Journal of Computer Science

Research Article

A Novel Time-Based Switch Migration Method (TSSM) for Load Balancing in Distributed Software-Defined Networking

Thangaraj E[1], Dinesh K[1], Prabhakaran Paulraj[2], A Barkathulla[1], Margaret Flora[3] and Karthik Elangovan[1]

[1]*Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, India*
[2]*Department of Electrical, Electronics, and Communications Engineering, St. Joseph University in Tanzania, Dar Es Salaam, Tanzania*
[3]*Department of Artificial Intelligence and Machine Learning, RMD Engineering College, Chennai, India*

Article history
Received: 16-06-2025
Revised: 25-12-2025
Accepted: 27-12-2025

Corresponding Author:
Prabhakaran Paulraj
Department of Electrical, Electronics, and Communications Engineering, St. Joseph University in Tanzania, Dar Es Salaam, Tanzania
Email: prabakaran.p@sjuit.ac.tz

Abstract: Switch migration is predominantly used for workload balancing in distributed Software Defined Network (SDN) controllers. The Time-Sharing Switch Migration (TSSM) technique allowed the switch's responsibility to be divided based on the time duration condition of overload, which resulted in fewer overload events, fewer ping-pong controller issues, and more efficient controllers. However, it has higher controller resource utilization during TSSM working hours, higher migration expenses, and requires more controllers to complete. Therefore, based on flow characteristics, we proposed a shortest-feasible open flow path algorithm that optimizes controller selection during the TSSM phase. The new TSSM approach maintains the benefits of TSSM while lowering controller resource consumption and migration expenses. An open network operating system is used to accomplish the suggested method for practical reasons. The testing results show that the proposed method lowers migration costs and controller resource utilization by about 33% compared with the existing system.

Keywords: Time-Sharing Switch Migration (TSSM), Ping-Pong Controller, Software Defined Networking (SDN), Load Balancing, Resource Consumption, Shortest-Feasible Open Flow Path Algorithm

Introduction

Network administration has become much more challenging because of the evolution of modern technology growth, such as cloud computing, data mining, Internet of Things (IoT), and data traffic. Each switch of the conventional network architecture system is divided into two planes: One is the control plane, and the other is the data plane, the former of which processes packets and the latter of which handles administration and decision-making. Because system administrators or employees must update each switch individually in the network, updating the existing algorithms with policies of switches is difficult and consumes more time for execution (Anerousis et al., 2021).

Modern SDN revolutionizes network administration by separating the control plane from individual switches and centralizing it within a key component called the controller. This architecture enables a single controller to oversee and manage numerous switches across the network. Consequently, monitoring and operating network switches has become more efficient than traditional methods, as the controller offers centralized and comprehensive insights into the status and operation of all connected switches.

Additionally, complex algorithms along with control policies can be rapidly implemented on switches by configuring numerous rules in the controller (Wang and Hu, 2019). SDN also supports various applications, such as:

i. Anonymous authentication,
ii. Rogue access point detection, and
iii. Cyber-attack protection, among others (Sahana and Brahmananda, 2023; Shahapure and Jayarekha, 2020)

A bottleneck in network management performance makes using a lone controller in a big network a challenging choice. Consequently, network applications require Distributed SDN Control (DSC), which is a viable approach for managing big networks with several switches (Bannour et al., 2018). Multiple controllers can collaborate through the DSC, enabling effective network management together. A subnet is a group of switches that

each controller oversees, and in order to promote collaboration, processes may be shared between controllers. Controller placement is the process by which each controller divides the subnets' workload and redistributes the load on its switches by periodically inspecting each subnet (Lu et al., 2019). The primary goals of the controller placement are load balance and so on. Such control tactics may have the effect of drastically changing the subnet's switches, which could lead to ping-pong instability. Additionally, it is believed that controller placement schemes do not work well for short-range streams like impulses and spread denial of facility (Wu et al., 2020; Wang and Wang, 2020; Tang et al., 2021).

Switch migration solves the issues and enables a more seamless subnet shift in less time. A switch migration approach evaluates the workload status of controllers at each time interval to determine whether a controller in the network is underutilized (able to take on additional tasks) or overloaded (currently busy).

When the network experiences high traffic, the migration technique transfers a switch from an overcrowded subnet controller to one that is less congested. One switch is moved at the beginning of the period, which is the smallest slice of the migration that is followed by the majority of switch migration techniques now in use. The switch remains the same in each subnet after migration until it is selected for the next period. More significantly, a controller is always needed to keep an eye on one switch during the conversion process when using these migration strategies. As a result, when elephant flow conditions occur, when the flow transfers a lot of packets, the controller in these systems has to deal with the serious problems of subnet instability and ping-pong (Chan et al., 2015).

Literature Review

Figure 1 depicts the numerous studies that have been conducted in the DSC network. Moreover, it highlights the various issues. A method for reducing the duration of service interruptions by simply switching between controllers was proposed by Aly (2019). In the event that the typical leader controller unit fails, Hu et al. (2017) explained how a controller that is not heavily loaded can function as a leader. The requirement for controllers to preserve fairness when distributing their tasks is emphasized in the discussion, which discusses controller placement approaches and difficulties. In contrast to earlier controller placement techniques, shortest-feasible open flow path Kim et al. (2019) proposed a robust deployment strategy to reduce packet loss and enhance the stability of the network. Wang and Chang (2020) introduced a method that improves the performance of a distributed data store within an Open Daylight controller cluster by regularly assigning shared leaders to cluster members. Additionally, the system described in Nithya et al. (2020) enable controllers to collaborate in diverting traffic, thereby alleviating congestion during periods of high traffic or network overload.

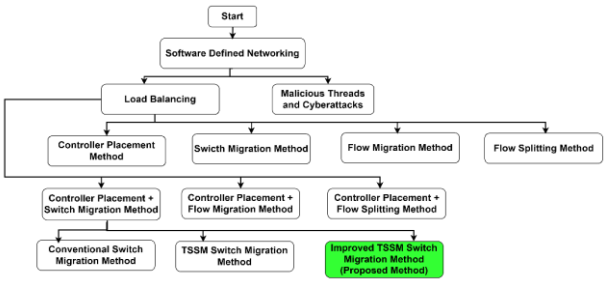


Fig. 1: Literature review of the paper

Sahoo et al. (2020a) have proposed a cyber-search system that utilizes a hybrid controller to manage cloudlets and local SDN. Dixit et al. (2013) introduce prediction-based controllers capable of estimating network demand and executing device migrations based on that demand. Research on controller placement indicates that certain techniques, such as workgroup control methods and deep reinforcement learning strategies, struggle to perform effectively during disruptions like distributed denial-of-service attacks.

By transferring switch management to less burdened controllers, switch migration alleviates the pressure on overworked controllers. Min et al. (2017) did not specify criteria for selecting destination controllers; it examined switch migration when a controller's CPU and memory utilization exceeded its threshold. Cui et al. (2018) applied the Q-learning technique to reduce the standard deviation of controller burden during switch migration. (Sahoo et al., 2020b) focused on the rapid transfer of switches from heavily loaded controllers, basing their approach on the controllers' response times. Hu et al. (2019) proposed selecting target controllers for switch migration by considering factors such as CPU usage, memory capacity, and bandwidth. Additionally, Hu et al. (2012) introduced a simulated annealing technique to identify the optimal controller while minimizing the cost of switch migration.

