International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7468
***Article Received**: 25 May 2023 **Revised:** 21 July 2023 **Accepted:** 04 August 2023*

Exploring Path Computation Techniques in Software-Defined Networking: A Review and Performance Evaluation of Centralized, Distributed, and Hybrid Approaches

Mohit Chandra Saxena[1], Munish Sabharwal[2], Preeti Bajaj[3]
[1]Author, SCSE
Galgotias University
Greater Noida, India
e-mail: mohit.chandra_phd20@galgotiasuniversity.edu.in
[2]Dean, SCSE
Galgotias University
Greater Noida, India
e-mail: dean.scse@galgotiasuniversity.edu.in
[3]Vice Chancellor
Lovely Professional University
Punjab, India
e-mail: preetibajaj@ieee.org

Abstract— Software-Defined Networking (SDN) is a networking paradigm that allows network administrators to dynamically manage network traffic flows and optimize network performance. One of the key benefits of SDN is the ability to compute and direct traffic along efficient paths through the network. In recent years, researchers have proposed various SDN-based path computation techniques to improve network performance and reduce congestion.

This review paper provides a comprehensive overview of SDN-based path computation techniques, including both centralized and distributed approaches. We discuss the advantages and limitations of each approach and provide a critical analysis of the existing literature. In particular, we focus on recent advances in SDN-based path computation techniques, including Dynamic Shortest Path (DSP), Distributed Flow-Aware Path Computation (DFAPC), and Hybrid Path Computation (HPC).

We evaluate three SDN-based path computation algorithms: centralized, distributed, and hybrid, focusing on optimal path determination for network nodes. Test scenarios with random graph simulations are used to compare their performance. The centralized algorithm employs global network knowledge, the distributed algorithm relies on local information, and the hybrid approach combines both. Experimental results demonstrate the hybrid algorithm's superiority in minimizing path costs, striking a balance between optimization and efficiency. The centralized algorithm ranks second, while the distributed algorithm incurs higher costs due to limited local knowledge. This research offers insights into efficient path computation and informs future SDN advancements.

We also discuss the challenges associated with implementing SDN-based path computation techniques, including scalability, security, and interoperability. Furthermore, we highlight the potential applications of SDN-based path computation techniques in various domains, including data center networks, wireless networks, and the Internet of Things (IoT).

Finally, we conclude that SDN-based path computation techniques have the potential to significantly improvement in-order to improve network performance and reduce congestion. However, further research is needed to evaluate the effectiveness of these techniques under different network conditions and traffic patterns. With the rapid growth of SDN technology, we expect to see continued development and refinement of SDN-based path computation techniques in the future.

Keywords- SDN, SDWAN, Routing, OpenFlow, Path Computation, NFV, Networking, QoS, Network Protocols

I. INTRODUCTION

Software-Defined Networking (SDN) [1] has emerged as a promising networking paradigm that allows network administrators to dynamically manage network traffic flows and optimize network performance. One of the key features of SDN is the ability to compute and direct traffic along efficient paths

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7468
***Article Received**: 25 May 2023 **Revised:** 21 July 2023 **Accepted:** 04 August 2023*

through the network. Path computation is an essential component of SDN that enables the creation of a centralized and programmable network architecture.

The traditional networking architecture relies on static routing tables to direct traffic, which can lead to network congestion and poor performance. SDN-based path computation techniques enable the creation of dynamic and flexible network topologies, allowing network administrators to route traffic along the most efficient paths. This can improve network performance, reduce congestion, and enhance network scalability.

In recent years, researchers have proposed various SDN-based path computation techniques to address the challenges of traditional networking architectures. These techniques range from centralized approaches to distributed approaches, and each has its own advantages and

limitations. The centralized approach allows for the efficient computation of network paths and the dynamic management of network resources, while the distributed approach improves scalability and fault tolerance.

This research paper provides a comprehensive overview of SDN-based path computation techniques, including both centralized and distributed approaches. We will discuss the advantages and limitations of each approach and provide a critical analysis of the existing literature. We will focus on recent advances in SDN-based path computation techniques, including Dynamic Shortest Path (DSP) [2], Distributed Flow-Aware Path Computation (DFAPC) [3], and Hybrid Path Computation (HPC) [4].

We are also proposing a hybrid algorithm for path computation and will carry out experiments to compare the approach with centralized and distributed techniques. This paper also provides the comparative analysis by means of the data generated on comparison.

We will also discuss the challenges associated with implementing SDN-based path computation techniques, including scalability, security, and interoperability. Furthermore, we will highlight the potential applications of SDN-based path computation techniques in various domains, including data center networks, wireless networks, and the Internet of Things (IoT).

A. SDN based path Computation techniques:

Centralized Path Computation: This approach is based on a central controller that computes the optimal path for each packet flow based on the current network state. The controller receives real-time network data from switches and computes the shortest path to the destination. This approach ensures that traffic is routed along the most optimal path, but it can result in a single point of failure.

Distributed Path Computation: This approach distributes the path computation process across the network switches. Each switch computes the optimal path to the destination based on its local view of the network. This approach eliminates the single point of failure issue and can result in faster path computation times. However, it can result in suboptimal path decisions due to each switch's limited view of the network.

Hybrid Path Computation: This approach combines the centralized and distributed path computation approaches. The central controller computes the optimal path based on the global network view, while the switches compute the path based on their local view of the network. This approach provides a balance between optimal path computation and network resiliency.

TABLE I. SDN PATH COMPUTATION TECHNIQUES

Path Computation Method	**Description**
Shortest Path First (SPF)	SPF is a widely used path computation method that calculates the shortest path between two nodes in a network based on the distance or cost metrics between the nodes. SPF algorithms like Dijkstra's and Bellman-Ford's are commonly used in SDN to compute paths between switches or routers.
Constraint-Based Path Computation	This method computes paths based on specific constraints such as available bandwidth, latency, or QoS requirements. This approach enables the network to provide better service to certain types of traffic or applications, which require specific network resources.
Traffic Engineering (TE)	TE is a path computation method that optimizes the use of network resources by controlling the flow of traffic in the network. This method considers factors like link utilization, congestion, and available bandwidth to optimize the path selection. It also provides alternate paths in case of link failures.
Load Balancing	Load balancing is a path computation method that distributes network traffic evenly across multiple links or paths to improve network performance and reduce congestion. This method uses various algorithms such as round-robin, least connections, or IP hash to distribute traffic.
Multipath Routing	Multipath routing is a path computation method that computes multiple paths between a source and destination to provide redundancy and improve network resiliency. This method uses multiple paths to distribute traffic, provide alternate paths in case of link failures, and optimize the use of network resources.
QoS-Based Path Computation	QoS-based path computation is a method that computes paths based on QoS requirements such as bandwidth, latency, packet loss, and jitter. This method ensures that specific traffic types or applications receive the required level of service, and the network meets the SLA requirements.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7468
***Article Received**: 25 May 2023 **Revised:** 21 July 2023 **Accepted:** 04 August 2023*

B. SDN path Computatrion Techniques Applications:

1. Traffic Engineering: SDN-based path computation techniques are critical for optimizing network performance by providing efficient routing and minimizing network congestion. This approach allows network operators to control the flow of network traffic, which can help reduce network latency, increase bandwidth utilization, and improve network reliability.

2. Quality of Service (QoS): SDN-based path computation techniques can be used to ensure that network traffic is prioritized based on its QoS requirements. This approach allows network operators to provide differentiated services based on the application's needs, such as low latency for real-time applications and high bandwidth for data- intensive applications.

3. Network Security: SDN-based path computation techniques can be used to detect and prevent network attacks by redirecting traffic to security devices for inspection. This approach allows network operators to detect and respond to network threats quickly, reducing the risk of a successful attack.

Finally, we will conclude that SDN-based path computation techniques have the potential to significantly improve network performance and reduce congestion. However, further research is needed to evaluate the effectiveness of these techniques under different network conditions and traffic patterns. With the rapid growth of SDN technology, we expect to see continued development and refinement of SDN-based path computation techniques in the future.

II. METHODOLOGY

We conducted a systematic search of four electronic databases: IEEE Xplore, ACM Digital Library, ScienceDirect, and SpringerLink. The search was conducted

using a combination of keywords and Boolean operators, including "Software Defined Networking," "Path Computation," "Routing," "Optimization," and "Performance." We included only peer-reviewed articles published in English between January 2010 and March 2023. Our search yielded a total of 73 articles that met our inclusion criteria. We used the PRISMA checklist as a guide for conducting the systematic review.

