Kafrelsheikh Journal of Information Sciences ISSN (Online): 2535-1478, ISSN (Print): 2537-0677 Volume 4, Issue 2, 2023, PP. 1–38

Software defined network traffic routing optimization: A systematic literature review

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

Abstract

The recent surge of interest in Software Defined Networking (SDN) technology is attributed to its centralized administration and control approach, which enhances network management and streamlines infrastructure maintenance. Despite its apparent sudden emergence, SDN is rooted in a lineage of endeavors aimed at enhancing network programmability. SDN offers real-time responsiveness and meets demanding high availability criteria. However, this novel paradigm encounters various technological challenges, some intrinsic and others inherited from pre-existing technologies. This study focuses on illuminating routing traffic concerns within the realm of SDN and provides insights into the forthcoming challenges that confront this transformative network model, encompassing both protocol and architecture perspectives. Additionally, we aim to explore diverse extant solutions and mitigation strategies that tackle issues of SDN scalability, elasticity, dependability, reliability, high availability, resiliency, and performance. This study entails a systematic analysis of 16 scholarly articles addressing routing traffic matters in the context of SDN. Through inductive analysis, this paper discerns and elucidates solutions for recurrently highlighted issues within academic discourse.

Keywords—Software Defined Networking, SDN, Traffic, Load-Balancing, Traffic Optimization and Path Selection

1. Introduction

Traditional routing protocols exhibit inefficiency within expansive and intricate networks, primarily due to their reliance on information flooding and localized routing computations [1], [2] . The Software-Defined Networking (SDN) paradigm effectively tackles these drawbacks 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 [3]. The control plane assumes responsibility for routing determinations in an entity known as the SDN controller [4], while the data plane undertakes the task of packet forwarding. This segregation empowers the control plane with a holistic network perspective, enabling more judicious routing choices [5]. The controller effectively manages flow control between the two planes by interacting through Application Programming Interface (API) between the two plans passing by the application layer as illustrated in Figure 1. The primary objectives of SDN are to simplify network and service management, reduce costs, and improve adaptability. SDN employs the OpenFlow protocol [6] to facilitate communication between the control and data planes. This strategic arrangement empowers the control plane to optimize traffic routing at the level of each distinct flow. As a consequence of these inherent benefits, SDN emerges as a propitious technology for elevating the efficiency and scalability of expansive, intricate networks.

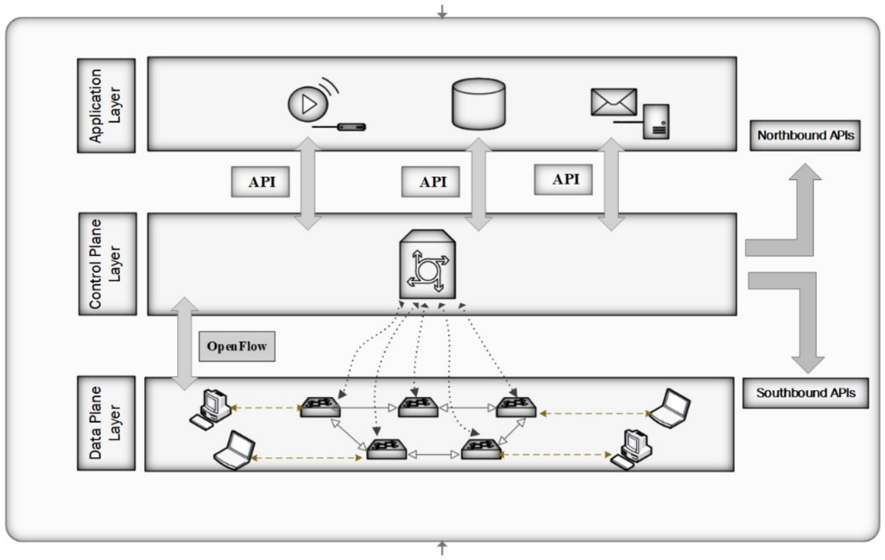

Figure 1. SDN paradigm architecture

SDN enables the utilization of two primary Application Programming Interfaces (APIs), namely Southbound APIs and Northbound APIs, which facilitate bidirectional communication and interaction with the control and data planes. Employing Northbound APIs within an SDN controller enables programmatic control over the network infrastructure, while Southbound APIs serve as the intermediary links between control and forwarding components [7]. The contemporary data landscape is characterized by high volumes of video traffic, the proliferation of large data centers, and the mobility of network users. These factors can lead to significant traffic congestion and performance problems for network operators. In addition, data center operators are facing challenges due to the rapid growth of server and virtual machine deployments, as well as the increasing amount of server-to-server communication traffic. SDNs can provide a centralized control plane that can optimize network traffic flows and dynamically adapt to changing conditions [8]. This can help to improve network performance and reliability and reduce the risk of traffic congestion. While software-defined networking (SDN) offers many advantages for improving network performance, its architecture also introduces some challenges. The structure of the paper is organized as follows. **Section 2** discusses the background of SDN traffic routing issues and strategies. **Section 3** discusses the Related literature where Research pertaining to traffic load balancing and traffic resilience is analyzed in **sub-section 3.1** and **sub- section 3.2** review the schemes designed for traffic optimization and path selection. **Section 4** explores a comparison of evaluation tools and performance results including the limitations. Finally, **section 5** concludes this paper and introduces future works.

2. SDN Traffic Routing issue and strategies

2.1. SDN Traffic Routing issue

The challenges and issues that are considered a paramount importance of traffic routing in the SDN environment are the following: The following are the most important challenges and issues of traffic routing in SDN environments [9]:

- Scalability remains vital: The capacity of SDN networks to handle extensive device counts and traffic flows is crucial. This poses a challenge as SDN controllers must manage substantial data and swiftly determine routes.
- Real-time adaptation to topology shifts: SDN networks must promptly respond to dynamic alterations in network structure. This is imperative since SDN controllers must swiftly reroute traffic to bypass failed connections or congested nodes.
- Meeting diverse QoS demands: Adhering to distinct application QoS requirements is essential for SDN networks. SDN controllers must effectively prioritize and route traffic, aligning with each application's distinct needs.
- Elevated security and privacy considerations: Ensuring the security and confidentiality of user data within SDN networks is paramount. The centralized control and data of SDN networks create susceptibility to potential breaches.
- Optimal resource utilization: Efficiently leveraging network resources is a core requirement for SDN networks. SDN controllers must formulate routing decisions that minimize congestion and maximize bandwidth deployment.

Despite these challenges, SDN is a promising technology with the potential to revolutionize the way networks are managed. As the technology matures, these challenges will likely be addressed, and SDN will become a more viable solution for a wide range of networks.

2.2. SDN Traffic Routing strategies

Software-Defined Networking (SDN) employs diverse strategies for traffic routing, the common strategies shown in figure 2 and categorized as follows [10]:

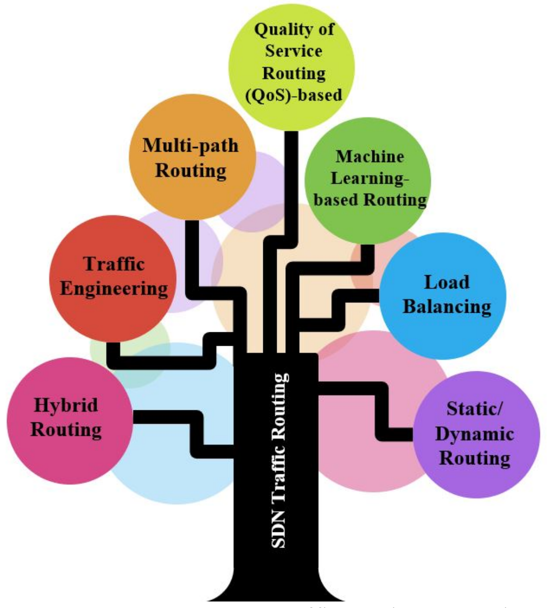

Figure 2. SDN common traffic routing strategies

Static Routing: Involves manually set routes, suitable for stable traffic patterns.
Dynamic Routing: Adapts routes dynamically using algorithms, excelling in dynamic traffic scenarios.
Hybrid Routing: Merges traditional and SDN-based approaches for combined benefits.
Load Balancing: Distributes traffic evenly across paths to enhance resource usage and avert congestion.
Traffic Engineering: Intelligently manages traffic flows to optimize performance and resource allocation.
Multi-path Routing: Improves network dependability by employing multiple routes simultaneously. This includes Equal-Cost Multipath (ECMP), which uniformly spreads traffic across equitably costed paths to optimize resource usage and balance loads, and Unequal-Cost Multipath (UCMP), which allocates distinct weights to paths according to their capacities, latencies, or expenses.
Machine Learning-based Routing: Utilizes machine learning for optimized routing decisions based on historical data and changing network conditions as predictive routing and reinforcement learning routing.
Quality of Service (QoS)-based Routing: Prioritizes traffic according to QoS needs like latency and packet loss.

3. Related literature

This paper provides an extensive survey of academic literature concerning the enhancement of traffic routing within the realm of Software-Defined Networking (SDN). The survey is organized into two primary sections. The first section critically reviews studies focused on traffic load balancing and traffic resilience, which aim to evenly distribute traffic across multiple links or paths to enhance network performance and maintain traffic delivery even during link failures. The second section analyzes research pertaining to traffic optimization and path selection, which is concerned with finding the best paths for traffic to take in order to minimize delay, maximize throughput, or minimize cost.

3.1.Traffic Load Balancing & Traffic Resilient

A method of load balancing based on server response time (LBBSRT) [11], aims to solve the problem of load balancing in the server cluster based on the server response time. LBBSRT chooses the server with minimum response time to provide services to users. The server response time is defined as the interval from accepting user requests to responding to user requests. Traditional schemes do not incorporate the server response times while balancing the server loads. The system model was formulated within the OpenFlow environment, illustrated in Figure 3.

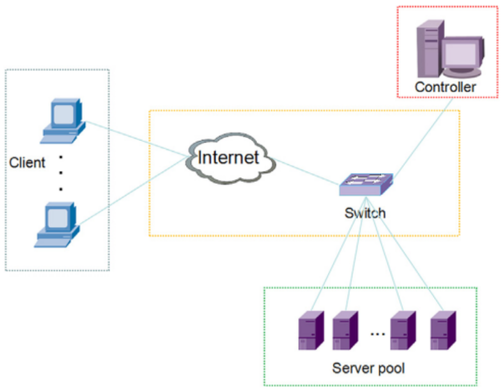

Figure 3. LBBSRT System model

LBBSRT uses the control plane to obtain the server response times accurately and effectively. The system is composed of two algorithms: one for real-time measuring of response time for each server in which, the server response time is obtained by parsing the Packet_in message which is sent by the switch. While the other for handling user requests, the controller handles the ARP messages (ARP_broadcast) that are sent by users and handle the user request by sending an ARP reply packet to users and then selecting the server with minimum or stable response time according to the obtained server data. Because the server response time directly reflects the server load capability, selecting a server based on the response times helps to send user requests to the servers operating under minimum server load to extract maximum performance. The longer the response time is, the higher the corresponding load is. The suggested LBBSRT method effectively leverages server resources, resulting in superior load-balancing outcomes and has been shown to have significant advantages in terms of overall response times and load balancing when contrasted with conventional Round Robin and Random strategies.

André and Fernando introduced a fault-tolerant controller framework for Software-Defined Networking (SDN) called RAMA [12]. The novelty of the solution lies in Rama not requiring changes to OpenFlow nor to the underlying hardware, allowing immediate deployment. The Rama controller framework adopts a primary/backup model to ensure SDN controller fault tolerance. RAMA, has a high-level architecture with OpenFlow-enabled switches, controllers managing the switches, and a coordination service. The model includes a primary controller and multiple backup controllers to tolerate faults. The primary/backup model allows for fault tolerance by electing a new leader when the master controller fails. The coordination service ensures strong consistency among controllers, but it becomes a system bottleneck due to the need for agreement between replicas. The protocol presented aims to handle switches' state consistently in the presence of faults. Furthermore, the RAMA controller framework guarantees three essential properties: (i) events are processed precisely once by the controllers, (ii) all controllers process events in the same order, ensuring they reach the same state, and (iii) switches process commands exactly once using Open Flow bundles [13].

