International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2023/223684 Volume 10, Issue 6, November – December (2023)

RESEARCH ARTICLE

Assessing a Real-time Adaptive Traffic Route Based on Ranking Software Defined Networking (SDN) Cluster of Controllers in a Datacenter

Omar M. Mohamed
Department of Computer Science, Faculty of Science, Minia University, Egypt.
omarmakram@minia.edu.eg

Tarek M. Mahmoud
Computer Science Department, Faculty of Computers and Artificial Intelligence, University of Sadat City, Egypt.
tarek@fcai.usc.edu.eg

Abdelmgeid A. Ali
Department of Computer Science, Faculty of Science, Minia University, Egypt.
a.ali@minia.edu.eg

Received: 20 July 2023 / Revised: 19 November 2023 / Accepted: 03 December 2023 / Published: 30 December 2023

Abstract – Software Defined Networking (SDN) is the new network model that uses the notion of centralized administration and control to promote network management. Although SDN offered many advantages to promote network performance, some challenges arose due to its architecture. A centralized controller must not constitute a single point of failure for the network to achieve high availability. This reveals the need for a Real-time fault tolerance mechanism. This mechanism aims to address potential failures by providing redundancy and failover capabilities within the network infrastructure. By distributing control and decision-making responsibilities across the cluster control, the system can continue to operate seamlessly by selecting the better controller by routing the traffic from the cluster to it even if one controller experiences a failure. This real-time fault tolerance mechanism plays a vital role in maintaining uninterrupted network operations and achieving the desired level of availability. This paper presents a Real-time Efficiently Adaptive Traffic Route RATR algorithm that ensures fault tolerance and load balancing which achieves high availability consistency based on real-time measurements of cluster members' performance monitoring and records the results as votes. Then calculate the collected votes for each performance metrics load which are CPU, Memory, Network traffic, and Response time. Finally, run the proposed grading mechanism to elect the leader controller and his vices from among all cluster members. Extensive experiments are conducted to prove the effectiveness of RATR. The results show that RATR achieves an optimal performance not only on the network throughput but also on the delay and packet loss compared with Round Robin and SMCLBRT algorithms.

Index Terms – Software Defined Networking, Controller, Cluster, Traffic, Load-Balancing, Fault-Tolerance Throughput, RATR.

1. INTRODUCTION

The proliferation of mobile devices and content, in conjunction with the widespread adoption of server virtualization and the advent of cloud services, necessitates a critical reevaluation of conventional network architectures within the networking industry. Although numerous networks still adhere to hierarchical structures, typified by tiers of Ethernet switches arranged in a tree-like configuration, this fixed design, which was well-suited for the era of client-server computing, fails to adequately meet the evolving computational and storage demands of modern enterprise data centers, campuses, and carrier environments [1]. Software Defined Networking (SDN) is a new modern approach in networking technology that eliminates the complex and static nature of traditional distributed network architectures by centralizing the control plane, thereby facilitating enhanced information exchange and utilization. This engenders a network architecture characterized by heightened flexibility and optimization, aptly aligned with contemporary application requisites. Specifically, SDN decouples the control plane from the data plane, yielding a more streamlined approach to packet management [2]. The basic structure of SDN is represented in (Figure 1). The SDN architecture consists of three distinct layers: the data plane layer, the control plane layer, and the application layer. The data plane layer, the control plane layer, and the application layer. The data plane layer is responsible for forwarding network traffic according to the assigned rules/policies [3]. The control plane layer plays a pivotal role in overseeing network infrastructure and deliberating on the optimal handling of network traffic. It achieves this by leveraging an SDN controller, which governs the overarching SDN functions. The responsibilities of the control plane encompass activities such as storing network topology information, configuring devices, disseminating state information updates, and determining the shortest path routing strategies. In essence, this layer serves as an intermediary, bridging the infrastructure layer and the application layer in the SDN architecture [4]. The controller is responsible for managing the entire traffic flow and solely takes decisions on routing, flow forwarding and packet dropping through programming [5]. The application layer, situated atop SDN architecture, houses business applications responsible for overseeing and enhancing business services. These applications employ control logic, informed by network state data from controller Northbound interfaces, to effect changes in network behaviour. Underlying this, the application layer forms the foundation, comprising both physical and virtual network components [6].

SDN empowers the utilization of two fundamental Application Programming Interfaces (APIs): Southbound APIs and Northbound APIs. These interfaces facilitate bidirectional communication and interaction between the control and data planes. By employing Northbound APIs within an SDN controller, one can exercise programmatic control over the network infrastructure. Concurrently, Southbound APIs operate as intermediary links between the control and forwarding components, establishing a seamless and efficient flow of information [3]. Although SDN remains in a state of rapid evolution, several issues have been discovered. The centralized network control inherent in the SDN paradigm presents two significant challenges. Firstly, there is limited reliability stemming from the presence of a single point of failure. Secondly, the control traffic between the infrastructure layer devices (switches) and the controller is confined to a single server with finite processing capacity, leading to scalability concerns. [7]. To ensure the reliability of a network, SDN must possess the capability to straightforwardly and gracefully execute failure recovery. The aforementioned challenges must be addressed while concurrently maintaining a logically centralized perspective of the network state, a critical aspect when harnessing a consortium of SDN controllers.

In a cluster architecture, communication with the switches is handled by several controllers (C1, C2, ……, Cn) as mentioned in (Figure 2). As a result of the distribution of the control traffic between the network devices and the controllers and the resulting advantageous load balancing, each controller's processing load decreases. Moreover, flexible methods can be established to boost network reliability in the event of one or more controller malfunctions [8].

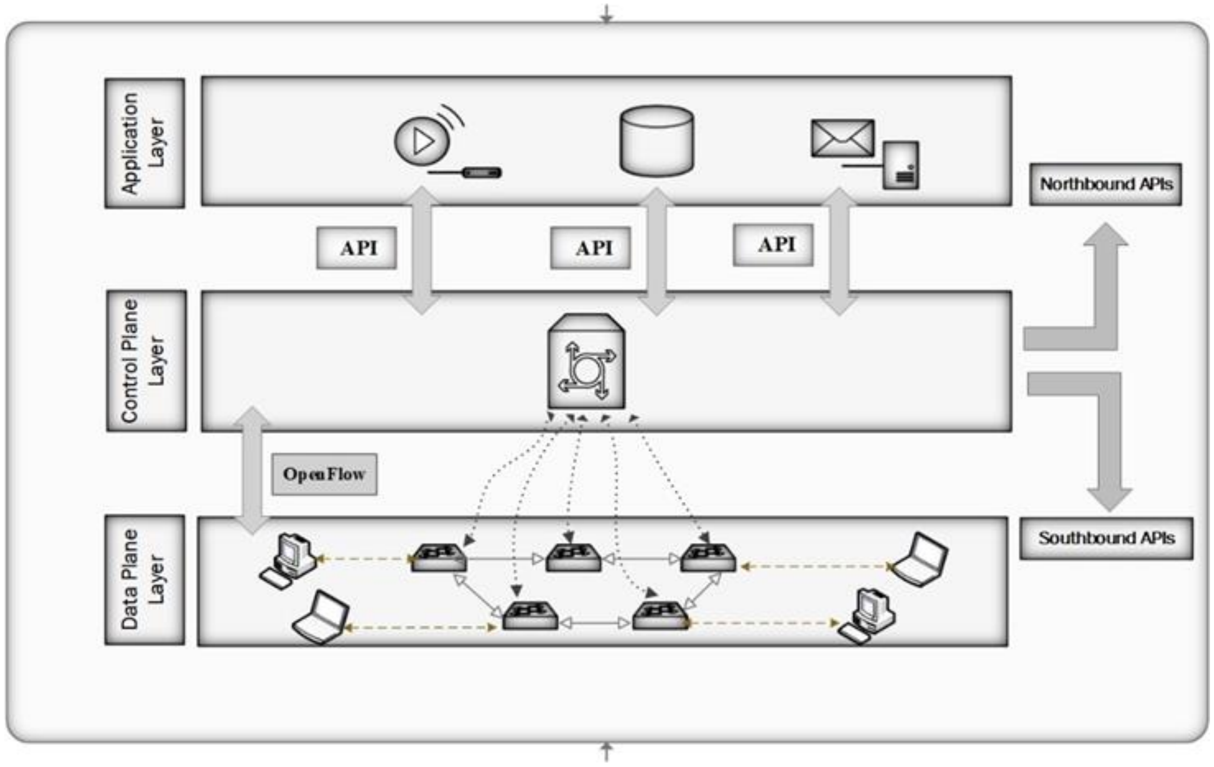

Figure 1 SDN Paradigm Architecture

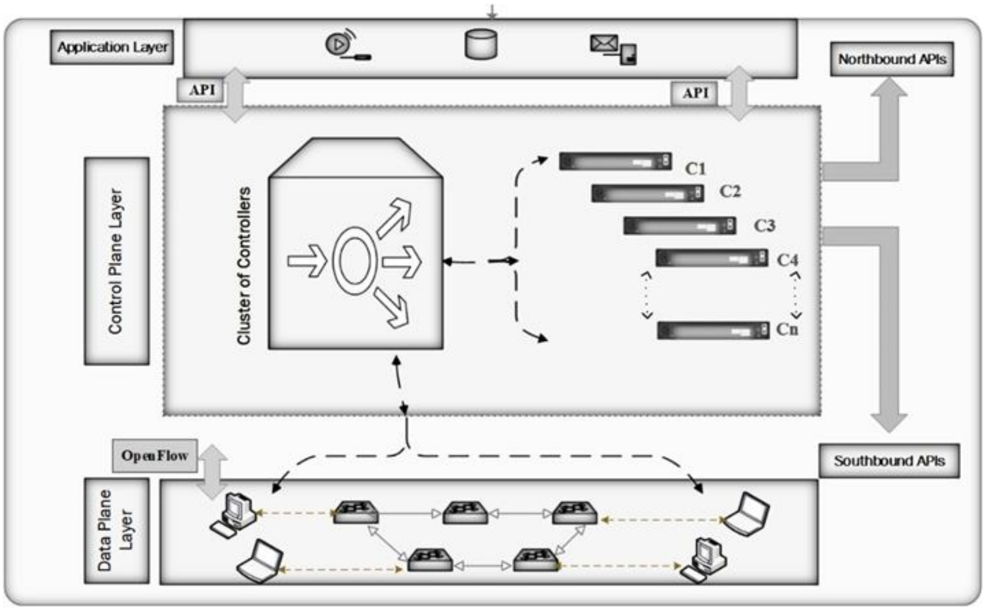

Figure 2 SDN Cluster Architecture

1.1. Theoretical Background

1.1.1. Fault –Tolerance

Fault tolerance is the property of failure adaptation that ensures system sustainability even if any of its components has failed. The proposed SDN controller cluster architecture represents a fault-tolerance design that maintains a pool of redundant controllers to promptly address any failure. [9]. A robust and fault-tolerant controller should be able to handle failures gracefully and recover quickly.

1.1.2. Load Balancing

Load balancing is the ability to distribute workloads more effectively among multiple controllers, ensuring that all available resources are used to their fullest potential while throughput is maximized, response times are kept to a minimum, and overload of any single controller is avoided [10].

1.1.3. Performance of Controller Load

Many attributes affected controller performance such as CPU load, memory load, network load which is the sum of IN/OUT bandwidth traffic, and the response time [11] which is the time it takes for the controller to respond to OpenFlow messages. Precisely determining the load status of the controller is the key aspect of load balancing, as distinct evaluation criteria will produce varying outcomes in terms of the load status.

