An Efficient Switch Migration Scheme for Load Balancing in Software Defined Networking

Original Scientific Paper

Thangaraj Ethilu
Annamalai University,
Department of Computer Science and Engineering, Tamilnadu, India.
ethilthangaraj@yahoo.co.in

AbiramiSathappan
Annamalai University
Department of Computer Science and Engineering, Tamilnadu, India.
reachabisv@gmail.com

Paul Rodrigues
King Khalid University
Department of Computer Engineering, Abha, Saudi Arabia
drpaulprof@gmail.com

Abstract *–Software-defined networking (SDN) provides increased flexibility to network management through distributed SDN control, and it has been a great breakthrough in network innovation. Switch migration is extensively used for workload balancing among distributed controllers. The time-sharing switch migration (TSSM) scheme proposes a strategy in which more than one controller is allowed to share the workload of a switch via time sharing during overloaded conditions, resulting in the mitigation of ping-pong controller difficulty, a reduced number of overload occurrences, and better controller efficiency. However, it has increased migration costs and higher controller resource consumption during the TSSM operation period because it requires more than one controller to perform. Therefore, we have proposed a strategy that optimizes the controller selection during the TSSM period based on flow characteristics through a greedy set coverage algorithm. The improved TSSM scheme provides reduced migration costs and lower controller resource consumption, as well as TSSM benefits. For its feasibility, the implementation of the proposed scheme is accomplished through an open network operating system. The experimental results show that the proposed improved TSSM scheme reduces the migration cost and lowers the controller resource consumption by about 36% and 34%, respectively, as compared with the conventional TSSM scheme.*

Keywords*: Quality of Service. Software-Defined Networks, Load-Balancing, Open Flow*

1. INTRODUCTION

The challenges in network management have tremendously increased due to the rapid deployment of cloud computing, big data applications, the internet of multimedia things, and increased data traffic. The traditional network architecture system combines a data plane and a control plane in each switch, with the former handling packet processing and the latter handling decision making and management. Therefore, updating the latest algorithms and new policies on the switches is very complex and time-consuming because all the switches involved in the given network need to be reconfigured one after another by system administrators or workers [1].

Currently, software-defined networking techniques create a unique view of network management in network applications where the control plane in the switches is shifted to a central unit known as the controller. Therefore, the controller can manage multiple switches in the network. In this modern approach, monitoring, and control of network switches are much simpler as compared with conventional network management techniques because the controller unit can provide such information about the switches. Furthermore, the latest algorithms and new control policies are easily updated to the switches via a set of rules in the controller [2]. Apart from this, SDN can support a wide range of applications, including (i) resisting cyber-attacks; (ii) identifying malicious access points; and (iii) providing anonymous authentication, etc. [3-7].

A lone controller in a large network will be a tough option since it creates a bottleneck in network management; therefore, distributed SDN control (DSC) is demanded in the network applications, and it acts as a promising solution in large network management with the several numbers of switches [8]. The DSC allows multiple controllers to coordinate with each other to manage the entire network. Where each controller is managing a subset of switches (i.e., a subnet), as well as workflow, these can be exchanged among controllers for the use of teamwork. Each controller involves distributing the workload for the subnets and reassigning its switches' workloads through the regular check-up of each subnet, called "controller placement" [9]. The placement of controllers is based mainly on load balancing, and it is applied through several techniques, including the workgroup control technique [10], the deep reinforcement learning technique [11], and so on. The outcome of such control techniques may widely alter the switches in the subnet and lead to instability in the subnet via ping-pong operation. Furthermore, controller placement techniques are not considered effective during short-lasting flows such as distributed denial of service and impulses [12].

Switch migration provides a smooth alteration of subnets with a lesser period and overcomes the above-mentioned issues. In each time frame (or time interval or period), a switch migration method is examining the workload status of each controller in the network to determine whether they are overloaded (busy) or lightly loaded (available to share other works). If it is overloaded, the migration method in a network relocates a switch from the busy controller subnet to a lightly loaded controller subnet. Most of the existing switch migration methods follow the smallest slice of the migration: one single switch, which is migrated at the beginning of the period. Once the switch is migrated, it remains in the latest subnet until the switch is selected for the next period. Most importantly, these migration methods always ask a controller to oversee one switch for a complete period. Therefore, the controller in these methods gets into the ping-pong difficulty of an "elephant flow situation (i.e., flow carries many packets) and goes into the serious trouble of a subnet that is unstable [13].

2. LITERATURE REVIEW

Over the years, several studies have detailed the various issues in the DSC network. Conventionally, controller load balancing is achieved through dynamic controller placement methods. Chan et al. [14] proposed a method that could minimize the service interruption time by smoothly transferring the workflow from the compromised controller to another controller. The leader controller redundancy is detailed in [15], where a lightly loaded controller can act as a leader in case of failure in the regular leader controller unit. Controller placement methods and challenges are reviewed in [9]. It has insisted that the controllers maintain fairness during the sharing of their workloads. Ref [16] proposed a reliable deployment method because of reducing packet loss and improving network stability, and it has achieved its objectives compared to other controller placement methods. Kim et al. [17] have proposed a method that improves the output of dispersed data stores in an Open Daylight controller cluster by consistently spreading the shared leaders to the cluster members. Ref [18] proposed a method in which controllers collaborate to reroute traffic to avoid congestion during a switch's busy or overloaded period. A software-defined cyber-seek framework is proposed in [19], where a hybrid controller is used for cloudlets and local networks. Prediction-based controllers are proposed in [20], and they predict the network load and perform the device transfer based on the prediction. The controller placement studies like the workgroup control technique and the deep reinforcement learning technique proposed in [10] and [11], respectively, show that these techniques are not effective during impulses, distributed denial of service, etc. Apart from the dynamic controller placement approach, methods for workload balancing for DSC are grouped into three categories: (i) switch migration, (ii) flow migration, and (iii) flow splitting.

Switch Migration: switch control can be transferred from overloaded controllers to lightly loaded controllers, considering workload reduction. The study [21] has discussed the switch migration because of CPU and memory allocation exceeding a controller's threshold level, but it does not define the way of choosing the targeted controllers. Switch migration using the Q-learning technique is discussed in [22], and it has reduced the standard deviation of the controller's workload. Cui et al. [23] have used the response time of the controller for switch migration. By using this technique, the switch is transferred with the largest load on the controller and the quickest reaction time. Ref. [24] proposed a method that targeted controller selection for switch migration based on CPU utilization, memory size, bandwidth, etc. Hu et al. [25] proposed a simulated annealing algorithm for selecting the targeted controller to reduce the switch migration cost.

Flow Migration: The flow migration method only transfers the hardness (i.e., flow beyond the threshold level) of the flow instead of migrating a whole switch. Hu et al. proposed a technique in which a "super controller administers every controller in the system and regulates the flow managed by them [26]. Ref. [27] proposed a game theory approach that managed the flow of each controller through workload exchange between them. Maity et al. [28] proposed a traffic-aware consistent approach for reducing the flow migration duration, and they achieved about a 15% reduction in flow migration time when compared with the conventional flow migration methods. Also, with the use of a traffic-aware flow migration approach, ref [29] has proposed a method to reduce the data plane load and achieved a 13% reduction when compared with the two-phase update approach.

Flow Splitting: This method allows a switch to be managed by more than one controller at the same time. Gorkemli et al. [30] discussed a method in which switches are required to negotiate with their controllers for flow splitting using a virtual overlay on the data plane. Ref. [31] proposed an approach based on convex quadratic programming for load balancing as well as reducing new switch-controller appointments through modeling the mapping between controllers and switches.

The control relation graph-based controller placement method for software-defined networking (SDN) is presented in [32]. It demonstrates that the proposed approach reduces management costs through load balancing and response time in LEO satellite networks. Zhang et al. proposed an SDN-based space-terrestrial integrated network architecture. In addition, it has presented an efficient dynamic controller placement and adjustment algorithm for better load balancing and response time [33]. Chen et al. proposed a dynamical control domain division problem to reduce the management cost. In addition, it has presented a heuristic algorithm to choose the best controller for better load balancing [34].

However, considering the practical viability of Open Flow, a switch cannot be controlled by more than one controller simultaneously considering synchronization and complex design. Therefore, flow migration and flow splitting methods are non-compliant to the OpenFlow protocol and cannot be implemented in the real-time controller platform.

