follows: given a set of switches S with their packet generation rate, a set of controllers C, the capacity for each controller (pre-determined threshold), hop count between switches and controllers, and the switch controllers relationship. We aim to minimize the delay while performing fractional switch migration among the controllers.

C. Demonstration

DSFSM is demonstrated with an example. For this purpose, we consider a network with three controllers and 13 switches, as shown in Fig. 1. The controllers c_1, c_2, and c_3 are connected with switches $\{s_1,\ s_2,\ s_3,\ s_4\}$, $\{s_5,\ s_6,\ s_7,\ s_8\}$, and $\{s_9,\ s_{10},\ s_{11},\ s_{12},\ s_{13}\}$ that are generating $packet\_in$ message rates of $\{140, 220, 150, 390\}$, $\{290, 170, 153, 270\}$, and $\{170, 250, 370, 110, 120\}$ packets/sec, respectively. As the switches are directly connected with their corresponding controllers, the hop count between switches and controllers is 1. We assume that for rule installation, the additional load added to the corresponding controllers is 0.1 of the total load on the particular controllers. For example, switch s_1 generates 140 packets/sec, and the load induced for rule installation $140 * 0.1\ (= 14)$ is added to the controller c_1. Thus, the total load generated by switch s_1 is 154 packets/sec. Therefore, the total load at the controllers c_1, c_2, and c_3 are 990, 971, and 1122 packets/sec based on the incoming $packet\_in$ messages and rule installation for that $packet\_in$ messages. Also, we assume that the processing capacities of the controllers are 1100, 1150, and 980 packets/sec, and the service rate of all controllers is the same, i.e., 2150 packets/sec.

The controller c_3 has a load greater than its processing capacity; it is considered overloaded. Consequently, it has to migrate the fraction of the switch's flow to the nearest controllers with a lower load, such as the controllers c_1 and c_2. After migration, the delay has to be minimal. Switch s_{10} from controller c_3 is selected, with fractions of 0.18 and 0.36 of switch s_{10} flows migrated to controllers c_1 and c_2, respectively. Migrating the flow of switch s_{10} results in a minimum delay of 105.04 milliseconds (ms). In contrast, migrating flows from other switches results in higher delay due to the higher hop counts between switches and controller c_3. After the migration of switch s_{10} flows, the load of the controllers c_1, c_2, and c_3 become 1039, 1070, and 973 packet/sec, respectively. Thus, with the completion of this process, no further migrations are required, and the controllers have a lesser load than their processing capacity.

III. Optimization Problem

A Multi-controller SDN consists of a set of controllers, defined as C, where $C = [1, |C|]$. The data plane comprises a set of switches, denoted by S, where $S = [1, |S|]$. The given parameters $p_{ij},\ h_{ij},\ \alpha_j,\ K,\ \eta,\ \beta$, and μ are used in DSFSM are explained as follows. The number $packet\_in$ messages sent from a switch $i \in S$ to controller $j \in C$ are denoted by p_{ij}. These messages are generated by a switch $i \in S$ to the connected controllers $j \in C$ only when the table missing entry is found in the OpenFlow table of that switch. The minimum number of hops between switch $i \in S$ and controller $j \in C$ is represented as h_{ij}. The variable h_{ij} helps to determine the lower propagation delay. The predefined threshold of the controller $j \in C$ is denoted as α_j, determining the controllers as overloaded or underloaded. Each switch $i \in S$ can be

connected to a maximum of K controllers in the network. η denotes the signal propagation speed, which is 2 × 108 m/sec. The cost of rule installation for each new flow is represented by β. The service rate of the controllers is represented by μ.