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 as figure 4.

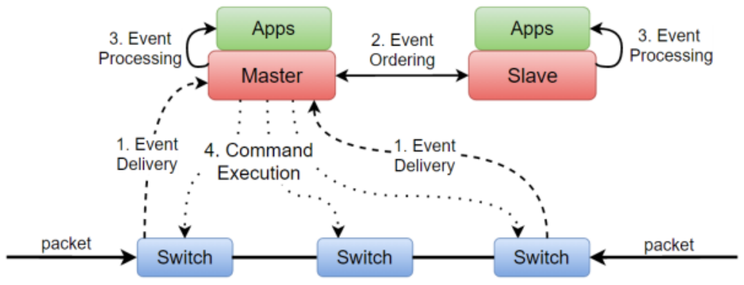

Figure 4. RAMA event processing structure [12]

RAMA offers significant advantages as it ensures consistent command and event processing, providing equivalent strong assurances as Ravana [14] without the need for any modifications to switches or the OpenFlow protocol. This quality makes RAMA a highly effective facilitator for the seamless implementation of fault-tolerant SDN solutions. RAMA is a robust SDN controller platform designed for fault tolerance, providing equivalent strong assurances as Ravana without the need for any modifications to switches or the OpenFlow protocol. On the other hand, RAMA does have some drawbacks compared to Ravana. The implementation of Rama results in higher costs due to increased network message exchanges and the introduction of additional mechanisms like bundles, which add to the overall overhead of the solution. Despite these drawbacks, the performance impact is relatively minor, and Rama's core value proposition of ensuring consistent command and event processing without requiring modifications to switches or the OpenFlow protocol remains compelling. Consequently, Rama remains a valuable enabler for the immediate adoption of fault-tolerant SDN solutions.

[15] Introduced a mechanism which aims to achieve efficiency by reducing host overhead and preventing packet reordering, without being restricted to a particular version of OpenFlow supported by the used controllers and switches. The proposed mechanism was assessed using MultiPath Transmission Control Protocol (MPTCP) [16] . The study explores the integration of SDN components, such as the SDN controller and virtual switch, within end-hosts to improve network performance. The MPTCP connection functions as a thin intermediary between the application and TCP layers. It enables the creation, management, and termination of TCP sub-flows, which start with the exchange of SYN, SYNACK, and ACK messages, as shown in Figure 5. The MPTCP protocol utilizes several types of messages, such as MP_CAPABLE,

ADD_ADDR/REMOVE_ADDR, DATA_FIN, FIN flag, RST/FIN, and MP_FASTCLOSE, to establish and terminate connections and ensure backward compatibility.

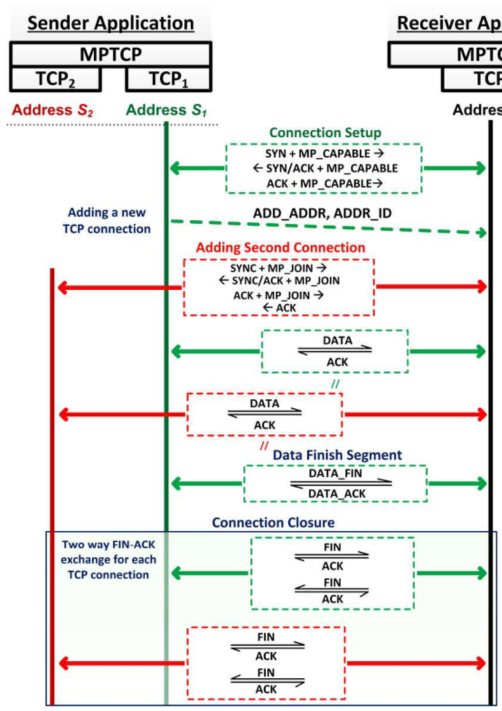

Figure 5. MPTCP Operations [15].

MPTCP connection must support the throughput of all flows without unfairly affecting normal TCP flows in the network. Additionally, both ends of the connection must be controlled during the connection's lifetime for the successful deployment of MPTCP. The introduced mechanism architecture setup involves a switch that controls the host's external network interfaces (eth0 and eth1). Two virtual network interfaces (veth0 and veth1) are connected, where veth0 is linked to the switch, and veth1 is an internal gateway for application traffic. The switch employs match-action rules set by the controller to implement traffic load balancing, selecting the external interface (eth0 or eth1) through which packets are forwarded as illustrated in figure 6.

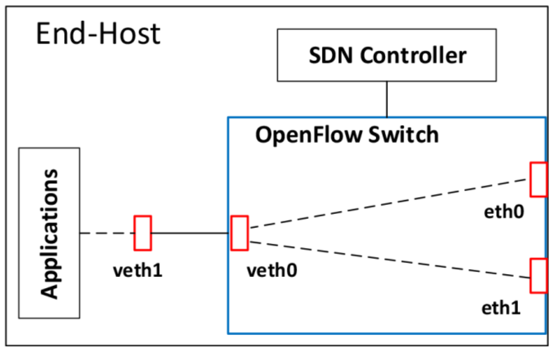

Figure 6. The architecture of host-based network load-balancing mechanism using OpenFlow [15].

This mechanism optimizes traffic distribution and load balancing within the network by performing the following steps:

- Packet Processing: The controller receives packets from the switch.
- TCP SYN Check: The controller checks if the received packet is a TCP SYN packet.
- Interface Assignment: If the packet is a TCP SYN packet, the controller employs a weighted round-robin load-balancing algorithm to assign an appropriate interface to the new flow.
- OpenFlow Rule Creation: Once the interface is determined, the controller creates an OpenFlow rule that matches TCP/IP packets and the specific source port of the received packet.
- Forwarding Instruction: The controller instructs the switch to forward the packet through the chosen interface as the output action.
- Rule Installation: Finally, the controller installs the newly created OpenFlow rule on the switch.

By carrying out these steps, the introduced mechanism enables dynamic interface assignment and traffic redirection based on OpenFlow rules, facilitating effective and efficient management of network flows. Implementing the proposed mechanism in existing SDN-based load-balancing approaches, it demonstrates excellent performance, not only when using single-link capacity but also using Multipath TCP (MPTCP) approaches.

Hamza et al. [17] Proposed the multiple threshold load balance (MTLB) switch migration scheme that aims to solve load imbalance and prevent controller overload. MTLB categorizes the load into various progressive levels that serve as the foundation for switching the migration process in cases where the load of a controller is dissimilar from that of others. This results in the threshold value being modified dynamically. The threshold value is subject to change based on the load status, with a dynamic approach that can shift from one value to another based on the average load. MTLB utilizes a trigger factor instead of periodic updates to update load information among controllers. The scheme initially categorizes the load into appropriate threshold levels for synchronization and migration handling. When a controller's load exceeds or approaches the threshold, it notifies the others to update their load information, reducing unwanted overhead from load information synchronization and handling the migration effectively. If there is a significant difference in load levels between controllers, the scheme executes a switch migration with careful consideration of the emigrant switch and target controller as mentioned in figure 7.

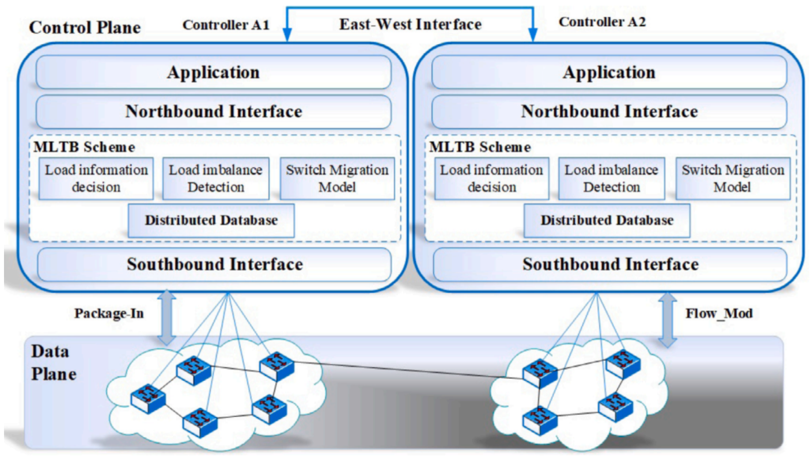

Figure 7. The architecture of the MTLB-Distributed SDN Model [17].

The scheme has three stages modules: checking for updates, detecting load imbalance, and selecting the suitable switch and controller for migration shown in figure 8.

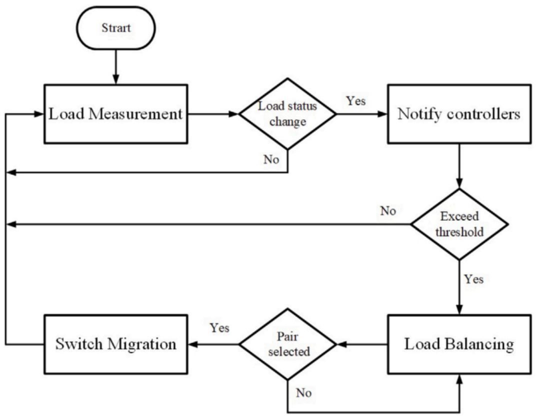

Figure 8. The MTLB scheme flowchart [17].

The system employs four status levels and three load thresholds. The highest threshold, "Overload," signifies that the controller has reached its full capacity and prompts immediate migration. Controllers in the "Highly Loaded" status can still operate for a limited duration but necessitate migration if other controllers are idle or in a normal state. "Normal" controllers are equipped to handle unexpected scenarios, while "Idle" controllers have the highest priority in receiving switches. The system's load management is thus designed to ensure efficient resource utilization and prevent performance bottlenecks. The MTLB scheme outperforms all other schemes with its high average throughput of approximately 1300 and 5000 packet/s. It also excels in having the lowest average packet delay compared to other schemes, while SMDM and EASM fall in between, and DDS experiences significant fluctuations and the highest peak delay. In terms of migration cost, SMDM has the highest, while EASM and DDS show similar costs due to their migration decision similarities. On the other hand, the MTLB scheme boasts the lowest migration cost and packet loss. The MTLB scheme is also superior in terms of communication overhead, being the lowest among all schemes. SMDM and EASM rank in the middle, while DDS exhibits high fluctuations based on traffic load. The MTLB scheme's effectiveness lies in its use of controller load status for efficient load information dissemination among controllers. Overall, the MTLB scheme offers the best performance and efficiency among the evaluated schemes.

To address the challenges of communication delay between controllers and switches, as well as inter-controller communication issues resulting from link failures in the network, Chunlin et.al [18] introduced a novel model based on task latency and dynamic constraints. The heuristic ant colony algorithm (HACA) [19] is employed for dynamically allocating computational resources, considering factors such as the traffic volume of each controller, available resources, the distance between controllers and switches, and the time delay between controllers caused by the communication delay occurring when controllers interact with each other. The proposed model leverages two key aspects: first, the development of a dependable controller placement method that optimizes latency and load considerations, improving the load optimization multi-controller placement (LOCP) algorithm. Secondly, the formulation of a resource allocation algorithm that takes into account task latency and reliability constraints, using the HACA. Improvement operation of a multi-controller placement (LOCP) algorithm illustrated in figure 9: **(1)** User devices connect to the edge computing layer via network access points (e.g., wireless access points, base stations) to access services. **(2)** A multi-access edge computing (MEC) server [20] located near the base station provides computing, and storage resources, and collects/analyzes information from end devices, reducing data and enhancing network service quality. **(3)** The MEC server connects to a local SDN controller through an OpenFlow switch for efficient network traffic and resource management. **(4)** A global controller oversees the local SDN controller, updating data matching and processing rules through operation status exchange. **(5)** Controllers communicate using an east-west interface, ensuring seamless coordination.