2.1. PROBLEM DESCRIPTION AND CONTRIBUTION

As discussed in the literature section, most of the switch migration methods are having issues with ping-pong difficulty. The ping-pong difficulty of the controller is explained in the following example. Let us consider two controllers [C_p and C_q] and three switches [S_a, S_b, S_c] in the network. The maximum manageable workload for each controller is 100 PIMS per second. The switches S_a, $S_{b,}$ and S_c produce 60, 80, and 60 PIMS per period, respectively. In time t, C_p handles switches S_a & S_b then controller C_q manages to switch S_c. Since $\alpha_{c\_p} = \delta_a(t) + \delta_b(t) = 60 + 80 > \beta_{c\_p}$(100 PIMS), C_p is overloaded and requires switch migration. In most of the switch migration methods, an overloaded controller will request and takeover a switch for a whole period from other controllers. Therefore, Switch S_a is transferred to controller C_q`s subnet at time t+1. Though at period t+1, $\alpha_{c\_q} = \delta_c(t) + \delta_a(t) = 60 + 60 > \beta_{c\_q}$ (100 PIMS), controller C_q will be overloaded. So, controller C_q *a*sks C_p to take over a switch again in time t+2, which makes ping-pong difficult.

Recently, W.K. Lai et al. [35] proposed a time-sharing switch migration scheme (TSSM) that mitigates the ping-pong difficulties in the controllers by sharing the workload of a switch that is supervised by two controllers at the same time during overloaded conditions. It proposes a strategy whereby switch migration is performed in a time-sharing manner, where the workload of the switch is divided between two controllers within a given period. Considering the previous example, at time t+1, controller C_p manages 20 PIMs of S_a, and the remaining 40 PIMs are handled by C_q through migration. During this time, both controllers C_p and C_q are managing the workload of switch S_a. Hence, C_p's workload becomes $\alpha_{c\_p} = \delta_a(t) + \delta_b(t) = 20 + 80 \leq \beta_{c\_p}$(100 PIMS) and, C_q's workload turns out to be $\alpha_{c\_q} = \delta_c(t) + \delta_a(t) = 60 + 40 \leq \beta_{c\_q}$(100 PIMS). Therefore, both controllers are not overloaded (busy) in period t+1. Similarly, at ti*me* t+2, C_q initially processed 40 PIMS, and the remaining 20 PIMS have been sent to the C_p controller subnet. In this approach, The TSSM scheme can successfully conquer the ping-pong difficulty of the controller.

Specifically, it proposes a strategy where two controllers, namely an overload controller (one) and a lightly loaded controller (it can be many, but this paper utilizes one), are combined, and the switch from an overloaded to a lightly loaded controller subnet is made at an adequate point in time. The outcome of this technique shows that it has considerably reduced overload occurrences of the controllers and effectively balanced the workload of all the controllers with improved controller efficiency as compared with the existing switch migration methods such as group-based dynamic controller placement [10], churn-triggered migration [30], and the "best-fit migration [32] method. Nevertheless, it is observed that more than one lightly loaded controller operation in the TSSM provides better controller efficacy than the original one (i.e., discussed in the paper) with the increased switch migration cost. In addition, this method has higher controller resource consumption during TSSM operation since the migration switch is managed (i.e., controlled) by more than one controller in the network.

Therefore, we proposed a strategy that optimizes the selection of lightly loaded controllers during the TSSM period and allows more than one lightly loaded controller for switch migration during the TSSM period without increasing migration costs. The controller is selected based on flow characteristics through a greedy set coverage algorithm, which reduces the controller's resource consumption by reducing the number of controllers participating in the flow processing. The improved TSSM scheme provides reduced migration costs and lower controller resource consumption, as well as TSSM benefits. The implementation of the proposed scheme is accomplished through an open network operating system (ONOS) for its feasibility, and it can respond to about one million flow processing requests per second.

In summary, software-defined networking (SDN) leads to an efficient administration process in network management through easy updating of network poli-

cies and the latest algorithms. Typically, distributed SDN is adopted in network management, considering bottleneck issues. Load balancing is a critical factor in the SDN, and it can be managed through (i) the dynamic controller placement method, (ii) switch migration, (iii) the flow splitting method, and (iv) the flow migration method. Considering the practical viability of Open Flow, a switch cannot be controlled by more than one controller simultaneously, considering synchronization and complex design. Therefore, flow migration and flow splitting methods are non-compliant with the OpenFlow protocol and cannot be implemented on the real-time controller platform. Considering the OpenFlow protocol and its implementation in the real-time controller platform, the dynamic controller placement method with switch migration is a better solution for load balancing.

The conventional switch migration methods suffer from ping-pong difficulty during the switch migration process because the whole single switch is migrated in the beginning period. It causes instability issues in the switch migration. The ping-pong difficulty is rectified by a time-sharing switch migration scheme. This method significantly reduces the overload occurrences of the controller, which leads to better load balancing. However, the selection of controllers during the TSSM period is random. So that it could increase the switch migration cost and higher controller resource consumption during TSSM operation since the migration switch is managed by more than one controller in the network. Therefore, our paper has proposed an improved TSSM scheme, and it has the following merits: (i) It contains all the merits of a conventional TSSM scheme, including the removal of ping-pong controller action during the switch migration process, a reduction in controller overload occurrences, and better controller efficiency. (ii) The selection of controllers during TSSM is specified and optimized through the greedy set algorithm, which reduces the switch migration cost and controller resource consumption. (iii) It provides better controller efficiency and load balancing compared with the conventional TSSM scheme. The structure of the paper is shown in Fig. 1.

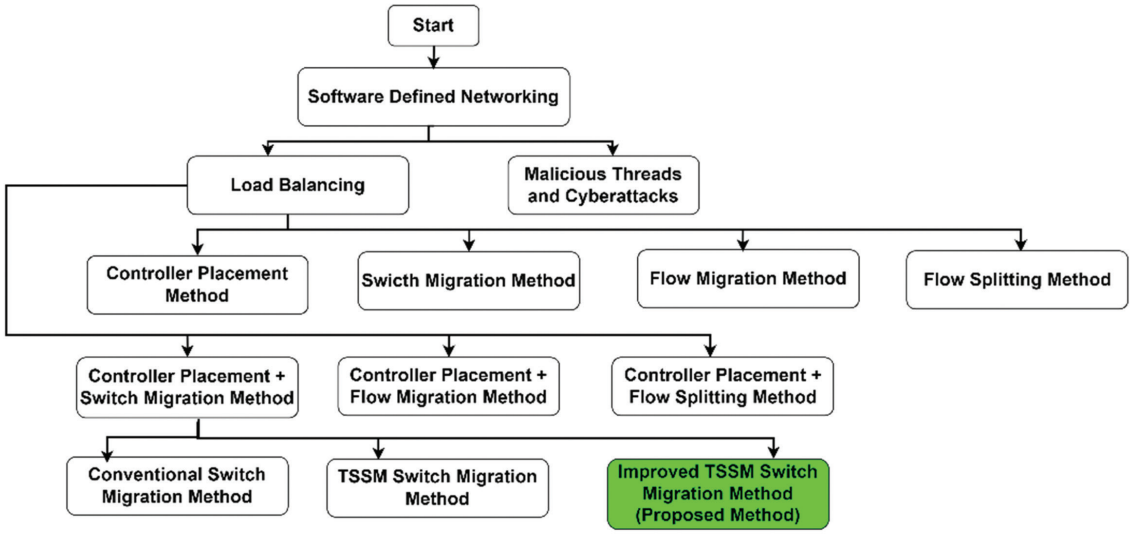


Fig. 1. Structure of the paper

2.2. ORGANIZATION OF THE PAPER

The paper is structured as follows: The literature review and problem description are covered in Section II of this paper. The background knowledge of the distributed SDN control network, OpenFlow protocol rules, and network model is detailed in Section III. The proposed improved TSSM scheme and matching algorithms are discussed in Section IV, and the performance evaluation of the proposed method is presented in Section V. Finally, the concluding statement is summarized in Section VI.

3. DISTRIBUTED SDN CONTROLLER

The architecture of the distributed SDN control network, the switch transfer procedure in the OpenFlow protocol, and network models are discussed in this section.