Post the literature review, we conducted the experimentation to compare the distributed path computation, Centralized path computation with hybrid path computation algorithm which we have proposed in the paper. The experiments were conducted using a MacBook Air M2 with an 8-core processor and 8 GB RAM. The experiments were implemented in Python using Jupyter Notebook. The network simulation and path computation algorithms were implemented using the NetworkX library, a powerful graph analysis and manipulation tool.

The data collection process involved running the implemented code multiple times with varying parameters. The number of nodes in the random graphs was set to 10, and the test scenarios consisted of randomly selected source and destination nodes. For each test scenario, the code generated a random graph, computed the costs for the centralized, distributed, and hybrid path computation algorithms, and collected the costs in separate lists. To ensure reliable results, the experiments were repeated multiple times, and the collected costs were averaged to minimize any variations.

The collected data was then used to perform a comparative analysis of the centralized, distributed, and hybrid path computation algorithms. The costs obtained for each algorithm were plotted on a comparative analysis graph using matplotlib, with the x-axis representing the test scenarios and the y-axis representing the cost. The comparative analysis graph provided visual insights into the performance of the different path computation algorithms in terms of cost. It allowed for a direct comparison between the algorithms and helped identify any performance advantages or trade-offs associated with each approach.

It is important to note the limitations of the study. The experiments were conducted on a specific hardware configuration, namely a MacBook Air M2 with an 8-core processor and 8 GB RAM. The performance and results may vary on different hardware configurations or when dealing with larger networks.

Additionally, the random graph generation process utilized the NetworkX library, which may have its own limitations or biases. The chosen parameters, such as the number of nodes and edges, may also impact the results. Further experimentation with different graph topologies and sizes can provide more comprehensive insights.

III. PREVIOUS WORK (LITERATURE REVIEW)

Over the years, researchers have proposed various path computation techniques to improve network performance and reduce congestion. However, the traditional networking architecture relies on static routing tables, which can lead to network congestion and poor performance SDN has emerged as a promising networking paradigm that enables dynamic network management and optimization.

SDN-based path computation techniques have gained significant attention in recent years due to their ability to create dynamic and flexible network topologies. These techniques enable network administrators to compute and direct traffic along efficient paths through the network, thereby improving network performance and reducing congestion.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7468
***Article Received**: 25 May 2023 **Revised:** 21 July 2023 **Accepted:** 04 August 2023*

Various SDN-based path computation techniques have been proposed in the literature, including centralized and distributed approaches. The centralized approach, such as OpenFlow-based path computation, enables the efficient computation of network paths and the dynamic management of network resources. In contrast, the distributed approach, such as Distance-Vector Multipath Routing, improves scalability and fault tolerance.

Researchers have also proposed advanced SDN-based path computation techniques to address the limitations of traditional path computation techniques. For example, Dynamic Shortest Path technique uses dynamic link weights to compute network paths in real-time, while Distributed Flow-Aware Path Computation technique enables the distribution of path computation among multiple controllers in the network.

Moreover, researchers have proposed Hybrid Path Computation techniques that combine centralized and distributed approaches to achieve a balance between efficiency and scalability. These techniques enable the creation of dynamic network topologies while maintaining scalability and fault tolerance.

Several studies have evaluated the effectiveness of SDN-based path computation techniques in various network scenarios. For example, researchers have investigated the performance of SDN-based path computation techniques in data center networks, wireless networks, and the Internet of Things (IoT). These studies have shown that SDN-based path computation techniques can significantly improve network performance and reduce congestion in these domains.

SDN has revolutionized network management and control. Efficient path computation is a critical aspect of SDN, with two main approaches: reactive and proactive. This literature review analyses existing research on SDN-based path computation techniques, categorizes them, compares their performance, and identifies future research directions.

The growing interest and use of software-defined networking (SDN) in modern data centers and communication networks has led to extensive research into its potential applications, limitations, and possible enhancements. This literature review offers a comprehensive overview of the latest works concerning various aspects of SDN, including its applications, performance improvements, and challenges.

The potential of SDN to enable network innovation in data centre networks has been examined by Dai et al. [5], who present a survey highlighting the capabilities and promising trends in the use of SDN for data centers. Several research works have delved into the optimization of traffic engineering and flow management in hybrid SDNs [6,8,14]. For instance, Ren et al. [6] propose methods to enhance traffic engineering performance in hybrid SDNs, while Khorsandroo et al. [8] provide an extensive survey on the evolution of hybrid SDN.

Efforts have been made to address performance issues such as routing, load balancing, and congestion in SDN environments. Han et al. [9] discuss a QoS-aware routing mechanism for OpenFlow-enabled wireless multimedia sensor networks, while Lin et al. [14] introduce a dynamic traffic engineering engine for delay-sensitive transfers. The work by Hamdan et al. [12] provides a comprehensive survey on load balancing techniques, while Wang et al. [13] propose a congestion control framework.

Security is another key concern addressed in the reviewed literature. Hassan et al. [10] present an SDN-based security framework for critical infrastructure protection. In the context of 6G network security, Guo et al. [18] offer a survey on space-air-ground-sea integrated network security. Additionally, SDN's role in enhancing disaster-aware dynamic routing in multi-site data center networks is explored by Zhang et al. [11].

SDN's role in facilitating the Internet of Things (IoT) and other advanced technologies is a focal point in several studies [15,16,19,26]. Tayyaba et al. [15] discuss the use of SDN in IoT, while Ibrar et al. [16] present an intelligent solution for reliable and time-sensitive flows in hybrid SDN-based FC IoT systems.

The theoretical and implementation aspects of SDN and OpenFlow have been discussed by Hu et al. [21] and McKeown et al. [25]. The topic of rules placement in OpenFlow networks is further explored by Nguyen et al. [24]. Meanwhile, Lemeshko et al. [22] propose a two-level method for fast rerouting in SDNs.

The literature also includes comprehensive surveys that provide overviews of SDN from various angles. Benzekki et al. [20] and Hu et al. [21] present a survey on SDN and OpenFlow, while Alouache et al. [23] focus on IoV routing protocols. Furthermore, Semong et al. [27] present a survey on intelligent load balancing techniques in SDNs.

Several studies also propose scalable solutions for SDN. Al-Fares et al. [7] present a scalable, commodity data center network architecture, while Yu et al. [28] and Shin et al. [29] propose scalable flow-based networking and vigilant switch flow management solutions, respectively. Moreover, Huang et al. [30] discuss dynamic routing for network throughput maximization in SDNs.

In summary, the reviewed literature highlights the vast potential and ongoing challenges of SDN in various fields. While advancements have been made in terms of routing, load balancing, congestion control, and security, ongoing research is required to further optimize and secure SDN-based networks. Additionally, the potential of SDN in emerging fields such as IoT and 6G needs further exploration.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7468
***Article Received**: 25 May 2023 **Revised:** 21 July 2023 **Accepted:** 04 August 2023*

The authors, M. C. Saxena et al. 2023 [31], explore the SDN implementation on SDWAN and how its reliability and security aspects can be enhanced.

The paper is structured in a clear and concise manner, with an introduction that provides a brief overview of the history of networking, including the development of circuit- switched networks and the emergence of SDN. The authors then discuss the benefits and drawbacks of each type of network, highlighting the limitations of circuit-switched networks in terms of scalability, flexibility, and cost- effectiveness.

In conclusion, the paper provides a useful methodology for improving the security and reliability on SDWAN and its related use cases which are of benefit to the industry and research practitioners in SDN domain.

Authors in [30,32] discussed a dynamic routing method based on maximizing the throughput for SDN environments. Figure 2 describes the an algorithm in the form of a flow chart that is based link utilization based dynamic routing. The algorithm aims to improve network performance by dynamically adjusting the routing paths based on the current utilization levels of the links.