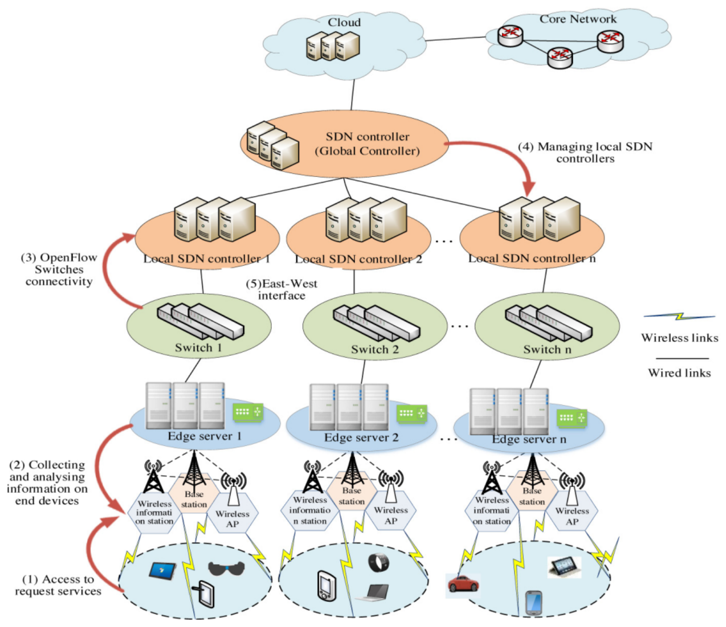

Figure 9. Dynamic controller placement and resource allocation in an SDN-based multi-access edge computing system [18].

By adopting this architecture, the approach optimizes network performance and resource utilization through edge computing and SDN-based control, resulting in enhanced user experiences and efficient network services. Regarding the controller placement problem, the research focused on three primary performance metrics. Firstly, it examines the delay between the controller and the switch. Secondly, it analyzes the time delay between controllers, due to inter-controller communication. Lastly, the research addresses load balancing to ensure equitable distribution of processing tasks among controllers. The study addressed the controller placement problem, focusing on delay and load optimization, while also considering network link connectivity. Experimental results demonstrate the improved LOCP algorithm's efficacy in achieving a balanced network load and reducing network overhead, particularly in small and medium-sized networks when compared to existing approaches. In addition, the improved HACA algorithm' solves the problem of work resource load allocation and effectively reduces the user resource response time as well as the average completion delay under the premise of ensuring the reliability constraint.

Jehad and Byeong-hee [21] Proposed a mathematical decision-making framework by calculating the optimal controller in terms of its features that enhance the performance of the Software-Defined Internet-of-Things (SD-IoT) using an analytical network decision-making process (ANDP) model [22]. The controller selection technique is based on a qualitative and quantitative examination of SDN controllers for SD-IoT. They determined ten considered characteristics of the controllers for the IoT environment listed in table 1. ANDP is used to determine the high-weight SD-IoT controller by computing weights for each controller, and then ANDP ranks the controllers with the best feature set for SD-IoT among others.

Calculate Controller Weights after applying the comparison matrix that is the outcome of all judgments of the controllers’ supporting features which are important in SDN based on the 10 characteristics listed in the following table.

Table 1. List of features for SD-IoT performance evaluation [21].

Serial#	**Notation**	Name	**Description**
1	**B_1**	OpenFlow-support	OpenFlow version1.0–1.5
2	**B_2**	GUI	Web based or Python-based
3	**B_3**	NB-API support	REST-API
4	**B_4**	Clustering support	To ensure reliability and performance
5	**B_5**	Openstack networking	Enabling different network technologies via quantum API
6	**B_6**	Synchronization	State synchronization of the clusters
7	**B_7**	Flow requests handling	The capability to handle the flow requests
8	**B_8**	Scalability	Adoptability in the extended networks
9	**B_0**	Platform support	Windows, Mac, Linux
10	**B_{10}**	Efficient energy management	The ability to utilize energy efficiently

The evaluation of controllers’ supporting features is represented according to the level of support in which G1 indicates extremely low support and G4 denotes very strong support. G2 indicates medium support, but G3 only reveals strong support, where the characteristics evaluation score from G1 to G4. After that comes the comparison stage of Controllers regarding their Features for SD-IoT, finally, the controller weights were calculated. This model presented a novel controller selection approach for SD-IoT environments, based on the Analytical Network Process (ANDP) model, is evaluated in terms of delay, throughput, CPU utilization, and reliability. Figure 10 illustrates the ANDP model for paired comparisons in selecting an SD-IoT controller. The figure represents the ranking model of the ANDP, comprising a features cluster (top one) and an alternatives cluster (bottom one). Additionally, a circular line indicates the interdependency among these features.

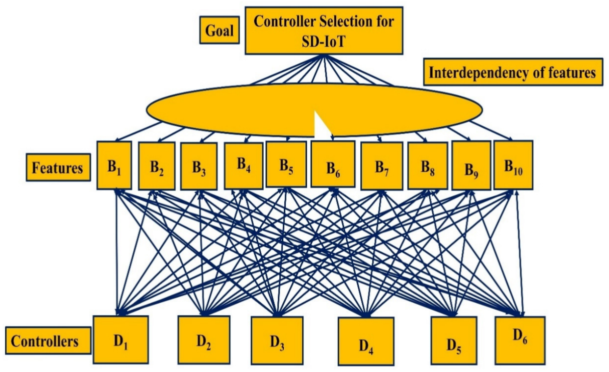

Figure10. The ANDP model for controller selection in SD-IoT [21].

The proposed model was compared with previous benchmark schemes, namely AHP [23] and EB-TOPSIS [24], through a series of experiments. Key Findings achieved that the proposed controller:

- Reduces delay in various traffic scenarios.
- Increases throughput while efficiently utilizing the CPU.
- Exhibits enhanced reliability during link failure recovery.

Jehad et.al. [25] Introduced ESCALB, a load-balancing scheme designed for multi-domain SDN-enabled IoT networks (SD-IoT). Its main goal is to efficiently migrate switches to controllers with available resources in a dynamic manner. ESCALB uses a hierarchical model for a control plane consisting of multiple domain controllers (DCs) and a global control (GC) plane. The GC plane comprises four sub-modules as figure 11, including the Load Calculation Module (LCM) and ANP Module (ANPM), which monitor load status by receiving information from the Distributed Control Plane (DCP) and rank controllers based on CPU usage, Flow Requests Capacity FRC, memory utilization, and the number of attached switches. In addition, the ANP Module (ANPM) utilizes the Analytic Network Process (ANP) model to prioritize slave controllers in the DCP. ANPM employs a mathematical procedure with a 9-point scale matrix to rank controllers, where 1 indicates equal importance and 9 signifies extreme significance. The Switches Migration Module (SMM) initiates switch migration to slave controllers if the master controller's load exceeds a predefined threshold. Flows Forwarding and Updating Module (FFUM) collaborates with ANPM and SMM to prioritize slave controllers, migrate switches, and forward flow requests to controllers with higher weights.

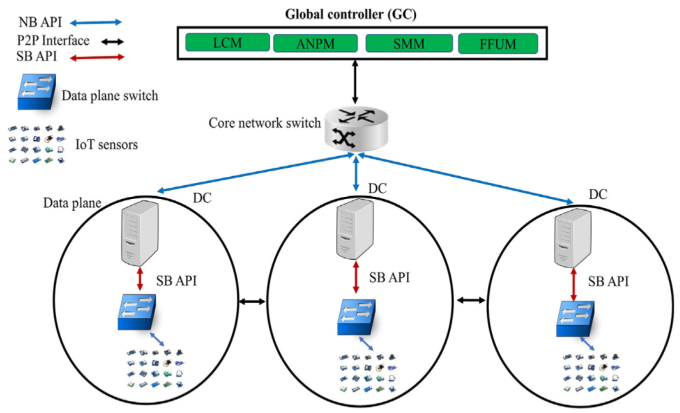

Figure 11. Proposed framework for load balancing using ANP module with SD-IoT [25].

The primary objective of the GC plan is to optimize network performance and resource utilization by utilizing ANP-based ranking, switch migration, and load-balancing mechanisms. The proposed scheme was compared with previous benchmark schemes, namely SASLB [26], DLB [27], SCLB [28], and SMLB [29]. ESCALB's effectiveness resides in its adeptness at intelligently selecting the most appropriate controller for load distribution, resulting in enhanced performance within SDN environments.

Table 2 outlines a comprehensive overview of studies pertaining to traffic load balancing and resilience, encompassing details regarding description, objectives, and traffic routing methods. In addition, Table 3 offers a comparative analysis of decision-making criteria and Implementation levels within the Data Plane and/or Control Plane.

Table 2. Traffic Load Balancing and Traffic Resiliency: Proposals description, objectives, and Techniques Method Employed.

Reference	Proposal description	Objectives	Techniques
Hong et. al. [11]	LBBSRT chooses the server characterized by minimum response time in order to provide services to the users.	Enhancing the load balancing effect by reducing the server response time.	Least Response Time Load Balancing
André and Fernando [12]	Rama controller framework is a promising approach to ensuring the fault tolerance of SDN controllers.	Prevent the controller from becoming a single point of failure, which necessitates the integration of the switch state into the fault-tolerant SDN framework.	Fault tolerance
Anees et. al. [15]	Network traffic control mechanism achieves efficiency by reducing host overhead and preventing packet reordering.	Improving Load Balancing and enhancing network efficiency by dynamically selecting the most suitable network interface for each traffic flow.	Least Connection Load Balancing
Hamza et al. [17]	MTLB is an efficient load balancing scheme for managing the load distribution among distributed Software-Defined Networking (SDN) controllers.	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.	Clustering

Chunlin et.al. [18]	A novel dynamic controller placement method that focuses on optimizing both delay and load factors by improving the multi-controller placement (LOCP) and ant colony algorithms.	Efficiently determine the optimal locations for placing controllers within a network to solve the multi-controller placement problem, enhancing the overall network service quality performance.	Clustering
Jehad and Byeong-hee [21]	Optimal SDN controller selection is critical to ensure optimal network usage, leading to improved Quality of Service (QoS) in SD-IoT, which involves assessing its attributes and validating its performance within the SD-IoT environment.	Enhancing throughput while efficiently utilizing the central processing unit (CPU) and minimizing recovery latency during network failures. Improving latency in both normal and heavy traffic scenarios	Clustering
Jehad et.al. [25]	A multi-criteria decision-making based slave controller selection strategy for SDN in IoT networks with dynamic switch migration.	Demonstrating the problem of static slave controller assignment to ensure effective migration of SDN switches.	Adaptive Load Balancing

Table 3. Traffic Load Balancing and Traffic Resiliency: Decision-Making Criteria and Implementation in the Data Plane and /or Control Plane.

Reference	Control plane	Data plane	Decision-Making based on
Hong et. al. [11]	✓	✓	The controller selects the server with minimum or stable response time according to obtained server data.
André and Fernando [12]	✓		Controller primary/backup Decisions and Coordination Service Decisions
Anees et. al. [15]		✓	enables dynamic interface assignment and traffic redirection based on OpenFlow rules.
Hamza et al. [17]	✓		The dynamic threshold value can be changed depending on the average load status.
Chunlin et.al. [18]	✓		Calculating the optimal placement of multiple controllers based on the network topology, traffic load, and available computational resources.
Jehad and Byeong-hee [21]	✓		Calculate Controller Weights utilizing the all-considered judgments of SD-IoT controllers' supporting features.
Jehad et.al. [25]	✓		Monitors the control plane in real-time, acquiring load information to assess and prioritize slave controllers and ensure successful switch migration.

3.2) Traffic Optimization and Path Selection

A novel traffic-aware QoS control mechanism for SDN-based virtualized networks was introduced [30], based on the single rate three color marker (srTCM) proposed by the Internet Engineering Task Force (IETF) [31]. This mechanism combines srTCM (Single rate three color marker) with two novel global token buckets (srTCM+ GTB) to effectively meter and mark packets from each virtual network, ensuring QoS in IP networks. Initially, the mechanism employs srTCM, which is based on Differentiated Services (DiffServ) principles [32], to classify and manage network traffic. DiffServ enables the provision of low latency for critical network traffic, such as voice or streaming media, while offering simple best-effort service to non-critical services like web traffic or file transfers. By using srTCM, packets within a stream are metered and categorized into three traffic parameters and add two Token Buckets (TB). The three considered traffic parameters are committed information rate (CIR), committed burst size (CBS), and excess burst size (EBS), which are respectively marked as green, yellow, or red.