When the strength of the flow exceeds a certain threshold, migration occurs. A super controller oversees all system controllers and manages the flows for which each controller is responsible, following a method proposed by Al-Sadi et al. (2016). Maity et al. (2020) established a shortest-feasible open flow path strategy to govern controller flows in an SDN network. Maity et al. (2021) suggested a standard flow migration methodology that achieved a 15% reduction in migration time while providing a consistent, traffic-aware approach to minimizing flow migration duration. Additionally, Gorkemli et al. (2018) enhanced data plane load by 13% over the two-phase update method by developing a traffic-aware flow migration mechanism.

Flow splitting is a technique that allows multiple controllers to manage a switch. This method, proposed by Lan et al. (2018), utilizes a virtual overlay on the data

plane to facilitate communication between switches and their controllers. Additionally, (Tyagi and Singh, 2025) introduced a convex quadratic programming-based approach to simulate the mapping between controllers and switches, aiming to minimize new switch-controller assignments while achieving load balancing. Chen et al. (2021a) have proposed a cluster-based routing method based on particle swarm optimization and a self-adaptive genetic methodology. Furthermore, by taking into account the most important factors, such as the energy and distance of network devices, it is possible to balance the cluster size by using a novel fitness function. It demonstrates how the suggested approach enhances fitness value, stability period, etc.

Zhang et al. (2020) presented a controller placement method for SDN that utilizes control relation graphs. Their approach demonstrates how this strategy can reduce management costs in LEO satellite networks by optimizing load balancing and response time. Chen et al. (2021b) proposed an SDN-architecture-based space-terrestrial network that improves load balance and reaction time. They also introduced an effective algorithm for dynamic controller placement and adjustment to reduce management costs. Lai et al. (2022) have proposed a dynamical control domain division problem. Furthermore, a heuristic technique for selecting the optimal controller for improved load balancing has been given. However, a switch cannot be simultaneously actuated by multiple controllers due to synchronization issues and design complexity. Zhang et al. (2017) have proposed the Minimum Weight Partial Connected Set Cover (PCSC) problem. Their approach involves transforming the PCSC problem into a Minimum Quota Node Weighted Steiner Tree (QNWST) problem and is applicable to the Maximum Budgeted Connected Set Cover (BCSC) problem.

Figure 1 illustrates a structured approach to improving Software Defined Networking (SDN) performance. It starts with SDN as the base and identifies key challenges such as traffic load imbalance and cybersecurity threats. Different strategies are introduced, including controller placement, switch migration, flow migration, and flow splitting. Some methods combine controller placement with these techniques to achieve better efficiency. Existing solutions like conventional switch migration and TSSM-based migration are included for comparison. The workflow ultimately leads to an improved TSSM switch migration technique, which is proposed as a more effective method. This approach aims to enhance network stability, scalability, and security without relying on traditional designs.

Major Issue and Paper Contribution

From the literature review, it is clear that most existing methods struggle with the ping-pong problem. This issue and the difficulty of the controller are explained clearly with an example. Imagine a network with three switches [SL, SM, and SN] and two controllers [CXI and CYI]. The maximum workload of each controller is 200 PIMs/s. Every period, switches (SLA, SMA, SNA) produce 120 PIMS/s, 160 PIMS/s, and 120 PIMS/s, respectively. At time t, controller CYI controls switch SNA, while CXI controls switches SLA and SMA. CXI is overloaded and needs switch migration since $YCX = \delta LA(t) + \delta MA(t) = 120 + 160 > \lambda$ (CXI) (200 PIMS). The majority of switch migration solutions involve an overloaded controller requesting and temporarily taking over a switch from another controller. Consequently, Switch SLA is relocated to controller CYI's subnet at time t+1. However, controller CYI will be overloaded if $YCYI = \delta NA(t) + \delta LA(t) = 120+120 > \lambda CYI$ (200 PIMS) at period t+1. Ping-pong becomes more complex as a result of controller CYI asking CXI to switch once more at t+2 times.

Lai et al. (2022) recently presented a Time-Sharing Switch Migration approach (TSSM) to distribute the load of a switch that is simultaneously monitored by two controllers in overloaded conditions, therefore reducing controller ping-pong. It recommends a switch migration plan where two controllers divide the workload of the switch for a set period of time. In the example above, time (t+1), CXI controls 40 PIMs/s of SLA, and CYI uses migration to manage the remaining 80 PIMs. The workload of switch SLA is currently controlled by both controllers, CXI and CYI. The controller is overloaded in this period. Therefore, CXI workload is $YCXI = \delta LA(t) + \delta MA(t) = 40+160 = \lambda CXI$ (200 PIMS), and CYI workload is $YCYI = \delta NA(t) + \delta LA(t) = 120+80 = \lambda CYI$ (200 PIMS) Likewise, CYI processes 80 PIMs at time t +2 and then forwards the excess 20 PIMs/s to the CX subnet controller. Using this tactic, the TSSM technique successfully resolves the ping-pong problem of the controller.

By merging an overloaded controller with a lightly loaded one into a single unit, the switch is transferred at the right time from the overloaded subnet to the less burdened one. In contrast to existing techniques, like best-fit migration (Chen et al., 2021), churn-triggered migration (Lan et al., 2018), and group-based dynamic controller placement (Wu et al., 2020), the outcomes of this technique show a notable decrease in instances of controller overload while effectively distributing the workload among all controllers and improving overall controller performance. The research indicates that lightly loaded controller activities in TSSM enhance controller efficacy, even though they lead to higher migration costs and necessitate additional controller resources for network management. The proposed solution minimizes migration costs and optimizes switch migration during TSSM time by utilizing several lightly loaded controllers and selecting controllers based on flow characteristics. In

addition to offering TSSM benefits, the new TSSM approach lowers controller resource consumption and migration expenses. The suggested plan is put into practice utilizing the ONOS, which can handle almost a million flow processing requests every second, to ensure its viability.

In conclusion, by making it simple to update network policies and the newest algorithms, software-defined networking (SDN) results in an effective administration procedure for network management. Usually, network management uses distributed SDN while taking bottleneck problems into account. Load balancing is a crucial component of SDN and can be controlled via:

(1) The flow splitting
(2) Switch migration
(3) Dynamic controller placement, and
(4) Flow migration

Given synchronization and intricate design, a switch cannot be controlled by many controllers at once due to OpenFlow's practicality. As a result, the real-time controller platform cannot use flow migration or flow splitting techniques since they do not adhere to the Open Flow standard. Traditional switch migration techniques encounter a ping-pong problem during the migration process because only one switch is moved initially. The switch migration becomes unstable as a result. A time-sharing switch migration strategy fixes the ping-pong problem. Better load balancing results from this method's large reduction in the controller's overload occurrences. Nonetheless, controllers are chosen at random throughout the TSSM period. Because the migration switch is controlled by many network controllers, it may result in higher controller resource consumption and switch migration costs during TSSM operation. As a result, this paper proposed an enhanced TSSM scheme that has some advantages:

(1) Improved controller efficiency, fewer instances of controller overload, and the elimination of issues in ping-pong controller action
(2) The approach is used to specify and optimize the controller selection during TSSM, lowering the cost of switch migration and consumption of controller resources
(3) In comparison, the traditional TSSM system offers superior controller efficiency and load balancing