1.2. Problem Statement

The inherent centralization of SDN control introduces potential limitations in terms of reliability and scalability. The reliance on a single controller creates a single point of failure, jeopardizing network resilience. Additionally, the control traffic between infrastructure layer devices and the centralized controller can overwhelm the processing capacity of a single server, hindering scalability as network size and complexity increase. The utilization of an SDN controller cluster represents a prevalent technique, offering a resolution to the inherent single point of failure issue associated with centralized controllers. The selection of an optimal controller from the cluster members is paramount to achieving peak network efficiency. Choosing the most capable controller to manage network traffic can lead to substantial performance gains. Consequently, the primary challenge lies in identifying and selecting the optimal controller among all controllers in the cluster to assume control of network traffic.

1.3. Contributions

To mitigate the single point of failure inherent in centralized controller architectures, an adaptive leader controller election mechanism and its associated vices are implemented to achieve load balancing and fault tolerance. The principal contribution of this paper can be outlined as follows:

- Real-time performance monitoring of cluster members is employed to actively track the performance and resource utilization of the controllers within the cluster, including parameters such as CPU utilization, memory allocation, network traffic, and response time.
- Network resource allocation is efficiently managed through the implementation of a controller load grading mechanism.

- Extensive simulations conducted using the Mininet framework demonstrate that the proposed mechanism significantly enhances network performance.

1.4. Organization

The remainder of this paper is organized as follows. Section 2 introduces the related works. The proposed framework model with algorithms is explained in Section 3. The conducted experiments and the obtained results are given in Section 4. Finally, section 5 concludes this paper and introduces the future works.

2. RELATED WORK

Commencing with a succinct overview of Software-Defined Networks (SDNs), this section delves into the examination of multi-controller scenarios. The subsequent section provides an introduction to a selection of established algorithms employed to mitigate the single point of failure inherent in SDN architecture. Jie et al. [12] implemented an SDN controller load-balancing scheme known as SMCLBRT, which focuses on response time metrics. Their study delved into the evolving characteristics of response times in relation to varying controller workloads. The processing of PACKET_IN messages, which are sent by switches to the master controller when they encounter packets without matching flow table entries, represents a substantial computational burden on the controller. The uneven distribution of these messages across controllers can lead to workload imbalances, with some controllers becoming overloaded and experiencing delays, while others remain idle or operate under normal conditions. The SMCLBRT scheme, specifically designed to address load balancing in distributed SDN control planes, particularly in scenarios with multiple overloaded controllers, utilizes a comprehensive assessment approach that focuses on overloaded controllers and their response times. Akin to other switch migration schemes, load balancing in the SDN control plane entails three fundamental phases: measuring the load imbalance among controllers, formulating strategic migration plans involving the selection of overloaded controllers, primary load switches, and immigration controllers, and finally, executing these migration plans to effectively redistribute the loads borne by the overloaded controllers.

The SMCLBRT architecture comprises four modules for centralized management: monitoring, load imbalance detection, switch migration decision, and migration execution. The monitoring module gathers controller response times and real-time load information, furnishing data to the load imbalance detection module. Upon detecting a load imbalance, the switch migration decision module generates migration actions, which are subsequently implemented by the migration execution module. The SMCLBRT scheme exhibits adaptability to diverse network environments. It results in a good-grained judgment on controllers' leads to revealing to the controller that might have a rapid increase in its response time at an appropriate time.

André and Fernando introduced a fault-tolerant controller framework for Software-Defined Networking (SDN) called RAMA [13]. The novelty of the RAMA solution lies in its ability to facilitate immediate deployment without necessitating alterations to OpenFlow or underlying hardware. The Rama controller framework employs a primary/backup model to ensure fault tolerance in SDN controllers. Rama's architecture includes OpenFlow-enabled switches, controllers managing these switches, and a coordination service. This model comprises a primary controller and multiple backup controllers, allowing for fault tolerance by electing a new leader in case of the master controller's failure. The coordination service, while ensuring strong consistency among controllers, can create a bottleneck due to the need for consensus between replicas. The proposed protocol focuses on maintaining consistent switch states in the presence of faults. Furthermore, the RAMA controller framework guarantees three crucial properties: (i) precise event processing by controllers, (ii) uniform event processing order across all controllers, ensuring they attain the same state, and (iii) switches processing commands exactly once using OpenFlow bundles [14]. The RAMA controller framework uses a two-stage replication protocol ensuring the consistency of the controller state. The first stage involves replicating the event to all replicas of the controller. The second stage involves verifying that the event has been processed successfully by all replicas of the controller. RAMA offers substantial advantages by ensuring consistent command and event processing, providing robust assurances comparable to Ravana [15], all without requiring modifications to switches or the OpenFlow protocol. This quality positions RAMA as a highly effective facilitator for seamlessly implementing fault-tolerant SDN solutions. Despite minor drawbacks, such as increased network message exchanges and additional mechanisms like bundles, leading to higher costs, RAMA's core proposition of ensuring consistent command and event processing without necessitating changes to switches or OpenFlow protocol remains compelling. Consequently, RAMA stands as a valuable enabler for the immediate adoption of fault-tolerant SDN solutions. Jehad et al. [16] Introduced an approach for selecting an optimal SDN controller to be considered a multi-criteria decision-making (MCDM) problem based on controller-supporting features. This study investigated ten essential auxiliary features that wield significant influence on the performance of Software-Defined Networking (SDN) systems and the meticulous process of controller selection. The OpenFlow protocol version facilitates comprehensive network monitoring through a user-friendly GUI. The Northbound REST API enables direct communication with the controller, reducing latency and enhancing throughput. Clustering and Quantum API empower controllers to leverage cloud computing for enhanced capabilities. Synchronization efficiency measures the controller's ability to manage responses, storage, and updates for OpenFlow switches. Productivity is linked to the programming language used in controller coding and influences application development ease. Partnership support involves evaluating technical prowess and financial resources associated with controller development. Platform support ensures compatibility with diverse operating systems, enabling features like multithreading and expedited memory access. Modularity support enhances controller robustness and performance in large-scale systems.

This approach depends on two steps for selecting an SDN controller. First, run the analytical network process (ANP) with features that affect the controllers' performance giving ranks for all controllers and second execute a performance comparison for verifying the QoS improvement according to the high-weight values. To assess the efficacy of the carefully chosen controllers, a rigorous quantitative evaluation of the proposed methodology was conducted in comparison with the Analytical Hierarchy Process (AHP). This evaluation involved measuring various Quality of Service (QoS) parameters for the two controllers, including topology discovery time, latency, throughput, and CPU utilization. The optimum controller selection using the ANP model conducts decreasing Topology time discovery and throughput improvement with sensible CPU utilization of the selected controller. Madhukrishna et al. [17] Propose a self-adaptive load balancing (SALB) scheme that dynamically distributes network traffic among multiple controllers by migrating switches from overloaded controllers to underutilized ones. The scheme considers both load distribution efficiency and switch-controller proximity to optimize network performance.

The SALB scheme employs a systematic process comprising load measurement, broadcast, evaluation, migration, and link reset phases. It utilizes real-time load data and adaptive threshold adjustments to maintain balanced workloads across controllers, thereby enhancing overall Software-Defined Network (SDN) performance. SALB's key component are Load Measurement Component: Which continuously monitors the load on each controller, providing real-time insights into processing capabilities and network traffic handling. Load Broadcast Component: Disseminates load information to all controllers, enabling them to assess peer workloads and make informed load-balancing decisions.

Load Balancing Component: Evaluates load distribution and initiates load balancing when significant imbalances arise. Load Migration Component: Identifies suitable controller-switch pairs for load balancing and migrates switches from overloaded (source) controllers to underutilized (target) controllers. It considers shortest path distances to ensure efficient data transfer. Link Reset Component: Resets controller-switch links involved in the migration process, ensuring seamless integration of switches with their respective controllers. SALB dynamically adjusts threshold values when highly loaded controllers cannot find suitable migration targets. This flexibility ensures effective load balancing under evolving network conditions. Evaluation results demonstrate that SALB is well-suited for large-scale real-time SDN applications compared to other algorithms.

Yi-Ren et. al. [18] addressed traffic engineering (TE) issues in SDN by proposing a reinforcement learning routing algorithm (RL-Routing) that utilizes an RL agent to predict future network behaviour and optimize routing paths. The RL agent interacts with the network by selecting actions (routing paths) based on real-time network state information and suggests improved routing paths between switches. The RL-Routing application consists of two primary modules: The Network Monitoring Module (NMM): Employs both passive and active network measurements to gather critical network device information, including link delay, throughput, and port speed. This data serves as the basis for state representation and reward computation. Action Translator Module (ATM): Converts the agent's selected action into a series of appropriate OpenFlow messages. These messages update the flow tables of switches when configuring new paths. To avoid Packet-In messages being sent to the controller, the ATM transmits these messages from the last switch of the path to the first switch. Finally, the old rules in the switches of the previous path are deleted.

RL-Routing has the potential to mitigate scalability issues in routing by automating path selection and reducing manual configuration and maintenance overhead. However, its effectiveness in addressing scalability challenges depends on factors such as network size and complexity, as well as the performance metrics being evaluated. The performance of RL-Routing was assessed on three well-known network topologies: Fat-tree, NSF Network (NSFNet), and Advanced Research Projects Agency Network (ARPANet). RL-Routing was compared against two widely used baseline solutions: Open Shortest Path First (OSPF) and Least Loaded routing algorithm (LL). The evaluation metrics included the reward function, a score calculated using network throughput and delay, and the utilization rate, calculated in the destination switch. The reward function can be adjusted to optimize either upward or downward network throughput.

Hong et al. [19] an assessing profit of prediction (APOP) scheme is proposed for achieving a load balancing in the control plane for multiple controllers. It relies on predicting the overloaded state and assessing the profitability. This scheme comprises three modules: monitor, load balancing trigger, and migration execution. The monitor module periodically collects statistical data on PACKET_IN messages received from managed switches per time slot and stores it in a database. The load balancing trigger module analyzes historical data to determine whether to initiate switch migration using Taylor's formula. To mitigate the detrimental impact of erroneous long-term flow predictions, APOP employs short-term prediction using Taylor's formula, which is computationally efficient and does not require training phases or long-term trend prediction. APOP's profit assessment mechanism then efficiently evaluates the short-term prediction results to minimize low-profit migrations. The migration execution module implements migration decisions made by the Profit Assessment Algorithm, selecting overloaded controllers, new master controllers, and switches to minimize loss. Through simulation results, APOP has demonstrated earlier migration execution, improved migration time, and better response time ensuring that the overloaded controllers successfully shift load before they catch their bottleneck.