IF packet size:

< CBS, then packet marked green; packets can be forwarded immediately.

<CBS & < EBS, then packet marked yellow; packets can be forwarded or dropped according to the link state.

Otherwise, then packet is marked red; packets will be dropped directly.

Two Token Buckets (TB): A token is added to the C-bucket every 1/CIR s.

If No. of Tokens in C-bucket > CBS, the extra token is added to E-bucket, then If No. of Tokens in E-bucket > EBS, then discard extra token as illustrated in figure 12.

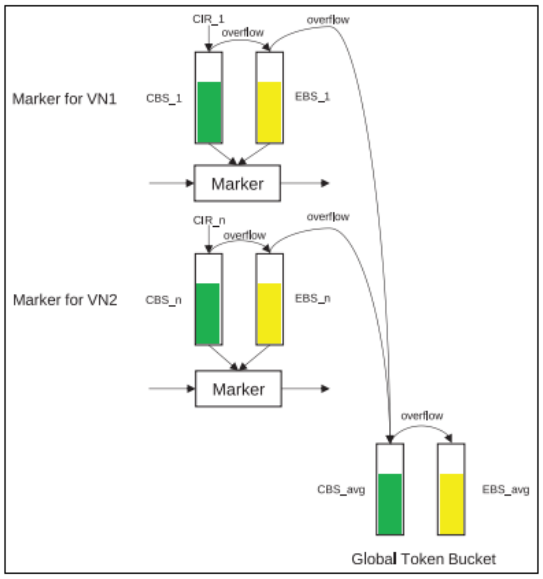

Figure 12. Token refill scenario of srTCM +GTB [30]

The proposed architecture of an autonomic managed network is illustrated in figure 13. A virtualization plane is introduced between the control plane and data plane to efficiently allocate the physical network based on the unique requirements of each virtual network. The virtual networks are categorized into different service levels using the differentiated services (DiffServ) approach. This involves color-marking packets according to the link status and inserting them into corresponding queues with different precedence using a weighted random early detection (WRED) queue [33].

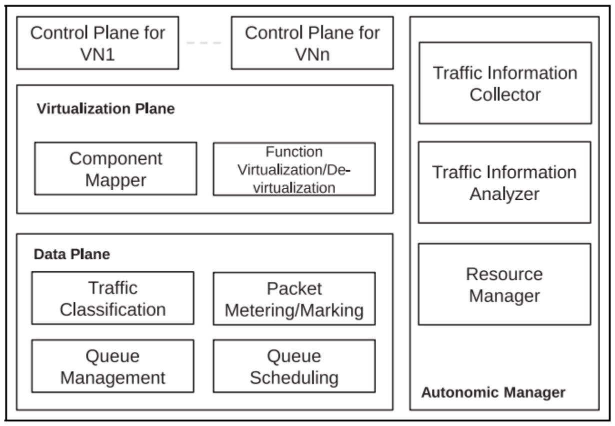

Figure 13. The architecture of autonomic managed network [30].

An autonomic manager is responsible for monitoring, collecting, and analyzing traffic and resource usage of both virtual and physical networks. It then dynamically redistributes resources among all virtual networks through a resource manager. To optimize resource utilization during periods of light traffic in some virtual networks, the srTCM and marking algorithm are extended. Two additional buckets, known as the global C bucket and global E bucket, are introduced. Similar to private token buckets of each virtual network, these are not automatically refilled, and their depths are not fixed. This adaptive approach ensures efficient resource allocation and enhances overall network performance. The performance evaluation was conducted by analyzing the TCP and UDP performance of the proposed srTCM+ GTB method compared to CBBPM. The proposed approach employs a traffic-aware quality-of-service control mechanism to enhance the quality of service for virtual machines in software-defined networking-based virtualized networks. The findings demonstrate the efficacy of this mechanism because srTCM+ GTB is better able to control congestion and ensure that all virtual networks receive their fair share of bandwidth which guarantees their isolation.

Junjie et.al. [34] Proposed a collaboration approach between MPTCP (Multipath TCP) and SR (Segment Routing) is aimed at addressing the resource consumption challenges in SDN-based Data Center Networks (DCNs) [35] [36]. The approach enables the simultaneous use of multiple paths for data transmission using MPTCP and provides a flexible and scalable mechanism for forwarding packets based on a segment identifier using Segment Routing (SR). In a large DCN or during a peak period, when a large flow arrives, it is divided into multiple subflows and transmitted using the MPTCP protocol. The SDN controller then allocates these subflows to specific transmission paths and maps each path into an SR path. This innovative method effectively manages traffic and reduces the demand for storage resources. The architecture proposed through the mentioned collaboration adopts a comprehensive four-layer approach for the DCN to ensure enhanced clarity and description. This four-layer architecture offers a comprehensive and well-structured approach to designing the DCN, incorporating advanced technologies such as SR and MPTCP while leveraging the capabilities of SDN for efficient traffic management. **MPTCP** offers a traffic-splitting capability, enabling the division of a flow into multiple subflows. This ensures that the flow can be simultaneously transmitted via multiple paths between peers. MPTCP proves beneficial in increasing the throughput of DCNs while reducing the likelihood of network failures, packet losses, and delays. In case of unavailability or poor performance of a path, MPTCP allows

the flow to be rerouted through an alternative path. MPTCP architecture comprises four essential components: path management, packet scheduling, subflow interface, and congestion control. Regarding the **SR** method, it operates using a series of segments as an ordered list of routing instructions for packets. SR necessitates Traffic Engineering (TE) decisions for the entire network. The transmission path is represented by two types of segments: node segments and adjacency segments. Node segments serve as unique identifiers for nodes within the network domain, and they possess global significance, enabling other nodes to transmit packets based on these identifications, often using Open Shortest Path First (OSPF) as the default protocol.
On the other hand, adjacency segments represent node-local interfaces that differ from node segments and are locally significant. These segments facilitate the transmission of packets to specific adjacent nodes through the associated interfaces.
A transmission example of MPTCP and SR is shown in figure 14.

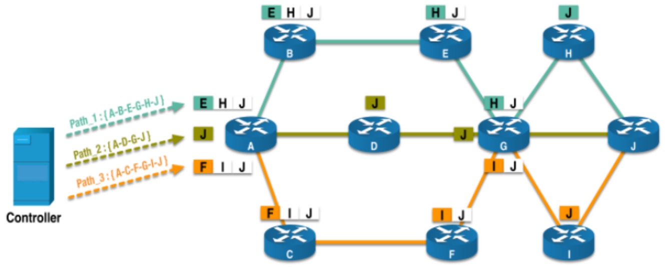

Figure 14. A transmission example of MPTCP and SR [34].

In the example, the controller selects three paths through the network for a packet to take. These paths are {A-B-E-G-H-J}, {A-D-G-J}, and {A-C-FG-I-J}. The SR technology then expresses these paths as a segment label list. The segment label list is a list of the segments that the packet must traverse in order to reach its destination. When the packet receives the segment label list from the controller, it follows the instructions in the list to complete the transmission process. The segment label list is a more efficient way to represent a path through the network than a traditional routing table. This is because the segment label list only includes the segments that the packet must traverse, while a routing table includes all of the possible paths through the network. As a result, the segment label list can be used to select the best path through the network for a packet, even if the network topology changes. In summary, the proposed MPTCP & SR showcases superior performance in terms of throughput maintenance and average link utilization under varying transmission demands. The dynamic path adjustment capability contributes to more efficient load balancing and response management, leading to more stable and improved network performance.

Shengru et. al. [37] introduced a forwarding technique that's independent of protocols, efficient in bandwidth usage and saves flow-table space. It utilizes POF-FIS, a protocol-oblivious forwarding instruction set. To implement POSR in an SDN network system, packet format is designed, and packet processing pipelines are developed to support unicast, multicast, and link failure recovery.
Packet format design
In consequence of the independent nature for POF protocol [38], the packet format design for POSR allows for a dedicated approach to accommodate the network scenario without the need to reuse existing protocol packet fields. The configuration of POSR packet header fields is illustrated in Figure 15.

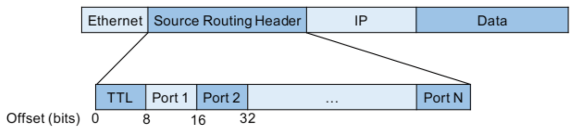	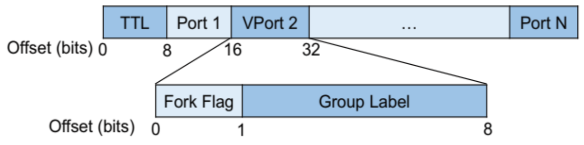
Figure 15. Header format design of POSR unicast packets [37]	**Figure 16.** Header format design of POSR multicast packets [37]

Specifically, the source routing header fields are introduced between the Ethernet and IP headers, and the Ethernet header Type field is defined as "0x0908" to signify a POSR packet. The Time-to-live (TTL) field indicates the remaining hops of the packet, while the Port field stores the designated output port on the switch for a given hop. To encode the routing path, a POSR packet incorporates multiple Port fields, where each intermediate switch extracts the outermost Port field to determine the packet's designated output port. For enabling POSR-based multicast, the POSR header's Port field is substituted with a VPort field, as depicted in Figure 16. The VPort field consists of multiple bits, with the first bit serving as the Fork Flag, indicating whether the corresponding switch acts as a fork node on the packet's multicast tree. The remaining bits in the VPort field represent the Group Label, which is assigned for each active multicast session. A fork node signifies a switch from which multiple branches originate within a multicast tree.

POSR Packet Processing Procedure

Upon the arrival of the first packet of a flow at the ingress POF switch, the switch sends a PacketIn message to the controller as no flow entry has been set up for the flow. The controller then calculates the flow path and sets up a flow entry in the ingress switch, which directs the switch to encode path information in the flow's packets using the POSR format. Given that the packet processing procedure for all POSR packets is identical in any intermediate switch, including destination switches, flow entries can be installed in all POF switches during network initialization, allowing POSR packets to share them.

FAST LINK FAILURE RECOVERY WITH POSR

To enable fast failover (FF) in POF switches, two entries are included in each FF group table to monitor switch port status and determine the backup path segment based on the packet's output port. The first entry in the FF group table forwards packets normally when the output port is up. If the output port is down, the second entry initiates POSR-based link failure recovery using the routing instructions of the backup path segment to replace those of the broken link in the headers of affected POSR packets, as illustrated in figure 17.

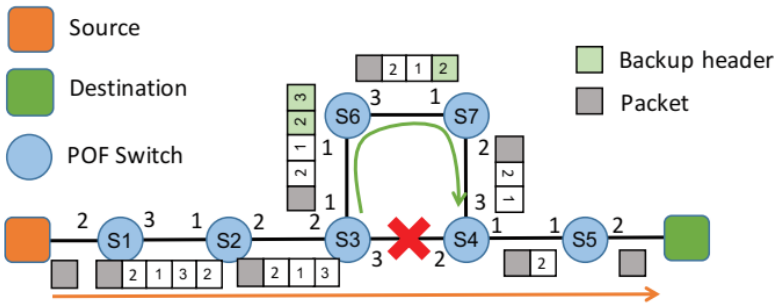

Figure. 17. POSR-based fast link failure recovery [37].

By leveraging the FF group table in POF switches, the system proactively monitors switch port status and identifies any port failures quickly. This approach enables affected packets to be swiftly rerouted through the backup path segment, minimizing network disruptions, and enhancing overall reliability in case of link failures. As the data transfer rate increases, OF-SP experiences a faster escalation in packet loss. The findings of this research suggest that POSR has the potential to be a viable solution for addressing the scalability challenges of SDN. POSR's ability to provide efficient source routing without relying on specific protocols allows for greater flexibility and adaptability, making it a promising option for large-scale SDN deployments.