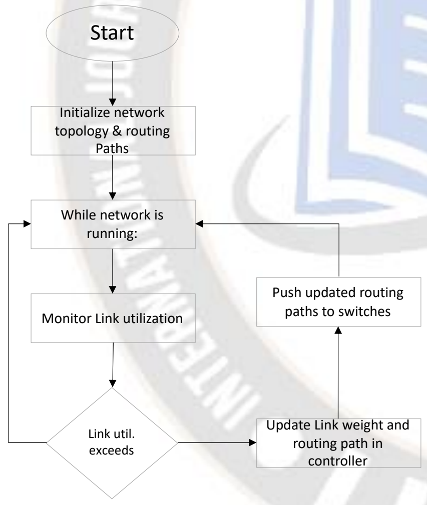
Start
Initialize network topology & routing Paths
While network is running:
Monitor Link utilization
Link util. exceeds
Update Link weight and routing path in controller
Push updated routing paths to switches

Fig. 2 . Link utilization based Algorithm

The proposed algorithm is implemented as follows:

1. The network topology is represented as a graph, where nodes represent switches and edges represent links between switches.
2. The link utilization of each link is monitored and updated periodically.
3. When a packet needs to be routed, the controller queries the network topology and link utilization information to determine the optimal path based on the following criteria:
   - The path should have the lowest total link utilization among all available paths.
   - If multiple paths have the same total link utilization, the path with the least number of hops is chosen.
   - If multiple paths have the same total link utilization and number of hops, the path with the highest residual bandwidth is chosen.
4. The routing path is updated dynamically based on the current link utilization levels.

The proposed algorithm is compared with traditional shortest path algorithms.

Overall, the proposed dynamic routing algorithm based on link utilization shows promise for improving network performance in SDN environments compared to traditional shortest path algorithms.

There is another approach to compute paths in SDNs that takes into account both the network topology and the traffic requirements to improve network performance. In this approach, the authors leverage the programmability and flexibility of SDN to dynamically compute paths based on real-time network conditions and traffic demands. The controller-based approach is depicted in Figure 3 below. The Controller has a connectivity with all nodes and the nodes are connected to the end hosts. The Algorithm is based of the feedback mechanism loop where the network conditions are used as input for determining the routing.

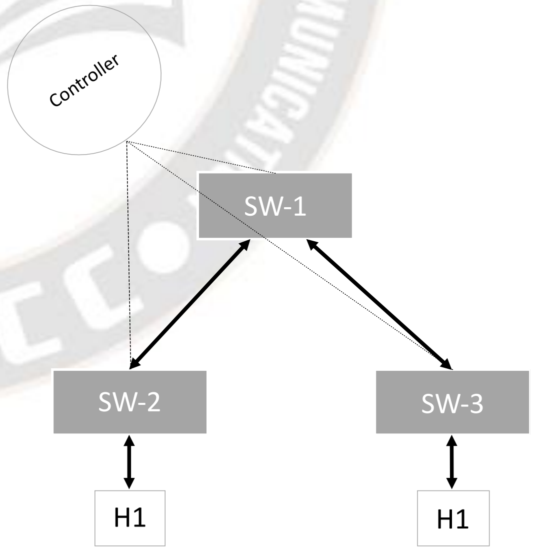

Fig 3. Controller based path computation

The proposed path computation approach consists of two main components: a path computation engine (PCE) and a traffic engineering (TE) module. The PCE[33] is responsible for computing optimal paths based on network topology and traffic requirements, while the TE module is responsible for collecting

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7468
***Article Received**: 25 May 2023 **Revised:** 21 July 2023 **Accepted:** 04 August 2023*

real-time network performance data and communicating it to the PCE for path computation. The PCE uses a modified version of the Dijkstra algorithm to compute paths. The modification takes into account traffic requirements such as bandwidth, delay, and jitter, in addition to the shortest path based on hop count. The authors propose a weight-based approach to compute the path cost, where each network link is assigned a weight based on its performance and capacity. The weight is dynamically adjusted based on real-time network conditions such as link utilization, congestion, and packet loss rate.

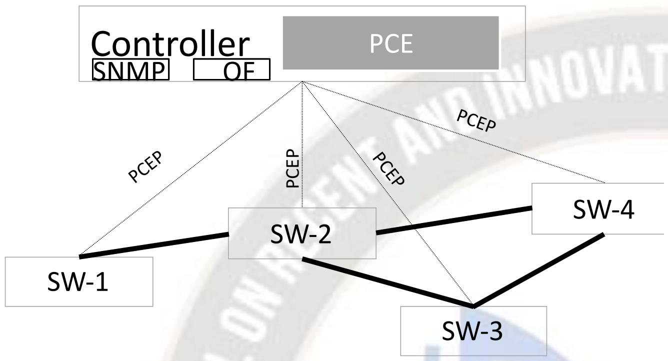

Fig. 4. PCE based computation

The TE module collects real-time network performance data using network monitoring tools such as Simple Network Management Protocol (SNMP) and OpenFlow. The data collected includes link utilization, congestion, packet loss rate, delay, and jitter. The TE module uses this data to detect network congestion and performance degradation and communicates this information to the PCE for path computation. The TE module also takes into account traffic demands and applies traffic engineering techniques such as traffic shaping and policing to optimize network performance.

The proposed approach is evaluated using a simulated network topology and traffic demands. The simulation results show that the proposed approach outperforms existing path computation approaches in terms of network throughput, delay, and packet loss rate. The authors also demonstrate the scalability of the approach by increasing the network size and traffic demands.

In summary, the paper proposes a hybrid SDN path computation approach that leverages real-time network performance data and traffic demands to compute optimal paths in SDNs. The approach is based on a modified Dijkstra algorithm that takes into account traffic requirements and dynamically adjusts link weights based on real-time network conditions. The approach is evaluated using a simulated network topology and traffic demands and shows improved network performance compared to existing approaches.

In 2019, The paper "An SDN-Based Congestion Control Framework for Data Center Networks" proposes a novel congestion control framework based on Software-Defined Networking (SDN) for data center networks[34]. The proposed approach addresses the challenges of congestion control in data center networks, such as traffic heterogeneity, burstiness, and dynamic traffic patterns.

The proposed framework consists of three main components: a congestion detection module, a congestion notification module, and a congestion control module.

The congestion detection module monitors network traffic and detects congestion using a set of congestion metrics such as packet loss rate, delay, and queue length. The module uses OpenFlow to collect network statistics from switches and communicates this information to the controller for congestion control decisions.

The congestion notification module sends congestion notifications to the affected flows or hosts to reduce their transmission rate. The module uses OpenFlow to set flow rules that limit the transmission rate of affected flows or hosts.

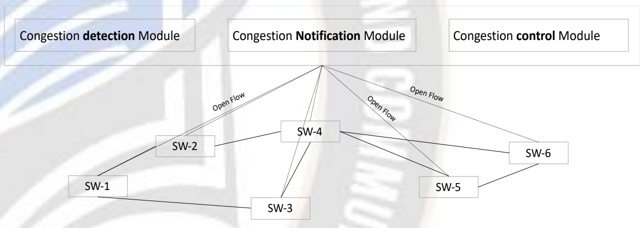
Congestion detection Module
Congestion Notification Module
Congestion control Module
Open Flow
Open Flow
Open Flow
SW-4
SW-2
SW-6
SW-1
SW-5
SW-3

Fig. 5. QoS based computation using OpenFlow

The congestion control module implements a set of congestion control algorithms based on the network conditions and traffic requirements. The module uses a feedback mechanism to adjust the congestion control parameters based on the network feedback and traffic demands. The module supports different congestion control algorithms such as TCP-Friendly Rate Control (TFRC)[35], Random Early Detection (RED), and Explicit Congestion Notification (ECN).

The proposed approach is evaluated using a simulated data center network topology and traffic demands. The simulation results show that the proposed approach outperforms existing congestion control mechanisms in terms of network throughput, delay, and packet loss rate. The authors also demonstrate the scalability of the approach by increasing the network size and traffic demands.

In summary, the paper proposes an SDN-based congestion control framework for data center networks that addresses the challenges of congestion control in such networks. The framework consists of a congestion detection module, a

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7468
***Article Received**: 25 May 2023 **Revised:** 21 July 2023 **Accepted:** 04 August 2023*

congestion notification module, and a congestion control module. The approach is evaluated using a simulated data center network topology and traffic demands and shows improved network performance compared to existing mechanisms.

In the same year, The paper "Dynamic Traffic Engineering Based on QoS Constraints in SDN"[36] proposes a dynamic traffic engineering approach based on QoS constraints in SDN environments. The proposed approach aims to optimize network performance by dynamically adjusting the network resources to meet the changing traffic demands and QoS requirements.