3.1. DISTRIBUTED SDN CONTROL NETWORK ARCHITECTURE

Two common control methods are typically followed in the distributed SDN control network, namely, (i) the hierarchical method and (ii) the flat control method, also called circular chain control [8]. In the hierarchical method, the central distributed controller (called the leader) has the idea of a network global view and is updating the network policies and latest algorithms to the sub-controllers, as shown in Fig. 2(a). The sub-controller takes control (is in charge) of the subnet of its switches, as well as reports its status to the leader. It is noted that the new leader will be selected if the original leader is broken down in the hierarchical method [15]. In the case of circular chain control, controllers have information about the local view of the network

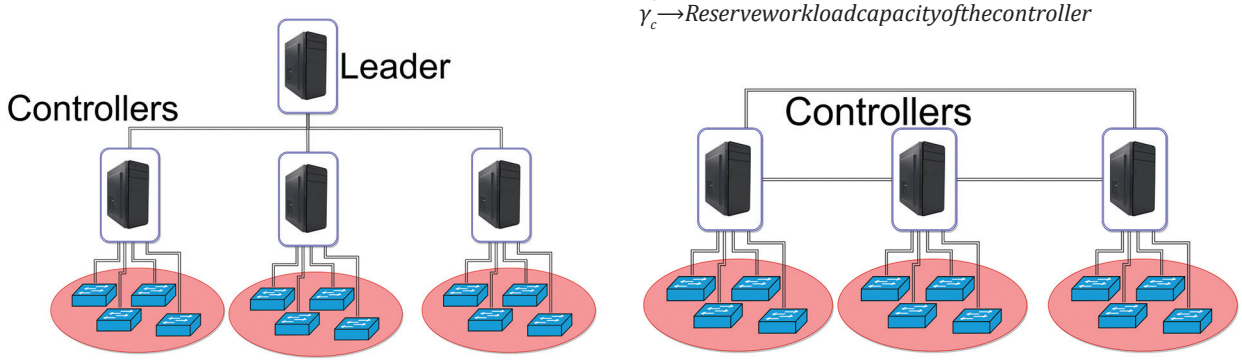
and authority over their own subnet. The involved controllers are swapping information among themselves in a distributed manner, as shown in Fig. 2(b).

The hierarchical method is considered in this paper to apply the proposed switch migration scheme. The leader is responsible for monitoring the status of each sub-controller as well as performing the TSSM scheme to select the lightly loaded controller for the overloaded controller during flow fluctuations, flow traffic, impulses, distributed denial of service, and so on. Afterward, two sub-controllers (overloaded and lightly loaded) are committed to sharing their workloads and migrating the switch where it is required.

Specifically, the threshold level of the sub controller is also defined in the leader to avoid unwanted switch migrations. When the workload of the controller is more than the threshold level, it is considered overloaded, and it is selected based on the maximum capacity and reserve capacity of the controller. Generally, the threshold level is selected between 90 and 95% of the maximum capacity, as recommended by network administrators. The threshold level of the controller is also noted as the maximum workload of the controller, and it is defined in eq. (1).

$$\varphi_c=\beta_c-\gamma_c \tag{1}$$

$\varphi_c \longrightarrow Thersholdworkloadlevelofthecontroller$
$\beta_c \longrightarrow Maximumworkloadcapacityofthecontroller$
$\gamma_c \longrightarrow Reserveworkloadcapacityofthecontroller$



(a) (b)

Fig. 2. Control methods for the DSC architecture: (a) Hierarchical method, (b) Flat method

3.2. TRANSFERRING PROCESS FOR SWITCHES IN OPENFLOW PROTOCOL

OpenFlow permits a switch transfer among various subnets and creates a connection with several controllers. Based on switch S_n's point of view.

The following roles are determined by each associated controller C_p.

- OFPCR_ROLE_EQUAL (Equal): This default role makes controller C_p to have full authority to switch S_n, and C_p can send commands to S_n and receive the status. Similarly, all the controllers have full access to S_n when it is acting in this role.
- OFPCR_ROLE_SLAVE (Slave): If the controller C_p role is changed to slave, then C_p can only read the status from switch S_n.
- OFPCR_ROLE_MASTER (Master): It is like as equal role and controller C_p has complete authority to S_n. Though, it is insisted that only one controller (e.g., C_p) is considered as a master controller for a switch Sn and other controllers are regarded as slaves to switch S_n.

Transferring process for the switches is defined in the OpenFlow protocol is shown in Fig. 3. Switch transferring process is initiated by the master controller since it has full authority over the switch. For example, controller's C_p and C_q are the master and targeted (slave) controllers respectively, for the switch S_n. It is insisted that overloaded controllers are transferring a switch to other controllers for workload balancing with the help of the leader (controller). Once the master controller (C_p) gets a command from the leader, it will then send a transfer request to switch S_n to targeted controller C_q.

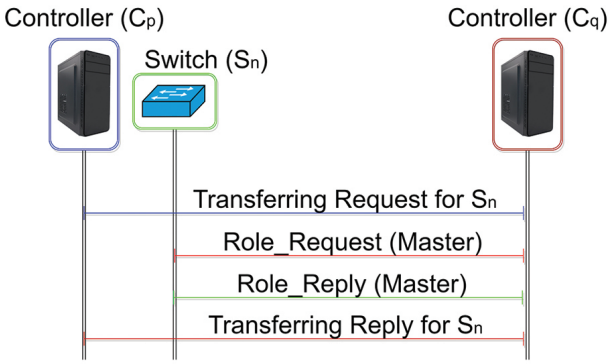


Fig. 3. Switch transferring process in OpenFlow Protocol

After that, controller C_q asks switch Sn to change the role of S_n control to master instead of slave through the Role_Request (Master) message, and switch S_n will provide a confirmation message to C_q via Role_Reply (Master). After all, C_q provides notification message to C_p for the victorious migration of switch S_n and then controller C_p acts as a slave controller for switch S_n.

The switch migration is supported by the OpenFlow protocol in versions 1.2, 1.3, 1.4, and 1.5 (most recent version). It is observed that OpenFlow regulation only instructs about how to alter (migrate) the switches between controllers for their roles and exchange messages between controllers. However, deciding target controllers and switches for migration is not defined by OpenFlow. The proposed improved TSSM scheme achieves optimized controller selection and determines when to execute switch migration during the TSSM period.

3.3. NETWORK MODEL

Let us predict an SDN-based network comprised of a collection S_n of switches and a group C_n of controllers. A switch (e.g., S_a) in S_n is controllable by a controller in C_n (e.g., C_p) with a model of one switch is controlled by a controller simultaneously recommended by OpenFlow, i.e., Cp is acting as a master controller for Sa, and it can be changed after the switch migration.

The workload of each controller is determined through Packet_In messages (PIMs) sent from the switches. Particularly, switches workload (δ(t)) are determined through the number of PIMs generated by a switch in each period ´*t*´. Subsequently, controller workload capacity is defined as the maximum amount of PIMS that can be handled in each period. For example, if switches S_a to S_z are administered by controller C_p then the workload of the controller C_p is calculated as,

$$WL_{pt} = \alpha_{c_p} = \sum_{S_a}^{S_z} \delta(t) \quad (2)$$

Generally, the maximum workload (αc) of the controller shall be less than the maximum capacity of the controller (βc) considering the requirement of reserve load during unwanted situations such as flow fluctuation, abrupt demand, etc. In this paper, hierarchical control of DSC architecture is considered; therefore, the leader collects workload from all the controllers at every period and directs the switch migration between controllers when required.