Conserving energy is crucial not just from a financial and ecological standpoint, but also for ensuring the sustainable expansion of the Internet. This is because the delivery of power to and the removal of heat from massive data centers present significant challenges [39].

[40] The academic work introduced a heuristic scheme named Exact Path Control (EPC) designed for the incremental deployment of SDN switches in hybrid SDNs. EPC involves flow-level explicit path control to power off redundant links and switches in the network, thereby conserving energy.

EPC algorithm procedures

The algorithm functions as follows: Firstly, it sorts all flows based on priority and then reroutes them one by one until all requests are arranged. Then, it calculates the energy savings by shutting down the active edges and switches. Additionally, it sorts all links according to traffic volume in ascending order and judges whether each link and switch can be turned off. It also checks whether a specific link can be powered off. The algorithm must ensure that the forwarding path's delay does not exceed the maximum delay and that the volume on each link is within its capacity limit. To enhance Network Congestion Avoidance (NCA), they proposed upgrading the least key nodes incrementally that all flows must pass through. This is achieved by SDN switches rerouting packets based on multiple MPLS labels that contain forwarding port numbers of switches on the route. Fine-grained flow scheduling is crucial for energy conservation, and encapsulating MPLS labels can help reroute flows and power off idle links and switches. When selecting switches to update, we consider the topologies' structure. Two optimization techniques have been identified: (1) isolating nodes without flows passing through them and retaining nodes with flows, and (2) separating specific sub-topologies from the original topology and selecting key nodes in the sub-topologies as SDN switches.

The provided academic content demonstrates an example of utilizing MPLS labels to achieve energy savings in a network as shown in figure 18. The scenario involves two flows, f1 from h1 to h3, and f2 from h2 to h3. The controller selects switch s2 to install flow entries for rerouting flow f1. Upon entering SDN switch s2, packets of flow f1 are encapsulated with 3 MPLS labels, each indicating a forwarding port along its route. As the packets traverse through the switch, they pop up one MPLS label at a time, determining their forwarding port. This rerouting results in a new path for flow f1, leading to energy savings by shutting down switches s3, s4, and the corresponding links.

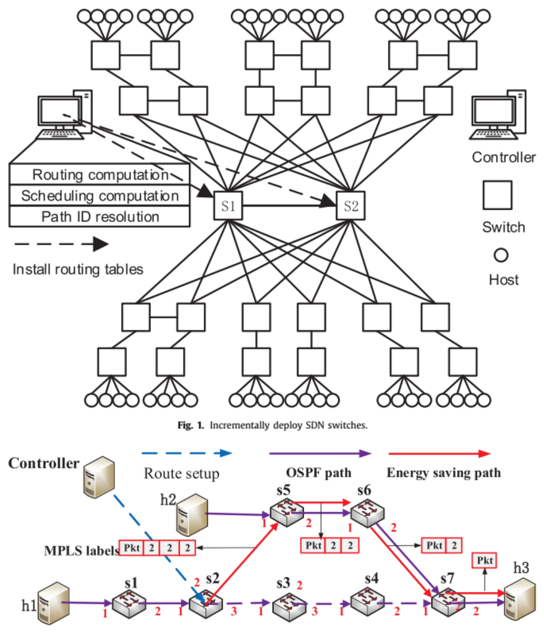

Figure 18. An example of Using MPLS label to achieve network energy saving [40]

The novel aspect of this approach is that, unlike traditional network MPLS, there is no need to maintain a path state in the forwarding path beyond the ingress node. This is because packets are now routed based on the list of labels they carry, enabling a more efficient and energy-saving routing strategy.

Wu et. al. addressed two critical issues in multi-path algorithms: scheduling efficiency and end-to-end delay fluctuations, which can impact video transmission quality. To tackle these challenges in media streaming applications, a dynamic and adaptive multi-path routing algorithm (DAMR) is proposed, utilizing SDN for centralized routing computations and real-time network state updates [41]. The key module of DAMR is the routing module, responsible for aggregating bandwidth based on monitoring information to ensure reliable transmission and avoid data packet loss due to link failures or congestion. By dynamically allocating data flows to multiple effective paths between nodes, DAMR optimizes network resource utilization, reducing congestion, packet loss rates, and end-to-end delays.
Three main algorithms constitute DAMR:

The network topology update algorithm optimizes the network topology and updates link information. *The network routing algorithm* handles packets in packets sent by OpenFlow switches and makes forwarding decisions based on routing decisions, acquiring QoS parameters from the monitor module for optimal path calculations. *The packet loss rate algorithm* ensures real-time measurement through periodic updates and calculates packet loss rates for each flow to determine optimal routes. The flowchart designed for the proposed algorithm is shown in Figure 19. Upon program initiation, the controller first checks for an optimal path to find, periodically querying the QoS parameters of the underlying network at 5-second intervals from the perspective of OpenvSwitch. When a packet reaches the controller, it calculates all accessible paths and their available bandwidths. The path with the maximum available bandwidth becomes the flow transmission path. If this path meets the delay constraint, it becomes the optimal path, and video streaming flows remain unchanged. However, if the path fails to meet the delay constraint, the controller utilizes the DAMR algorithm to find the optimal path. If no such optimal path exists due to unmet delay constraints, no rerouting occurs. Nevertheless, when an optimal path is found, the controller acquires QoS information from the monitor module and calculates link weights between node pairs. The link-state status information of each link on the flow's path is then updated. By utilizing the topology management module from the floodlight controller, link-state information, including connected nodes and ports, is stored in clusters. By traversing each node and its connected nodes, the full path from the source to the destination node is obtained, ensuring the optimal path that meets the delay constraint is achieved.

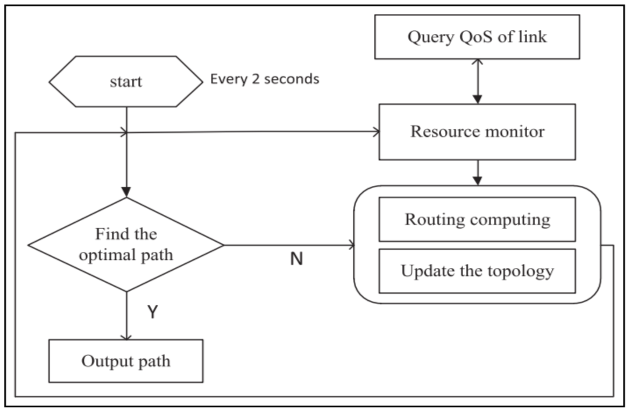

Figure 19. The flowchart design of the DAMR algorithm [41]

DAMR algorithm is compared with a single-path algorithm (e.g., Dijkstra and Bellman-Ford) that selects a subset of paths for video streaming based on additive cost properties like hop counts. In contrast, the ECMP algorithm [42]distributes traffic equally among multiple equal-cost paths, involving all links in the network. Experimental results show DAMR overcomes computational overhead, effectively adapting to dynamic network changes, and improving link utilization and user service quality. DAMR utilizes OpenFlow centralized control to optimize resource allocation and aggregate bandwidth resources by employing optimization theory.

Muteb and Abdelmounaam presented POX-PLUS [43] is an upgraded version of the commonly used POX controller in SDN networks, which includes a novel routing module called DRAPSP (Dynamic Routing based on All Pairs Shortest Paths). DRAPSP is responsible for calculating and effectively managing the shortest routes between nodes in the SDN. POX-PLUS consists of three primary components as shown in figure 20:

• DR-APSP: This module is the main routing component that operates on our dynamic routing all-pairs shortest path (APSP) algorithm [44]
• POX Controller is the original POX controller that has been modified to take on the additional responsibility of updating DR-APSP with information regarding the initial topology, as well as any subsequent modifications to the network.
• Data Structures are utilized to store information related to the topology and its shortest paths.

These data structures comprise the Topology Graph, Hosts, and a forest of shortest paths trees (SPT).

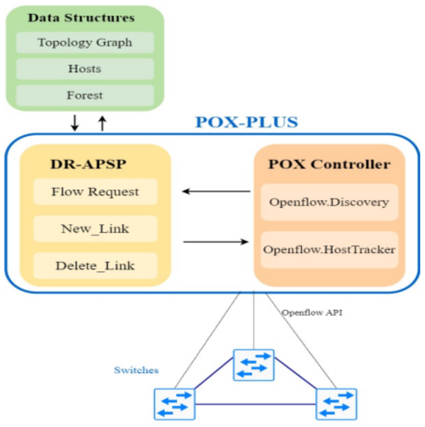

Figure 20. POX-PLUS Architecture [43].

The controller detects any link event in the network utilizing link deletion and link insertion algorithms. When detecting a link deletion, DR-APSP starts a loop over all shortest path trees (SPTs), that consists of two main phases. In the first phase, the algorithm identifies the affected switches in each SPT, and subsequently removes affected flows that selected the deleted edge, for each of the affected switches. Then the controller will apply a fresh flow rule in response to subsequent requests. In the second phase update the current SPT for future requests using APSP approach. The Link Insertion algorithm is triggered by a link insertion event, following which it begins a loop over all vertices. The loop comprises two primary steps: the first step entails updating the current SPT, while the second step involves removing impacted flows from affected switches. The measured performance parameters include:

"init time" for initializing flow tables at all switches, measuring the duration of data packet transmission between every host pair in the network. ***"Add time"*** indicates the time taken by an application to transmit a data packet between any two hosts after incorporating new network links. ***"Delete time"*** represents the time required for an application to send a data packet between any two hosts following the removal of network links. ***"Selected paths time"*** measures the duration taken by an application to transmit a data packet between any two hosts after configuring flow tables at all switches.
DR-APSP module is compared to the three routing schemes offered by the POX controller (l2 learning, l2 multi, and l3 learning). DR-APSP is the sole method that preserves the shortest paths and flow rule tables without recomputing paths. The research shows that l2 learning and l3 learning involve a high cost due to a short idle time for flows in flow tables, which is necessary for fast recovery.
Yi-Ren et. al. [45] provided a solution that helps to solve a traffic engineering (TE) problem of SDN in terms of throughput and delay. This solution developed a reinforcement learning routing algorithm (RL-Routing) which predicts the future behaviour of the underlying network using an RL agent to learn the optimal routing paths in a network. The RL agent interacts with the network by selecting actions (routing paths) based on the current state of the network and suggests better routing paths between switches. The RL-Routing application comprises two main modules are

shown in figure 21. The first module, the Network Monitoring Module (NMM), utilizes both passive and active network measurements to collect crucial information about network devices, including link delay, throughput, and port speed. This data is utilized to represent states and compute rewards. The second module, known as the Action Translator Module (ATM), converts the selected action by the agent into a series of appropriate OpenFlow messages.

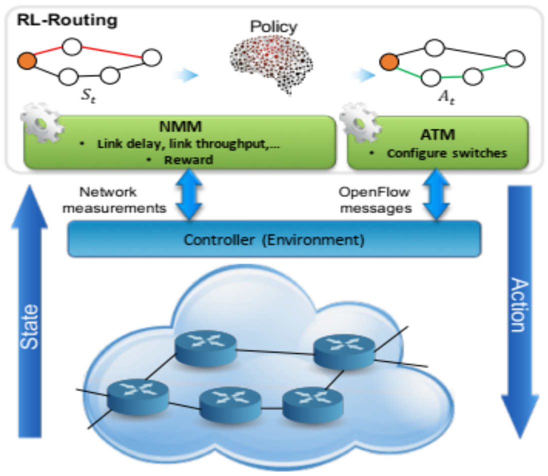

Figure 21. The components of RL-Routing [45].