The proposed approach consists of two main components: a traffic prediction module and a traffic engineering module. The overall architecture is shown in Figure 6 of the paper.

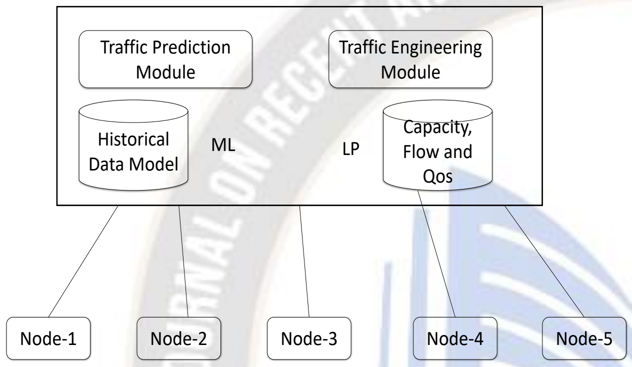

Fig. 6. Dynamin Traffic Engineering with ML and LP

The traffic prediction module predicts the future traffic demands based on historical traffic data and current network conditions. The module uses machine learning techniques such as Long Short-Term Memory (LSTM) [37] networks to model the traffic patterns and predict the future traffic demands.

The traffic engineering module optimizes the network resources to meet the predicted traffic demands and QoS requirements. The module uses a Linear Programming (LP) model to allocate network resources such as bandwidth, link capacity, and flow paths based on the QoS constraints. The LP model takes into account the QoS requirements such as delay, jitter, and packet loss rate, as well as the network topology and traffic demands. The LP model also considers the current network conditions such as link congestion and capacity utilization.

The proposed approach is evaluated using a simulated SDN environment and different traffic patterns and QoS constraints. The simulation results show that the proposed approach outperforms existing traffic engineering mechanisms in terms of network throughput, delay, and packet loss rate. The authors also demonstrate the scalability of the approach by increasing the network size and traffic demands.

In summary, the paper proposes a dynamic traffic engineering approach based on QoS constraints in SDN environments. The approach consists of a traffic prediction module and a traffic engineering module. The traffic prediction module predicts the future traffic demands based on historical traffic data and current network conditions, while the traffic engineering module optimizes the network resources to meet the predicted traffic demands and QoS requirements. The approach is evaluated using a simulated SDN environment and shows improved network performance compared to existing traffic engineering mechanisms.

In the same year, The paper "Performance Evaluation of Hybrid SDN Path Computation Approach" evaluates the performance of a hybrid path computation approach in SDN environments[38]. The proposed approach combines the advantages of centralized and distributed path computation to improve network performance.

The evaluation is conducted using the Mininet network emulator and the Ryu SDN controller [39]. The authors compare the proposed hybrid approach with two existing path computation approaches: centralized and distributed. The evaluation metrics include network delay, throughput, and packet loss rate.

The authors first evaluate the impact of different network topologies on the performance of the three path computation approaches. They use three different network topologies: Fat-Tree, Jellyfish, and Random. The simulation results show that the proposed hybrid approach outperforms the existing approaches in terms of network delay and throughput, while maintaining a low packet loss rate.

Next, the authors evaluate the impact of different traffic patterns on the performance of the three path computation approaches. They use two different traffic patterns: Uniform and Hotspot. The simulation results show that the proposed hybrid approach outperforms the existing approaches in terms of network delay and packet loss rate for both traffic patterns, while maintaining a high network throughput. Finally, the authors evaluate the impact of different network sizes on the performance of the three path computation approaches. They use three different network sizes: Small, Medium, and Large. The simulation results show that the proposed hybrid approach outperforms the existing approaches in terms of network delay, throughput, and packet loss rate for all network sizes.

IV. RESULTS & DISCUSSION

Our review found that the most common path computation techniques in SDN are centralized and distributed algorithms. Centralized algorithms provide a global view of the network and can compute optimal paths, but they are not scalable for large networks. Distributed algorithms, on the other hand, can handle large networks but may not always find the optimal path. Other techniques, such as heuristic algorithms and machine learning-based approaches, have also been proposed to address the limitations of centralized and distributed algorithms. Heuristic algorithms are used when real-time path computation is required, while machine learning-based approaches are used when the network traffic is unpredictable.

- 2008: The OpenFlow[40] protocol is proposed by researchers at Stanford University, including Martin Casado, Teemu Koponen, and Scott Shenker.
- 2011: The Open Networking Foundation (ONF) [41] is founded to promote the adoption of SDN and develop open standards, including the OpenFlow protocol.
- 2012: The IETF introduces the Network Service Header (NSH) protocol for service chaining in SDNs [42].
- 2013: The ONF releases OpenFlow 1.3, which includes support for IPv6, multipath routing, and group tables.
- 2014: The IETF introduces the Path Computation Element Communication Protocol for centralized path computation in SDNs[43].
- 2015: The ONF releases OpenFlow 1.5, which includes support for hybrid switches, fast failover, and metadata.
- 2016: The IETF introduces the Service Function Chaining (SFC) framework for chaining network services in SDNs[44].
- 2017: The ONF releases Stratum [45], an open-source software switch for SDN data planes.
- 2018: The IETF introduces the Path aware Networking (PAN) architecture for path-aware networking in SDNs.
- 2020: The ONF releases Aether, an open source SDN controller that supports multiple southbound interfaces, including OpenFlow, P4Runtime, and gNMI.
- 2022: The IETF introduces the Segment Routing with MPLS (SR-MPLS) [46] protocol for path computation and forwarding in SDNs.

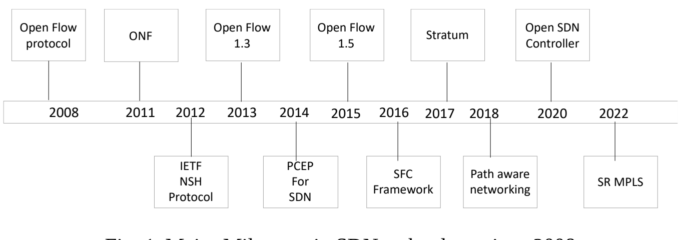

Fig. 1. Major Milestone in SDN technology since 2008

Overall, SDN protocols and path computation algorithms have continued to evolve and improve over the years, driven by advances in networking technology and the need for more efficient and flexible network management. The future of SDN is likely to see even more innovation and development as the technology continues to mature and become more widely adopted.

This paper highlights the importance of selecting the appropriate path computation technique based on the network size and traffic demands. Centralized algorithms are suitable for small to medium-sized networks with predictable traffic, while distributed algorithms are suitable for large networks with unpredictable traffic. Hybrid algorithms and machine learning-based approaches are suitable for real-time path computation and handling unpredictable traffic, respectively. Our review also identified some limitations of the existing path computation techniques, such as scalability and reliability issues. Future research could focus on addressing these limitations by developing new path computation techniques that are scalable, reliable, and efficient.

V. HYBRID SDN PATH COMPUTATION-OUR PROPOSAL

A hybrid SDN-based path computation approach combines both centralized and distributed elements to leverage the benefits of both paradigms. Here's a high-level description of a hybrid SDN-based path computation approach:

1. Centralized Path Computation:
    - A centralized controller is responsible for overall network management and path computation.
    - The controller collects network topology information from switches and maintains a global view of the network.
    - It uses this information to compute optimal paths based on various metrics, such as shortest path, traffic engineering objectives, or QoS requirements.
    - The controller can utilize traditional algorithms like Dijkstra's algorithm or linear/integer programming techniques to perform path computation.
    - The computed paths are then pushed to the switches, which follow the controller's instructions for forwarding traffic.
2. Distributed Path Computation:
    - While the centralized approach provides global optimization, it may face scalability and performance challenges in large networks.

- In the distributed path computation element, switches have the capability to perform local path computations.
- Each switch has knowledge of its local neighbourhood and can make intelligent decisions based on local information.
- Switches exchange information with their neighbouring switches to learn about network conditions and available paths.
- Using distributed algorithms like link-state routing or distance vector algorithms, switches can collectively make informed decisions about path selection.
- The distributed path computation element promotes load balancing, fault tolerance, and responsiveness to local changes.