The decision variables x_i^j, b_i^j, and $f_i^{jj'}$ are used to formulate DSFSM are described as follows. A binary variable $x_i^j = 1$ denotes the connection between the switches and controllers, where $x_i^j = 1$ if switch $i \in S$ is associated with controller $j \in C$, and 0 otherwise. A variable $b_i^j \in [0,1]$ represents the traffic distribution weight between controllers $j \in C$ and switches $i \in S$. Note that the traffic distribution weight between controllers $j \in C$ and switches $i \in S$ is summed to exactly one. A variable $f_i^{jj'} \in [0,1]$ denotes the fraction of switch $i \in S$ flow migrated from controller j to j'. When $0 < f_i^{jj'} < 1$, it denotes that a switch $i \in S$ flow is migrated from controller j to j'. Thus, tracing variable $f_i^{jj'}$ and counting the flow migration gives the number of upgrades that happen for balancing load among the controllers, also called synchronization cost. Controller $j \in C$ obtains load due to $packet\_in$ messages (or new flows) received from the connected switch $i \in S$ and the instantiation of the rules for those new flows. Thus, the load on controller $j \in C$ is calculated by using $L_j = \sum_{i \in S} p_{ij} \times x_i^j + \sum_{i \in S} \sum_{j \in C} p_{ij} \times x_i^j \times \beta,\ \forall j \in C$, where β is a given parameter that represents the cost of rule instantiation. The first term indicates that controller $j \in C$ receives load due to $packet\_in$ messages, and the second term is due to the instantiation of the rules for those new flows.

The total delay that occurs while performing the fractional switch migration for distributing load among the controllers contains three different folds. First, the switch-controller propagation delay (γ_{prop}) is estimated based on the shortest path between the switches $i \in S$ and controllers $c \in C$. Where $\gamma_{\text{prop}} = \frac{\sum_{i \in S} h_{ij} \times x_i^j}{\eta},\ \forall j \in C$ and η denotes the signal propagation speed, which is 2×10^8 m/sec [11]. Second, at each controller, modeling the queuing of the $packets\_in$ messages generated by switch $i \in S$ to the connected controller $j \in C$ is M/M/1 queue [12]. Thus, this process is referred to as the controller processing delay and is represented as γ_{proc}. Where $\gamma_{\text{proc}} = \frac{1}{\mu - \sum_{i \in S} p_{ij} \times x_i^j},\ \ \forall j \in C$ and μ represents the service rate of the controller $j \in C$. Last is the synchronization delay, which occurs when the controllers upgrade their load information with another controller. Synchronization delay is represented as γ_{sync}, where $\gamma_{\text{sync}} = \sum_{i \in S} b_i^j \times x_i^j,\ \ \forall j \in C$. Therefore, the total delay is $\gamma_{\text{prop}} + \gamma_{\text{proc}} + \gamma_{\text{sync}}$.

To minimize the delay while performing the fractional switch migration among the controllers is obtained by using the following optimization problem.

$$\min: \ \gamma_{\text{prop}} + \gamma_{\text{proc}} + \gamma_{\text{sync}} \tag{1a}$$

$$s.t. \ \sum_{j \in C} x_i^j \ \leq \ K, \ \ \forall i \in S, \tag{1b}$$

$$b_i^j \ \leq \ x_i^j, \ \ \forall i \in S, j \in C, \tag{1c}$$

$$\sum_{j \in C} b_i^j = 1, \ \ \forall i \in S, \tag{1d}$$

$$f_i^{jj'}(1 - x_i^j) \geq 0, \ \ \forall i \in S, j \in C, \tag{1e}$$

$$\sum_{i \in S} p_{ij}(b_i^j - b_i^j \times x_i^j) \leq \alpha_j, \ \ \forall j \in C, \tag{1f}$$

$$\sum_{i \in S} p_{ij} \times x_i^j + \sum_{j' \in C | j \neq j'} \sum_{i \in S} p_{ij} \times f_i^{j'j} \times x_i^j - \sum_{j' \in C | j \neq j'} \sum_{i \in S} p_{ij} \times f_i^{jj'} \times x_i^j \leq \alpha_j, \ \ \forall j \in C, \tag{1g}$$