These messages are employed to update the flow tables of switches when configuring a new path. To prevent Packet-In messages from 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 address scalability issues in routing by automating the process of path selection and reducing the need for manual configuration and maintenance. However, the effectiveness of RL-Routing in addressing scalability issues can depend on various factors, such as the size and complexity of the network and the performance metrics being evaluated. The performance of RL-Routing was evaluated on three well-known network topologies: Fat-tree, NSF Network (NSFNet), and Advanced Research Projects Agency Network (ARPANet), comparing RL-Routing with two widely used baseline solutions: Open Shortest Path First (OSPF) and Least Loaded routing algorithm (LL). The evaluation metrics are the reward function, which is a score computed using network throughput and delay, and the utilization rate, which is calculated in the destination switch. The reward function can be adjusted to optimize either upward or downward network throughput. Gururaj et.al. introduced the resource-efficient multicast tree construction model (REMTC) [46] that uses Dijkstra’s Shortest Path algorithm for initial tree formation, identifies a multicast path, and processes the Shortest Path Tree to reduce the overall hop count and path cost. Aims to reduce tree alteration using more common paths to reach the devices by tree optimization algorithm which enables the dynamic join and leaves of participating devices. In this model, the multicast tree construction algorithm continuously monitors the network and user requirements and adapts the multicast tree accordingly. For example, if a link becomes congested or unavailable, the algorithm can dynamically reroute traffic to other available links to maximize network efficiency and meet user requirements. REMTC model seeks to construct a tree comprised of nodes and links with the lowest possible values for both total path cost and total hop count. The REMTC algorithm is designed to identify the optimal values for both of these factors during the initial construction of the multicast tree. The trunk path that connects the maximum number of end devices in the shortest path tree (SPT) is then selected as it improves route stability. The construction of the multicast tree utilizes level information to enhance resource utilization. The tree optimization algorithm works as follows, the collection of participant information is performed by the SDN controller during the establishment of a session between end devices and a server. The dynamic joining/leaving of the participants from/to the multicast group is managed through Session Initiation Protocol (SIP) and Session Description Protocol (SDP) messages [47]. To evaluate the bandwidth availability, the SDN controller centrally computes the link cost based on switch statistics gathered at regular intervals, including the number of packets transmitted and received by each switch port. Dijkstra's shortest-path tree algorithm uses this link metric to construct the shortest-path tree, which is then utilized to determine the trunk path and level formation. The performance of REMTC was evaluated by comparing it with SPT, ST, BAERA [48], and OBSTA [49] for the following metrics:

- The "number of links" refers to the edges that are utilized in the multicast tree to forward the multicast data from the source node to the participating nodes.
- "Processing Latency" represents the amount of time it takes for the algorithm to construct a multicast tree for the set of participants.
- The "Rerouting Cost" is the delay that is incurred when constructing or rearranging the multicast tree due to a dynamic join or leave of a participant.

The REMTC model is capable of maintaining stable bandwidth consumption during dynamic join and leave events, which is not achieved by ST and SPT methods. The REMTC model only modifies the multicast tree to accommodate dynamic participants, thereby avoiding changes to the forwarding table of SDN switches and achieving multicast tree stability.

Majda et.al. presented a new model called Video Streaming Adaptive QoS Routing with Resource Reservation (VQoSRR) for Software Defined Networking (SDN) networks [50]. This model is designed to enhance the quality of video streaming services in SDN networks by offering adaptive QoS routing and resource reservation. The VQoSRR model employs a centralized management architecture to offer adaptive QoS routing and resource reservation tailored to video streaming services. By utilizing a feedback mechanism, the VQoSRR model dynamically adjusts QoS routing according to network conditions. Network state information, including bandwidth availability, delay, and packet loss, is considered to determine the optimal path for video streaming traffic. Furthermore, the model ensures resource reservations to guarantee sufficient resources for video streaming services. To achieve this, the VQoSRR model divides the network into multiple virtual networks and allocates resources to each one. It employs a bandwidth reservation mechanism to secure the required bandwidth for video streaming traffic and a packet scheduling mechanism to prioritize video streaming traffic over other types of traffic. Upon initiation of a new video stream flow by the server, the switch sends the first packet copy to the controller's QoS routing manager for determining the routing path. The VQoSRR employs two routing paths to balance between frequent dynamic network state updates and reduced routing computation overheads. One path is used as the primary route for the current flow, while the other serves as an alternative path for potential rerouting. These two paths are selected based on a weighted graph, where link characteristics such as packet loss rate and available bandwidth are determined using QoS routing manager algorithms with the aid of topology manager and statistics collector modules. In addition, this study introduced the Dynamic Traffic Rerouting Algorithm (DR-RA), which adapts to network conditions to ensure video QoS compliance, ultimately enhancing video quality. DR-RA updates the route's cache and reroutes traffic using alternative paths or generating new ones. The algorithm operates periodically, reading network statistics at predefined intervals. In case the current path violates the flow's QoS requirements, its flow entries are deleted from the switch, and an alternative route is examined. If the alternative path also fails to meet quality requirements, the algorithm calculates another path. Employing the alternative path offers several advantages:

- Reduces time spent recalculating the routing path by utilizing the alternative route instead of rerunning the routing algorithm.
- Improves response times for flow installation.
- Enhances network resilience by providing two paths, which can be beneficial in the event of a path failure.

Experiments showed that VQoSRR improved user perception of video quality and offered better control over routing and resource allocation and the videos delivered with DR-RA had higher quality than those without DR-RA or VQoS-RR.

Table 4 outlines a comprehensive overview of studies pertaining to Traffic Optimization and Path Selection, encompassing details regarding description, objectives, and traffic routing methods. In addition, Table 5 offers a comparative analysis of decision-making criteria and Implementation levels within the Data Plane and/or Control Plane.

Table 4. Traffic Optimization and Path Selection: Proposals description, Objectives, and Techniques method Employed.

Reference	Proposal description	Objectives	Techniques
Jiameng and Sang-Hwa [30]	srTCM+ GTB is a traffic-aware quality-of-service control mechanism for software-defined networking-based virtualized networks.	To provide a traffic-aware quality-of-service control mechanism for software-defined networking-based virtualized networks.	Traffic-Aware Multi-Path Routing with (QoS)-based Routing
Junjie et.al. [34]	Develop a collaborative traffic transmission mechanism utilizing MPTCP and SR to address resource consumption issues in an SDN-based DCN.	Providing a better traffic management solution, which is still effective in DCNs' peak hours.	Hybrid Multi-Path Routing
Shengru et. al. [37]	Design protocol oblivious source routing (POSR) by proposing a protocol-agnostic, bandwidth-optimized, and flow-table-conserving packet forwarding approach.	Addressing the issue of scalability in SDN to significantly mitigate flow-table consumption, minimize path setup latency, and expedite link failure recovery.	Source-Based Multi-Path Routing
Xuya et. al. [40]	Introduces a heuristic approach for the gradual deployment of SDN switches in hybrid SDNs to intelligently power off redundant	Attain energy efficiency through the proficient rerouting of flows and the optimal	Traffic-Aware Multi-Path Routing with Energy-Efficient

	links and switches in the network to effectively conserve energy resources.	shutdown of idle links and switches whenever feasible.	
Wu et. al. [41]	DAMR is a dynamic and adaptive multi-path routing algorithm that addresses packet loss, time delay, and bandwidth constraints in multimedia applications through centralized routing computations and real-time network state updates.	Improving network performance by dynamically selecting multiple paths while considering network congestion and link quality.	Dynamic Multi-Path Routing with (QoS)-based Routing
Muteb and Abdelmounaam [43]	DR-APSP is an efficient routing module devised to elevate the POX controller version to POX-PLUS, thereby facilitating the computation and perpetual maintenance of inter-node paths within the Software-Defined Networking (SDN) framework.	Preserving the integrity of the shortest paths and flow rule tables following an update operation, obviating the necessity of recomputing paths.	Dynamic Routing
Yi-Ren et. al. [45]	RL-Routing represents a proficient approach to routing within SDN, employing Reinforcement Learning (RL) to predict the future behaviour of the underlying network and suggests optimal routing paths between switches.	Mitigate scalability challenges in routing through the automation of path selection, thereby minimizing the reliance on manual configuration and upkeep.	Machine Learning-based Routing
Gururaj et.al. [46]	The REMTC algorithm is a resource-efficient multicast tree construction model that constructs a reliable and scalable multicast tree for multiple receivers. This, in turn, optimizes resource consumption and enhances bandwidth efficiency.	Improving network resources utilization and communication through Constructing a stable multicast tree with more common paths to reach multicast participants.	Traffic Engineering
Majda et.al. [50]	VQoSRR is a novel model utilizing queue mechanisms to meet bandwidth guarantees for video traffic, while also offering two routing paths between the source and destination to cater to multiple QoE constraints.	Improving the quality of video streaming services in SDN networks by providing adaptive QoS routing and resource reservation.	Quality of Service (QoS)-based Routing

Table 5. Traffic Optimization and Path Selection: Decision-Making Criteria and Implementation in the Data Plane and /or Control Plane.

Reference	Control plane	Data plane	Decision Making based on
Jiameng and Sang-Hwa [30]	✓		The traffic load, bandwidth and latency of each virtual machine.
Junjie et.al. [34]	✓	✓	Dividing the main flow into multiple subflows, and transmitting using MPTCP, while implementing an ordered list of routing instructions for packets using SR.
Shengru et. al. [37]		✓	Redesign the source packet header format and design a packet processing pipeline that is optimized for unicast and multicast traffic.
Xuya et. al. [40]		✓	Sorting all flows based on their priority and then sequentially rerouting them until all requests are organized. This entails selecting SDN switches for deployment based on flow requirements and the associated switch costs.
Wu et. al. [41]	✓	✓	Calculating the aggregated bandwidth using real-time monitoring information and a periodic scheduling strategy, taking into consideration both congested and uncongested scenarios.
Muteb and Abdelmounaam [43]	✓		Detection of network link events, which subsequently trigger the removal of all impacted and obsolete flow rules that do not contribute to any of the established shortest paths following the update operation.
Yi-Ren et. al. [45]	✓		The RL agent learns to select paths that optimize network performance metrics, such as latency, throughput, and packet loss. The agent anticipates future network behaviour to propose improved routing paths between switches.
Gururaj et.al. [46]	✓	✓	During the tree formation phase, the algorithm calculates the shortest path based on link cost.
Majda et.al. [50]	✓	✓	Dividing the network into multiple virtual networks and allocating resources to each virtual network. A bandwidth reservation mechanism reserves the required bandwidth for video streaming traffic, and a packet scheduling mechanism prioritizes video streaming traffic over other traffic types.

4) Evaluation Tools and Performance Results

4.1) Evaluation Tools

Table 6 provides a comprehensive overview of simulation model tools, datasets, and metrics used to evaluate the proposed research in traffic load balancing and traffic resilience. The table is organized by simulation model tool, followed by dataset and metrics [51].

Table 6. Summarizes the simulation model tools, dataset and metrics used to evaluate traffic load balancing and traffic resilience.

Reference	simulation model tools	Dataset	Measured metrics
Hong et. al. [11]	Mininet [52], Floodlight controller [53] and OpenvSwitch [54].	12:30 clients	Server response time, CPU utilization and Memory utilization.
André and Fernando [12]	Mininet, OpenvSwitch, ZooKeeper 3.4.8 (REF) and iperf.	Two controllers and 2:64 switches	Throughput and failover time.
Anees et. al. [15]	Mininet, GNS3, Ryu controller, OpenvSwitch and Linux traffic control tool 'tc'.	single network interface and multiple network interfaces	Throughput and file download time vs different file sizes.
Hamza et al. [17]	Mininet and Floodlight controller [53]	Two real network topologies, **Cernet**: 36 nodes & 53 links. and **DFN**:58 nodes & 87 links.	Throughput - Packet loss - Response time - Migration cost Overhead.
Chunlin et.al. [18]	Mininet, Ryu controller [53]	Internet2 OS3E: 34 nodes and 42 links. Janetbackbon: 22 nodes and 35 links.	Controller-to- switch delay - Propagation delay between controllers - Average task completion time – Load balancing degree.
Jehad and Byeong-hee [21]	Mininet and OpenvSwitch	linear topology up to 500 sensor nodes	Throughput - E2E Delay – CPU utilization - recovery latency in case of link failure
Jehad et.al. [25]	Mininet, ONOS controller [55], iperf [56]traffic generation.	Network Topology **DFN**: 58 switches, 87 links. **OS3E**: 34 switches, 42 links. **RedIris**: 19 switches, 32 links. **Interoute**:110 switches, 149 links. **Abilene**: 11 switches ,1 link.	E2E latency - communication cost - load curve for controllers- Migration time - Response time - CPU utilization – Jain's fairness index (JFI).