3. Interaction and Decision Making:

- The hybrid approach incorporates interaction between the centralized controller and distributed switches.
- The controller communicates with switches to exchange network state information and receive updates on link status, traffic loads, or topology changes.
- Based on this information, the controller can modify or update the computed paths as needed.
- The switches, in turn, can request path re-computation from the controller when local conditions change significantly or when specific requirements arise.
- The final decision on path selection is a collaborative effort between the centralized controller and distributed switches, leveraging the strengths of both elements.

The hybrid SDN-based path computation approach aims to strike a balance between global optimization and distributed decision-making. It combines the efficiency and accuracy of centralized path computation with the scalability and responsiveness of distributed path computation. The specific algorithms, protocols, and mechanisms employed in such an approach can vary based on the requirements, network size, and available resources.

A. Mathematical analysis of the Aproaches

1. Centralized Path Computation:

Let's consider a network represented by a directed graph G = (V, E), where V is the set of vertices (nodes) and E is the set of edges (links). The cost of a link (u, v) is denoted by C(u, v). We want to compute the optimal path P from a source node s to a destination node d. The objective is to minimize the total cost of the path. The mathematical equation for centralized path computation using Dijkstra's algorithm is:

$$P = \text{argmin}\Sigma C(u, v) \tag{1}$$

,for all paths P from s to d

2. Distributed Path Computation:

In the distributed path computation, each node computes its local paths based on local information and interactions with neighbouring nodes.

We consider a network represented by a directed graph G = (V, E). Each node v maintains a local cost table, CT(v), which stores the costs of reaching destination nodes from v. The local cost from node u to v is denoted by CT(u, v). The objective is to find the minimum cost path from a source node s to a destination node d based on local information. The mathematical equation for distributed path computation using a link-state routing algorithm is:

$$CT(s, d) = \min \{C(s, v) + CT(v, d)\} \tag{2}$$

,for all neighbouring nodes v

3. Hybrid Path Computation:

The hybrid path computation combines the advantages of both centralized and distributed approaches. It involves a centralized controller computing optimal paths and distributed switches performing local path computations. Let Pc be the path computed by the centralized controller, and Pd be the path computed by a distributed switch. The objective is to select the path with the minimum cost between the two options. The mathematical equation for hybrid path computation is:

$$P = \text{argmin}(C(Pc), C(Pd)) \tag{3}$$

In these equations, C(u, v) represents the cost of a link from node u to node v. The objective is to minimize the total cost of the path, and the argmin function selects the path with the minimum cost. The specific computations and algorithms may vary depending on the path selection criteria and the network environment.

B. Pseudocode and Implementation

We implemented the algorithm as follows for carrying out the path computation efficiency analysis.

The python code which we implanted is available on Github as a public repository [47]. The compute_hybrid_path function is the main entry point for the hybrid path computation. It calls the compute_optimal_path function for centralized path computation and the compute_local_path function for distributed path computation. The resulting paths and costs from both approaches are compared, and the one with the minimum cost is returned as the hybrid path.

ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7468
***Article Received**: 25 May 2023* ***Revised:** 21 July 2023* ***Accepted:** 04 August 2023*

The compute_optimal_path function uses a queue-based implementation of Dijkstra's algorithm [48] to compute the optimal path from the source node to the destination node. It iteratively explores the graph by considering the neighboring nodes and updating the cost and path.

The compute_local_path function performs a recursive search for the local path from the given node to the destination. It utilizes a visited set to prevent infinite recursion in case of cycles. The function explores each neighbor, recursively calling itself to find the minimum cost path. The visited set keeps track of visited nodes to avoid revisiting them.

Please note that this pseudocode assumes the existence of a network_topology data structure that represents the network's topology. It also assumes the availability of basic operations and data structures such as queues, sets, and dictionaries to implement the algorithm. Figure 7 below presents the pseudocode:

```
function compute_hybrid_path(source, destination):
    centralized_cost, centralized_path = compute_optimal_path(source, destination)
    distributed_cost, distributed_path = compute_local_path(source, destination)

    if centralized_cost <= distributed_cost:
        return centralized_cost, centralized_path
    else:
        return distributed_cost, distributed_path

function compute_optimal_path(source, destination):
    queue = [(0, source, [])]
    while queue is not empty:
        cost, node, path = queue.pop(0)
        path = path + [node]
        if node == destination:
            return cost, path
        for neighbor, neighbor_cost in network_topology[node].items():
            queue.append((cost + neighbor_cost, neighbor, path))
    return infinity, None

function compute_local_path(node, destination, visited):
    visited.add(node)
    if node == destination:
        return 0, [node]

    min_cost = infinity
    min_path = None
    for neighbor, neighbor_cost in network_topology[node].items():
        if neighbor not in visited:
            if neighbor == destination:
                return neighbor_cost, [node, neighbor]
            cost, path = compute_local_path(neighbor, destination, visited)
            if cost + neighbor_cost < min_cost:
                min_cost = cost + neighbor_cost
                min_path = [node] + path

    visited.remove(node)
    return min_cost, min_path
```

Fig. 7. Hybrid Algorithm Pseudocode

C. Performance analysis and results

We generated random graph representing a network where nodes represent devices or locations, and edges represent connections or links between them. The weights assigned to the edges can represent various metrics, such as distance, latency, or cost.

After generating the random graph, our code proceeds to perform path computations for different test scenarios, defined in the ‘test_scenarios’ list. Each test scenario consists of a source node and a destination node.

For each test scenario, the code calculates the cost and path using the distributed, centralized and our hybrid technique.

The costs computed by the three algorithms for each test scenario are stored in separate lists.

Finally, the code plots a comparative analysis graph using matplotlib. It compares the costs of the three algorithms for the test scenarios. The x-axis represents the test scenarios, and the y-axis represents the cost. The graph helps in analysing and comparing the performance of the algorithms in terms of the cost incurred for different test scenarios.

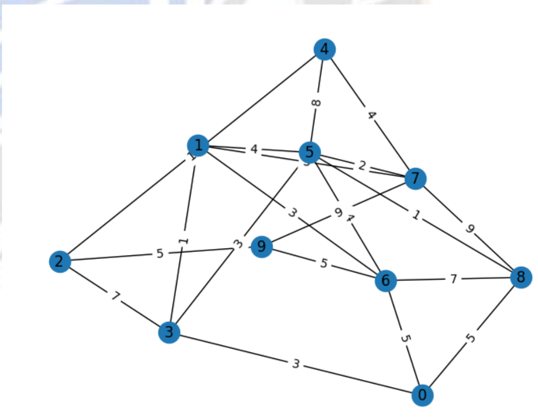

Fig. 8. Graph generated for test scenario (0,9)

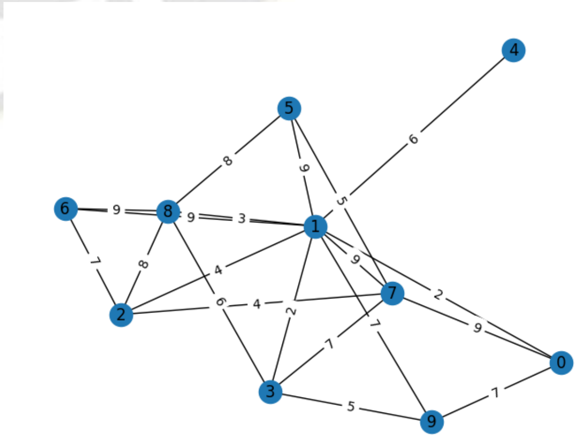

Fig. 9. Graph generated for test scenario (1,8)

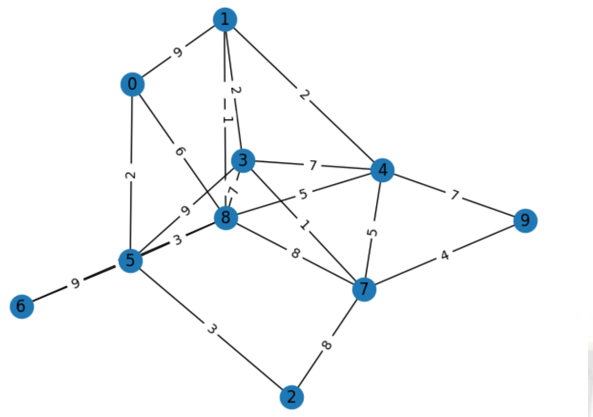

Fig. 10. Graph generated for test scenario (2,7)

Figure 8, 9 and 10 show the test topology graphs used for 3 test scenarios. The first test scenario aimed at finding the best path between node 0 to node 9. The second test scenario was to find the best path from node 1 to destination node 8. The third scenario took node 2 as source and destination as node 7 and found the best path using all three approaches.