Organization of the Paper

Distributed Software-Defined Network Controller

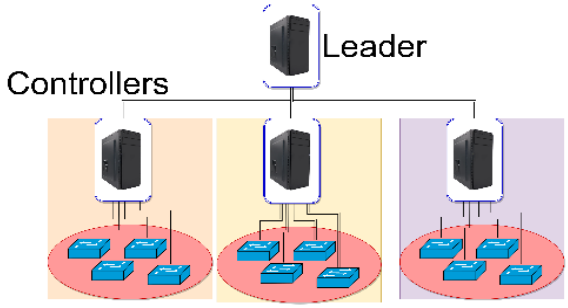
This section covers network architectures, the distributed SDN control network architecture, and the switch transmission method of the Open-Flow protocol. The proposed network architecture under the TSSM framework consists of a two-layer design: the data plane with Open Flow switches and the control plane with distributed SDN controllers. Switches initially connect to their nearest controller, but controller workloads are continuously monitored. The TSSM module evaluates traffic volume, controller utilization, and overhead cost to determine migration decisions. Busy controllers (C busy) offload selected switches to light controllers (C light) while ensuring that capacity constraints are not violated. This interaction ensures adaptive and stable load balancing.

Distributed Software-Defined Network Control Network Architecture

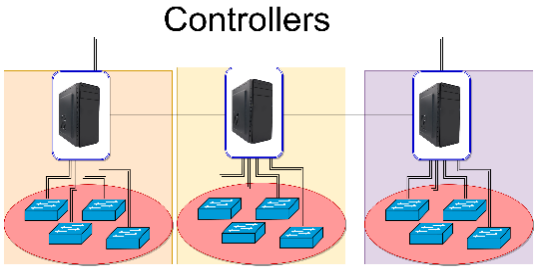
In a distributed SDN control network, two prevalent control strategies are:

(i) Flat control, often referred to as circular chain control
(ii) Hierarchical control (Bannour et al., 2018)

A central element of the hierarchical control paradigm is the leader, a distributed controller that possesses a global view of the network, as illustrated in Figure 2. The leader is responsible for delivering the latest network policies and algorithms to the sub-controllers. The sub-controller communicates its status to the leader while managing its own subnet of switches. Notably, if the current leader is decommissioned, a new leader will be selected according to the hierarchical structure (Hu et al., 2017).



(a) Ordered method



(b) Horizontal method

Fig. 2: Techniques for controlling the DSC architecture

Aspect	Ordered Method	Horizontal Method
Execution	Sequential	Parallel
Speed	Slower	Faster
Stability	High (less conflict)	Lower (risk of conflicts)
Complexity	Lower	Higher (needs coordination)
Use case	Small–medium scale, predictable loads	Large-scale, dynamic traffic

In contrast, with circular chain control, each controller possesses a local perspective of the network and manages its own subnet independently.

This article implements the proposed switch migration methodology using a hierarchical strategy. A leader is responsible for monitoring and controlling the state of each sub-controller and applying the TSSM scheme. This scheme selects the least burdened controller during fluctuations in control of flow, congestion, distributed denial of service (DDoS) attacks, and other scenarios. Subsequently, two sub-controllers one heavily loaded and the other lightly loaded collaborate to share responsibilities and migrate the switch as needed. Network administrators set a controller's threshold level between 90-95% of full capacity to prevent switch migrations. Overloading occurs when the controller's workload exceeds this level, and the controller is chosen based on its maximum and standby capacity specified in Equation (1):

$$A_C = o_C - p_C \quad (1)$$

Where:

$A_C \longrightarrow$ Threshold workload level of the controller

$o_C \longrightarrow$ Maximum workload capacity of the controller

$p_C \longrightarrow$ standby workload capacity of the controller

Switch Transfer Process in Open Flow

Open-Flow connects to numerous controllers and permits switch transfers between subnets. From the standpoint of the switch, each associated controller chooses the following responsibilities.

OFPCR Role Equal: This defaulting role gives the controller total control over switching and enables the sending and receiving of instructions and status data. Now, in a similar vein, it is fully accessible to all controllers when functioning in this capacity. OFPCR Role Slave (Slave): This function can only read the switch's status when the controller function is fixed to slave. The OFPCR role master, also called the master, is comparable to an equal role and grants the controller complete authority over something. - However, it is emphasized that only one controller, for example, is recognized as the controller of one switch that is referred to as the master switch, while all other controllers are deemed slaves to that switch. It has total control over the switch; the master controller starts the switch transferring process. For instance, the master switch and slave switch controllers C_{XI} and C_{YI}, respectively. In order to balance the workload, it is required that overworked controllers employ a leader to transfer a switch to other controllers. The targeted controller, C_{YI}, receives a transfer request for switch S_{LA} from the master controller, C_{XI}, after an instruction from the leader. Then, the master switch sends a role request message, controller C_q asks to change the S_{LA} control role from slave to master. S_{LA} replies to C_{YI} with the reply message to Master. Ultimately, C_{YI} notifies C_{XI} that the switch S_{LA} migration was successful. Controller C_{XI} operates a slave and switch S_{LA}.

Open-Flow does not, however, outline the process for selecting target switches and controllers for migration. The suggested improved TSSM approach chooses the best controller and establishes the best time for switch migration to take place within the TSSM period.

Network Modelling

Consider an SDN network consisting of multiple switches (S_{NA}) and controllers (C_{NA}). Within this framework, a controller (e.g., C_{XA}) can manage a switch (e.g., S_{LA}). According to the OpenFlow model, each switch can be controlled by only one controller at a time. This means that CXI serves as the master controller for S_{LA}, and this controller-switch assignment can change during switch migration. The workload of each controller is determined by the PIMs (in-messages) received from the switches. Specifically, the workload of a switch $(\delta(t))$ is based on the number of PIMs it generates during a given time period *'t'*. Additionally, each controller has a defined workload capacity, which is the maximum number of PIMs it can process during separate periods. For instance, the workload of controller C_{XI} is assessed when it manages switches ranging from *SAa* to *SAz*:

$$\Upsilon_{CXI} = \sum_{SA_a}^{SA} \delta(t) \quad (2)$$

Generally, the maximum workload of the controller (Y_C) is less than the maximum capacity (λ_C) to allow a reserve load in case of unforeseen circumstances, such as flow fluctuations or unexpected demand. This study investigates the hierarchical control of DSC architecture, where the master continuously receives workload data from every controller in the network and manages the migration of switches between them as needed.