$$x_i^j \in \{0,1\}, \ \ \forall i \in S, \ j \in C, \tag{1h}$$

$$b_i^j \in [0,1], \ \ \forall i \in S, \ j \in C, \tag{1i}$$

$$f_i^{jj'} \in [0,1], \ \ \forall i \in S, \ (j,j') \in C. \tag{1j}$$

Equation (1a) minimizes delay, where the first term represents propagation, the second term denotes processing, and the last represents synchronization delays. Equation (1b) assures that a switch is connected to at most K controllers. Equation (1c) indicates that the relationship between traffic distribution weight b_i^j and binary variable x_i^j and ensuring that b_i^j is non-zero when $x_i^j = 1$. Equation (1d) denotes that the traffic from switch $i \in S$ is completely distributed to its connected controllers, and the sum of the traffic distribution weight of switch $i \in S$ to the connected controllers is one. Equation (1e) provides that a fraction of the $packet\_in$ messages from switch $i \in S$ is sent to controller $j \in C$ only if the switch is mapped to that controller. Equation (1f) represents the portion of a switch's load that has to migrate from controller $j \in C$ to another, and the remaining load on that controller $j \in C$ can not exceed the pre-determined threshold of controller $j \in C$. Equation (1g) represents the load at the controller $j \in C$ after completion of the migration process, either receiving or migrating from one controller to another is less than or equal to the pre-determined threshold of the controller $j \in C$. Where the first term in (1g) denotes the experienced load from the connected switches, the second term indicates that the added load at controller $j \in C$ due to the received migrated switch's flow from controller $j' \in C$, and the third term represents reduced (or migrated) load from controller $j \in C$ due to the migration of the switch's flows from controller $j \in C$. Equation (1h) denotes a binary variable. Equation (1i) is a variable that represents the traffic distribution weight of switch $i \in S$ to controller $j \in C$. Equation (1j) denotes a variable representing the fraction of a switch $i \in S$ flow that has to migrate from controller $j \in C$ to $j' \in C$.

The controller processing delay in (1a) i.e., $\gamma_{\text{proc}} = \frac{1}{\mu - \sum_{i \in S} p_{ij} \times x_i^j},\ \ \forall j \in C$ is a non-linear function, as it involves the ratio of binary variable. Thus, we formulate it as a linear expression by introducing the following variable:

$$m_j = \frac{1}{\mu - \sum_{i \in S} p_{ij} \times x_i^j}, \ \ \forall j \in C. \tag{2}$$

Equation (2) can further be rewritten as:

$$m_j \times \mu - m_j \times \sum_{i \in S} p_{ij} \times x_i^j = 1, \ \ \forall j \in C. \tag{3}$$

The second part in (3) becomes non-linear as the product of two variables. Thus, we transform it into a linear form by satisfying (5a) as follows.

$$m_j \times \mu - \sum_{i \in S} p_{ij} \times t_j = 1, \quad \forall j \in C. \tag{4}$$

Similarly, (1e)-(1g) are non-linear as these equations contain products of two variables. Thus, we transform these equations into linear ones with the help of the following variables.

$$t_j = m^j \ \wedge \ x_i^j, \quad \forall j \in C, \tag{5a}$$
$$d = b_i^j \ \wedge \ x_i^j, \quad \forall i \in S, j \in C, \tag{5b}$$
$$e = x_i^j \ \wedge \ f_i^{j'j}, \quad \forall i \in S, \ (j, j') \in C, \tag{5c}$$
$$f = x_i^j \ \wedge \ f_i^{jj'}, \quad \forall i \in S, \ (j, j') \in C. \tag{5d}$$

where $\wedge$ expresses the logical 'AND' operation. For example, let $u \in [0, 1]$, $v \in [0, 1]$, and w is a binary variable. $u = v \wedge w$ is linearized by: $u \leq v$, $u \leq w$, $u \geq v + w - 1$, and $u \geq 0$.