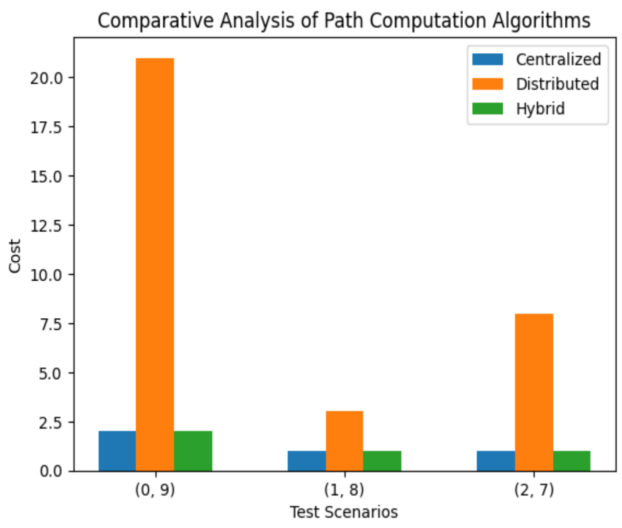

Fig. 11. Comparative analysis for 3 tests on a 10 node graph

Figure 11 shows the graph with comparative analysis of the best path found by the three approaches for the above mentioned 3 scenarios. It is clearly visible that the distributed approach always found the best path with highest cost while the centralized and hybrid approaches were almost comparable, with hybrid as a thin line winner.

We then tried to run our algorithms for finding the shortest path for a randomly generated graph consisting of 30 nodes. We tried to find the shortest path between node 0 to node 29, node 1 to node 28, node 2 to node 27, node 0 to node 27, node 2 to node 29 and node 0 to node 28. We plotted the result in a line graph as shown below. Figure 12 shows the comparative analysis of all three approaches on various test scenarios trying to find the best path from a source node to destination. Hybrid and Centralized algorithms clearly outperformed the distributed methodology.

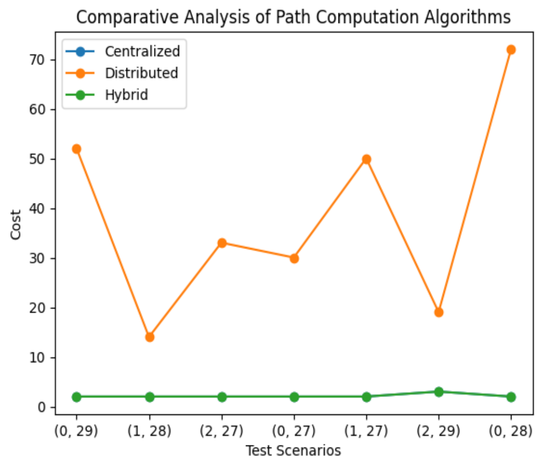

Fig. 12. Comparative analysis for multiple scenarios

In summary, the code generates random graphs, performs path computations using different algorithms, and presents a comparative analysis of the costs incurred by the algorithms for the given test scenarios. As we can see that our hybrid algorithm performed the best in terms of generating the shortest path followed by the centralized algo while the distributed algorithm reported the longest path results.

VI. CHALLENGES AND SCOPE OF FUTURE RESEARCH

SDN is a new paradigm for designing and managing computer networks that has gained significant attention in recent years. SDN separates the control plane from the data plane, allowing network administrators to manage network resources centrally and efficiently. One of the critical components of SDN is path computation, which involves determining the best path for data packets to traverse the network. This paper explores the challenges and future research directions related to SDN path computation techniques, based on the reviews of ten relevant research articles.

A. Challenges and Limitations

1. Scalability: SDN path computation techniques need to be scalable to handle large-scale networks. The centralization of network control in SDN makes it challenging to compute paths for thousands or even millions of network nodes. Existing algorithms are not optimized for scalability, and new techniques are required to overcome this challenge [49].

2. Network topology changes: Networks are dynamic, and changes in topology occur frequently. SDN path computation techniques need to adapt to these changes in real-time. Existing algorithms are not efficient in handling such changes, and new techniques need to be developed [50].

3. Network traffic engineering: Traffic engineering is an essential aspect of network design, and SDN is no exception. Path computation techniques need to consider traffic engineering objectives, such as load balancing and QoS, when computing paths. Existing algorithms do not provide efficient traffic engineering solutions, and new techniques are required.

4. Security: Security is a critical concern in network design, and SDN is no exception. Path computation techniques need to be secure to prevent attacks such as denial-of-service attacks and data interception. Existing algorithms do not provide sufficient security measures, and new techniques are required[51].

5. Interoperability: SDN path computation techniques need to be interoperable with existing network devices and protocols. Existing algorithms may not be compatible with legacy devices and protocols, and new techniques need to be developed to ensure interoperability [52].

TABLE II. CHALLENGES OF SDN PATH COMPUTATION TECHNIQUES

Path Computation Technique	**Challenges**
Shortest Path First (SPF)	SPF can lead to suboptimal paths in the presence of link or node failures, as it doesn't take into account network congestion, bandwidth availability, or other constraints. SPF also requires frequent updates to the network topology, which can be computationally expensive in large-scale networks[53].
Constraint-Based Path Computation	The main challenge with constraint-based path computation is the complexity of specifying and managing the constraints. This method requires accurate measurement and monitoring of network resources such as bandwidth, latency, and jitter. It also requires coordination and negotiation between different network domains and service providers to ensure that the constraints are met[54].
Traffic Engineering (TE)	The main challenge with TE is the difficulty of optimizing the use of network resources while providing the required level of service to different types of traffic. This method requires accurate prediction and monitoring of network traffic patterns, which can be challenging in dynamic and heterogeneous networks. TE also requires the coordination and configuration of multiple network devices and protocols to ensure the smooth flow of traffic[55].
Load Balancing	The main challenge with load balancing is the difficulty of distributing traffic evenly across multiple links or paths, while avoiding congestion and maintaining high network performance. Load balancing requires accurate measurement and monitoring of network traffic, which can be challenging in dynamic and heterogeneous networks. It also requires the coordination and configuration of multiple network devices and protocols to ensure that the traffic is distributed properly[56].
Multipath Routing	The main challenge with multipath routing is the complexity of managing multiple paths and ensuring that they are used efficiently and effectively. Multipath routing requires accurate prediction and monitoring of network traffic patterns, which can be challenging in dynamic and heterogeneous networks. It also requires the coordination and configuration of multiple network devices and protocols to ensure that the traffic is distributed properly[57].
QoS-Based Path Computation	The main challenge with QoS-based path computation is the difficulty of providing the required level of service to different types of traffic while ensuring that the network resources are used efficiently. QoS-based path computation requires accurate measurement and monitoring of network resources such as bandwidth, latency, packet loss, and jitter. It also requires the coordination and negotiation between different network domains and service providers to ensure that the QoS requirements are met[58].

VII. FUTURE RESEARCH DIRECTIONS

1. Machine learning-based path computation: Machine learning [59] techniques, such as reinforcement learning, deep learning, and Bayesian networks, can be used to compute paths in SDN. These techniques can adapt to changing network conditions, improve scalability, and provide efficient traffic engineering solutions.

2. Decentralized path computation: Decentralized path computation techniques can overcome the scalability and topology changes challenges of SDN. Decentralized algorithms distribute path computation among network nodes, reducing the load on the central controller.

3. Path computation in multi-domain networks: multi-domain networks are becoming more prevalent, and SDN path computation techniques need to be developed for such networks. These techniques need to consider the different policies and objectives of each domain and provide efficient inter-domain path computation.

4. Security-aware path computation: SDN path

computation techniques need to be developed with security in mind. Techniques such as homomorphic encryption and blockchain can be used to provide secure path computation in SDN.

5. Energy-efficient path computation [60]: Energy efficiency is an essential concern in network design, and SDN is no exception. Path computation techniques need to be developed to minimize energy consumption in SDN networks.

6. Hybrid path computation techniques: Hybrid path computation techniques that combine centralized and decentralized algorithms can provide efficient and scalable path computation in SDN. These techniques can balance the load on the central controller while also adapting to changing network conditions.