Switch Migration

- Meaning
- In SDN, switches are assigned to controllers. When a controller is overloaded, some switches are reassigned (migrated) to another controller

- Goal: Balance the load across controllers
- Key aspect: A planned reallocation of switch-to-controller mapping, usually driven by traffic demand or controller capacity

Controller Ping-Pong

- Meaning
- A negative phenomenon where a switch keeps migrating back and forth between controllers too frequently
- Cause: Poor migration policies (e.g., threshold too sensitive, lack of hysteresis)
- Consequence: Increases overhead, instability, and delays instead of improving load balance
- Example: Switch S1 moves from C1 → C2 due to temporary load, but then moves back to C1 when conditions change slightly

Load Balancing

- Meaning
- A broader concept that refers to distributing traffic and processing fairly across multiple controllers to prevent overload
- Techniques include
  - Switch migration (reassigning switches)
  - Dynamic flow rerouting (diverting flows without moving switches)
  - Hierarchical delegation (some controllers acting as leaders)
- Goal: Achieve fairness, efficiency, and reliability across the network

Suggested Switch Migration Methodology

The network switches are initially set up using the controller placement technique or by network operators, with a master controller overseeing each switch. Conventional switch migration techniques involve moving a switch at the beginning of the period and moving the entire switch, even if it is not required, as explained in the section above. Because of this, the connection between controllers and switches doesn't change over time. Time-sharing is used in TSSM to enable switch migration, allowing network switches to dynamically switch at any time. Additionally, section 2.1 discussed that the controller ping-pong problem is effectively resolved by the TSSM technique. However, because the method permits several controllers to share their loads in the time-sharing switch migration period, it consumes a higher time period, potentially increasing migration cost in comparison with the remaining migration techniques. It is found that the number of controllers and switches used determines the approximate migration expenses. Consequently, based on flow characteristics, this study proposed a method that significantly reduced the number of controllers connected to switches during time-sharing migration. During the time-sharing migration phase, we created a shortest-feasible open flow path algorithm to determine the optimal connection between switches and controllers. This reduces the number of controllers connected to the switch, which lowers the cost of migration and controller resource consumption. The following methods are meant to guarantee that the suggested switch migration technique is completed successfully. Instead of assigning a switch to any random light controller, the algorithm chooses the controller that offers the shortest feasible control path while satisfying capacity constraints.

How it Works

1. Input:

- Network topology (switch–controller distances)
- Flow demand characteristics (traffic volume, delay sensitivity)
- Controller status (capacity, load)

2. Candidate Controller Set Generation:

- For a migrating switch *Si*, identify all controllers {*Cj*} that are currently underloaded (C-light)

3. Path Cost Evaluation (Shortest Open Flow Path):

- Compute the shortest communication path (hop count, latency, or bandwidth-weighted distance) between *Si* and each feasible controller *Cj*
- This is typically done using Dijkstra's algorithm or a minimum-hop BFS on the network graph

4. Selection Rule:

- Among all feasible controllers, choose the one with the minimum path cost (shortest feasible Open Flow path)
- This ensures low latency and efficient control traffic

The diagram explains a step-by-step process for balancing load in an SDN environment using multiple algorithms. It begins by optimally placing controllers with the help of a deep learning approach and identifying both overloaded and lightly loaded controllers. These controllers are then ranked based on their load levels. A full switch migration is attempted from heavily loaded

controllers to lightly loaded ones. Once migration conditions are satisfied, the Time-Sharing Switch Migration (TSSM) scheme is activated. During TSSM, controller selection and switch associations are optimized to ensure efficient flow paths. The process ends after successful execution of the TSSM scheme, achieving improved load distribution and network performance.

Algorithm 1: Finding Overloaded and Lightly Loaded Controllers

1 $C_{busy} \leftarrow$ ø and $C_{light} \leftarrow$ ø ;
2 **Foreach** $C_{XI} \in C$ **do**
3 $\Upsilon_{C_{XI}} \leftarrow 0$;
4 **foreach** $S_{LA} \in S_X$ **do**
5 $\Upsilon_{C_{XI}} \leftarrow \Upsilon_{C_{XI}} + \delta_{LA,t}^{(X)}$;
6 If $\Upsilon_{C_X} > \Phi_{C_X}$ then
7 $C_{busy} \leftarrow C_{busy} \cup \{C_X\}$;
8 else if $\Upsilon_{C_{XI}} < \mu \times \Phi_{C_{XI}}$ **then**
9 $C_{light} \leftarrow C_{light} \cup \{C_{XI}\}$;
10 If $C_{busy} \neq$ ø and $C_{light} \neq$ ø **then**

Algorithm 1: Finding Overloaded and Lightly Loaded Controllers

As indicated by C_{busy} and C_{light}, this technique ensures that every overloaded (busy) and lightly loaded (assistant or target) controller in the specified network is identified. The burden on each controller, represented as Y_{CX}, is assessed using an Equation (2). This equation, outlined in the method code, spans 3 to 5 lines and is calculated by summation of loads in every switch within the subnet switch(e.g., $\delta_{LA,t}^{(XI)} + \delta_{MA,t}^{(XI)} + \cdots$After that, the threshold level (A_{C_XI}) is compared to the controller workload (e.g., Y_{CXI}.)workload crosses the threshold value, the controller is overloaded, then it is entered into the overloaded controllers entity in the leader (as mentioned in lines 6 7) Line 8 elects light load controllers that support a lightly loaded coefficient, denoted as 'μ', which network managers specify to be between 0.8 and 0.85. The threshold value is then multiplied by this lightly loaded coefficient. It means that the controllers are lightly loaded if their job is less than the result. They are then added to the leader's lightly loaded controller unit. As shown in line 10, switch migration must take place when neither the C_{busy} nor the C_{light} controllers are empty.

Algorithm 2: Balancing the load for migration segment switch

1 SORT (C_{busy}, $\Upsilon_{C_{XI}} - \Phi_{C\_XI}$);
2 SORT (C_{light}, $\Phi_{C\_YI} - \Upsilon_{C_{YI}}$);
3 foreach $C_{XI} \in C_{busy}$ do
4 SORT (S_{XI}, $\delta_{LA,t}^{(XI)}$);
5 while $\Upsilon_{C_{XI}} > \Phi_{C\_XI}$ do
6 if $C_{light} =$ ø then
7 Cease above module ;
8 select the improved controllers $[C_{YI1}, C_{YI2},\ldots]$ from C_{light};
(Controller to Matrix of switch relationship)
9 ← Algorithm 3 (Request PIM´s of Switch, Switches from C_{busy})
(S_{LA}, $[\tau_1, \tau_2, \ldots]$, $[\rho_1, \rho_2, \ldots]$)
10 ← Algorithm 4 (C_{XI} , $[C_{YI1}, C_{YI2},\ldots]$) ;
11 Transfer S_L to $[C_{Y1}, C_{Y2},\ldots]$ subnet next $[\tau_1, \tau_2, \ldots]$ units of time;
12 $\Upsilon_{C_{XI}} \leftarrow \Upsilon_{C_{XI}} - [\rho_1, \rho_2, \ldots]$;
13 $\Upsilon_{C_{YI\_1}} \leftarrow \Phi_{C\_YI} + [\rho_1, \rho_2, \ldots]$;
$\Upsilon_{C_{YI\_2}} \leftarrow \Phi_{C\_YI} + [n_1, n_2, \ldots]$;
14 **if** $\Upsilon_{C_{YI[1,2,\ldots]}} \geq \mu \times \Phi_{C\_YI[1,2,\ldots]}$ **then**
15 $C_{light} \leftarrow C_{light} \setminus [C_{YI1}, C_{YI2},\ldots]$;
16 **Else**
17 SORT (C_{light}, $\Phi_{C\_YI} - \Upsilon_{C_{YI}}$) ;

Algorithm 2: Ordering the Overloaded and Assisting Controllers, As Well as Switch Migration