4. PROPOSED SWITCH MIGRATION SCHEME

During the initial stage, controller placement methods or network operators are used to configure the network switches, where each switch is controlled by a master controller. As discussed in the previous section, conventional switch migration methods include migrating a switch at the beginning of the period as well as a complete part of a switch even though it is not required. Thus, connections between controllers and switches are static for the whole period. In the case of TSSM, switch migration is allowed through time-sharing, and switches in the network can dynamically alter their connections with the controller in each period. In addition, the TSSM scheme effectively overcomes the controller ping-pong difficulty, as discussed in Section 2.1. Nevertheless, controller resource consumption is higher during the TSSM period, which could increase the migration cost of the method compared to other migration methods since it allows more than one controller to share their switch loads during the TSSM period. It is observed that migration costs are estimated based on the utilization of controllers and switches. Therefore, this paper has proposed an algorithm that significantly reduces the number of controllers associated with the switches based on flow characteristics during time-sharing migration. We have introduced a greedy set coverage algorithm to achieve the optimal association between the switches and controllers during the time-sharing migration period, such that the number of controllers associated with the switch is reduced, which subsequently reduces controller resource consumption and lowers the migration cost. The following algorithms are designed for the successful

Algorithm 1: Locating Overloaded and Lightly Loaded Controllers

1. $C_{over} \leftarrow$ ø and $C_{light} \leftarrow$ ø ;
2. **foreach** $C_p \in C$ **do**
3. $\alpha_{c\_p} \leftarrow 0$;
4. **foreach** $S_a \in S_p$ *do*
5. $\alpha_{c\_p} \leftarrow \alpha_{c\_p} + \delta_{a,t}^{(p)}$;
6. if $\alpha_{c\_p} > \varphi_{c\_p}$ then
7. $C_{over} \leftarrow C_{over}$ U {Cp} ;
8. else if $\alpha_{c\_p} < \lambda \times \varphi_{c\_p}$ **then**
9. $C_{light} \leftarrow C_{light}$ U {Cp} ;
10. If $C_{over} \neq$ ø and $C_{light} \neq$ ø then
11. Use Algorithm 2 for load balancing between C_{over} and C_{light} ;

Algorithm 1: Locating Overloaded and Lightly Loaded Controllers

This algorithm is ensured to find all the overloaded (called busy) and lightly loaded controllers (called assistant or target controllers) in the given network, symbolized by *Cover* and *Clight*, respectively. The workload of each controller (e.g., α_{c_p}) is estimated based on Eq. (2) through adding the loads of each switch (e.g., $\delta_{a,t}^{(p)} + \delta_{b,t}^{(p)} + \cdots$) in the subnet, it is described in the algorithm code between 3 and 5 lines. Afterward, controller workload (e.g., α_{c_p}) is compared with the threshold level (φ_{c_p}) and if it is more than the threshold level then it is considered as an overloaded controller and included in the overload controllers (characterized in lines 6 -7) unit in the leader. Then lightly loaded controllers are determined based on a lightly loaded coefficient 'λ', value between 0.9 and 0.95 (selected by network administrators) and it is included in line 8. Afterwards, lightly loaded coefficient is multiplied with the threshold value, and if the workload of the controllers is less than the multiply value, then it is considered a lightly

loaded controller, and it is added to the lightly loaded controller unit in the leader. It is insisted that switch migration be carried out when both C_{over} and C_{light} controllers are non-empty, checked in line 10.

Lemma 1: Let assume overloaded and lightly controllers are subset of main controller $\ddot{C}$ (i.e., $\xi Cover \& \xi Clight \in \ddot{C}$) and all the switches are included within this domain is represented as $\ddot{S}$ (i.e, $\xi S \in \ddot{S}$), then complexity of the time computation for algorithm 1 is estimated as O ($\xi C_{over} + \xi C_{light} + \xi S$) + T_2, T_2 is the computation time of algorithm 2.

Proof: In algorithm 1, Line 1 requires a constant amount of time to initialize both C_{over} and C_{light}. Then, in lines 2-9, the outer for-loop has iterations similar to the number of controllers positioned in this domain, but lines 3, 6, 7, 8, and 9 all need O (1) time.

Lines 4-5's inner for-loop (together with the outer for-loop) examines every switch in overloaded controller S_p. As a result, the outer for-loop takes ($\xi C_{over} + \xi C_{light}$)× O (1) + O ($\xi S$) = O ($\xi C_{over} + \xi C_{light} + \xi S$). Line 11 then performs Algorithm 2 and uses T_2 time. To summarize, Algo. time complexity is O ($\xi C_{over} + \xi C_{light} + \xi S$) + T_2.

Algorithm 2: Switch Migration Segment for Load Balancing

1. SORT (C_{over}, α_{c_p} - φ_{c_p});
2. SORT (C_{light}, φ_{c_q} - α_{c_q});
3. **foreach** $C_p \in C_{over}$ do
4. SORT (Sp , $\delta_{a,t}^{(p)}$);
5. **while** $\alpha_{c\_p} > \varphi_{c\_p}$ do
6. **if** $C_{light} =$ ø then
7. Cease this module ;
8. Pick the optimized controllers C_{q_1}, C_{q_2},… from C_{light} ;
9. (Controller-Switch Association Matrix) ← Algorithm 3 (Request PIM´s of Switch, Switches from Cover)
10. (S_a, [τ_1 , τ_2 , …] , [n_1 , n_2 , …]) ← Algorithm 4 (C_p , [C_{q1}, C_{q2}, …]) ;
11. Transfer S_a to [C_{q1}, C_{q2}, …]'s subnet after [τ_1 , τ_2 , …] units of time ;
12. $\alpha_{c\_p} \leftarrow \alpha_{c\_p}$ - [n_1, n_2, …];
13. $\alpha_{c\_q1} \leftarrow \alpha_{c\_q}$ + [n_1, n_2, …] ;
    $\alpha_{c\_q2} \leftarrow \alpha_{c\_q}$ + [n_1, n_2, …] ;
14. **if** $\alpha_{c_{q[1,2,\ldots]}} \geq \lambda \times \varphi_{c_{q[1,2,\ldots]}}$ **then**
15. $C_{light} \leftarrow C_{light} \setminus \{C_q [1,2, \ldots]\}$;
16. **else**
17. SORT ($\boldsymbol{C_{light}}$, φ_{c_q} - α_{c_q}) ;

Algorithm 2: Ordering the pair of overloaded and assistant controllers and switch migration.

The aim of this algorithm is to share the workload between controllers by locating the pair of overloaded and lightly loaded controllers. The SORT function helps sort the overloaded and lightly loaded controllers in decreasing workload order. The overload controllers are sorted in code line 1, whereas line 2 provides the sorted information about the lightly loaded controller. Hence, a controller with extremely leftover capacity will be considered the first to contribute to the workload of an overloaded (busy) controller. The code in lines between 3 and 17 handles each controller in the network through for-loop by most overloaded controller to the lowest overloaded one. Line 4 sorts of the switches under C_p administration based on their workload in conjunction with decreasing order. The while loop in line 5-16 keep on decreasing the workload of the C_p by migrating a switch until it gets below threshold workload. However, if there is no assistant controller to help (i.e., C_{light} is empty), and more overload controllers are still in the domain then algorithm 2 terminates as given in line 6 -7. Otherwise, if we want to select again a lightly loaded controller C_q for sharing workload then time sharing switch migration scheme is to be activated. For that initially, Algorithm 3 is executed to find the optimum controllers [C_{q1} , C_{q2} , …] for TSSM in view of reduced controller resources consumption and lower migration cost. Afterward, once the optimized controllers are discovered then TSSM scheme is executed based on Algorithm. 4. The output of Algorithm. 4 provides three output parameters as noticed in line 10. In which, ´τ´ gives the information about what time switch S_a should migrate to other controllers, whereas ´n´ provides the information of how much of PIMs to be migrated to each controller. Afterward, workload updates of C_p and [C_{q1} , C_{q2}, …] is performed in line 11 to 13 and if [C_{q1}, C_{q2}, …] is exceeded the threshold level then these controllers are removed from the lightly loaded controllers as given in line 14, otherwise these controllers are again going for the sorted function in the lightly loaded controller unit as given in line 17 and line 2.

Lemma 2: This property proves that algorithm 2 must be converge and it does not run forever due to the finite number of overloaded controllers. Let consider sum of lightly loaded controllers, and switches are represented as $|C_{light}| = \xi_{light}$, $|S| = \xi_S$, respectively. In the worst scenario, algorithm 2 takes ξ_S ($T_3 + T_4$ + O (ξ_{light} + $\log_2 \xi_S$)) time, where T_3 and T_4 is the calculation time of Algorithm 3 and 4.