Hamza et al. [20] proposed a Multiple Threshold Load Balance (MTLB) Switch Migration Scheme, to address the issue of load imbalance and controller overload in distributed Software-Defined Networking (SDN) environments. MTLB effectively categorizes the load into several progressive levels, serving as the foundation for initiating switch migration when load discrepancies arise between controllers. This dynamic approach dynamically adjusts threshold values based on the load status, ensuring efficient load balancing, and preventing performance bottlenecks. MTLB employs a trigger factor rather than periodic updates to synchronize load information among controllers, reducing unnecessary overhead. The scheme initially categorizes the load into appropriate threshold levels for synchronization and migration handling. Upon detecting a controller's load exceeding or approaching a threshold, it notifies other controllers to update their load information, effectively managing load synchronization and handling migration efficiently. The MTLB scheme comprises three stages: 1) Checking for Updates: Monitors controller load status and triggers synchronization when necessary. 2) Detecting Load Imbalance: Identifies load discrepancies between controllers and initiates migration if required. 3) Selecting Suitable Switch and Controller: Select the most appropriate switch for migration and identify the optimal target controller. The MTLB system architecture employs four status levels and three load thresholds to effectively manage controller load:

- Overload: Indicates that the controller has reached its full capacity, prompting immediate migration.
- Highly Loaded: Controllers in this status can still operate for a limited duration but require migration if other controllers are idle or in a normal state.
- Normal: Controllers equipped to handle unexpected scenarios and prioritize receiving switches when necessary.
- Idle: Controllers with the highest priority to receive switches, ensuring efficient resource utilization.

The MTLB scheme effectively addresses load imbalance and controller overload in distributed SDN environments, outperforming other schemes in terms of throughput, packet delay, migration cost, and communication overhead. Its effectiveness stems from its use of controller load status for efficient load information dissemination and its three-stage module design for seamless migration handling.

Chunlin et.al [21] introduced a novel model based on task latency and dynamic constraints, to address the challenges of communication latency between controllers and switches, as well as inter-controller communication issues resulting from link failures. Dynamic allocation of computational resources using the heuristic ant colony algorithm (HACA) [22] is employed to optimize controller placement and load balancing in distributed SDN networks. The model leverages two key aspects: 1) A dependable controller placement method that considers latency and load considerations, improving the load optimization multi-controller placement (LOCP) algorithm. 2) A resource allocation algorithm using HACA that takes into account task latency and reliability constraints.

The process for improving a multi-controller placement (LOCP) algorithm is described as follows:

Firstly, user devices establish connections to the edge computing layer employing network access points, such as wireless access points or base stations, in order to access services. Secondly, a multi-access edge computing (MEC) server [23], situated close to the base station, provides computing and storage resources, while also gathering and analyzing information from end devices. This helps in reducing data and improving the quality of network services. Thirdly, the MEC server establishes a connection with a local software-defined networking (SDN) controller through an OpenFlow switch, enabling efficient management of network traffic and resources. Fourthly, a global controller oversees the operations of the local SDN controller, ensuring that data matching and processing rules are updated through the exchange of operational status information.

Finally, the controllers communicate with each other using an east-west interface, thereby facilitating seamless coordination. The model adopts an edge computing and SDN-based control architecture, optimizing network performance and resource utilization, enhancing user experiences, and providing efficient network services. The controller placement problem is addressed by focusing on three primary performance metrics: controller-switch delay, inter-controller delay, and load balancing, while also considering network link connectivity. Jehad and Byeong-hee [24] developed a mathematical decision-making framework for selecting the optimal controller for Software-Defined Internet-of-Things (SD-IoT) based on its performance-enhancing features using the Analytical Network Decision-Making Process (ANDP) model [25]. This controller selection technique integrates qualitative and quantitative assessments of SD-IoT controllers. The authors identified ten relevant controller characteristics for the IoT environment, as listed in Table 1. They then used ANDP to calculate weights for each controller after applying the comparison matrix which is the outcome of all judgments of the controllers' supporting features and ranking them based on their feature sets. The controller with the highest weight was selected as the optimal controller for SD-IoT.

Table 1 List of features for SD-IoT performance evaluation [24]

Serial#	Notation	Name	Description
1	B1	OpenFlow-support	OpenFlow version1.0–1.5
2	B2	GUI	Web based or Python-based
3	B3	NB-API support	REST-API
4	B4	Clustering support	To ensure reliability and performance
5	B5	Openstack networking	Enabling different network technologies via quantum API
6	B6	Synchronization	State synchronization of the clusters
7	B7	Flow requests handling	The capability to handle the flow requests
8	B8	Scalability	Adoptability in the extended networks
9	B0	Platform support	Windows, Mac, Linux
10	B10	Efficient energy management	The ability to utilize energy efficiently

The evaluation of controller features is represented by a four-level scale, with G1 indicating extremely low support and G4 denoting very strong support. G2 indicates medium support, and G3 indicates strong support. The controller evaluation score ranges from G1 to G4. Next, the controllers are compared based on their features for SD-IoT. Finally, the controller weights are calculated. This model presents a novel controller selection approach for SD-IoT environments based on the Analytical Network Process (ANDP) model.

The proposed controller is evaluated in terms of delay, throughput, CPU utilization, and reliability. Jehad et.al. [26] presented ESCALB, a load-balancing scheme specifically designed for multi-domain Software-Defined Networking-enabled Internet of Things (SD-IoT) networks. The primary objective of ESCALB is to facilitate the efficient migration of switches to controllers with available resources in a dynamic manner. To achieve this, ESCALB employs a hierarchical control plane model comprising multiple domain controllers (DCs) and a global control (GC) plane. Within the GC plane, four sub-modules are implemented, including the Load Calculation Module (LCM) and the ANP Module (ANPM).

These modules monitor the load status by receiving information from the Distributed Control Plane (DCP) and rank the controllers based on factors such as CPU usage, Flow Requests Capacity (FRC), memory utilization, and the number of attached switches. The ANPM utilizes the Analytic Network Process (ANP) model, employing a mathematical procedure with a 9-point scale matrix to prioritize slave controllers in the DCP, where 1 indicates equal importance and 9 signifies extreme significance.

The Switches Migration Module (SMM) initiates the migration of switches to slave controllers if the load on the master controller exceeds a predefined threshold. Collaborating with the ANPM and SMM, the Flows Forwarding and Updating Module (FFUM) prioritizes slave controllers, facilitates switch migration, and forwards flow requests to controllers with higher weights. Overall, the GC plane aims to optimize network performance and resource utilization through ANP-based ranking, switch migration, and load-balancing mechanisms.

The effectiveness of ESCALB lies in its ability to select the most suitable controller intelligently and strategically for load distribution, thereby leading to improved performance within Software-Defined Networking (SDN) environments. Table 2 provides a comprehensive overview of the proposed methodologies, along with their respective advantages and disadvantages, as discerned through an exhaustive analysis of the related literature review.

Table 2 Summary of Related Works

Author & Reference	Proposed Methodology	Advantages	Disadvantages / Limitations
Jie et al. [12]	Effectively address load balancing in distributed SDN control planes with multiple overloaded controllers based on an assessment approach centered on the response times of overloaded controllers	Balances the load across multiple SDN controllers and concurrently executes distinct switch migration operations.	Does not accommodate the majority of load distribution patterns
André and Fernando [13].	The Rama controller framework is a promising approach to ensuring the fault tolerance of SDN controllers.	Perform better throughput, ensuring fault tolerance without necessitating modifications to switches or the OpenFlow protocol.	Network overhead due to increased network message exchanges. Need to evaluate the impact of network latency.
Jehad et al. [16]	Utilize the ANP model to optimize controller selection based on controller-supporting features.	The optimal selection of a controller using the ANP model leads to a reduction in topology discovery time and enhances throughput.	Modestly elevate the CPU utilization of the chosen controller
Madhukrishna et al. [17]	Distribute the load across multiple controllers through the migration of switches from source controllers to target controllers, particularly under conditions of high network traffic.	Preventing controller overload, reducing latency, and optimizing load balancing time and loss rate.	The SALB algorithm may not be suitable for situations requiring load balancing beyond the predefined maximum base threshold.
Yi-Ren et. al. [18]	RL-Routing utilizes Reinforcement Learning (RL) to forecast network behaviour in SDN, enabling efficient routing by suggesting optimal paths between switches.	Optimizing file transmission times by avoiding congested paths and minimizing packet retransmission.	Deploy RL-Routing in a real network environment and evaluate it on other topologies.
Hong et al. [19]	APOP leverages short-term flow prediction utilizing Taylor's formula to mitigate long-term prediction errors and profit assessment to avoid low-profit migrations.	Improve resource utilization and reduce switch migration-induced fluctuation during load balancing.	Other quality of service (QoS) metrics have not been considered.
Hamza et al. [20]	Employing a switch migration scheme to address load imbalance and prevent controller overload involves categorizing the load into multiple progressive tiers and dynamically adapting the threshold value.	A dynamic threshold value makes the algorithm more responsive to load changes and prevents unnecessary traffic migration.	The current load distribution scheme is reactive, triggering only when thresholds are exceeded, which can lead to spikes in controller load and performance degradation. the scheme does not consider the variability in flow processing times.

Chunlin et.al [21]	Calculate the optimal placement of multiple controllers based on the network topology, traffic load, and available computational resources.	The improvement of LOCP algorithm slightly outperforms in load balancing while ensuring a lower propagation delay and queuing delay.	To enhance the evaluation of MEC servers, incorporate additional metrics like network throughput, address critical aspects like security, mobile device motion trajectory, and task priority, and assess performance in large-scale networks.
Jehad and Byeong-hee [24]	A mathematical decision-making framework that utilizes the ANDP model to calculate the optimal controller based on its features, aiming to enhance the performance of the SD-IoT.	The optimal controller reduces delay, improves throughput, utilizes CPU efficiently, and recovers from link failures seamlessly.	Controller selection alone may not be enough for SD-IoT, as other important aspects such as security and energy efficiency need to be considered.
Jehad et.al. [26]	Monitor the control plane in real-time and acquire load information to assess based on CPU usage, Memory usage, flow requests and no. of switches then prioritize slave controllers, ensuring successful switch migration to controllers with idle resources.	Response time analysis results demonstrate ESCALB's effectiveness due to its superior slave controller selection mechanism, consistently outperforming alternatives with significantly lower end-to-end latency across various network topologies.	Failed to achieve optimal utilization of CPU resources.

3. THE PROPOSED FRAMEWORK MODEL

This paper proposes a new framework model of high availability represented in route traffic assuring fault-tolerance and load balancing in SDN based upon a real-time measurement of the controller's performance load metrics. Measured metrics are CPU load, memory load, network load, and Response time. This framework model aims to optimize load balancing by increasing throughput, reducing server response time and packet loss, and achieving reliability by providing redundant ordered controllers in the event of a failure. The proposed model is implemented as an SDN application using the REST API (Representational State Transfer API) as the northbound API [27]. The REST API collects real-time information about the network topology, devices, traffic, and performance. The proposed RATR framework aims to address the centralization challenge in SDN controllers by leveraging a smart cluster of controllers. It strives to provide a more efficient and effective solution to the centralization problem in SDN controller environments. This model aims to achieve enhanced throughput performance while minimizing the associated overhead, in contrast to conventional cluster setups.

3.1. Framework Design Overview

The proposed framework relies on the creation of a Smart Controllers Cluster (SCC), which owns and manages a group of controllers. The SCC has a virtual engine called the Virtual Cluster Engine (VCE), which controls all cluster events. The VCE executes an efficient centralized algorithm with a global knowledge of the real-time performance of all controllers in the cluster. This study presents two heuristic phases.

Phase 1: Plans the real-time performance monitoring measurements of cluster members of controllers, as shown in Algorithm 1.

Phase 2: Plans the voting election process to elect the leader controller and its associated vice controllers from all cluster members, based on the collected votes from Phase 1, as shown in Algorithm 2. The elected leader controller is responsible for managing route traffic flow control to network devices and hosts, delegated to it by the SCC. In order to ascertain the optimal metric factors that reflect the prevailing levels for each metric under investigation, individual performance tests were conducted for each measure over ten rounds spanning different time slots. Statistical data was meticulously recorded. The findings of this study indicate that each metric examined has a significant impact on improving network performance.