By noticing a couple of over-loaded and lightly-loaded controllers, this technique aims to divide the workload across the controllers. In order to reduce the workload order, the SORT function helps to organize controllers that are overloaded and lightly loaded. While line 2 of the code sorts the information about the lightly loaded controller, line 1 sorts the overload controllers. Consequently, the task of an overloaded (busy) controller will be given priority over a controller with extremely excess capacity. Lines 3 17 of the code use a for-loop to address each controller in the network, going from the maximum overloaded to the final. Line 4 assembles the switches below C_{YI} administration in downward order according to their load. Lines 5-16 implement a while Loop to decrease the load on C_{YI} by shifting a switch till it touches an exact threshold. The algorithm terminates in Lines 6 7, but only if there is no accessible assistant controller (i.e., C_{light} is empty) and there are still also several controllers in the domain. To address this, the TSSM method is employed to select a lightly weighted controller, C_{YI}, for load distribution. First, the best controllers for TSSM are identified using Algorithm 3, which considers controller resource consumption and migration costs. Once the best controllers are determined, the TSSM scheme is implemented according to Algorithm 4. Algorithm 4 produces three output parameters, as shown in line 10. Here, 'ρ' indicates the number of PIMs to be migrated to each controller, while 'τ'

specifies the period during which switch SL should migrate to other controllers. Lines 11 13 update the workload for C_{XI}, and if the workload exceeds the threshold value, those controllers are deleted from the list of minimally loaded controllers, as indicated by line 14. Conversely, if the workload remains below the threshold, they are reintegrated into the minimally loaded controller, as demonstrated in terms 17 and 2.

Algorithm 3:	Selection of Optimized Controller for TSSM Scheme
1	**Input:** Organized light and busy controllers ={C_{busy}}, {C_{light}} obtained from Algorithm 2 ;
2	Sort ($\delta_{LA,t}^{(XI)}$,$\delta_{MA,t}^{(XI)}$, $\delta_{NA,t}^{(XI)}$) ;
3	Volume and redundant weight for every controller below a leader
4	Traffic Matrix: $TR^{'}$ = [TR_{XY}]
5	**Initialization:**TR = [TR_{XIYI}], C_{NA} = [$C_{XIYI}^{'}$],o_C, p_C
6	**repeat**
7	Each switch accomplishes its maximum desired migration.
8	Early migration pair S_{LA}: $C_{XI} \rightarrow C_{YI}$;
9	**for all** controllers **do**:
10	**if** S_L: $C_{XI} \rightarrow C_{YI}$; and $\Upsilon_{C_{YI}} \leq o_{C\_YI}$. p_{C_YI}: Migration satisfaction fixes do not go against capacity restrictions.
11	**If** migration value (S_n, C_n) < 0: Take into account the relative importance of control traffic overhead and control resource utilization.
12	Put the switch migration choice into operation. $S_{LA} \rightarrow C_{YI}$
13	Update C_{NA} = [$C_{XIYI}^{'}$];
14	**end if**
15	**end if**
16	**end for**
17	**Until** the switches have not offered any suggestions.

Algorithm 3: Optimization of the Controller for the TSSM Scheme to Save Migration Costs

The objective of this proposed technique is to generate effective controllers for TSSM operation. In order to reduce controller resource utilization and, consequently, switch migration cost, the optimal controller is selected based on flow characteristics. Algorithm 3 displays the Shortest-Feasible Open Flow Path, which is utilized for the best controller selection. The PIMs of every switch in the overloaded controller, along with the threshold level, flow needs, and other details of the controller, are necessary for this approach. The sort function organizes a series of flows along each path in ascending order from lines 3 to 12. A controller that manages most of the switches along the route is selected and executed in lines 4-6.

This pseudo-code describes a switch migration algorithm for SDN controller load balancing:

1. Identify busy and light controllers
2. Evaluate traffic loads and capacity
3. Iteratively migrate switches from overloaded to underloaded controllers
4. Ensure migrations don't violate capacity constraints
5. Accept migrations only if they improve efficiency
6. Stop when no switch suggests migration

Algorithm 4:	Time to Switch Migration Estimating Segment
1	$\Delta_{over} \leftarrow \min(\Upsilon_{C_{XI}} - \Phi_{C\_XI})$ & $\Delta_{light} \leftarrow \max(\Phi_{C\_YI} - \Upsilon_{C_{YI}})$;
2	$S_{XI}^{\chi I} \leftarrow \emptyset$ and $S_{XI}^{\psi} \leftarrow \emptyset$;
3	**foreach** $S_{LA} \in S_{XI}$ **do**
4	**if** $\delta_{LA,t}^{(XI)} \geq \Delta_{over}$ **then**
5	$S_{XI}^{\chi I} \leftarrow S_{XI}^{\chi} \cup \{S_L\}$;
6	**Else**
7	$S_{XI}^{\psi} \leftarrow S_{XI}^{\psi} \cup \{S_{LA}\}$;
8	**if** $S_{XI}^{\chi} \neq \emptyset$ **then**
9	$S_{LA} \leftarrow$ the final switch of S_{XI}^{χ} ;
10	$b\tau = \left[\%\ of\ \Delta_{light}\ with respect to\ \Delta_{over}\right] \times (L_t)$;
11	**else**
12	$S_{LA} \leftarrow$the initial switch of S_{XI}^{ψ};
13	$\tau \leftarrow$ 0 then$\rho \leftarrow \delta_{LA,t}^{(XI)}$;
14	$\delta_{LA,t}^{(XI)} \leftarrow \delta_{LA,t}^{(XI)} - \rho$ and$\delta_{LA,t}^{(YI)} \leftarrow \rho$;
15	**return** (S_{LA}, τ, ρ);

Algorithm 4: Time to Switch Migration Estimating Segment

Following the definition of the optimal lightly loaded controllers (C_{YI1}, C_{YI2},...) in Algorithm 3, Algorithm 4 pairs these controllers with a highly-loaded controller to achieve three jobs. These tasks are:

i) Selecting a switch from the overloaded controller
ii) Calculating the migration time (τ) of the switch, and
iii) Determining numerous PIMs (ρ) that manage the lightly loaded controllers

The algorithm divides switch in high-loaded controllers into two sub-nets, 'Δover' and 'Δunder', based on their load levels. If a switch's load exceeds, it is placed in decreasing order. To reduce the number of migrations (as performed in lines 8 9), switches positioned near the highly-loaded ones exactly, the final switch according to the load particular order are chosen for migration within the subnet. This method is effective since a small amount of high load from the high-loaded controllers can be simply moved to the minimally loaded controllers. The projected migration periods for every switch are found by the number of packets in messages created in that switch, the ideal minimally loaded controllers, and then the switch itself. If Δlight is half of Δthe over value and the PIM formation rate remains constant, the switch migration period should be half the duration specified in

Equation (3). Line 13 statuses that switch migration initiates at the beginning of the period when $\tau = 0$. Additionally, as shown in lines 11 and 12, the subnet is assessed for improved load balancing even when it is not highly loaded, provided the subnet's switches are empty. This process, utilizing the optimum controller finding algorithm (Algorithm 3), will continue till every controller is load stable for every switch in the time allotment method. After this, Algorithm 2 will be applied once more.

Materials

Time domain simulation analysis is used to calculate the achievement of the suggested switch migration technique. The ONOS platform serves as the test platform, as shown in Figures 3-4, and the experimental network, which has 24 switches and 7 controllers, is designed using a hierarchical DSC architecture.