Proof: Lines 1 and 2 of algorithm 2 take time required for the sorting of overload O ($\xi_{over} + log_2\, \xi_{over}$) and lightly loaded O ($\xi_{light} + log_2\, \xi_{light}$) controllers. We choose an overloaded controller C_p(i.e., line 3), an assistance controller C_q (i.e., line 8), and shift the load of a switch Sp from C_p to C_q (i.e., lines 9-12) in the for-loop. Except for lines 4, 9, 10, and 16, each of the residual statements

inside the for-loop takes O (1) time. Then, line 4 takes the time to sort the switch Sp and it is estimated as O ($|S_p|$ $log2$ $|S_p|$). Line 9 detects a switch and the time (T_3) for finding the lightly loaded controller optimization by algorithm 3 and line 10 takes the time (T_4) required for the switch migration by algorithm 4. Therefore, considering all the time taken by each line then the total time complexity of the algorithm 2 is estimated as, ξ_S (T_3+T_3+O(ξ_{light}+log_2 ξ_S)).

Algorithm 3: Selection of Optimised Controller for TSSM Scheme

1. **Initialization**: controller-switch association{};
   set switches ={};
2. SORT (S_p, $\delta_{a,t}^{(p)}$; SORT (S_q, $\delta_{a,t}^{(q)}$);
3. **end-to-end traffic distribution**:
   Flow_pair = Flow_sort (flow)
4. **while** C_p in the Flow_pair:
   Traversing traffic on the network
5. Path_swicth = Dijkstra (Network Topology, C_p);
   Calculate the flow path
6. **while** Constantly Traversing Controller and Switch Path Sets
7. S is a set, {S_a, S_b, … S_z} is a subset of S, and $US_a = S$
8. **if** $S_a = S$, S_a is selected as the optimal coverage set of S.
9. **if** an element x satisfies $x \in S$, and $x \in S_a$, then S_a is the part of the optimal coverage set of S.
10. **if** $S_a \subset S_b$ exists, S_a is removed from { S_a, S_b, … S_z}
    Let S_n (x) denote a set from{S_a, S_b, … S_z} satisfying $x \in$ Sa, and
11. **if** S_n (x) $\subset$ S_n (y), element y is removed from S.
12. Perform the Corresponding Action
13. **else**:
14. Implement a Greedy Strategy to Select the Controller that Covers the Most Switches
15. **end if**
16. **end while**
17. **end while**
18. This Algorithm ceases until all switches are associated.
19. Use Algorithm 4 for TSSM scheme;

Algorithm 3: Selection of optimized controller for TSSM scheme in view of reduced migration cost

The objective of this algorithm is to provide optimized controllers for the lightly loaded controllers in the TSSM operation. The optimized controller is selected based on flow characteristics such that it reduces the controller's resource consumption and, subsequently, the switch migration cost. The greedy set coverage algorithm [36] is utilized for the optimized controller selection and is presented in Algorithm 3. This algorithm requires the PIMs of each switch in the overloaded controller C_p, as well as the controller's threshold level, network topology map, and so on. The Flow_sort function in line between 3 and 12 estimates the total amount of flow in each path and sorts it in descending order. In lines 4-6, execute and select a controller that covers most of the switches in the path. The sorted path set will continue to be covered until all switches are associated. Subsequently, the optimized controllers are selected, and then Algorithm 4 is executed for the TSSM strategy.

Lemma 3: Let assume all the lightly loaded controllers are sorted in the lightly loaded controller domain, it is represented as $|C_{light}|$=ξ_{light}. Let $C_{q1} > C_{q2}$, then in any optimal solution exists based on optimal flow that $X_1 <$ 1, or $X_2 = 0$. then complexity of the time computation for algorithm 3 is estimated as O ((C_{light})log^2(C_{light})).

Proof: In algorithm 1, Line 1 requires a constant amount of time to initialize controller-switch association. Line 2 takes the time required to sort the overload and lightly loaded controllers, represented as O (ξ_{over} + log^2 ξ_{over}), and O (ξ_{light} + log^2 ξ_{light}) respectively. Line 5 requires a time to estimate the flow path between switches and lightly loaded controllers, represented as O (log C_{light}). Then the total complexity of the time is computed as O ((C_{light})log^2(C_{light})).

Algorithm 4: Time to Switch Migration Estimating Segment

1. $\Delta_{over} \leftarrow min\ (\alpha_{c\_p} - \varphi_{c\_p})$ & $\Delta_{light} \leftarrow$ $max\ (\varphi_{c\_q} - \alpha_{c\_q})$;
2. $S_p^{\mu} \leftarrow$ ø and $S_p^{v} \leftarrow$ ø;
3. **foreach** $S_a \in S_p$ **do**
4. **if** $\delta_{a,t}^{(p)} \geq \Delta_{over}$ **then**
5. $S_p^{\mu} \leftarrow S_p^{\mu} \cup \{S_a\}$;
6. **else**
7. $S_p^{v} \leftarrow S_p^{v} \cup \{S_a\}$;
8. **if** $S_p^{\mu} \neq$ ø **then**
9. $S_a \leftarrow$ the last switch of S_p^{μ};
10. τ=[% of Δ_{light} *with respect to* Δ_{over}]×(L_t) ;
11. **else**

12. $S_a \leftarrow$the first switch of S_p^{ν};

13. $\tau \leftarrow 0$ then $n \leftarrow \delta_{a,t}^{(p)}$;

14. $\delta_{a,t}^{(p)} \leftarrow \delta_{a,t}^{(p)} - n$ and $\delta_{a,t}^{(p)} \leftarrow$ n ;

15. **return** (S_a, τ, n);

Algorithm 4: Time to Switch Migration Estimating Segment

After the optimum lightly loaded controllers (C_q [1,2, …]) are defined from Algorithm 3, they are combined with overloaded controllers to perform three tasks via the execution of Algorithm 4. The responsibilities are, (i) select a switch (from an overloaded controller) for sharing their workload with lightly loaded controllers, (ii) determine the switch migration time (τ), and (iii) calculate the number of PIMS (n) that lightly loaded controllers will process. Based on Algorithm 4, line 1, execute and consider 'Δ_{light}' be the remaining capacity of the lightly loaded controllers, and 'Δ_{over}' is believed to be the minimum amount of overload in the overloaded controllers. After that, switches in the overloaded controllers are split into two subnets namely S_p^{μ} and S_p^{ν}, respectively, where switch load is more than 'Δ' then it is sorted in S_p^{μ} with decreasing load order and S_p^{ν} includes remaining switches in the overloaded controllers, respecting codes are given in line 2-7. At first, switches near to 'Δ' (might be the very last switch in Spμ based on load soring order) is selected in the S_p^{μ} subnet for migration in view of reducing number of migrations (executed in line 8 - 9), because minimum amount of overload in the overloaded controllers can be easily getting placed in the lightly loaded controllers. The estimation of switch migration time depends on amount of PIM's generation in the switch, Δ_{light} in the optimum lightly loaded controllers, Δ_{over} in the switch. For example, if Δ_{light} is half of the Δ_{over} value and PIMs produced rate is constant then switch migration time is estimated as half of the period length as given in eq. (3). If τ=0, then switch migration occurs at beginning of the period as executed in line 13. Furthermore, once the switches in S_p^{μ} subnet is empty then S_p^{ν} subnet is considered for the better load balancing even though it is not overloaded, it is executed in line 11 - 12. This process will be repeated until all the controllers are load balanced via optimal controller finding (Algorithm 3) for each switch in the time-sharing scheme and then finally back to Algorithm 2.

Lemma 4: In the worst scenario, given ξS switches in S, Algo. 3 takes O(ξS) time.

Proof: The first two lines of Algo. 3 require a consistent amount of time to initialize. Because $S_p \subseteq S$, the for-loop in lines 3-7 repeats at most ξS times, and each statement takes O (1) time. Each statement in lines 8-15 obviously requires O (1) time. To summarize, Algo. 3's temporal complexity is O (1) + ξS O (1) + O (1) = O(ξS).

The relationship between all the algorithms is summarized in Fig. 4, and Table. 1. It is shown that the Algorithm 1 is used to locate the overload and lightly loaded controllers in the SDN domain. The whole switch migration is performed through an algorithm. 2. The optimization of controller selection for the TSSM scheme is achieved through Algorithm 3, and Algorithm 4 handles the TSSM process.