Table 7 presents a comprehensive overview of simulation model tools, datasets, and metrics utilized for evaluating the proposed research in traffic optimization and path selection. The table is organized based on the simulation model tool, followed by the dataset and metrics.

Table 7. Summarizes the simulation model tools, dataset and metrics used to evaluate traffic optimization and path selection.

Reference	Simulation model tools	Dataset	Measured metrics
Jiameng and Sang-Hwa [30]	MiniNet, OVX virtualization platform and iperf	TCP and UDP performance for randomly three virtual networks	TCP & UDP throughputs and data arrival rate
Junjie et.al. [34]	NS-3.26 [57]	*4k* fat tree topology with 20 switches and 64 hosts	Throughput, average link utilization
Shengru et. al. [37]	POF switches, POF controller and iPerf.	14 POF switches	Throughput, Path Setup Latency, Packet Loss Ratio for Link

			Failure and Average Failure Recovery Time
Xuya et. al. [40]		ISP 1755 and the ISP 3967 network topologies [37,38]	Energy saving ratio – Number of controlled flow - Upgrading cost
Wu et. al. [41]	Mininet, Foodlight1.2 controller and OpenvSwitch	Custom network topology with 100 Mbps for all links.	Throughput -Time delay – Packet loss rate
Muteb and Abdelmounaam [43]	Mininet, POX controller [55]	**SYNTH1**: 100 switches, 11 hosts, and 129 randomly generated links. **AS:** 100 switches, 2 hosts, and 205 randomly generated links.	CPU time, init time, add time, delete time and deleted paths time.
Yi-Ren et. al. [45]	Mininet, Ryu controller	Fat-tree, NSFNet, and ARPANet network topologies	Reward is a score computed - Utilization rate is calculated in the destination switch
Gururaj et.al. [46]	Mininet	250, 500, 750,1000,2000 and 3000 nodes with 10,25,50,75 and 100 Destinations	number of links, processing latency and rerouting cost
Majda et.al. [50]	Mininet, Floodlight controller and Iperf	Three video types: Video 1, SD, Video 2, HD And Video 3, HD	Structural Similarity Index Metric (SSIM), Mean Opinion Score (MOS) and Video Multimethod Assessment Fusion (VMAF)

4.2) Performance results analysis

This section presents a comprehensive overview of the compared schemes, demonstrated results, and drawbacks and limitations of the proposed traffic load balancing & traffic resilience solution and traffic optimization and path selection solution as shown in table 8 and table 9 respectively.

Table 8. Describes the demonstrated results, and drawbacks of the proposed traffic load balancing and traffic resilience solution.

Reference	Compared with	The demonstrated results	Drawbacks& limitations
Hong et. al. [11]	Round Robin, Random	LBBSRT has significant advantages compared to other schemes in terms of the overall response times and achieves a much better effect of load balancing. Overcomes the drawbacks of traditional methods, including high cost, low reliability, and poor extensibility.	A slight difference in memory utilization at 50% and CPU utilization at 75% for the servers that had been experimented. Using a server response time as the main metric for load balancing might not be universally suitable for all network environments and applications.
André and Fernando [12]	Ravana	Rama, a resilient SDN controller platform, offers comparable robustness as Ravana, rather, it performed better throughput, ensuring fault tolerance without	Network overhead due to increased network message exchanges. Need to evaluate the impact of network latency.

		necessitating modifications to switches or the OpenFlow protocol.	
Anees et. al. [15]	MPTCP approach and traditional WRR, RR and MBW approach.	a significant 55% increase in achieved throughput compared to the traditional single network approach, surpassing the performance of Multipath TCP (MPTCP) by 10%. This improvement can be attributed to the Weighted Round Robin (WRR) method, which allocates TCP flows based on computed link weights, resulting in better performance than MPTCP when using a single network interface.	Network Dependency: Any network failures or communication issues between the end hosts and the controller could disrupt traffic control and lead to service interruptions due to the proposed mechanism based on the external network interfaces modification of the data plane switch.
Hamza et al. [17]	DDS, SMDM, EASM [	The MTLB scheme stands out among all other evaluated schemes with its high average throughput. It also achieves the lowest average packet delay, outperforming SMDM and EASM. MTLB scheme demonstrates the lowest migration cost, packet loss and communication overhead	- Further studies are required to precisely determine the optimal intervals and identify efficiency levels. - The current load distribution operates reactively, triggered only when thresholds are exceeded, potentially resulting in spikes in controller load and performance degradation. - Additionally, the scheme does not consider the variability in flow processing times, where certain flows demand more processing than others, leading to uneven load distribution.
Chunlin et.al. [18]	K-means, SA and ECMP. LBRA, CARA and SCA	- The improvement of LOCP algorithm slightly outperforms in propagation delay, queuing delay, and load balancing degree in large-scale networks, resulting in an average improvement of 18.36% in load balancing while ensuring a lower propagation delay and queuing delay. -The average user completion delay growth rate of the HACA improved algorithm is lower than other algorithms, including 50.39% lower than that of the LBRA and 32.29% lower than that of the SCA algorithm.	- Consider more metrics to be evaluated like network throughput. - - The experimental setup lacks consideration for crucial factors such as security concerns, the influence of mobile devices' motion trajectory, and the task priority of mobile devices on MEC servers. - An implementation in the case of network failures is required to evaluate the reliability of algorithms' improvement.
Jehad and Byeong-hee [21]	AHP and EB-TOPSIS	The results show that the proposed controller outperforms AHP and EB-TOPSIS strategies in terms of delay reduction in flow request management and load balancing features. The proposed controller maintains a consistent throughput, exhibits a faster start, and demonstrates better CPU utilization, even with increased traffic. The recovery latency time of the proposed controller is smaller than that	Focusing on controller selection may limit the applicability of the scheme to a specific aspect of SD-IoT, and it may not address other important aspects such as security or energy efficiency.

		of the other strategies in case of link failures.	
Jehad et.al. [25]	SASLB, DLB, SCLB and SMLB	The response time results indicate ESCALB's efficacy, attributed to its efficient slave controller selection. ESCALB consistently outperforms other schemes in terms of end-to-end (E2E) latency for all network topologies. On the other hand, SASLB demonstrates lower communication costs but suffers from uneven packet distribution among SDN controllers, unlike ESCALB.	With a rising number of nodes and links in the network of the five topologies, the JFI, is becoming lower. However, ESCALB strategy preserves it as near to 1 as possible, it may cause ineffective packet distribution among controllers. ESCALB did not achieve optimal CPU utilization

Table 9. Describes the demonstrated results, and drawbacks of the proposed traffic optimization and path selection solution.

Reference	Compared with	The demonstrated results	Drawbacks& Limitations
Jiameng and Sang-Hwa [30]	CBBPM	- TCP throughput achieves significant improvements in VN1 (160%), VN2 (36%), and VN3 (17%). - VN1 shows 30%-100% better UDP throughput using the proposed srTCM+ GTB compared to CBBPM. Similarly, VN2 and VN3 experience approximately 4%-25% improvements with proposed srTCM+ GTB over CBBPM. - Packet loss demonstrates the superior performance of srTCM+ GTB.	- Sensitive to network latency. - Difficult to scale to large networks. - It is not as well-documented as some other bandwidth management solutions.
Junjie et.al. [34]	SingleTCP and MPTCP	Proposed MPTCP & SR demonstrated a positive correlation with increasing transmission demand, outperforming SingleTCP and MPTCP, which showed slower increases and throughput reduction, respectively. the proposed method demonstrated higher average link utilization compared to MPTCP, achieved through dynamic path adjustment, leading to better load distribution and reduced delays.	Increased overhead by increasing the packet header size. Need a Multi-controller implementation ensures high scalability for DCNs.
Shengru et. al. [37]	Traditional OpenFlow-based OF-SP.	- Achieves 100% receiving throughput with fewer flow entries. - The ability to share flow entries on intermediate switches reduces the number of flow entries required. - Achieves shorter path setup latency, especially under higher traffic loads. - Outperforms OF-SP in terms of packet loss ratio. - Achieves shorter average failure recovery time than OpenFlow.	- Improve the performance of POF Switch to make process packets more efficient and work smoothly for switches equipped with 10GbE NICs. - May increase SDN management complexity for network admins, and introduce computational or memory overhead, affecting SDN performance.
Xuya et. al. [40]	PLSP and EA-FA.	- EPC achieves effective control over 95% of flows with only 10% of the upgrading cost, saving an additional	Focusing only on the energy-saving aspect of hybrid SDN networks, it may not provide a

		10% of total power consumption compared to existing solutions. - The EPC consistently outperforms both PLSP and EA-FA in energy-saving ratio.	comprehensive analysis of all the factors that affect the performance of these networks.
Wu et. al. [41]		- DAMR outperforms ECMP and single-path methods in congested scenarios. - DAMR provides smooth, jitter-free performance, increased bandwidth, and improved system throughput. - 35%:70% improvement in quality-of-service with DAMR.	- The primary emphasis is on unicast algorithms to ensure end-to-end Quality of Service (QoS) while not addressing multicast algorithms. - Extend the research to incorporate varying priorities for different flows, including prioritizing specific flows and implementing diverse QoS strategies to meet bandwidth or delay constraints.
Muteb and Abdelmounaam [43]	l2 learning, l2 multi, and l3 learning	DR-APSP outperforms other approaches in terms of speed by a factor of 4 to 10, while also being more cost-effective in terms of recomputing shortest paths.	- DR-APSP's routing approach based on the APSP algorithm may not be optimal for all network types and traffic patterns due to its limited versatility. Additionally, its computationally intensive dynamic routing algorithm demands significant processing power and memory, potentially compromising performance in large networks.
Yi-Ren et. al. [45]	OSPF and LL.	- RL-Routing obtains higher rewards on all three network topologies. - RL-Routing minimizes the file transmission time on all three network topologies. - RL-Routing avoids congested paths. - Hosts re-transfer fewer packets with RL-Routing than with baseline solutions.	Deploy RL-Routing in a real network environment and evaluate it on other topologies.
Gururaj et.al. [46]	SPT, ST, BAERA, and OBSTA	- REMTC constructs an optimal, scalable, and stable multicast topology. - REMTC selects the trunk path based on the device count, unlike other methods that rely on optimization techniques. - REMTC's rerouting time is notably quicker than ST, taking only 20-25% of the time. This improved efficiency is due to the trunk path algorithm, which prioritizes common links to reach multicast participants. - REMTC's distinctive features contribute to reducing the alteration time during dynamic join and leave events.	- Applying a machine learning-based approach to predict optimal paths and adapt to changing network conditions. - The adaptation of REMTC for dynamic traffic in SDN requires the processing of real-time network data and the construction of multicast trees in real time. This can be computationally expensive, especially in large and complex networks, and may not scale well.
Majda et.al. [50]	No VQoSRR, VQoS-R and VQoS-RR	- VQoS-RR increased SSIM quality values for SD and HD videos. - VQoSRR improved average MOS by 36.0% for SD and 56% for HD.	- Not covered the impact of network failures and on the performance of the VQoSRR model.

	Subjective MOS evaluations better-reflected differences compared to objective SSIM when comparing results with the proposed VQoSRR. - DR-RA achieved the highest HD SSIM at 7 seconds and the best SD result at 5 seconds, suggesting longer intervals for HD rerouting. - DR-RA achieved high-quality videos with improved SSIM average values.	- The experimental results are limited to a specific set of video durations. Further analysis is needed to assess the performance of the proposed model for long-duration videos and different resolutions. - The integration of the proposed model with the multicast transmission is a promising direction for enhancing smart city tools and applications, particularly for surveillance systems in hospitals and civil defence organizations. - Extend the research to investigate the scalability of the proposed model in large-scale network environments.