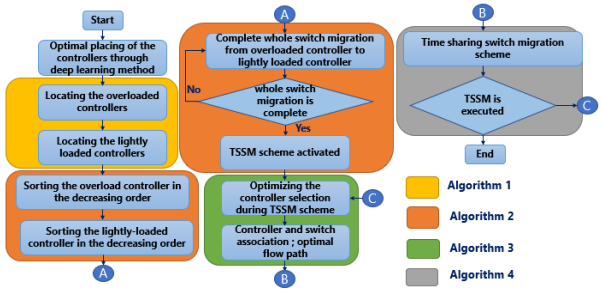
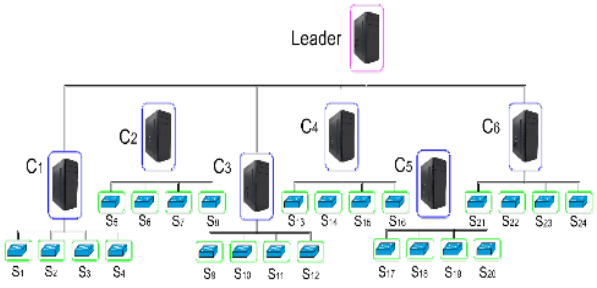
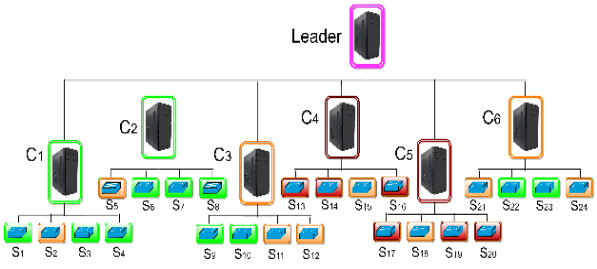


Fig. 3: Relationships between the enhanced TSSM scheme algorithms



(a) Network configuration at "0" sec



(b) Network configuration at "16th" sec

Fig. 4: The network topology utilized by the platform for the simulated test

As a result, one controller is served by a leader, overseeing the remaining six controllers in the network.

This leader does not contribute to switching organization; instead, the six secondary controllers are responsible for managing their respective switches within their subnet. The test platform indicates that the 200-second simulation is divided into 40 segments. Each secondary controller can process 640,000 PIMs at 5-second intervals.

Furthermore, each controller has a fixed capacity of 512,000 PIMs per month, allowing it to handle a total of 3.072 million PIMs per period. There are three levels of switch loads: Low, medium, and large. Under light stress, each switch generates approximately 13,600 PIMs per second, while under medium load, it generates around 26,800 PIMs per second. Nonetheless, a switch is deemed to be a heavy load if it produces more than 40,800 PIMs per second. The entire controller's affordable load, or roughly 48% of the total controller's affordable load, is 1.6 106 PIMs per period. Switches are loaded lightly. The entire load, however, is 4.808 106 PIMs per period if all switches are considered heavy loads, which is far more than the total controller's affordable load.

Results and Discussion

In this proposed work, the simulation begins with a modest load on all switches to evaluate the effectiveness of the switch migration approach. The c-bench technique is employed to gradually increase the load on the switches throughout the simulation. At the sixteenth second, eight switches (S2, S5, S11, S12, S15, S18, S21, and S24) are generating 26,800 PIMs/s. Meanwhile, the remaining six switches (S13, S14, S16, S17, S19, and S20) are processing 40,800 PIMs/s, and the ten switches (S1, S3, S4, S6, S7, S8, S9, S10, S22, and S23) are producing approximately 13,600 PIMs/s concurrently. Switch migration must therefore take place utilizing both the conventional (full switch) and TSSM techniques, with the overall controller burden being 3.0176 106 PIMs each period. The performance of the proposed method is evaluated using three examples:

(1) Controller workloads
(2) Overload events
(3) Consumption of the resource controller

As mentioned earlier, every controller has the capacity to process up to 512,000 PIMs per period. A controller is considered overloaded if it processes more than 102,400 PIMs per second. This test examines two standard techniques:

(i) Open-Flow
(ii) TSSM schemes

The test outcomes are associated with the suggested methodology for estimating efficiency.

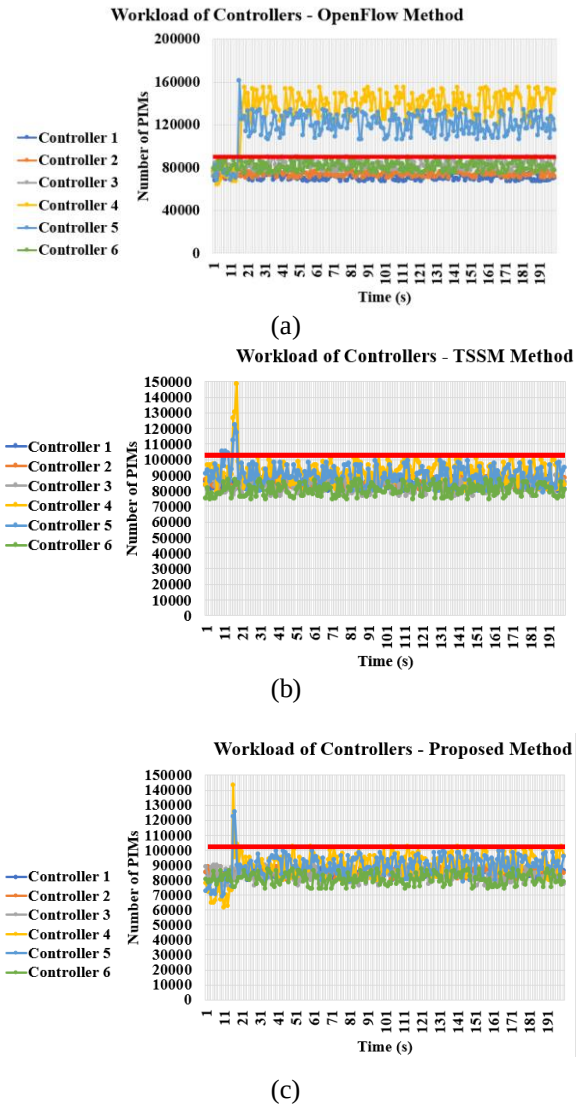


Fig. 5: Controller workload comparison: (a) Open-Flow approach, (b) traditional TSSM, and (c) the suggested approach

(a) Open-Flow approach
(b) traditional TSSM
(c) the suggested approach

Based on PIMs produced in the switches, Fig. 5a shows that controllers C4 and C5 are severely overloaded since the migration of the switch is not carried out using the Open-Flow technique. C4, C5 controllers are required to manage roughly 745,600 PIMs every period during this time, surpassing their maximum capacity of 640,000 PIMs/S period. This leads to unexpected challenges in the networking domain. As shown in Figure 5b, the TSSM system divides the workload among the controllers using time-sharing migration and makes sure that every controller is below its threshold limitations. Additionally, the test results do not show the Ping-Pong issue, which is characterized by no high leaps and often communicated switches being handled as nil. The test results for the proposed switch migration technique are displayed in Fig. 5c. Load sharing between controllers is far flatter than in the TSSM scheme; that is, practically all controllers share a similar load, increasing efficiency and lowering maintenance or downtime.