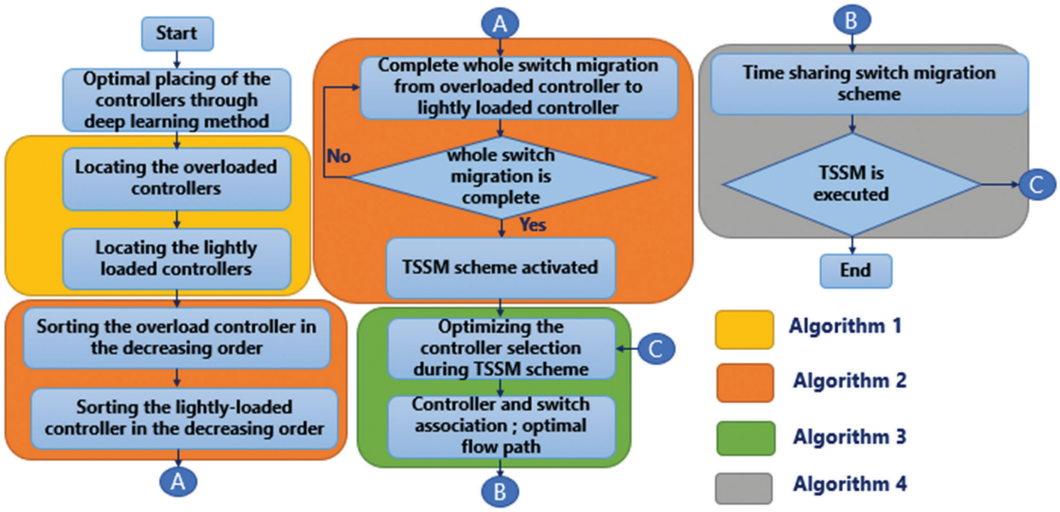


Fig. 4. Relationship between algorithms used in the improved TSSM scheme

Table. 1. Functions of Algorithms used in the improved TSSM scheme

Algorithms	Process
1	It is used to locate the overloaded and lightly loaded controllers in the SDN domain.
2	Initially, it is sorting the overloaded and lightly loaded controllers based on their overloading and PIMS accessibility. After that, it performs the whole switch migration from overloaded controllers to lightly loaded controllers.
3	It achieves optimized controller selection based on flow path through a greedy set algorithm for the TSSM operation.
4	It performs the TSSM operation and achieves better load balancing.

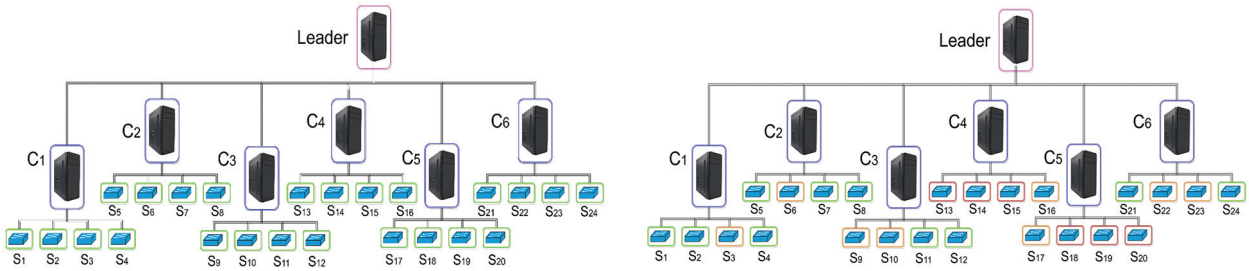


Fig. 5. Network topology used in the simulation test platform: (a) at '0' second, (b) at 21st second

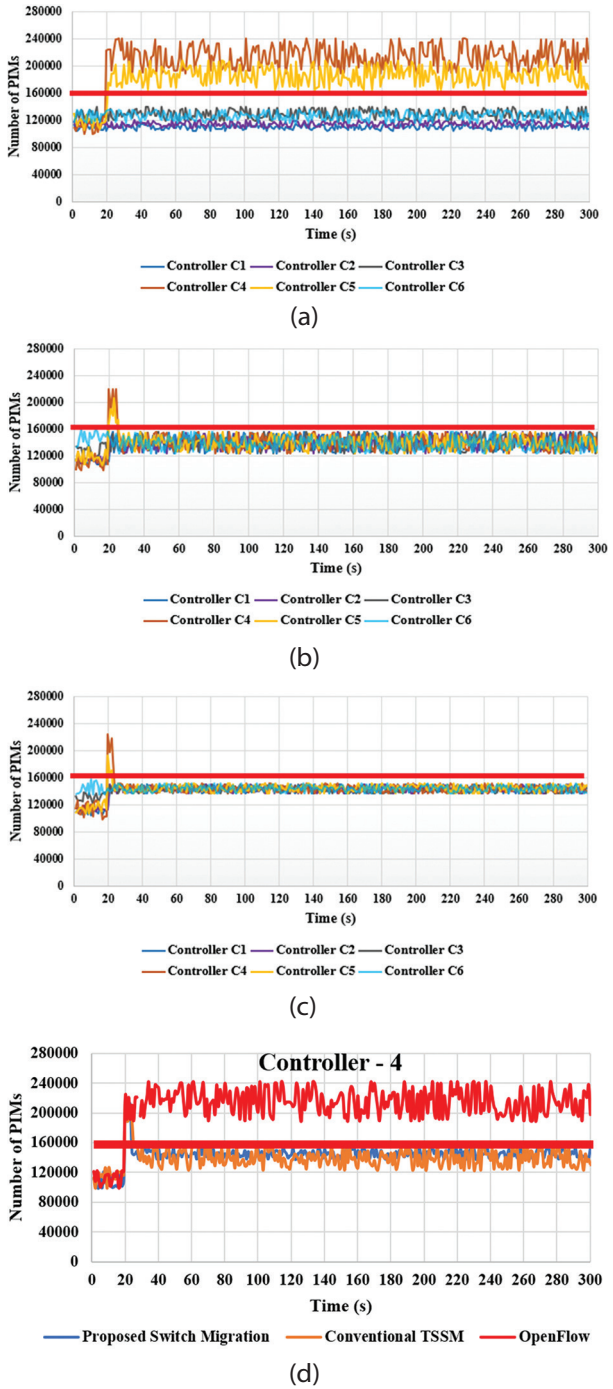


Fig. 6. Comparison of workload of controllers: (a) OpenFlow Method, (b)Conventional TSSM method, (c) Proposed Switch migration method, (d) Controller-4 comparsion results.

5. EVALUATION AND ANALYSIS

The performance of the proposed switch migration scheme is evaluated through time domain simulation analysis. The ONOS platform is considered the test platform, and hierarchical DSC architecture is adopted for the experimental network, it has seven controllers and 24 switches, as shown in Fig. 5. Therefore, one controller is acting as a leader, and its primary role is managing the other six controllers in the network; it does not involve itself in switch management; six secondary controllers are controlling their switches in its subnet. In this test platform, simulation time is considered 300 seconds and is divided into 60 periods. Each secondary controller has a PIMs handling capacity of 10^6 PIMs per period length of 5 seconds. In addition, the threshold for each controller is set at 800,000 PIMs per period. As a result, the total affordable load for the controller is estimated to be 4.8×10^6 PIMs per period. The loads of the switches are classified into three levels, namely, (i) light load, (ii) medium load, and (iii) large load. During light load, each switch generates around 21000 PIMs per second, whereas if a switch produces 42,000 PIMs per second, it is called medium load. However, if a switch is generating more than 63,500 PIMs per second, then it is called a large load. It is observed that if all the switches are lightly loaded, then the total affordable load for the controller is 2.52 × 106 PIMs per period, which is about 52.5 % of the total affordable load. But if all switches are considered as large loads, then the total load will be 7.62×10^6 PIMs per period, which is much higher than the total affordable load for the controller. Therefore, in this simulation study, the simulation starts with a small load in all switches, and the load will be randomly increased in the switches by cbench tool as simulation time increases, for evaluating the performance of the switch migration method. For example, at 21st seconds, 10 switches (S1, S2, S4, S5, S7, S8, S11, S12, S21, S24,) are generating about 21,500 PIMS/s, and 8 switches (S3, S6, S9, S10, S16, S17, S22, S23,) are generating 43,000 PIMs/s, and the remaining switches (S13, S14, S15, S18, S19, S20,) are producing 64,000 PIMs/s. Therefore, total controller workload is 4.715×10^6 PIMs per period, and there must be switch migration by both conventional (complete switch) and TSSM scheme. For evaluating the performance of the proposed method, three cases are considered: (i) work loads of occurrences, (ii) occurrences of overload, (iii) controller resource consumption, and (iv) migration cost.