Therefore, (1a) can be expressed by (6).

$$\min : \gamma = \gamma_{\text{prop}} + \text{m}_\text{j} + \text{d}. \tag{6}$$

Equations (1a)-(1j) are transform into following MILP problem.

$$\min : \ \gamma \tag{7a}$$
$$s.t. \ Eqs. \ (1b) - (1d), Eq. \ (4) \tag{7b}$$
$$f_i^{jj'} - d \geq 0, \quad \forall i \in S, j \in C, \tag{7c}$$
$$\sum_{i \in S} p_{ij}(b_i^j - d) \leq \alpha_j, \quad \forall j \in C, \tag{7d}$$
$$\sum_{i \in S} p_{ij} \times x_i^j + \sum_{j' \in C | j \neq j'} \sum_{i \in S} p_{ij} \times e - \sum_{j' \in C | j \neq j'} \sum_{i \in S} p_{ij} \times f \leq \alpha_j, \quad \forall j \in C, \tag{7e}$$
$$Eqs.(5a) - (5d) \tag{7f}$$
$$Eqs.(1h) - (1j) \tag{7g}$$
$$t_j, d, e, f \in [0, 1], \quad \forall j \in C. \tag{7h}$$

IV. Heuristic Approach

When the introduced MILP problem in Section III becomes intractable, a heuristic algorithm is considered to solve the same. The heuristic algorithm starts by estimating the load of each controller based on the *packet_in* messages and rule installation for each messages. It then determines overloaded or underloaded controllers by comparing the evaluated load to a specified threshold, represented by α_j. To redistribute load using fraction switch migration, the selection of the underloaded and overloaded controllers is achieved greedily in an iterative approach employing the introduced MILP formulation (7a)-(7h). Following each repetition, the controller's load is updated, and convergence is obtained when all controllers' loads fall below the pre-determined threshold. By utilizing a greedy approach, the introduced heuristic algorithm minimizes the delay associated with MILP. The detailed steps of the heuristic algorithm are presented in Algorithm 1.

Algorithm 1: Heuristic algorithm

Input : Same as the MILP problem in Section III.
Output: Same as the MILP problem in Section III.
1 Estimate each controller's load $L_j, \forall \ j \in C$.
2 Determine the controllers as overloaded and underloaded based on the predefined threshold $\alpha_j, \forall \ j \in C$.
3 Sort the overloaded controllers in a non-increasing order of their load.
4 **for** *each j in overloaded controllers* **do**
5 Solve the MILP using (7a)-(7h) following all the underloaded controllers and return the delay.
6 Update each controller's load.
7 **end**

V. Performance Evaluation

The performance of DSFSM is estimated in terms of delay and synchronization cost (SC). The delay is the summation of the propagation, processing, and synchronization delays while performing fractional switch migration among the controllers. The synchronization cost is defined as the number of upgrades required to install flow rules in the OpenFlow tables [8]. We assume four networks: the National Science Foundation Network (NSFNET), ARN, FORTHNET, and TataIND, containing 13, 28, 60, and 140 nodes, respectively, with each node representing a switch [13]. We use three, five, seven, and ten controllers for NSFNET. We employ three, five, seven, ten, and fifteen controllers for the ARN topology. Likewise, for FORTHNET and TataIND, three, five, seven, 10, 15, and 20 controllers are employed. Due to space and page limit, we show an ARN topology with 5 controllers as shown in Fig. 2. Each switch randomly generates *packet_in* messages in the range of [50, 500]. Additionally, each controller has a randomly assigned pre-determined threshold in the range of [500, 4500] . A controller triggers fractional switch migration when it exceeds the given threshold. The machine utilized for simulation is designed with the Intel Core i7 3.20GHz processor with 16GB RAM. The optimization problem is solved using PuLP-2.7.0, a Python-based library [14].

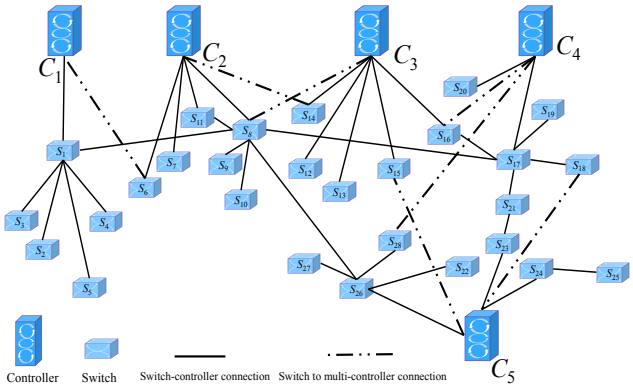

Fig. 2: Topology with 5 controllers used for performance evaluation.