3.1.1. Virtual Cluster Engine (VCE)

The Virtual Cluster Engine (VCE) undertakes various functions as outlined below:

- It verifies the accessibility of all cluster members by transmitting an ICMP packet to each of them.

- It monitors the performance of all active cluster members, ensuring their continued operational status.
- It facilitates the election of a leader controller, along with the corresponding 1st and 2nd associated vice controllers for the leader position.
- At regular intervals referred to as time slots, the SCC initiates the recurring voting election process.

3.1.2. Voting Election Process

- Determine the count of alive (active) cluster members and assign them the variable (n). Subsequently, the evaluation proceeds to assign votes to each controller.
- SCC retains specifically the votes designated for each controller, considering a total of three controllers within the scope of this study.
- Collect the performance load metrics for each controller in the cluster for every time slot, including CPU utilization, memory utilization, network traffic, and response time.
- For each designated time slot, encompassing a set of five records, compute the respective averages and subsequently arrange them in ascending order to identify the optimal values.
- To evaluate the performance of the controllers, a grading mechanism is implemented for each performance metric. The controllers are compared based on their considered load metrics. The evaluation of controller load metrics is represented by an **n**-level voting system, with "n" indicating the highest grade, "n-1" denoting the second highest grade, "n-2" implying the third highest grade and so on until the vote equals "1" which denotes the lowest grade. In this study, the considered cluster members are three controllers, so the highest grade has "3" votes, the next one in order has "2" votes, and the lowest grade has "1" votes.
- The optimal ranking of ordered region cluster members is determined through the calculation of Grade Average Votes (GAV) using the gained metric factor. This process involves several steps: Firstly, the grade for each cluster member is computed for every performance metric using the previous grading mechanism. Secondly, the corresponding adaptive metric factor is multiplied by each grade. Thirdly, the weighted grades for each cluster member are averaged, resulting in the computation of the GAV for each member. Fourthly, the cluster members are ranked in descending order based on their GAV, thereby establishing the optimal ranking of the ordered region cluster members ex: "Region123," signifies that "C1" assumes the role of a leader, "C2" serves as the first associated vice to the leader, and "C3" functions as the second associated vice to the leader.
- Upon electing a leader controller, the Smart Controllers Cluster (SCC) routes all traffic to the leader controller. In the event of a leader controller failure, the SCC is prepared to reroute all traffic to the first associated vice controller. The proposed RATR architecture is illustrated in. (Figure 3).

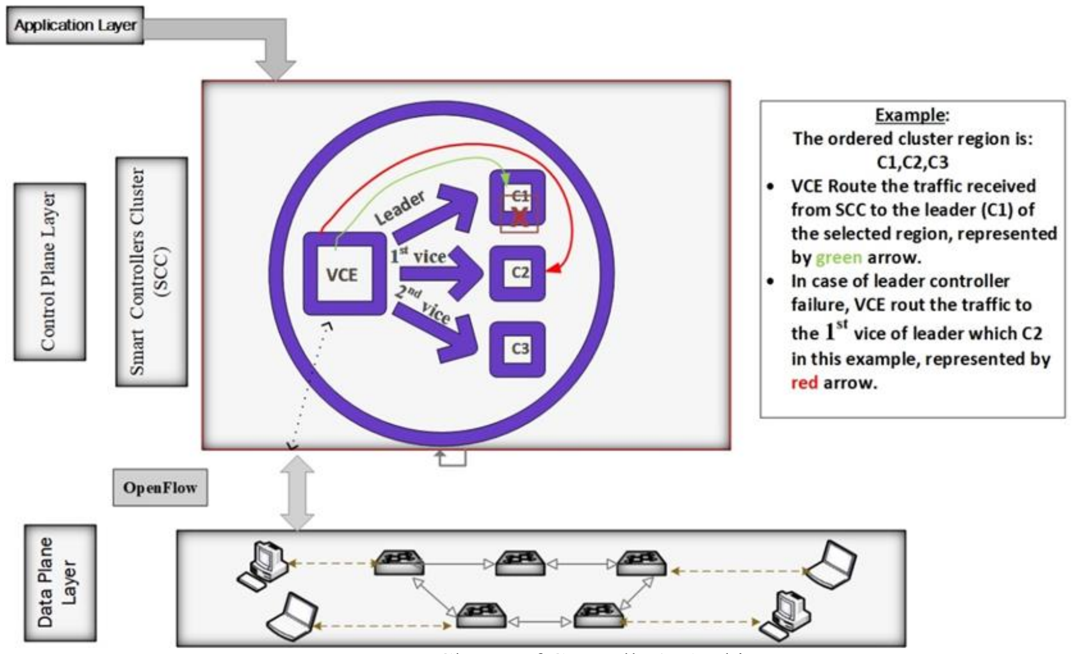
Application Layer
Control Plane Layer
Smart Controllers Cluster (SCC)
VCE
Leader
1st vice
2nd vice
C1
C2
C3
OpenFlow
Data Plane Layer
Example:
The ordered cluster region is: C1,C2,C3
VCE Route the traffic received from SCC to the leader (C1) of the selected region, represented by green arrow.
In case of leader controller failure, VCE rout the traffic to the 1st vice of leader which C2 in this example, represented by red arrow.

Figure 3 RATR Cluster of Controller's Architecture

As proof of concept, simulation and discussion of different scenarios will also be provided. Table 3 describes the notation used in RATR algorithms.

```
Controller Health (Cn)
Input: Controller set (N), Time_slot= TJ
Output entry for every controller
    For each n in N do
    $T_{J\ ==}$ 0
    Start Real-time monitoring application.
    While $T_J >= 0$
        "Record measurement metrics (votes)"
        insert Cn [CU, MU, NT, RT, TJ]
        Return Cn $_{(TJ)}$ [$CU_n$, $MU_n$ ,$NT_n$ ,$RT_n$]
        TJ == TJ +1
        End
    End
```

Algorithm 1 Real-Time Performance Monitoring Measurement Procedure

```
Cluster (1st, 2nd, 3rd, ……, n)
Input: Metric_list: M = Cn $_{(TJ)}$ [$CU_n$, $MU_n$ ,$NT_n$ ,$RT_n$], MF
Output: Region Cluster (1st, 2nd, 3rd, ……, n) = {$L_0$, $L_1$, $L_2$, …., $L_N$}
Calculate average Cn $_{(TJ)}$ [$CU_n$, $MU_n$ ,$NT_n$ $RT_n$]
For each metric in the Metric _list
Sort_min [Metric _list, TJ] {C1, C2, …. Cn}
Grade Average Votes = GAV
    "Grading metrics (votes) = GV"
        1st minimum value = n
        2nd minimum value = n-1
        3rd minimum value = n-2
        …..
        …..
        Apply metric factor to calculate total GV.
        $MG_{IM}$ (t) = $GV_{im}$ * $MF_m$
Calculate Grade Average Votes (GAV)
    $GAV\ (t) = \sum_{\substack{i=0 \\ m \in M}}^{n} MG_{im}\ (t) \Big/ \sum_{m \in M} MF_m$
Sort Max (GAV)
$L_0$ = 1st Highest GAV = Leader controller
$L_1$ = 2nd Highest GAV= 1st vice Leader controller
$L_2$ = 3rd Highest GAV = 2nd vice Leader controller
• Get the token value needed for calling the elected region.
Run region Curl function that is responsible for routing the traffic to the leader controller according to the resulting order.
```

Algorithm 2 Process for Electing the Leader and Vice Controllers Based on Votes

Table 3 Proposed RATR Framework Notations

Notation	Description
CU	CPU Utilization
MU	Memory Utilization
NT	Network Traffic
RT	Response Time
M	Metric list
MF	Metric Factor
MG	Metric Grade
GAV	Grade Average Votes

4. EXPERIMENTAL RESULTS AND DISCUSSION

To validate the effectiveness of the proposed RATR. The conducted simulation and emulation experiments were performed using a MININET emulator [28] with three identical virtual machines for HPE VAN SDN controllers [29] and that is through a workstation, Intel Core i7, 2.70 GHz with 16GB RAM. The proposed RATR is implemented as an SDN application. Utilizing the REST API (Representable State Transfer API) [27] is considered to be the northbound API. The REST API gathers needed real-time performance statistics on the controllers of the cluster and intended topology resulting from monitoring the system resources and analyzing the usage patterns. In addition to current throughput traffic and latency statistics using the Iperf tool which is an open-source tool that you can use to measure network throughput, packet loss, and delay [30]. It can generate and analyze TCP and UDP traffic, making it useful for testing router performance. Iperf is a robust tool that offers valuable insights into network performance, making it highly favoured among network administrators and engineers for testing and optimizing network capabilities. However, utilizing Iperf effectively and correctly interpreting the results often necessitates a certain level of technical expertise.

This section presents a thorough analysis of key performance metrics—specifically, average throughput, average latency, and packet loss percentage—across the considered algorithms. The algorithms under scrutiny encompass Round Robin [31], SMCLBRT [12], ESCALB [26], and the Proposed RATR. The evaluation is integral to understanding the effectiveness and limitations of these algorithms within a simulated environment.

Average throughput is defined as the sum of successfully transmitted data per time unit. It is controlled by available bandwidth, usually measured by the unit of bits per second (bps). Average latency is defined as the average time employed for a data packet to be delivered from the source node to the destination node, also known as a delay. Latency is usually measured in milliseconds (ms) and is an important factor in determining the performance of a network. Average packet loss is defined as the percentage of a lost data packet concerning packets that were sent [32] [33].

Experimental results were implemented using a simulation model of linear Topology [28] network and generating random HTTP traffic as client/server. Linear Topology-A consists of 4 switches and 4 hosts, linear Topology-B consists of 6 switches and 6 hosts, linear Topology-C consists of 8 switches and 8 hosts, and linear Topology-D consists of 10 switches and 10 hosts.

4.1. Linear Topology-A

Table 4 delineates the selection of the primary controller and associated subordinates in the context of the Round Robin, SMCLBRT, ESCALB, and the proposed RATR algorithms.