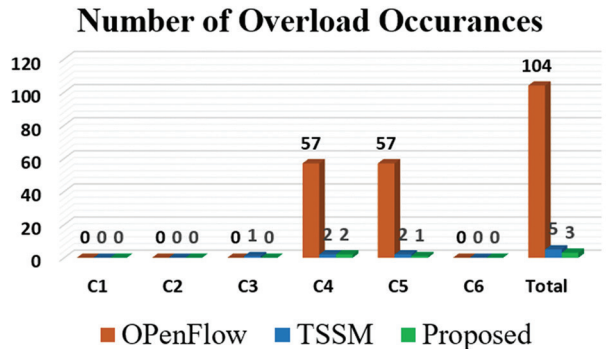


Fig. 7. Comparison of number of overload occurrences in conventional and proposed method

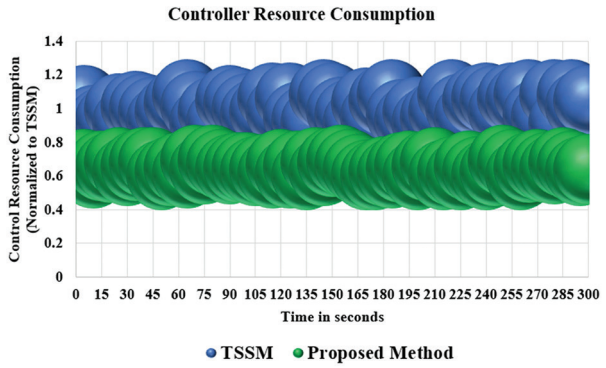


Fig. 8. Comparison of controller resource consumption between TSSM and proposed switch migration method

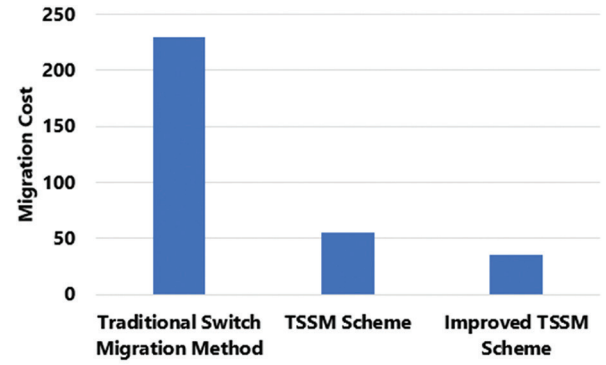


Fig. 9. Comparison of migration cost between conventional method and proposed method

Test 1: Workload of Controllers

As we have considered, each controller can process up to 800,000 PIMs per period, and if the controller has processed more than 160,000 PIMs/s then it is considered an overloaded controller. In this test, two conventional methods such as OpenFlow and TSSM are considered, and their test results are compared with the proposed method for evaluating the performance.

It is noted that switch migration is not performed in the OpenFlow method, and therefore, controllers C4, and C5 are heavily overloaded, as shown in Fig. 6a, based on PIMs generated in the switches. During this period, controllers C4 and C5 must handle about 1,165,000 PIMs/period, which is more than their maximum capacity (106 PIMs per period) and leads to unexpected issues in the networking domain. In the case of the TSSM scheme, it shares the workload between controllers through time sharing migration and ensures that all the controllers are within their threshold limits, as shown in Fig. 6b. In addition, the Ping-Pong issue (there are no high jumps, and frequently transferred switches are considered nil) is not noticed in the test results. The test results of the proposed switch migration scheme are presented in Fig. 6c. It is observed that load sharing between the controllers is much flatter (i.e., mostly all the controllers are sharing about the same load, which improves the efficiency and reduces the downtime or maintenance activity of the controllers) as compared with the TSSM scheme.

Test 2: Number of Overload Occurrences

This test is used to evaluate the number of overload occurrences for the controllers in the entire period (300 s), and it is useful for finding the performance of the switch migration method. The comparison of overload occurrence for all three methods is given in Fig. 7. It shows that the OpenFlow method provides a high number of overload occurrences since there is no action of switch migration, and therefore, controllers C1, C2, C3, and C6 are in the lightly loaded range, whereas C4, and C5 are highly loaded, and these controllers are completely overloaded during the given period.

In cases of TSSM, it has significantly reduced the number of overload occurrences for the controller since it avoids ping-pong difficulty and therefore switches that are repeatedly migrated are neglected. In the case of the proposed method, it further reduces the number of overload occurrences compared with the TSSM scheme. The proposed method uses more than one optimum controller as a lightly loaded controller during time sharing migration, which could reduce the number of overload occurrences. Because, if one controller is not enough to share the load of the switch (this controller may be initially considered excess in this situation), then it is again processed to find another controller for switch sharing in the conventional TSSM method, this might happen when the requirement for load sharing is high in the overloaded controller and lightly loaded single converters are not enough to handle this load. However, the proposed method chooses more optimum controllers based on the load sharing and avoids excessive processing and overload occurrences.

Test 3: Controller Resource Consumption

Controller resource consumption describes the utilization of a given number of controllers and switches. It is noted that if we reduce the number of controllers associated with the switches, that certainly reduces the switch migration cost of the network. OpenFlow is not performed with the switch migration event; thus, it is omitted for this evaluation study. The migration cost of the conventional TSSM is lower when compared with other switch migration methods. However, it is higher as re-

lated to the proposed switch migration method because the proposed method chooses the optimal controllers for sharing the workload based on flow characteristics, which reduces the controller's resource consumption and reduces the switch migration cost. The control resource consumption of the switch migration method is given in Fig. 8. It is observed that the proposed switch migration method provides about 34% less switch migration cost as compared with the conventional TSSM scheme.

Test 4: Migration Cost

The total number of switches that must be moved between various subnets to pass the migration cost test. Usually, controllers reassign switches to make sure loads are as evenly distributed as feasible. As switch migration is not carried out through the OpenFlow approach, it is not included. In the case of TSSM, it chooses switches for migration at random whose loads are near to tiny changes and permits two controllers to share their loads at the same time (i.e., time-sharing migration). The improved TSSM technique optimizes controller selection and enables controller workload sharing via time-sharing switch migration. The findings (Fig. 9) demonstrate that the enhanced TSSM program offers a 36% reduction in migration cost when compared with the traditional TSSM scheme.

6. CONCLUSION

This paper has presented an improved TSSM approach and mitigates the issue of increased switch migration cost in the conventional TSSM method by finding more than one optimal target controller during the time-sharing period. It has utilized the flow characteristics for finding the optimal controllers through a greedy set coverage algorithm. In addition, the proposed switch migration approach has TSSM benefits, which have conquered the ping-pong controller difficulty. The ONOS platform is considered for the performance evaluation of this study, which found that the improved TSSM scheme has better performance than the conventional TSSM method in terms of workload sharing among controllers, number of overload occurrences, and reduced controller resource consumption. Specifically, it reduces the controller's resource consumption by 34% when compared with the conventional TSSM.

ACKNOWLEDGEMENT

The authors wish to thank Annamalai University, India, for providing laboratory and experimentation resources.

7. REFERENCES

[1] N. Anerousis, P. Chemouil, A. A. Lazar, N. Mihai, S. B. Weinstein, "The origin and evolution of open programmable networks and SDN", IEEE Communications Surveys and Tutorials., Vol. 23, No. 3, 2021, pp. 1956-1971.

[2] Y.-C. Wang, H. Hu, "An adaptive broadcast and multicast traffic cutting framework to improve Ethernet efficiency by SDN", Journal of Information Science and Engineering., Vol. 35, No. 2, 2019, pp. 375-392.

[3] M. Alsaeedi, M. M. Mohamad, A. A. Al-Roubaiey, "Toward adaptive and scalable OpenFlow-SDN flow control: A survey", IEEE Access, Vol. 7, 2019, pp. 1073-1079.