Amount of High-Load Scenario

In this evaluation, it is crucial to assess the migration techniques of the switch performance since it counts the number of times the controllers experience overload over the course of 200 seconds. A comparison of overload occurrence for each of the three ways is shown in Figure 6. Because there is no switch migration action, it shows that the Open Flow approach has resulted in numerous overload events. As a result, during the specified period, controllers C4 and C5 experience heavy load and are completely overloaded, while [C1, C2, C3, and C6] controllers remain lightly loaded.

This examination is crucial for assessing the effectiveness and migration technique of the switch, as it measures how many times the controllers experience overload over a 200-second period. Figure 6 illustrates the frequency of overload occurrences for each of the three methods. The lack of switch migration action shows that the Open Flow approach results in a considerable number of overload events. As a result, during this period, controllers C4 and C5 are heavily loaded and completely overloaded, while controllers [C1, C2, C3, and C6] remain in a light-loaded state. This situation can arise if the lightly loaded single converters cannot support the demand, leading to a high load-sharing requirement on the overloaded controller. In contrast, the suggested approach reduces superfluous processing and overload scenarios while choosing more efficient controllers based on load sharing.

Controller Resource Consumption

This test evaluates the resource consumption of the controller to assess the migration costs associated with switch migration schemes. The number of controllers and switches used is described by controller resource usage. It should be mentioned that the cost of switch migration is reduced when the number of controllers connected to the switches is kept to a minimum. Open-Flow is not included in this assessment study since it is not carried out during the switch migration event. The typical TSSM has a lower migration cost than other switch migration techniques. However, it is higher than the suggested switch migration technique because the latter optimizes controller resource consumption and switch migration costs by choosing the right controllers to share the workload. The switch migration strategy's control resource utilization is shown in Figure 7.

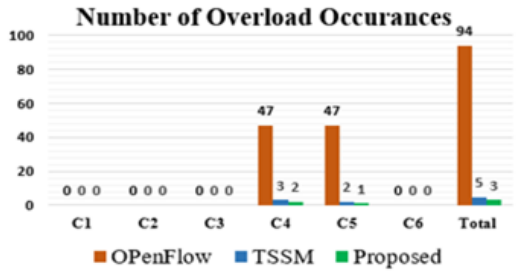


Fig. 6: Comparison of Controller Overload Events Across Switch Migration Techniques

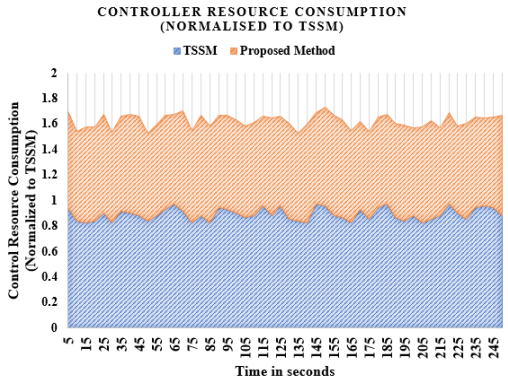


Fig. 7: Controller resource consumption comparison between the suggested switch migration method and TSSM

The recommended switch migration strategy reduces switch migration expenses by about 33% in comparison to the traditional TSSM system. Table 1 shows how effective the suggested approach is overall when compared to the current approaches.

Conclusion

This proposed work addresses the issue of high switch migration costs in the traditional TSSM strategy by selecting several optimal target controllers during the time-sharing phase. It employed a shortest-feasible open flow path algorithm method to identify the optimal controllers based on flow characteristics. Additionally, the ping-pong controller problem has been resolved by the proposed switch migration technique, which offers TSSM advantages. The ONOS platform was utilized to assess the study's performance, revealing the proposed improved TSSM method. This improvement was evident in workload sharing, controller resource usage, and the frequency of overload incidents. Specifically, the modified TSSM reduced controller resource utilization by 33% compared to the standard TSSM. By minimizing the ping-pong issues between the controllers and optimizing the migration cost, the suggested paper enhances load balancing. However, in certain cases, choosing the controllers based on flow characteristics increases the calculation time (i.e., memory). Deep learning methods may be used in the future to speed up processing and reduce migration costs. Even though the proposed method has a lot of merits, it has a limitation of high computational time during switch migration.

Table 1: Evaluation of the suggested method's efficacy in comparison to current approaches

	Load Balancing Strategy	OpenFlow Complaint	Time Sharing	Controller Ping-pong difficulty	Switch Migration Cost
(Cui et al., 2018)	Switch Migration	Y	N	Y	High
(Hu et al., 2019)	Switch Migration	Y	N	Y	High
(Hu e al.,2012)	Switch Migration	Y	N	Y	Medium
(Lai et al., 2022)	Switch Migration	Y	Y	N	Medium
Proposed Method	TSSM-Switch Migration	Y	Y	N	Low

Acknowledgment

The authors are grateful to the publisher for their constant support in the publication of this manuscript. The author, Dr. E. Thangaraj, expresses his deep sense of gratitude to the Veltech University Management for their constant encouragement that enabled him to work on this research, and all authors are thankful to everyone who contributed to the successful publication of this research paper.

Funding Information

This study was not supported by any funding agencies. The funders had no role in study design, decision to publish, or preparation of the manuscript.

Authors Contributions

Thangaraj E: Drafted the manuscript and compiled information from the literature.

Dinesh K: Gathered information from the literature and revised the manuscript.

Prabhakaran Paulraj: Supervised the manuscript.

Barkathulla A and Margaret Flora: Designed the figures and tables.

Karthik Elangovan: Reviewed the manuscript.

Ethics

This article does not contain any studies with human participants or animals performed by any of the authors.

References

Al-Sadi, A. M., Al-Sherbaz, A., Xue, J., & Turner, S. (2016). *Routing algorithm optimization for software defined network WAN*. 1–6. https://doi.org/10.1109/aic-mitcsa.2016.7759945

Aly, W. H. F. (2019). *Controller Adaptive Load Balancing for SDN Networks*. 2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN), Zagreb, Croatia. https://doi.org/10.1109/icufn.2019.8805922

Anerousis, N., Chemouil, P., Lazar, A. A., Mihai, N., & Weinstein, S. B. (2021). The Origin and Evolution of Open Programmable Networks and SDN. *IEEE Communications Surveys and Tutorials*, *23*(3), 1956–1971. https://doi.org/10.1109/comst.2021.3060582

Bannour, F., Souihi, S., & Mellouk, A. (2018). Distributed SDN Control: Survey, Taxonomy, and Challenges. *IEEE Communications Surveys & Tutorials*, *20*(1), 333–354. https://doi.org/10.1109/comst.2017.2782482

Chan, Y.-C., Wang, K., & Hsu, Y.-H. (2015). *Fast Controller Failover for Multi-domain Software-Defined Networks*. 2015 European Conference on Networks and Communications (EuCNC), Paris, France. https://doi.org/10.1109/eucnc.2015.7194101

Chen, L., Tang, F., & Li, X. (2021a). *Mobility-and load-adaptive controller placement and assignment in leo satellite networks*. 1–10. https://doi.org/10.1109/infocom42981.2021.9488806

Chen, L., Tang, F., Li, X., Yang, L. T., Cao, L., Yu, J., Fu, L., Li, Z., & Kong, L. (2021b). Dynamical Control Domain Division for Software-Defined Satellite-Ground Integrated Vehicular Networks. *IEEE Transactions on Network Science and Engineering*, *8*(4), 2732–2741. https://doi.org/10.1109/tnse.2021.3050213