Table 4 Controllers Selected Based on Evaluated Algorithms under Linear Topology-A

Controller election	TS1				TS2				TS3				TS4				TS5			
	Round Robin	SMCLBRT	ESCALB	Proposed RATR	Round Robin	SMCLBRT	ESCALB	Proposed RATR	Round Robin	SMCLBRT	ESCALB	Proposed RATR	Round Robin	SMCLBRT	ESCALB	Proposed RATR	Round Robin	SMCLBRT	ESCALB	Proposed RATR
Leader	C1	C3	C1	C1	C2	C1	C1	C1	C3	C2	C1	C2	C1	C3	C3	C2	C2	C1	C1	C3
1st vice	■	C2	C3	C3	■	C3	C3	C3	■	C1	C3	C3	■	C2	C2	C3	■	C2	C3	C2
2nd vice	■	C1	C2	C1	■	C2	C2	C2	■	C3	C2	C1	■	C1	C1	C1	■	C3	C2	C1

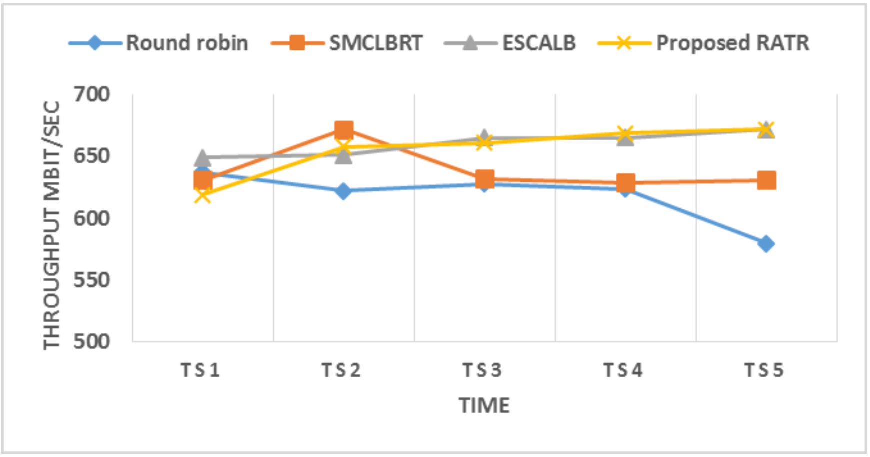

Figure 4 Average Throughput vs Time in the case of Liner Topology-A

Figures 4, 5, and 6 represent the simulation results of the considered algorithms. In each Figure, the time slots are considered versus the average throughput, latency, and packet loss percentage respectively. An analysis of Figure 4 reveals that the average throughput of the proposed RATR and ESCALB algorithms consistently increased across the examined time slots. In contrast, the throughput of the Round Robin and SMCLBRT algorithms exhibited fluctuations, increasing and decreasing during the evaluated time periods. The proposed RATR demonstrated performance closely resembling that of ESCALB, notably surpassing ESCALB in terms of throughput during time slots TS2 and TS4. This upward trajectory in throughput observed in the proposed RATR algorithm suggests an enhancement in network performance, a direct consequence of the effective electoral process involving the leader controller and their associated vices.

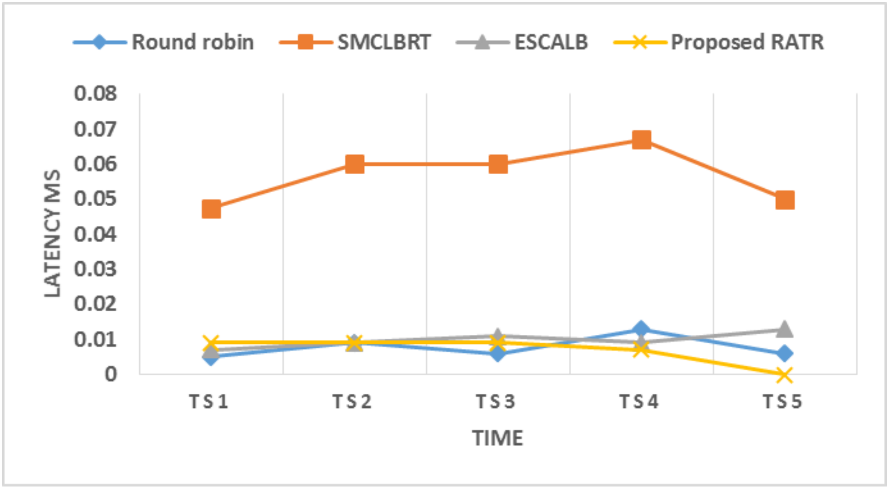

Figure 5 Average Latency vs Time in the Case of Liner Topology-A

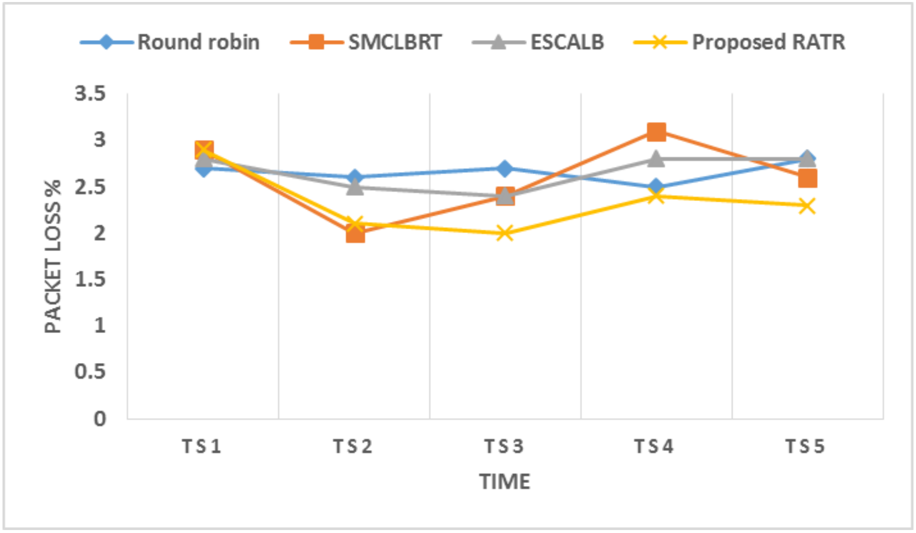

Figure 6 Average Packet Loss vs Time in the Case of Liner Topology-A

The Round Robin algorithm demonstrates fluctuating performance across various time slots; however, its relatively lower throughput implies that it may not be the most optimal choice for maximizing network capacity. SMCLBRT demonstrated competitive throughput, but its performance inconsistencies across time slots highlight its limitations. ESCALB outperformed the other algorithms in terms of throughput, attributable to its efficient traffic load balancing capabilities. The proposed RATR algorithm, although slightly inferior in terms of throughput, presents promising results considering its novel approach.

Figure 5 illustrates the average latency measurements obtained for each algorithm across the five evaluated time slots. The Round Robin algorithm consistently yields the lowest latency values, ranging from 0.005 ms to 0.013 ms. SMCLBRT exhibits marginally higher latency values, ranging from 0.0475 ms to 0.067 ms. ESCALB and the proposed RATR algorithms demonstrate comparable latency performance, with ESCALB ranging from 0.007 ms to 0.013 ms and the proposed RATR ranging from 0.007 ms to 0.009 ms.

The Round Robin algorithm achieves the lowest latency due to its straightforward scheduling approach, which ensures an equitable distribution of packets. However, this algorithm's susceptibility to increased latency during heavy network congestion remains a concern. ESCALB and the proposed RATR algorithms consistently exhibit similar latency values, indicating their ability to maintain low latency even under fluctuating network conditions. Figure 6 illustrates that the proposed RATR algorithm consistently exhibits the lowest average packet loss percentage, decreasing across the examined time slots except for TS4 and TS5. Nevertheless, even during these two time slots, the proposed RATR algorithm maintains the smallest packet loss values. The proposed RATR algorithm's competitive packet loss performance underscores its ability to efficiently handle network traffic with minimal data loss.

The evaluation of the four considered algorithms in the case of Linear Topology-A, reveals their respective advantages and limitations. Round Robin provides fairness in packet distribution but lacks efficiency in terms of throughput. SMCLBRT demonstrates competitive throughput but exhibits higher latency and packet loss. ESCALB outperforms other algorithms in terms of throughput and maintains low latency and packet loss. The Proposed RATR algorithm presents a novel approach with promising results in terms of latency and packet loss, although it slightly lags in throughput.

4.2. Linear Topology-B

Table 5 delineates the selection of the primary controller and associated subordinates in the context of the Round Robin, SMCLBRT, ESCALB, and the proposed RATR algorithms. Figures 7, 8, and 9 represent the simulation results of the considered algorithms. In each Figure, the time slots are considered versus the average throughput, latency, and packet loss percentage respectively.

Table 5 Controllers Selected Based on Evaluated Algorithms under Linear Topology-B

Controller election	TS1				TS2				TS3				TS4				TS5			
	Round Robin	SMCLBRT	ESCALB	Proposed RATR	Round Robin	SMCLBRT	ESCALB	Proposed RATR	Round Robin	SMCLBRT	ESCALB	Proposed RATR	Round Robin	SMCLBRT	ESCALB	Proposed RATR	Round Robin	SMCLBRT	ESCALB	Proposed RATR
Leader	C1	C2	C1	C2	C2	C3	C1	C3	C3	C3	C2	C3	C1	C3	C3	C3	C2	C2	C1	C2
1st vice	■	C1	C3	C3	■	C1	C3	C1	■	C1	C1	C2	■	C1	C2	C2	■	C1	C3	C3
2nd vice	■	C3	C2	C1	■	C2	C2	C2	■	C2	C3	C1	■	C2	C1	C1	■	C3	C2	C1