[4] J. H. Cox, R. Clark, H. Owen, "Leveraging SDN and WebRTC for rogue access point security", IEEE Transactions on Network and Service Management., Vol. 14, No. 3, 2017, pp. 756-770.

[5] Y.-C. Wang, S.-Y. You, "An efficient route management framework for load balance and overhead reduction in SDN-based data center networks", IEEE Transactions on Network and Service Management., Vol. 15, No. 4, 2018, pp. 1422-1434.

[6] W. Iqbal et al. "ALAM: Anonymous lightweight authentication mechnism for SDN-enabled smart homes", IEEE Internet of Things Journal., Vol. 8, No. 12, 2021, pp. 9622-9633.

[7] Y.-C. Wang, R.-X. Ye, "Credibility-based countermeasure against slow HTTP DoS attacks by using SDN", Proceedings of the IEEE Annual Computing and Communication Workshop and Conference, NV, USA, 27-30 January 2021, pp. 890-895.

[8] F. Bannour, S. Souihi, A. Mellouk, "Distributed SDN control: Survey, taxonomy, and challenges", IEEE Communications Surveys and Tutorials. Vol. 20, No. 1, 2018, pp. 333-354.

[9] J. Lu, Z. Zhang, T. Hu, P. Yi, J. Lan, "A survey of controller placement problem in software-defined networking", IEEE Access, Vol. 7, 2019, pp. 24290-24307.

[10] H. Sufiev, Y. Haddad, L. Barenboim, J. Soler, "Dynamic SDN controller load balancing", Future Internet, Vol. 11, No. 3, 2019, pp. 1-21.

[11] Y. Wu, S. Zhou, Y. Wei, S. Leng, "Deep reinforcement learning for controller placement in software defined network", Proceedings of the. IEEE Conference on Computer Communications Workshops, Toronto, ON, Canada, 2020, pp. 1254-1259.

[12] Y.-C. Wang, Y.-C. Wang, "Efficient and low-cost defense against distributed denial-of-service attacks

in SDN-based networks", International Journal of Communication Systems., Vol. 33, No. 14, 2020, pp. 1-24.

[13] F. Tang, H. Zhang, L. T. Yang, L. Chen, "Elephant flow detectionand load-balanced routing with efficient sampling and classification", IEEE Transactions on Cloud Computing., Vol. 9, No. 3, 2021, pp. 1022-1036.

[14] Y.-C. Chan, K. Wang, Y.-H. Hsu, "Fast controller failover for multidomain software-defined networks", Proceedings of the. European Conference on Networks and Communications., Paris, France, 29 June - 02 July 2015, pp. 370-374.

[15] W. H. F. Aly, "Controller adaptive load balancing for SDN networks", Proceedings of the International Conference on Ubiquitous and Future Networks., Zagreb, Croatia, 2-5 July 2019, pp. 514-519.

[16] T. Hu, J. Zhang, L. Cao, J. Gao, "A reliable controller deployment strategy based on network condition evaluation in SDN", Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, Beijing, China, 24-26 November 2017, pp. 367-370.

[17] T. Kim, J. Myung, S.-E. Yoo, "Load balancing of distributed datastore in OpenDaylight controller cluster", IEEE Transactions on Network and Service Management, Vol. 16, No. 1, 2019, pp. 72-83.

[18] Y.-C. Wang, E.-J. Chang, "Cooperative flow management in multidomain SDN-based networks with multiple controllers", Proceedings of the International Conference on Smart Communities: Improving Quality of Life Using ICT, IoT and AI, Charlotte, NC, USA, 14-16 December 2020, pp. 82-86.

[19] S. Nithya, M. Sangeetha, K. N. A. Prethi, K. S. Sahoo, S. K. Panda, A. H. Gandomi, "SDCF: A software-defined cyber foraging framework for cloudlet environment", IEEE Transactions on Network and Service Management, Vol. 17, No. 4, 2020, pp. 2423-2435.

[20] K. S. Sahoo, P. Mishra, M. Tiwary, S. Ramasubbareddy, B. Balusamy, A. H. Gandomi, "Improving end-users' utility in software-defined wide area network systems", IEEE Transactions on Network and Service Management, Vol. 17, No. 2, 2020, pp. 696-707.

[21] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, R. Kompella, "Towards an elastic distributed SDN controller", ACM Special Interest Group on Data Communication., Vol. 43, No. 4, 2013, pp. 7-12.

[22] Z. Min, Q. Hua, Z. Jihong, "Dynamic switch migration algorithm with Q-learning towards scalable SDN control plane", Proceedings of the International Conference on Wireless Communications and Signal Processing, Nanjing, China, 11-13 October 2017, pp. 1-4.

[23] J. Cui, Q. Lu, H. Zhong, M. Tian, L. Liu, "SMCLBRT: A novel load-balancing strategy of multiple SDN controllers based on response time", Proceedings of the IEEE International Conference on High Performance Computing and Communications, Exeter, UK, 28-30 June 2018, pp. 541-546.

[24] K. S. Sahoo et al. "ESMLB: Efficient switch migration-based load balancing for multicontroller SDN in IoT", IEEE Internet of Things Journal, Vol. 7, No. 7, 2020, pp. 5852-5860.

[25] T. Hu, J. Lan, J. Zhang, W. Zhao, "EASM: Efficiency-aware switch migration for balancing controller loads in software-defined networking", Peer-to-Peer Networking and Applications., Vol. 12, 2019, pp. 452-464.

[26] Y. Hu, W. Wang, X. Gong, X. Que, S. Cheng, "BalanceFlow: Controller load balancing for OpenFlow networks", Proceedings of the IEEE International Conference on Cloud Computing and Intelligence Systems., Hangzhou, China, 30 October - 1 November2012, pp. 780-785.

[27] W. Lan, F. Li, X. Liu, Y. Qiu, "A dynamic load balancing mechanism for distributed controllers in software-defined networking", Proceedings of the International Conference on Measuring Technology and Mechatronics Automation, Changsha, China, 10-11 February 2018, pp. 259-262.

[28] S. M. Maity, C. Mandal, "Traffic-Aware Consistent Flow Migration in SDN", Proceedings of the IEEE International Conference on Communications, Dublin, Ireland, 7-11 June 2020, pp. 1-6.

[29] S. M. Maity, C. Mandal, "DART: Data Plane Load Reduction for Traffic Flow Migration in SDN," IEEE Transactions on Communications, Vol. 69, No. 3, 2021, pp. 1765-1774.

[30] B. Gorkemli, S. Tatlcıoglu, A. M. Tekalp, S. Civanlar, E. Lokman, ˇ "Dynamic control plane for SDN at scale", IEEE Journal on Selected Areas in Communications, Vol. 36, No. 12, 2018, pp. 2688-2701.

[31] F. Al-Tam, N. Correia, "Fractional switch migration in multicontroller software-defined networking", Computer Networks., Vol. 157, 2019, pp. 1-10.

[32] L. Chen, F. Tang, X. Li, "Mobility-and load-adaptive controller placement and assignment in LEO satellite networks", Proceedings of the IEEE Conference on Computer Communications, Vancouver, BC, Canada, 10-13 May 2021, pp. 1-10.

[33] X. Zhang et al. "Dynamical controller placement among SDN space-terrestrial integrated networks", Proceedings of the IEEE 22nd International Conference on High Performance Computing and Communications, 14-16 December 2020, pp. 352-359.

[34] L. Chen et al. "Dynamical control domain division for software-defined satellite-ground integrated vehicular networks", IEEE Transaction on Network Science and Engineering, Vol. 8, No.4, 2021, pp. 2732-2741.

[35] W.-K. Lai, Y.-C. Wang, Y.-C. Chen, Z.-T. Tsai, "TSSM: Time-Sharing Switch Migration to Balance Loads of Distributed SDN Controllers", IEEE Transaction on Network and Service Management, Vol. 19, No. 2, 2022, pp. 1585-1597.

[36] Y. Zhang, Y. Ran, Z. Zhang, "A simple approximation algorithm for minimum weight partial connected set cover", Journal of Combinatorial Optimization., Vol. 34, No. 3, 2017, pp. 956-963.