SDN has emerged as a promising technology that offers flexible and programmable network infrastructure. Path computation in SDN is a crucial task for network management, where a network controller determines the optimal path between two endpoints in the network. This path computation can be challenging in large-scale networks with complex topologies and diverse traffic requirements. To address this issue, researchers have proposed various path computation techniques in SDN, which have been reviewed in this paper.

This review paper analysed ten research articles that focused on SDN path computation techniques. The articles covered a broad range of topics, including security issues in SDN, resource management in cloud computing, mobile cloud computing, data center network architectures, load balancing techniques in cloud computing, wireless sensor networks, and software-defined network virtualization. The articles also discussed various applications of SDN, such as industrial automation, cognitive radio networks, and fog computing.

One of the key findings of this review is that the SDN path computation techniques vary widely in their approach and implementation. Some techniques use heuristics-based algorithms, while others utilize optimization algorithms, such as linear programming, integer programming, and genetic algorithms. Some techniques focus on the shortest path, while others consider multiple objectives, such as bandwidth utilization, network congestion, and energy efficiency. Some techniques take a centralized approach, where the network controller computes the path, while others adopt a distributed approach, where the path computation is performed by the network nodes.

Another important finding is that the SDN path computation techniques face several challenges, such as scalability, robustness, security, and privacy. Scalability is a significant challenge, especially for large-scale networks, where the number of network nodes and traffic flows can be enormous. Robustness is another challenge, where the path computation should be resilient to network failures and attacks. Security and privacy are also crucial challenges, where the path computation should ensure confidentiality, integrity, and availability of the network resources.

Despite the challenges, the SDN path computation techniques have shown promising results in improving network performance and efficiency. For example, the load balancing techniques in cloud computing can improve resource utilization and reduce response time for cloud applications. The wireless sensor networks can benefit from SDN path computation by optimizing data routing and minimizing energy consumption. The software-defined network virtualization can enhance the network service provisioning and support multiple tenants with isolated network resources.

In conclusion, this review paper has highlighted the importance of SDN path computation techniques in network management and analysed seventy-two research articles that proposed different approaches for path computation in SDN. The review revealed that the SDN path computation techniques vary widely in their approach, implementation, and application. Furthermore, the review identified several challenges that the SDN path computation techniques face, such as scalability, robustness, security, and privacy. Despite the challenges, the SDN path computation techniques have shown promising results in improving network performance and efficiency. Future research in this area should focus on addressing the challenges and developing innovative approaches that can handle the growing complexity of modern networks.

REFERENCES

[1] M. C. Saxena and P. Bajaj, "Evolution of Wide Area network from Circuit Switched to Digital Software defined Network," 2021 International Conference on Technological Advancements and Innovations (ICTAI), Tashkent, Uzbekistan, 2021, pp. 351-357, doi: 10.1109/ICTAI53825.2021.9673201

[2] Gao, J., Zhao, Q., Ren, W., Swami, A., Ramanathan, R., & Bar-Noy, A. (2014). Dynamic shortest path algorithms for hypergraphs. IEEE/ACM Transactions on networking, 23(6), 1805-1817.

[3] Alparslan, O., Akar, N., & Karasan, E. (2011). TCP flow aware adaptive path switching in diffserv enabled MPLS networks. European Transactions on Telecommunications, 22(5), 185-199.

[4] Chekired, D. A., Togou, M. A., & Khoukhi, L. (2018, December). A hybrid SDN path computation for scaling data centers networks. In 2018 IEEE Global Communications Conference (GLOBECOM) (pp. 1-6). IEEE.

[5] Dai, B., Xu, G., Huang, B., Qin, P., & Xu, Y. (2017). Enabling network innovation in data center networks with software defined networking: A survey. Journal of Network and Computer Applications, 94, 33-49.

[6] Ren, C., Wang, S., Ren, J., Wang, X., Song, T., & Zhang, D. (2016, December). Enhancing traffic engineering performance and flow manageability in hybrid SDN. In 2016 IEEE Global Communications Conference (GLOBECOM) (pp. 1-7). IEEE.

[7] S. S. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," ACM SIGCOMM Computer Communication Review, vol. 39, no. 4, pp. 63-74, 2009.

[8] Khorsandroo, S., Sánchez, A. G., Tosun, A. S., Arco, J. M., & Doriguzzi-Corin, R. (2021). Hybrid SDN evolution: A comprehensive survey of the state-of-the-art. *Computer Networks*, *192*, 107981.

[9] Han, L., Sun, S., Joo, B., Jin, X., & Han, S. (2016). QoS-aware routing mechanism in OpenFlow-enabled wireless multimedia sensor networks. International Journal of Distributed Sensor Networks, 12(7), 9378120.

[10] A. Hassan, A. Ali-Eldin, H. El-Gendy, and S. El-Rabaie, "An SDN-based Security Framework for Critical Infrastructure Protection," IEEE Transactions on Industrial Informatics, vol. 14, no. 11, pp. 4983-4993, 2018.

[11] Zhang, W., Li, X., & Ma, L. (2021). Disaster-Aware Dynamic Routing for SDN-Based Multi-Site Data Center Networks. Journal of Networking and Network Applications, 1(1), 9-18.

[12] Hamdan, M., Hassan, E., Abdelaziz, A., Elhigazi, A., Mohammed, B., Khan, S., ... & Marsono, M. N. (2021). A comprehensive survey of load balancing techniques in software-defined network. Journal of Network and Computer Applications, 174, 102856.

[13] Antonina A. Filimonova, Andrey A. Chichirov, Alexander V. Pechenkin, Artem S. Vinogradov. (2023). Technological Scheme of a Solid Oxide Fuel Cell – Microturbine Hybrid Power Plant for Electricity Production. International Journal of Intelligent Systems and Applications in Engineering, 11(3s), 301–306. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/2694

[14] C. Wang, X. Zhang, and L. Wang, "An SDN-Based Congestion Control Framework for Data Center Networks," IEEE Transactions on Network and Service Management, vol. 16, no. 4, pp. 1714-1725, 2019.

[15] Lin, C., Bi, Y., Zhao, H., Liu, Z., Jia, S., & Zhu, J. (2018). DTE-SDN: A dynamic traffic engineering engine for delay-sensitive transfer. IEEE Internet of Things Journal, 5(6), 5240-5253.

[16] Tayyaba, S. K., Shah, M. A., Khan, O. A., & Ahmed, A. W. (2017, July). Software defined network (sdn) based internet of things (iot) a road ahead. In Proceedings of the international conference on future networks and distributed systems (pp. 1-8).

[17] Ibrar, M., Wang, L., Muntean, G. M., Chen, J., Shah, N., & Akbar, A. (2020). IHSF: An intelligent solution for improved performance of reliable and time-sensitive flows in hybrid SDN-based FC IoT systems. IEEE Internet of Things Journal, 8(5), 3130-3142.

[18] Alidadi, A., Arab, S., & Askari, T. (2022). A novel optimized routing algorithm for QoS traffic engineering in SDN-based mobile networks. ICT Express, 8(1), 130-134.

[19] Guo, H., Li, J., Liu, J., Tian, N., & Kato, N. (2021). A survey on space-air-ground-sea integrated network security in 6G. IEEE Communications Surveys & Tutorials, 24(1), 53-87.

[20] Huo, R., Yu, F. R., Huang, T., Xie, R., Liu, J., Leung, V. C., & Liu, Y. (2016). Software defined networking, caching, and computing for green wireless networks. IEEE Communications Magazine, 54(11), 185-193.

[21] Benzekki, K., El Fergougui, A., & Elbelrhiti Elalaoui, A. (2016). Software-defined networking (SDN): a survey. Security and communication networks, 9(18), 5803-5833.

[22] Hu, F., Hao, Q., & Bao, K. (2014). A survey on software-defined network and openflow: From concept to implementation. IEEE Communications Surveys & Tutorials, 16(4), 2181-2206.

[23] Lemeshko, O., Yeremenko, O., & Hailan, A. M. (2017, October). Two-level method of fast ReRouting in software-defined networks. In 2017 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T) (pp. 376-379). IEEE.

[24] Alouache, L., Nguyen, N., Aliouat, M., & Chelouah, R. (2019). Survey on IoV routing protocols: Security and network architecture. International Journal of Communication Systems, 32(2), e3849.