5) Conclusion and Future Directions

In conclusion, Software-Defined Networks (SDN) have wrought a transformative impact on network management through the segregation of the control layer and forwarding devices, yielding centralized oversight and adaptable traffic control. This paper has highlighted the significance of Load Balancing & Resilient traffic and traffic optimization in the context of Traffic Routing within SDN. By means of a comprehensive evaluation and juxtaposition of scholarly discourse, a diverse array of techniques and resolutions for traffic routing within SDN has been explored. Nonetheless, it is paramount to recognize that the domain of SDN is characterized by perpetual evolution, ushering in novel challenges and prospects in tandem with the progress of SDN development. The establishment of standardized SDN components and the assimilation of protocols tailored to SDN constitute pivotal undertakings, indispensable for surmounting legacy network-related issues. Subsequent research ought to center around the control plane, endeavouring to conceive innovative controller solutions, which function as the linchpin of the SDN architecture. Mitigating the control plane's susceptibility as a solitary point of failure and effectuating robust security measures stand as imperative requisites. Additionally, the integration of high availability (HA) mechanisms and performance enhancement strategies to adhere to the stipulations of service level agreements (SLAs) and to efficaciously furnish services should be of paramount concern. In summation, this exposition furnishes a comprehensive survey of traffic routing methodologies within SDN. Nonetheless, the road ahead remains replete with imperatives encompassing standardization, control plane augmentation, fortification of security protocols, and optimization of performance. It is incumbent upon forthcoming research endeavours to address these challenges, in order to further advance the capabilities and reliability of SDN networks.

REFERENCES

[1] D. M. Casas-Velasco, O. M. C. Rendon and N. L. S. d. Fonseca, "DRSIR: A Deep Reinforcement Learning Approach for Routing in Software-Defined Networking," in *IEEE Transactions on Network and Service Management,* 2022.

[2] I. Wahid, S. Tanvir, A. Hameed and M. Ahmad, "Software-Defined Networks and Named Data Networks in Vehicular Ad Hoc Network Routing: Comparative Study and Future Directions," *Security and Communication Networks,* vol. 2022, pp. 1-14, 2022.

[3] 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.

[4] Y. Li, D. Zhang, J. Taheri and K. Li, "SDN components and OpenFlow," in *Big Data and Software Defined Networks*, London, IET Digital Library, 2018, p. 49–68.

[5] A. Nayyer, A. K. Sharma and L. K. Awasthi, "Issues with Routing in Software Defined Networks," in *2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, Waknaghat, India, 2020.

[6] ONF, "Open Networking Foundation," 15 April 2015. [Online]. Available: https://opennetworking.org/wp-content/uploads/2013/07/OpenFlow1.3.4TestSpecification-Basic.pdf. [Accessed 22 October 2022].

[7] A. 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.

[8] M. Jammal, T. Singh, A. Shami, R. Asal and Y. Li, "Software defined networking: State of the art and research challenges," *Computer Networks,* vol. 72, pp. 74-98, 2014.

[9] Q. Waseem, S. S. Alshamrani, K. Nisar, W. I. Sofiah, W. Din and A. S. Alghamdi, "Software-Defined Network (SDN) Forensic. Symmetry," *Future Technology,* vol. 13, no. 5, pp. 767 - 785, 2021.

[10] A. Hodaei and S. Babaie1, "A Survey on Traffic Management in Software-Defined Networks: Challenges, Effective Approaches, and Potential Measures," *Wireless Personal Communications,* vol. 118, no. 2, p. 507–1534, 2021.

[11] H. Zhong and J. C. Yaming Fang, "Reprint of ‘‘LBBSRT: An efficient SDN load balancing scheme based on server response time’’," *Future Generation Computer Systems,* vol. 80, p. 409–416, 2018.

[12] A. Mantas and F. M. V. Ramos, "Rama: Controller Fault Tolerance in Software-Defined Networking Made Practical," 2019.

[13] 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.

[14] 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.

[15] A. Al-Najjar, F. H. Khan and M. Portmann, "Network traffic control for multi-homed endhosts via SDN," *IET Communications,* vol. 14, no. 19, pp. 3312-3323, 2020.

[16] M. Scharf and A. Ford, "Multipath TCP (MPTCP) Application Interface Considerations," RFC6897, IETF, 2013.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] m. Belkadi and Y. Laaziz, "A Systematic and Generic Method for Choosing A SDN Controller," *Journal of Computer Networks and Communications,* vol. 5, pp. 239-247, 2017.

[24] D. Kannan and R. Thiyagarajan, "Entropy based TOPSIS method for controller selection in software defined networking," *oncurrency and Computation: Practice and Experience,* vol. 34, 2021.

[25] 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.

[26] A. Abdelaziz, A. T. Fong, A. Gani, U. Garba, S. Khan, A. Akhunzada, H. Talebian and K.-K. R. Choo, "Distributed controller clustering in software defined networks," *PLoS ONE,* vol. 12, no. 4, pp. 1-12, 2017.

[27] Y. Zhou, M. Zhu, L. Xiao, L. Ruan, W. Duan, D. Li, R. Liu and M. Zhu, "A Load Balancing Strategy of SDN Controller Based on Distributed Decision," in *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, Beijing, China, 2014.

[28] T. Hu, P. Yi, Z. Guo, J. Lan and Y. Hu, "Dynamic slave controller assignment for enhancing control plane robustness in software-defined networks," *Future Generation Computer Systems,* vol. 95, p. 681–693, 2019.

[29] S. K. Sagar, P. Deepak, T. Mayank, U. Muhammad, S. Bibhudatta and W. Zhenyu, "ESMLB: Efficient Switch Migration-Based Load Balancing for Multicontroller SDN in IoT," *Internet of Things Journal,* vol. 7, no. 7, pp. 5852-5860, 2020.

[30] J. Shi and S.-H. Chung, "A traffic-aware quality-of-service control mechanism for software-defined networking-based virtualized networks," *International Journal of Distributed Sensor Networks,* vol. 13, no. 3, 2017.

[31] J. Heinanen and R. Guerin, "A Single Rate Three Color Marker," RFC 2697, IETF, 1999.

[32] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "An Architecture for Differentiated Services," RFC 2475, IETF,1998.

[33] L. B. Lim, L. Guan, A. Grigg, I. W. Phillips, X. Wang and I. U. Awan, "RED and WRED Performance Analysis Based on Superposition of N MMBP Arrival Proccess," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, Perth, WA, Australia, 2010.

[34] J. PANG, G. XU and X. FU, "SDN-Based Data Center Networking With Collaboration of Multipath TCP and Segment Routing," *IEEE Access,* vol. 5, pp. 9764-9773, 2017.

[35] A. Ford, C. Raiciu, M. Handley and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses," RFC 6824, IETF, 2013.

[36] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona and P. Francois, "The Segment Routing Architecture," in *2015 IEEE Global Communications Conference (GLOBECOM)*, San Diego, CA, USA, 2015.

[37] S. Li, K. Han, F. Nirwan Ansari, Q. Bao, D. Hu, J. Liu, S. Yu and Z. Zhu, "Improving SDN Scalability with Protocol-Oblivious Source Routing: A System-Level Study," *IEEE Transactions on Network and Service Management,* vol. 15, no. 1, pp. 275-288, March 2018.

[38] S. Li, K. Han, H. Huang, Q. Sun, J. Liu, S. Zhao and Z. Zhu, "SR-PVX: A Source Routing Based Network Virtualization Hypervisor to Enable POF-FIS Programmability in vSDNs," *IEEE Access,* vol. 5, pp. 7659-7666.

[39] N. Huin, M. Rifai, F. Giroire, D. L. Pachec, G. Urvoy-Keller and J. Moulierac, "Bringing Energy Aware Routing Closer to Reality with SDN Hybrid Networks," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference, Singapore*, Singapore, 2017.

[40] X. Jia, Y. Jiang, Z. Guo, G. Shen and L. Wang, "Intelligent path control for energy-saving in hybrid SDN networks," *Computer Networks,* vol. 131, pp. 65-76, 2018.

[41] W. Jiawei, Q. Xiuquan and N. Guoshun, "Dynamic and adaptive multi-path routing algorithm based on software-defined network," *International Journal of Distributed Sensor Networks,* vol. 14, no. 10, 2018.

[42] "Equal-Cost Multipath Routing in Data Center Network Based on Software Defined Network," in *2018 6th International Conference on Information and Communication Technology (ICoICT)*, Bandung, Indonesia, 2018.

[43] M. Alshammari and A. Rezgui, "POX-PLUS: An SDN Controller with Dynamic Shortest Path Routing," in *IEEE 9th International Conference on Cloud Networking (CloudNet)*, Piscataway, NJ, USA, 2020.

[44] M. Alshammari and A. Rezgui, "An All Pairs Shortest Path Algorithm for Dynamic Graphs," *International Journal of Mathematics and Computer Science,* vol. 15, no. 1, p. 347–365, 2020.

[45] 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.

[46] G. Bijur, M. Ramakrishna and K. A. Kotegar, "Multicast tree construction algorithm for dynamic traffic on software defined networks," *Scientific Reports,* vol. 11, no. 1, 2021.

[47] M. Ramakrishna and A. K. Karunakar, "SIP and SDP based content adaptation during real-time video streaming in Future Internets," *Multimedia Tools and Applications,* vol. 76, no. 20, p. 21171–21191, 2017.

[48] L.-H. Huang, H.-J. Hung, C.-C. Lin and D.-N. Yang, "Scalable Steiner Tree for Multicast Communications in Software-Defined Networking," *arXiv:1404.3454,* 2014.

[49] S.-H. Chiang, J.-J. Kuo, S.-H. Shen, D.-N. Yang and W.-T. Chen, "Online Multicast Traffic Engineering for Software-Defined Networks," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, , Honolulu, HI, USA, 2018.

[50] M. O. Elbasheer, A. Aldegheishem, N. Alrajeh and J. Lloret, "Video Streaming Adaptive QoS Routing with Resource Reservation (VQoSRR) Model for SDN Networks," *Electronics,* vol. 8, no. 11, pp. 1252 - 1275, 2022.

[51] M. S. Islam, M. Al-Mukhtar, M. R. K. Khan and M. Hossain, "A Survey on SDN and SDCN Traffic Measurement: " Eng 4, no. 2: 1071-1115. https://doi.org/10.3390/eng4020063," *Existing Approaches and Research Challenges,* vol. 4, no. 2, pp. 1071-1115, 2023.

[52] N. Gupta, M. S. Maashi, S. Tanwar, S. Badotra, M. Aljebreen and S. Bharany, "A Comparative Study of Software Defined Networking Controllers Using Mininet.," *Electronics,* vol. 11, no. 7, pp. 2715-2752, 2022.

[53] Y. Li, X. Guo, X. Pang, B. Peng, X. Li and P. Zhang, "Performance Analysis of Floodlight and Ryu SDN Controllers under Mininet Simulator," in *2020 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, Chongqing, China, 2020.

[54] X. Zhang, K. Salamatian and G. Xie, "Improving Open Virtual Switch Performance Through Tuple Merge Relaxation in Software Defined Networks," *IEEE Transactions on Network and Service Management,* vol. 19, no. 3, pp. 2078-2091, 2022.

[55] A. L. Stancu, S. Halunga, A. Vulpe, G. Suciu, O. Fratu and E. C. Popovici, "A comparison between several Software Defined Networking controllers," in*," 2015 12th International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS)*, Nis, Serbia, 2015.

[56] J. Dugan, S. Elliott, B. A. Mah, J. Poskanzer and K. Prabhu, "iperf.fr," iperf, [Online]. Available: https://iperf.fr/. [Accessed 2022].

[57] "Ns-3.26 Release Notes.," Available online: https://www.nsnam.org/releases/ns-3-26, accessed on 1/8/2023.

[58] Y. Zongying, L. Haoming and Z. Qiaoying, "The China Education Research Network (CERNET) and library services," *MCB UP Ltd,* 1998.

[59] L. Jing and O. Stephansson, "Discrete Fracture Network (DFN) Method," *Developments in Geotechnical Engineering,* vol. 85, pp. 365-398, 2007.