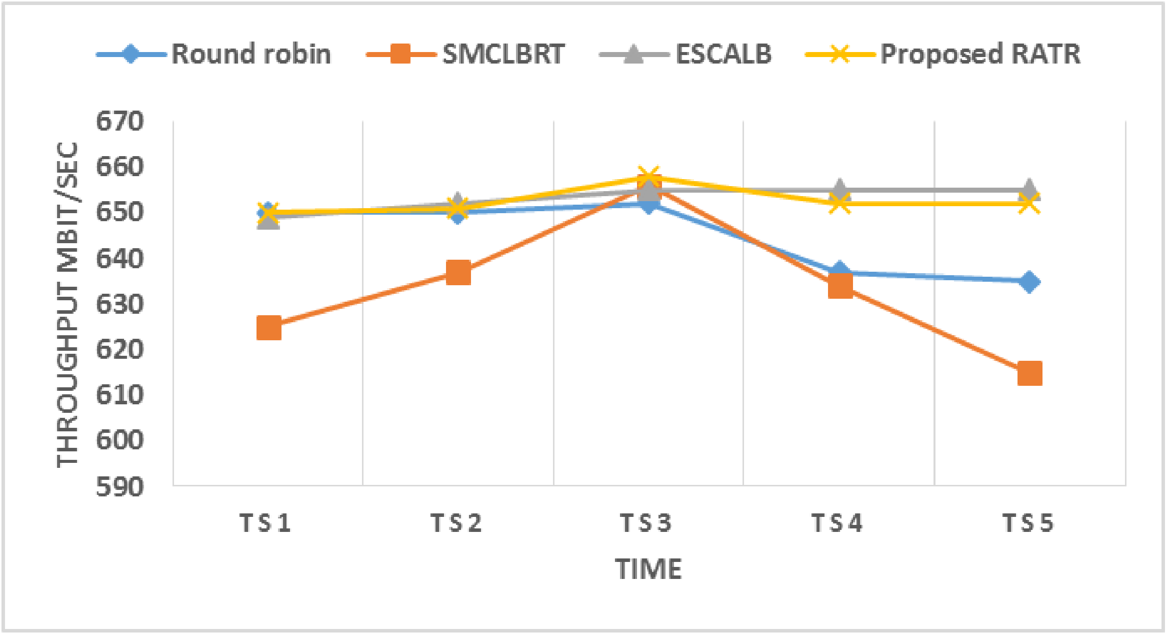

Figure 7 Average Throughput vs Time in the Case of Liner Topology-B

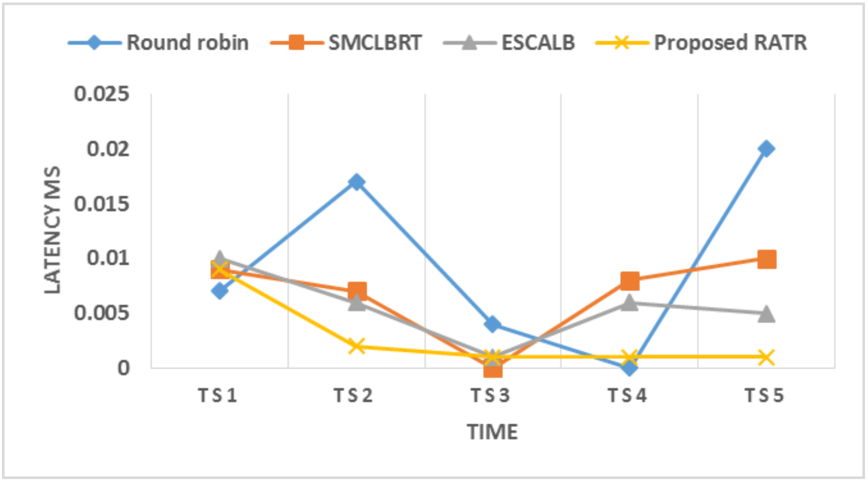

Figure 8 Average Latency vs Time in the Case of Liner Topology-B

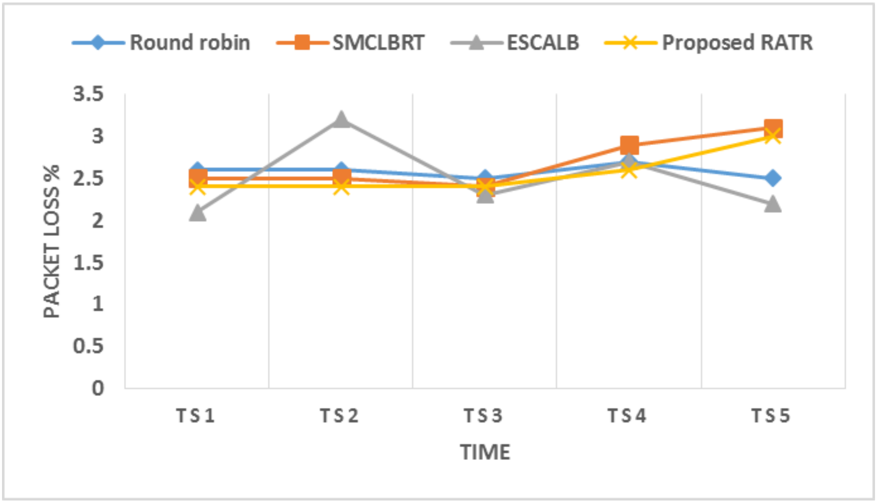

Figure 9 Average Packet Loss vs Time in the Case of Liner Topology-B

Figure 7 clearly reveals the superior average throughput attained by the proposed RATR algorithm in comparison to Round Robin and SMCLBRT. Remarkably, RATR's throughput exhibits a close resemblance to that of ESCALB, with ESCALB demonstrating a marginal advantage during time slots TS4 and TS5. This observation highlights the proposed RATR's effectiveness in managing network traffic and achieving high throughput, comparable to the best-performing algorithm.

The Round Robin algorithm consistently delivers stable performance across the examined time slots, ensuring fair packet distribution. However, its relatively lower throughput suggests that it may not be the optimal choice for maximizing network capacity. SMCLBRT demonstrates competitive throughput; however, its performance inconsistencies across time slots pose a limitation. ESCALB outperforms the other algorithms in terms of throughput, attributable to its efficient load-balancing mechanism. The proposed RATR algorithm achieves comparable throughput, indicating its potential as a viable alternative.

Examining Figure 8, the average latency measurements of the proposed RATR exhibit stability within the range of 0.001 ms to 0.009 ms that decreased across the specified time slots. In contrast, the Round Robin algorithm displays more significant fluctuations, with average latency peaking at 0.017 ms during TS2 and reaching 0.02 ms at TS5. While SMCLBRT maintains an average latency range between 0 ms and 0.01 ms, ESCALB's average latency falls between 0.001 ms and 0.01 ms.

The present findings demonstrate that the Round Robin algorithm experiences fluctuating average latency, making it susceptible to increased latency during periods of heavy network congestion. In contrast, the SMCLBRT algorithm exhibits lower latency, which can be attributed to its optimized scheduling mechanism. The outcomes further reveal that the Proposed RATR achieves a slightly lower average latency compared to both SMCLBRT and ESCALB, emphasizing its adeptness in handling time-sensitive data packets efficiently.

Moreover, as depicted in Figure 9, the proposed RATR demonstrates the most minimal average packet loss percentage, with a marginal increase observed solely in the final time slot in comparison to the Round Robin algorithm. The Proposed RATR algorithm attains noteworthy competitiveness in packet loss percentages, signifying its proficiency in managing network traffic with minimal data loss.

An assessment of the four considered algorithms under the Linear Topology-B configuration reveals their distinct strengths and shortcomings. Round Robin ensures fairness in packet distribution but may not fully utilize network capacity. SMCLBRT exhibits competitive throughput, lower latency, and minimized packet loss, making it a promising option. ESCALB emerges as the top performer in terms of throughput, maintaining consistently low latency and stable packet loss. The proposed RATR algorithm demonstrates comparable performance to ESCALB, establishing its potential as an alternative with improved latency and packet loss characteristics. The proposed RATR's adaptive mechanism effectively balances network traffic and prioritizes critical data packets, resulting in stable average latency performance. This conclusion is supported by the evaluation results, which consistently demonstrate the proposed RATR's ability to maintain low latency and stable packet loss across the examined time slots.

4.3. Linear Topology-C

Table 6 presents the selection of the leader controller and their vices for the Round Robin, SMCLBRT, ESCALB, and the proposed RATR algorithms evaluated under Linear Topology-C. Figures 10, 11, and 12 represent the simulation results of the considered algorithms. In each Figure, the time slots are considered versus the average throughput, latency, and packet loss percentage respectively.

Table 6 Controllers Selected Based on Evaluated Algorithms under Linear Topology-C

Controller election	TS1				TS2				TS3				TS4				TS5			
	Round Robin	SMCLBRT	ESCALB	Proposed RATR	Round Robin	SMCLBRT	ESCALB	Proposed RATR	Round Robin	SMCLBRT	ESCALB	Proposed RATR	Round Robin	SMCLBRT	ESCALB	Proposed RATR	Round Robin	SMCLBRT	ESCALB	Proposed RATR
Leader	C1	C3	C1	C3	C2	C3	C2	C2	C3	C1	C2	C2	C1	C1	C3	C1	C2	C2	C3	C2
1st vice	■	C1	C2	C1	■	C2	C3	C1	■	C3	C3	C1	■	C3	C1	C2	■	C3	C1	C1
2nd vice	■	C2	C3	C2	■	C1	C1	C3	■	C2	C1	C3	■	C2	C2	C3	■	C1	C2	C3