Figure 3 shows delay while performing fractional switch migrations among the controllers. The flow of *packet_in* messages reach to the controller increases as the network size increases, resulting in higher processing times for that controllers due to performing the fractional switch migrations. The delay is comparatively lower when employing the MILP than the other approaches. This is because MILP can execute the optimal fractions of switches for migrations to achieve an

efficient load distribution among the controllers. Additionally, the heuristics approach gives efficient delay compared to those of FractioalLB. FractioalLB is used to distribute *packet_in* messages of a switch among multiple controllers without considering the hop count between switches and controllers. The flow of a switch is migrated to distant controllers, leading to a higher delay (hop count directly impacts the propagation delay) as switch-controllers communication increases. It is noteworthy that the delay positively correlates with the network size.

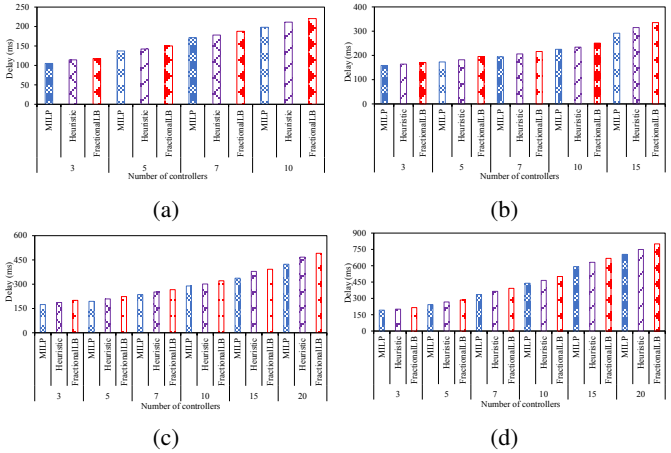

Fig. 3: Delay using different topologies for (a) NSFNET, (b) ARN, (c) FORTHNET, and (d) TataIND.

Figure 4 shows the synchronization costs using different approaches with various numbers of controllers for numerous network topologies. The synchronization costs are lower than those of other approaches when utilizing MILP. This is because MILP can execute the optimal fractions of switches with a minimum hop count for migrations to achieve an efficient load distribution among the controllers. The heuristic approach has effective synchronization costs comparable to FractionalLB. FractionalLB is used to distribute the fraction of *packet_in* messages of a switch among many controllers without considering the hop count. Migration of a distant switch directly impacts controller processing, as a higher hop count requires more flow rule upgrades across the controllers along the path. It is essential to highlight that the synchronization cost increases with the network size. Likewise, the possibility of switch migrations rises as the network grows.

VI. Conclusion

The proposed approach is based on the fractional switch migration, named DSFSM, to redistribute the switch's flow among the controllers in the network. DSFSM minimizes the delay while performing fractional switch migrations among controllers. To achieve this, DSFSM redistributes the fraction of a switch's flow when the controller's load exceeds its predetermined threshold to other controllers while ensuring the hop count and processing capacity constraint are satisfied. An optimization problem is formulated for DSFSM using a mixed-integer linear program (MILP). Numerical results demonstrated that MILP minimizes the delay with lower synchronization costs. However, it needs a higher computation time compared to the heuristic approach. The numerical results

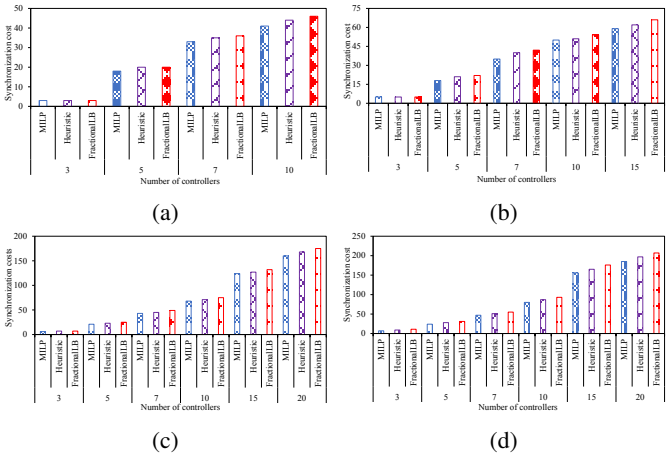

Fig. 4: Synchronization cost using different topologies for (a) NSFNET, (b) ARN, (c) FORTHNET, and (d) TataIND.