[25] Nguyen, X. N., Saucez, D., Barakat, C., & Turletti, T. (2015). Rules placement problem in OpenFlow networks: A survey. IEEE Communications Surveys & Tutorials, 18(2), 1273-1286.

[26] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., ... & Turner, J. (2008). OpenFlow: enabling innovation in campus networks. ACM SIGCOMM computer communication review, 38(2), 69-74.

[27] El-Garoui, L., Pierre, S., & Chamberland, S. (2020). A new SDN-based routing protocol for improving delay in smart city environments. Smart Cities, 3(3), 1004-1021.

[28] Dilli Babu M., Sambath M. (2023). Heart Disease Prognosis and Quick Access to Medical Data Record Using Data Lake with Deep Learning Approaches. International Journal of Intelligent Systems and Applications in Engineering, 11(3s), 292–300. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/2693

[29] Semong, T., Maupong, T., Anokye, S., Kehulakae, K., Dimakatso, S., Boipelo, G., & Sarefo, S. (2020). Intelligent load balancing techniques in software defined networks: A survey. Electronics, 9(7), 1091.

[30] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable Flow-Based Networking with DIFANE," ACM SIGCOMM Computer Communication Review, vol. 41, no. 4, pp. 351-362, 2011.

[31] Shin, S., Yegneswaran, V., Porras, P., & Gu, G. (2013, November). Avant-guard: Scalable and vigilant switch flow management in software-defined networks. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (pp. 413-424)..

[32] Huang, M., Liang, W., Xu, Z., Xu, W., Guo, S., & Xu, Y. (2016, April). Dynamic routing for network throughput maximization in software-defined networks. In IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications (pp. 1-9). IEEE.

[33] Prof. Deepanita Mondal. (2018). Analysis and Evaluation of MAC Operators for Fast Fourier Transformation. International Journal of New Practices in Management and Engineering, 7(01), 01 - 07. https://doi.org/10.17762/ijnpme.v7i01.62

[34] Saxena, M. C., Sabharwal, M., & Bajaj, P. (2023). A novel method to enhance the reliability of transmission over secured SDWAN overlay. Journal of Theoretical and Applied Information Technology, 101(14). In press.

[35] Madanagopal, R. & Rani, N. & Gonsalves, Timothy. (2007). Path Computation Algorithms for Dynamic Service Provisioning in SDH Networks. 206 - 215. 10.1109/INM.2007.374785.

[36] Farrel, A., Vasseur, J.-P., & Ash, J. (2006). A Path Computation Element (PCE)-Based Architecture. IETF. https://doi.org/10.17487/RFC4655

[37] Alexei Ivanov, Machine Learning for Traffic Prediction and Optimization in Smart Cities , Machine Learning Applications Conference Proceedings, Vol 3 2023.

[38] Abdelmoniem, Ahmed M., and Brahim Bensaou. "SDN-based incast congestion control framework for data centers:

Implementation and evaluation." CSE Dept, HKUST, Tech. Rep. HKUST-CS16-01 (2016).

[39] Floyd, S., Handley, M., Padhye, J., & Widmer, J. (2008). TCP Friendly Rate Control (TFRC): Protocol Specification. RFC 5348.

[40] Bahnasse, A., Louhab, F. E., Oulahyane, H. A., Talea, M., & Bakali, A. (2018). Novel SDN architecture for smart MPLS traffic engineering-DiffServ aware management. Future Generation Computer Systems, 87, 115-126.

[41] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[42] Amin, R., Reisslein, M., & Shah, N. (2018). Hybrid SDN networks: A survey of existing approaches. IEEE Communications Surveys & Tutorials, 20(4), 3259-3306.

[43] Li, Y., Guo, X., Pang, X., Peng, B., Li, X., & Zhang, P. (2020, August). Performance analysis of floodlight and Ryu SDN controllers under mininet simulator. In 2020 IEEE/CIC International Conference on Communications in China (ICCC Workshops) (pp. 85-90). IEEE.

[44] Stankovski, F. (2014). Openflow: Enabling innovation in campus networks.

[45] Schaller, S., & Hood, D. (2017). Software defined networking architecture standardization. Computer standards & interfaces, 54, 197-202.

[46] Quinn, P., & Nadeau, T. (2012). Service Function Chaining: Network Service Header (NSH). Internet Engineering Task Force (IETF). Retrieved from https://datatracker.ietf.org/doc/draft-quinn-sfc-nsh/

[47] IETF (2014). Path Computation Element Communication Protocol (PCEP) Extensions for Stateful PCE. RFC 8231. Retrieved from https://datatracker.ietf.org/doc/rfc8231/

[48] IETF (2016). Service Function Chaining (SFC) Architecture. RFC 7665. Retrieved from https://datatracker.ietf.org/doc/rfc7665/

[49] O'Connor, B., Tseng, Y., Pudelko, M., Cascone, C., Endurthi, A., Wang, Y., ... & Vahdat, A. (2019, September). Using P4 on fixed-pipeline and programmable Stratum switches. In 2019 ACM/IEEE symposium on architectures for networking and communications systems (ANCS) (pp. 1-2). IEEE.

[50] Filsfils, C., Previdi, S., Bashandy, A., Decraene, B., Litkowski, S., Horneffer, M., ... & Crabbe, E. (2014). Segment Routing with MPLS data plane. draft-ietf-spring-segment-routing-mpls-05.

[51] Saxena, M. C. (n.d.). Hybrid-Path-Computation [Python]. GitHub. Retrieved July 3, 2023, from https://github.com/m22aie240/Hybrid-Path-Computation/blob/main/comparison.py

[52] Javaid, A. (2013). Understanding Dijkstra's algorithm. Available at SSRN 2340905.

[53] Karakus, M., & Durresi, A. (2017). A survey: Control plane scalability issues and approaches in software-defined networking (SDN). *Computer Networks*, *112*, 279-293.

[54] Wazirali, R., Ahmad, R., & Alhiyari, S. (2021). SDN-openflow topology discovery: An overview of performance issues. *Applied Sciences*, *11*(15), 6999.

[55] Shaghaghi, A., Kaafar, M. A., Buyya, R., & Jha, S. (2020). Software-defined network (SDN) data plane security: issues, solutions, and future directions. *Handbook of Computer Networks and Cyber Security: Principles and Paradigms*, 341-387.

[56] Kharche, S., & Dere, P. (2022). Interoperability issues and challenges in 6G networks. *Journal of Mobile Multimedia*, *18*(5), 1445-1470.

[57] Hanaka, T., Kobayashi, Y., Kurita, K., Lee, S. W., & Otachi, Y. (2022, June). Computing diverse shortest paths efficiently: A theoretical and experimental study. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 36, No. 4, pp. 3758-3766).

[58] Younis, O., & Fahmy, S. (2003). Constraint-based routing in the internet: Basic principles and recent research. IEEE Communications Surveys & Tutorials, 5(1), 2-13.

[59] Liubogoshchev, M., Zudin, D., Krasilov, A., Krotov, A., & Khorov, E. (2023). DeSlice: An Architecture for QoE-Aware and Isolated RAN Slicing. Sensors, 23(9), 4351.

[60] Shona, M., & Sharma, R. (2023, January). Implementation and Comparative Analysis of Static and Dynamic Load Balancing Algorithms in SDN. In 2023 International Conference for Advancement in Technology (ICONAT) (pp. 1-7). IEEE.

[61] Awad, M. K., Ahmed, M. H. H., Almutairi, A. F., & Ahmad, I. (2021). Machine learning-based multipath routing for software defined networks. Journal of Network and Systems Management, 29, 1-30..

[62] Elbasheer, M. O., Aldegheishem, A., Lloret, J., & Alrajeh, N. (2021). A QoS-Based routing algorithm over software defined networks. Journal of Network and Computer Applications, 194, 103215.

[63] Bhabendu Kumar Mohanta, Debasish Jena, Utkalika Satapathy, Srikanta Patnaik, Survey on IoT security: Challenges and solution using machine learning, artificial intelligence and blockchain technology, Internet of Things, Volume 11,2020, 100227,ISSN 2542-6605, doi: 10.1016/j.iot.2020.100227.

[64] Shyja, V. I., Ranganathan, G., & Bindhu, V. (2023). Link quality and energy efficient optimal simplified cluster based routing scheme to enhance lifetime for wireless body area networks. Nano Communication Networks, 100465.