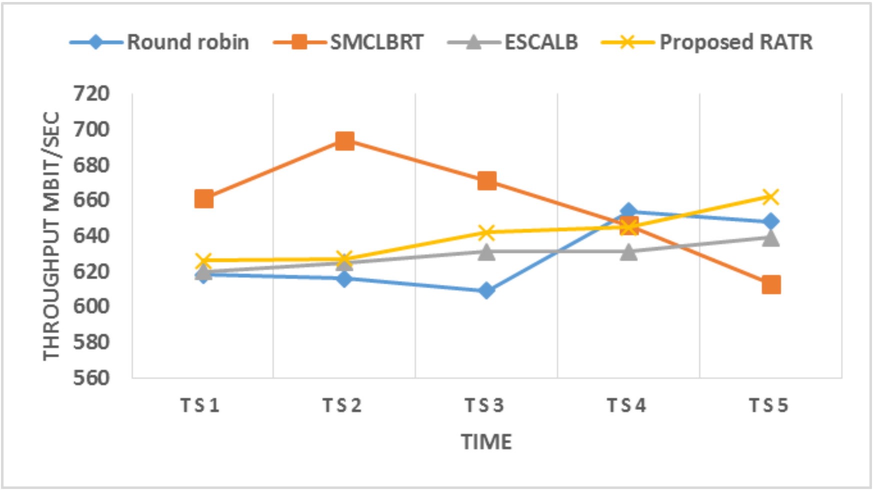

Figure 10 Average Throughput vs Time in the Case of Liner Topology-C

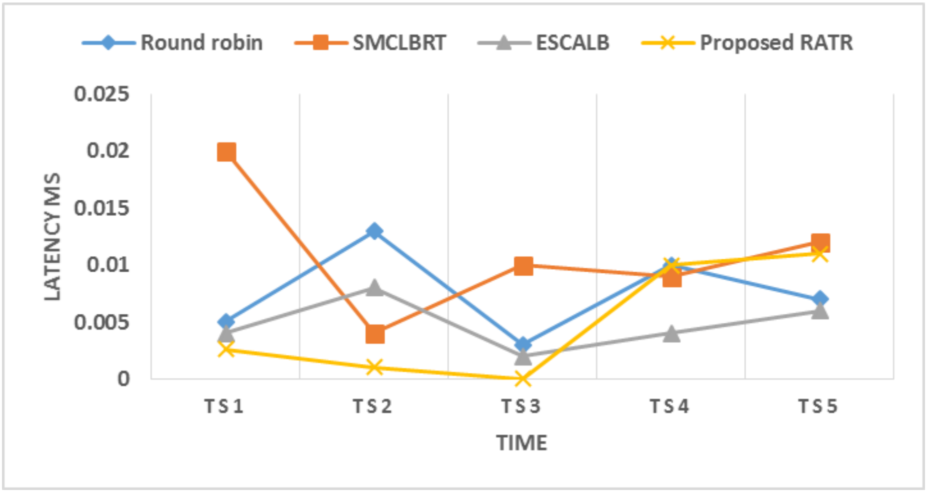

Figure 11 Average Latency vs Time in the Case of Liner Topology-C

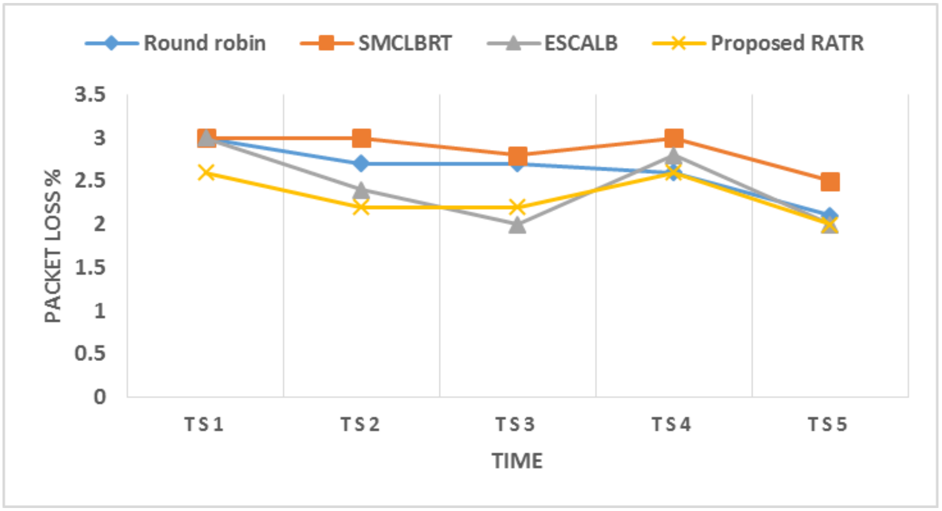

Figure 12 Average Packet Loss vs Time in the Case of Liner Topology-C

Figure 10 clearly demonstrates that the average throughput of the proposed RATR and ESCALB algorithms exhibits a consistent upward trend across the examined time slots, while the Round Robin and SMCLBRT algorithms exhibit fluctuating throughput, alternating between increases and decreases. The proposed RATR algorithm consistently outperforms ESCALB in terms of average throughput across all evaluated time slots. This consistent increase in throughput observed in the proposed RATR algorithm suggests a significant enhancement in network performance, attributable to the effectiveness of its adaptive mechanism in optimizing traffic flow. The Round Robin algorithm demonstrates inconsistent performance across the time slots, highlighting its limitations in maximizing network capacity due to its simplistic scheduling approach. SMCLBRT exhibits competitive throughput during the first two time slots; however, its declining throughput in subsequent time slots indicates its inability to effectively handle high-traffic scenarios. ESCALB achieves comparable throughput to Round Robin, while the proposed RATR algorithm consistently delivers superior throughput, establishing its potential as a viable alternative. The proposed RATR algorithm's superior performance can be directly attributed to its effective election process for selecting the leader and their associated vices. The successful collaboration and contributions of the elected individuals lead to an overall improvement in network performance, resulting in increased throughput and enhanced efficiency. Figure 11 demonstrates the proposed RATR algorithm's superior average latency performance compared to the Round Robin, SMCLBRT, and ESCALB algorithms across the first three time slots. The proposed RATR algorithm maintains consistently low average latency throughout the evaluated time slots, highlighting its ability to facilitate efficient communication with minimal delays. This underscores the RATR algorithm's effectiveness in minimizing latency and enhancing overall system responsiveness. Furthermore, in terms of packet loss, Figure 12 illustrates that the proposed RATR algorithm consistently exhibits the lowest average packet loss percentage across the observed time slots compared to the alternative algorithms. However, it is worth noting that in TS3, ESCALB records the lowest packet loss value. This nuanced result suggests that while RATR generally excels in mitigating packet loss, there are specific instances where ESCALB exhibits superior performance. The Round Robin and SMCLBRT algorithms maintain relatively higher packet loss percentages, indicating potential limitations in their ability to handle network congestion effectively. ESCALB exhibits lower packet loss, demonstrating its effectiveness in reducing dropped packets. The proposed RATR algorithm achieves competitive packet loss percentages, indicating its capability to handle traffic with minimal loss. The evaluation of the four considered algorithms under the Linear Topology-C configuration reveals their distinct strengths and weaknesses. Round Robin ensures fairness in packet distribution but may not fully utilize the network capacity, resulting in higher latency and packet loss. SMCLBRT demonstrates superior throughput but also incurs higher latency and packet loss. ESCALB achieves a comparable throughput to Round Robin, with lower latency and moderate packet loss. The proposed RATR algorithm showcases promising results, indicating its potential as an alternative with improved latency and packet loss characteristics.

4.4. Linear Topology-D

Table 7 illustrates the selection of the leader controller and their vices for the Round Robin, SMCLBRT, ESCALB, and the proposed RATR algorithms evaluated under Linear Topology-D. Figures 13, 14, and 15 represent the simulation results of the considered algorithms. In each Figure, the time slots are considered versus the average throughput, latency, and packet loss percentage respectively.

Table 7 Controllers Selected Based on Evaluated Algorithms under Linear Topology-D

Controller election	TS1				TS2				TS3				TS4				TS5			
	Round Robin	SMCLBRT	ESCALB	Proposed RATR	Round Robin	SMCLBRT	ESCALB	Proposed RATR	Round Robin	SMCLBRT	ESCALB	Proposed RATR	Round Robin	SMCLBRT	ESCALB	Proposed RATR	Round Robin	SMCLBRT	ESCALB	Proposed RATR
Leader	C1	C3	C1	C1	C2	C3	C1	C3	C3	C3	C1	C3	C1	C1	C2	C3	C2	C1	C1	C3
1st vice	■	C2	C2	C3	■	C1	C3	C2	■	C1	C2	C2	■	C3	C1	C2	■	C3	C3	C2
2nd vice	■	C1	C3	C1	■	C2	C2	C1	■	C2	C3	C1	■	C2	C3	C1	■	C2	C2	C1

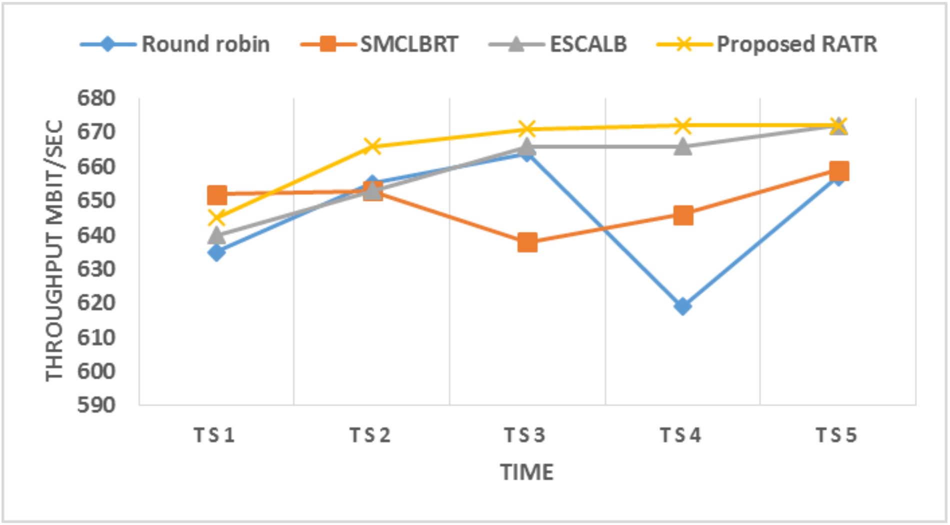

Figure 13 Average Throughput vs Time in the Case of Liner Topology-D

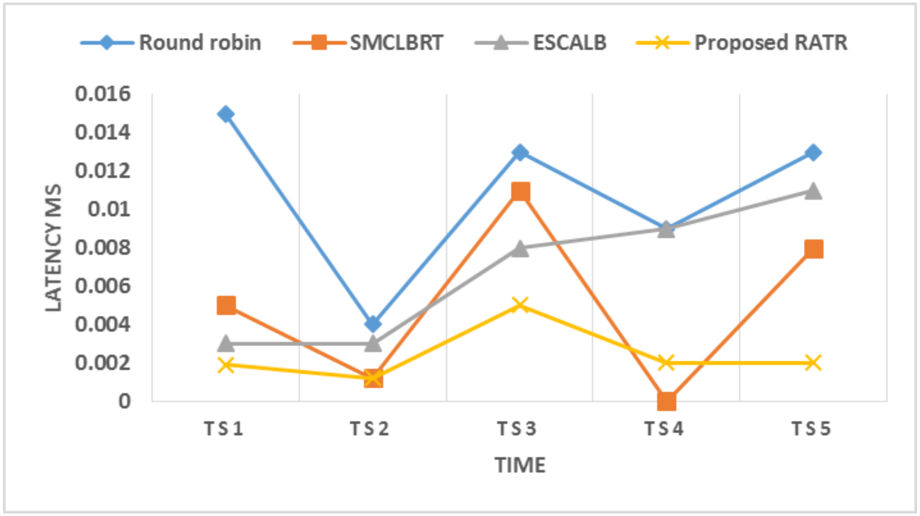

Figure 14 Average Latency vs Time in the Case of Liner Topology-D

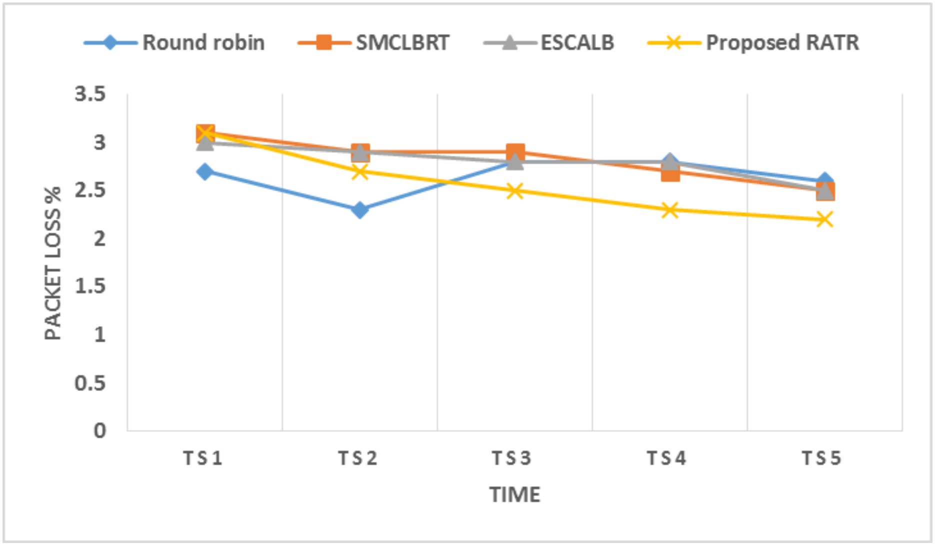

Figure 15 Average Packet Loss vs Time in the Case of Liner Topology-D

Figure 13 demonstrates the proposed RATR algorithm's superior average throughput compared to all other considered algorithms. The proposed RATR and ESCALB algorithms consistently exhibit an upward trend in throughput across the examined time slots, while the Round Robin and SMCLBRT algorithms exhibit fluctuating throughput, characterized by alternating increases and decreases. These fluctuations in performance highlight the limitations of Round Robin and SMCLBRT in effectively managing higher traffic loads. Figure 14 illustrates the Proposed RATR's remarkable consistency in maintaining an average latency within the impressive range of 0.0019 ms to 0.002 ms across all assessed time slots. In stark contrast, SMCLBRT exhibits significant latency variability, reaching peaks of 0.011 ms in TS3 and 0.008 ms in TS5. ESCALB exhibits an escalating trend in average latency across time slots, particularly evident from TS2 to TS5. Unlike both Round Robin and SMCLBRT, which display latency fluctuations across diverse test scenarios, the Proposed RATR maintains a consistently low average latency throughout the scrutinized time slots. Compared to the aforementioned algorithms, it exhibits noteworthy reductions in latency, particularly evident in the final two time slots of the evaluation. Figure 15 highlights the Proposed RATR's superior performance in terms of average packet loss percentage, showcasing a decreasing trend across all scrutinized time slots compared to the Round Robin and SMCLBRT algorithms. This exceptional performance can be directly attributed to the Proposed RATR algorithm's effective election process for selecting the leader and their associated roles. The successful collaboration and contributions of the elected individuals lead to an overall enhancement in network performance, resulting in consistently increased throughput and improved efficiency. The evaluation of the four considered algorithms under the Linear Topology-D reveals their distinct strengths and weaknesses. Round Robin, while ensuring equitable packet distribution, falls short in terms of throughput, exhibiting higher latency and moderate packet loss. SMCLBRT, while demonstrating improved throughput, also experiences higher latency and packet loss. ESCALB achieves a comparable throughput to SMCLBRT, maintaining consistently low latency and moderate packet loss. The proposed RATR algorithm's superior performance can be directly attributed to its effective election process for selecting the leader and their associated vices. The successful collaboration and contributions of the elected individuals lead to an overall improvement in network performance, resulting in consistently increased throughput, lower average latency, and competitive packet loss percentages and enhanced efficiency.