Cui, J., Lu, Q., Zhong, H., Tian, M., & Liu, L. (2018). *SMCLBRT: A Novel Load-Balancing Strategy of Multiple SDN Controllers Based on Response Time*. 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; Exeter, United Kingdom. https://doi.org/10.1109/hpcc/smartcity/dss.2018.00102

Dixit, A., Hao, F., Mukherjee, S., Lakshman, T. V., & Kompella, R. (2013). Towards an elastic distributed SDN controller. *ACM SIGCOMM Computer Communication Review*, *43*(4), 7–12. https://doi.org/10.1145/2534169.2491193

Gorkemli, B., Tatlicioglu, S., Tekalp, A. M., Civanlar, S., & Lokman, E. (2018). Dynamic Control Plane for SDN at Scale. *IEEE Journal on Selected Areas in Communications*, *36*(12), 2688–2701. https://doi.org/10.1109/jsac.2018.2871308

Hu, T., Lan, J., Zhang, J., & Zhao, W. (2019). EASM: Efficiency-aware switch migration for balancing controller loads in software-defined networking. *Peer-to-Peer Networking and Applications*, *12*(2), 452–464. https://doi.org/10.1007/s12083-018-0632-6

Hu, T., Zhang, J., Cao, L., & Gao, J. (2017). *A reliable controller deployment strategy based on network condition evaluation in SDN*. 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China. https://doi.org/10.1109/icsess.2017.8342934

Hu, Y., Wang, W., Gong, X., Que, X., & Cheng, S. (2012). *BalanceFlow: Controller load balancing for OpenFlow networks*. 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems (CCIS), Hangzhou, China. https://doi.org/10.1109/ccis.2012.6664282

Kim, T., Myung, J., & Yoo, S.-E. (2019). Load Balancing of Distributed Datastore in OpenDaylight Controller Cluster. *IEEE Transactions on Network and Service Management*, *16*(1), 72–83. https://doi.org/10.1109/tnsm.2019.2891592

Lai, W.-K., Wang, Y.-C., Chen, Y.-C., & Tsai, Z.-T. (2022). TSSM: Time-Sharing Switch Migration to Balance Loads of Distributed SDN Controllers. *IEEE Transactions on Network and Service Management*, *19*(2), 1585–1597. https://doi.org/10.1109/tnsm.2022.3146834

Lan, W., Li, F., Liu, X., & Qiu, Y. (2018). *A Dynamic Load Balancing Mechanism for Distributed Controllers in Software-Defined Networking*. 259–262. https://doi.org/10.1109/icmtma.2018.00069

Lu, J., Zhang, Z., Hu, T., Yi, P., & Lan, J. (2019). A Survey of Controller Placement Problem in Software-Defined Networking. *IEEE Access*, *7*, 24290–24307. https://doi.org/10.1109/access.2019.2893283

Maity, I., Misra, S., & Mandal, C. (2020). *Traffic-Aware Consistent Flow Migration in SDN*. ICC 2020 - 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland. https://doi.org/10.1109/icc40277.2020.9148983

Maity, I., Misra, S., & Mandal, C. (2021). DART: Data Plane Load Reduction for Traffic Flow Migration in SDN. *IEEE Transactions on Communications*, *69*(3), 1765-1774. https://doi.org/10.1109/tcomm.2020.3042271

Min, Z., Hua, Q., & Jihong, Z. (2017). *Dynamic switch migration algorithm with Q-learning towards scalable SDN control plane*. 1–4. https://doi.org/10.1109/wcsp.2017.8171121

Nithya, S., Sangeetha, M., Prethi, K. N. A., Sahoo, K. S., Panda, S. K., & Gandomi, A. H. (2020). SDCF: A Software-Defined Cyber Foraging Framework for Cloudlet Environment. *IEEE Transactions on Network and Service Management*, *17*(4), 2423–2435. https://doi.org/10.1109/tnsm.2020.3015657

Sahana, S. D., & Brahmananda, S. H. (2023). Secure Authentication Framework for SDN-IoT network using Keccak-256 and Bliss-B algorithms. *International Journal of Information Technology*, *15*(1), 335–344. https://doi.org/10.1007/s41870-022-01074-w

Sahoo, K. S., Mishra, P., Tiwary, M., Ramasubbareddy, S., Balusamy, B., & Gandomi, A. H. (2020a). Improving End-Users Utility in Software-Defined Wide Area Network Systems. *IEEE Transactions on Network and Service Management*, *17*(2), 696–707. https://doi.org/10.1109/tnsm.2019.2953621

Sahoo, K. S., Puthal, D., Tiwary, M., Usman, M., Sahoo, B., Wen, Z., Sahoo, B. P. S., & Ranjan, R. (2020b). ESMLB: Efficient Switch Migration-Based Load Balancing for Multicontroller SDN in IoT. *IEEE Internet of Things Journal*, *7*(7), 5852–5860. https://doi.org/10.1109/jiot.2019.2952527

Shahapure, N. H., & Jayarekha, P. (2020). Virtual machine migration ased load balancing for resource management and scalability in cloud environment. *International Journal of Information Technology*, *12*(4), 1331–1342.yu https://doi.org/10.1007/s41870-018-0216-y

Tang, F., Zhang, H., Yang, L. T., & Chen, L. (2021). Elephant Flow Detection and Load-Balanced Routing with Efficient Sampling and Classification. *IEEE Transactions on Cloud Computing*, *9*(3), 1022–1036. https://doi.org/10.1109/tcc.2019.2901669

Tyagi, V., & Singh, S. (2025). A novel energy efficient routing technique for SDN-enabled underwater WSNs using free-space optical communication. *Journal of Optical Communications*, *45*(s1), s777–s790. https://doi.org/10.1515/joc-2022-0204

Wang, Y., & Wang, Y. (2020). Efficient and low-cost defense against distributed denial-of-service attacks in SDN-based networks. *International Journal of Communication Systems*, *33*(14). https://doi.org/10.1002/dac.4461

Wang, Y.-C., & Chang, E.-J. (2020). *Multi-domain SDN-based Networks with Multiple Controllers*. 82–86. https://doi.org/10.1109/honet50430.2020.9322815

Wang, Y.C., & Hu, H. (2019). An Adaptive Broadcast and Multicast Traffic Cutting Framework to Improve Ethernet Efficiency by SDN. *Journal of Information Science & Engineering*, *35*(2), 375–392.

Wu, Y., Zhou, S., Wei, Y., & Leng, S. (2020). *Deep Reinforcement Learning for Controller Placement in Software Defined Network*. 1254–1259. https://doi.org/10.1109/infocomwkshps50562.2020.9162977

Zhang, X., Tang, F., Cao, L., Chen, L., Yu, J., Xu, W., Zhang, X., Lei, J., & Wang, Z. (2020). *Dynamical Controller Placement Among SDN Space-Terrestrial Integrated Networks*. 352–359. https://doi.org/10.1109/hpcc-smartcity-dss50907.2020.00043

Zhang, Y., Ran, Y., & Zhang, Z. (2017). A simple approximation algorithm for minimum weight partial connected set cover. *Journal of Combinatorial Optimization*, *34*(3), 956-963. https://doi.org/10.1007/s10878-017-0122-4