indicated that the delay using MILP in NSFNET, ARN, FORTHNET, and TataIND are 20%, 25%, 32%, and 37%, respectively, lesser than that of FractionalLB.

The proposed heuristic approach in Section III, which includes MILP, can be employed in a dynamic network environment regarding frequent changes to bursts of network traffic. The analysis of the fractional switch migration in DSFSM, considering network dynamics, is left as part of future works.

References

[1] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," *Proc. IEEE,* vol. 103, no. 1, pp. 14–76, 2015.

[2] U. Prajapati, B.C. Chatterjee, and A. Banerjee, "GLBSM: Greedy-based load balancing by reducing switch migrations in software-defined networks," in *Proc. IEEE ANTS,* Gandhinagar, India, 2022, pp. 1–6.

[3] M. Priyadarsini, J. C. Mukherjee, P. Bera, S. Kumar, A. Jakaria, and M. A. Rahman, "An adaptive load balancing scheme for software-defined network controllers," *Comput. Netw.*, vol. 164, pp. 106918, 2019.

[4] U. Prajapati, B.C. Chatterjee, and A. Banerjee, "GLBMF: Greedy-Based Load Balancing in SDN by Reducing Switch Migrations and Prioritizing Mice Flow Traffic," *Wireless Pers. Commun.*, pp. 1-24, 2024.

[5] S. Zafar, L. Zefeng, N. H. Zaydi, M. Ibrar, and X. Hu "DSMLB: Dynamic switch-migration based load balancing for software-defined IoT network," *Comput. Netw.,* vol. 214. pp. 109145, 2022.

[6] U. Prajapati, B.C. Chatterjee, and A. Banerjee, "OptiGSM: Greedy-based load balancing with minimum switch migrations in software-defined networks," in *IEEE Trans. on Netw. and Serv. Manage.*, vol. 21, no. 2, pp. 2200-2210, 2023.

[7] F. Al-Tam and N. Correia, "Fractional switch migration in multi-controller software-defined networking," *Comput. Netw.*, vol. 157, pp. 1-10, 2019.

[8] U. Prajapati, B. C. Chatterjee, and A. Banerjee, "FractionalLB: Controller load balancing using fractional switch migration in software-defined networks," *IEEE Netw. Letters*, 2024.

[9] W. K. Lai, Y. C. Wang, Y. C. Chen, and Z.T. Tsai, "TSSM: Time-sharing switch migration to balance loads of distributed SDN controllers," *IEEE Trans. on Netw. and Serv. Manage.*, vol. 19, no. 2 pp. 1585-1597, 2022.

[10] X. Yang, H. Xu, S. Chen, and H. Huang, "Indirect multi-mapping for burstiness management in software defined networks," *IEEE/ACM Trans. Netw.*, vol. 29, no. 5, pp. 2059-2072, 2021.

[11] J. Chen, Y. J. Xiong, X. Qiu, D. He, H. Yin, and C. Xiao, "A cross entropy based approach to minimum propagation delay for controller placement in software defined network," *Elsevier Compt. Commun.*, vol. 191, pp. 133–144, 2022.

[12] T. Das, and M. Gurusamy, "Multi-objective control plane dimensioning in hybrid SDN/legacy networks" *IEEE Trans. on Netw. and Serv. Manage.*, vol. 18, no. 3, pp. 2929-2942,2021 .

[13] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden and M. Roughan, "The internet topology zoo", IEEE J. Sel. Areas in Commun., vol. 29, no. 9, pp. 1765-1775, 2011.

[14] "PuLP" Accessed: Sep-2024.[Online] Available: https://pypi.org/project/PuLP/