In conclusion, the findings of this study underscore RATR's remarkable ability to enhance throughput, minimize latency, and curtail packet loss, while acknowledging its minor limitations in specific scenarios, particularly linear topology-A and linear topology-B. Despite ESCALB's marginally superior throughput in linear topology-A and linear topology-B, RATR exhibits exceptional performance in linear topology-C and linear topology-D across all evaluated metrics. This superiority highlights RATR's adeptness in handling high-traffic scenarios. In contrast to Round Robin's fluctuating latency across different test scenarios, RATR maintains consistently low latency, demonstrating its reliability in diverse network sizes. The dynamic leader and their associated vices election process, coupled with the collaborative efforts of the elected individuals, underpin RATR's overall improvement in network performance. RATR's efficient resource management and scheduling strategies effectively minimize packet collisions and ensure seamless data transmission, resulting in superior performance across all evaluated metrics.

5. CONCLUSION AND FUTURE WORK

This paper concentrates on examining the issue of centralized controllers and the resultant load in the context of employing multiple controllers to achieve load balancing. This paper presents a comprehensive performance evaluation of four traffic scheduling algorithms in SDN: Round Robin, SMCLBRT, ESCALB, and Proposed RATR. The performance variations observed among the investigated algorithms can be attributed to differences in their respective measurement criteria. The Round Robin algorithm prioritizes an equitable distribution of packets for leader selection. In contrast, the SMCLBRT algorithm considers the response times of overloaded controllers when computing loads. The ESCALB algorithm incorporates CPU usage, memory usage, flow requests, and the number of switches into its load calculations. The proposed RATR algorithm, on the other hand, relies on real-time measurements of CPU load, memory load, network load, and response time of controllers within the cluster. The study provides evidence that the mentioned load metrics have a direct impact on network performance. The simulation findings indicate that the proposed RATR is more effective than the other compared algorithms. The proposed RATR achieves better results in average throughput, delay, and packet loss in most of the studied time slots. In general, the impact of the controller's load with compared algorithms on the performance parameters shows the success of the proposed RATR electing the better order of the members of cluster according to the examined load metrics. In future work, the integration between some load-balancing algorithms and traffic routed for clusters with members should be addressed to enhance the performance of the RATR and to study more load metrics as uptime of controller that is a critical metric, irrespective of its direct impact on performance.

REFERENCES

[1] H. Facchini, R. B. S. Perez, B. Roberti and R. Azcarate, "Experimental performance contrast between SDN and traditional networks," in 2021 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON), Valparaíso, Chile, 2021.

[2] H. Leqing, "How to Realize the Smooth Transition From Traditional Network Architecture to SDN," in 2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE), Harbin, China, 2020.

[3] Alghamdi, D. Paul and E. Sadgrove, "A RESTful Northbound Interface for Applications in Software Defined Networks," in 17th International Conference on Web Information Systems and Technologies (WEBIST 2021), 2021.

[4] Y. Li, D. Zhang, J. Taheri and K. Li, "SDN components and OpenFlow," in Big Data and Software Defined Networks, IET Digital Library, 2018, pp. 49-67.

[5] S. AZODOLMOLKY and O. Coker, Software-Defined Networking with OpenFlow, Second ed., Packt, 2017, pp. 248, ISNB:9781783984282.

[6] C. Lin, J. HU, G. Li and L. Cui, "A Review on the Architecture of Software Defined Network," Chinese Journal of Electronics, vol. 27, no. 6, pp. 1111-1117, 2018.

[7] Y. Zhanga, L. Cui, W. Wangc and Y. Zhanga, "A survey on software defined networking with multiple controllers," Journal of Network and Computer Applications, vol. 103, pp. 101-118, 2018.

[8] A. H. Alhilali and A. Montazerolghaem, "Artificial intelligence based load balancing in SDN: A comprehensive survey," Internet of Things, vol. 22, 2023.

[9] B. Yamansavascilar, A. C. Baktir, A. Ozgovdeb and C. Ersoya, "Fault tolerance in SDN data plane considering network and application based metrics," Journal of Network and Computer Applications, sciencedirect, vol. 170, 2020.

[10] N. Joshi and D. and Gupta, "A Comparative Study on Load Balancing Algorithms in Software Defined Networking," in Ubiquitous Communications and Network Computing, Springer International Publishing, 2019, pp. 142--150.

[11] J. Ali, R. H. Jhaveri, M. Alswailim and B.-h. Roh, "ESCALB: An effective slave controller allocation-based load balancing scheme for

multi-domain SDN-enabled-IoT networks," Journal of King Saud University – Computer and Information Sciences, vol. 35, no. 6, 2023.
[12] J. Cui, Q. Lu, H. Zhong, M. Tian and a. L. Liu, "A Load-Balancing Mechanism for Distributed SDN Control Plane Using Response Time," IEEE Transactions on Network and Service Management, vol. 15, no. 4, pp. 1197-1206, 2018.
[13] A. Mantas and F. M. V. Ramos, "Rama: Controller Fault Tolerance in Software-Defined Networking Made Practical," 2019.
[14] H. Wang, L. X. H. Zhu, W. Xie and G. Lu, "Modeling and Verifying OpenFlow Scheduled Bundle Mechanism Using CSP," in 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, Japan, 2018.
[15] N. Katta, H. Zhang, M. Freedman and J. Rexford, "Ravana: Controller fault-tolerance software-defined networking," in In Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research, 2015.
[16] J. AliID, B.-h. Roh and S. LeeID, "QoS improvement with an optimum controller selection for software-defined networks," PLoS ONE, pp. 1-37, 2019.
[17] M. Priyadarsini, J. C. Mukherjee, P. B. S. Kumar, A. H. M. Jakaria and M. A. Rahman, "An adaptive load balancing scheme for software-defined network controllers," Computer Networks, Elsevier ScienceDirect, vol. 164, 2019.
[18] Y.-R. Chen, A. Rezapour, W.-G. Tzeng and S.-C. Tsai, "RL-Routing: An SDN Routing Algorithm Based on Deep Reinforcement Learning," IEEE Transactions on Network Science and Engineering, vol. 7, no. 4, pp. 3185-3199, 2020.
[19] H. Zhong, J. Fan, J. Cui, Y. Xu and Lu Liu, "Assessing Profit of Prediction for SDN controllers load balancing," Computer Networks, vol. 191, 2021.
[20] A. Mokhtar, X. Di, Y. Zhou, Z. M. A. Alzubair Hassan and S. Musa, "Multiple-level threshold load balancing in distributed SDN controllers," Computer Networks, vol. 198, pp. 108369 - 108385, 2021.
[21] C. Li, K. Jiang and Y. Luo, "Dynamic placement of multiple controllers based on SDN and allocation of computational resources based on heuristic ant colony algorithm," Knowledge-Based Systems, vol. 241, pp. 108330 - 108348, 2022.
[22] C. Li, S. Liang, J. Zhang, Q.-e. Wang and Y. Luo, "Blockchain-based Data Trading in Edge-cloud Computing Environment," Information Processing & Management, vol. 59, no. 1, pp. 102786-102808, 2022.
[23] K. Antevski, C. J. Bernardos, L. Cominardi, A. d. l. Oliva and Alain Mourad, "On the integration of NFV and MEC technologies: architecture analysis and benefits for edge robotics," Computer Networks, vol. 175, pp. 107274-107291, 2020.
[24] J. Ali and B.-h. Roh, "A Novel Scheme for Controller Selection in Software-Defined Internet-of-Things (SD-IoT).," Sensors, pp. 3591-3608, 2022.
[25] Z. C. S. O. O. Prince Boateng, "An Analytical Network Process model for risks prioritisation in megaprojects," International Journal of Project Management, vol. 33, no. 8, pp. 1795-1811, 2015.
[26] J. Ali, R. H. Jhaveri, M. Alswailim and Byeong-hee Roh, "ESCALB: An effective slave controller allocation-based load balancing scheme for multi-domain SDN-enabled-IoT networks," Journal of King Saud University - Computer and Information Sciences, vol. 35, no. 6, pp. 101566 - 101577, 2023.
[27] H. P. Enterprise, HPE VAN SDN Controller 2.7 REST API Reference, Hewlett Packard Enterprise Development LP, 2016.
[28] M. P. Contributors, "Mininet," Mininet, 2016. [Online]. Available: http://mininet.org/. [Accessed 2017].
[29] H. Packard, HPE VAN SDN Controller 2.7 Programming Guide, US: Hewlett Packard Enterprise Development LP, 2016.
[30] J. Dugan, S. Elliott, B. A. Mah, J. Poskanzer and K. Prabhu, "iperf.fr," iperf, [Online]. Available: https://iperf.fr/. [Accessed 2022].
[31] S. Kaur, K. Kumar, J. Singh and N. S. Ghumman, "Round-robin based load balancing in Software Defined Networking," in 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2015.
[32] M. S. Islam, M. Al-Mukhtar, M. R. K. Khan and a. M. Hossain, "A Survey on SDN and SDCN Traffic Measurement: Existing Approaches and Research Challenges," Eng, vol. 4, pp. 1071-1115, 2023.
[33] M. Omran, A. Alssaheli, Z. Z. Abidin, N. A. Zakaria and Z. A. Abas, "Implementation of Network Traffic Monitoring using Software Defined Networking Ryu Controller," WSEAS Transactions on Systems and Control, vol. 16, pp. 270-277, 2021.

Authors

Omar M. Mohamed received the Bachelor’s degree in Mathematics and Computer Science from Minia University, Egypt in 1999 and the Master of Sciences degree from Minia University, Egypt in 2015. He is currently IT Coordinator at Minia University, Egypt. His research interests include computer Networks, computer security, IoT, software engineering computing, pattern recognition, neural networks and artificial intelligence.

Tarek M. Mahmoud received the Bachelor's degree in mathematics from Minia University, Egypt, in 1984, the M.Sc. degree from Assiut University, Egypt, in 1991, and the Philosophy Doctorate (Dr.Ing.) degree in computer science (computer networks) from Bremen University, Germany, in 1997. He was a Professor of computer science with the Computer Science Department, Faculty of Science, Minia University. He is currently a Professor of computer science with the Faculty of Computers and Artificial Intelligence, University of Sadat City, Egypt. His research interests include pattern recognition, social networks analytics, web and text mining, artificial intelligence, natural language processing, and computer networks.

Abdelmgeid A. Ali received the PhD degree in Computer Science in 1996. Currently, he works as a Professor in Computer Science Department, Minia University, El Minia , Egypt. He has published over 80 research papers in prestigious international journals and conference proceedings. He has supervised over 60 Ph.D. and M. Sc. Students. Prof Ali is a member of the International Journal of Information Theories and Applications (ITA). Prof Ali interests are Information Retrieval, Software Engineering, Image Processing, Data security, metaheuristics, IOT, Digital Image Steganography, and Data Warehousing.

How to cite this article:

Omar M. Mohamed, Tarek M. Mahmoud, Abdelmgeid A. Ali, “Assessing a Real-time Adaptive Traffic Route Based on Ranking Software Defined Networking (SDN) Cluster of Controllers in a Datacenter”, International Journal of Computer Networks and Applications (IJCNA), 10(6), PP: 868-888, 2023, DOI: 10.22247/ijcna/2023/223684 .