IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 12, 2022, accepted May 27, 2022, date of publication June 6, 2022, date of current version June 9, 2022.

*Digital Object Identifier 10.1109/ACCESS.2022.3180333*

# Effective Flow Table Space Management Using Policy-Based Routing Approach in Hybrid SDN Network

**MANISH PALIWAL[1] AND KAPIL KUMAR NAGWANSHI[2], (Senior Member, IEEE)**
[1]Department of Computer Science Engineering, School of Technology, Pandit Deendayal Energy University, Gandhinagar, Raysan, Gujarat 382007, India
[2]ASET, Amity University Rajasthan, Jaipur, Rajasthan 303002, India
Corresponding author: Kapil Kumar Nagwanshi (dr.kapil@ieee.org)

**ABSTRACT** Software-defined networking makes forwarding devices easier to manage and provides centralized control. Because of the centralization, a network administrator can programme the devices cheaply. Network administrators make attempts to convert their entire network into SDN-compatible switches. A good balance of SDN and legacy switching functions can lead to a successful network scenario in network architecture. In this study, a hybrid network scenario is provided in which the external boundary forwarding devices of the service provider network are replaced with SDN devices. Still, the other internal forwarding devices continue to operate traditionally. The benefits of both SDN and legacy network design are combined, allowing the network administrator to reap the benefits of both. The network architecture employs a policy-based routing algorithm that takes advantage of free IP addresses from the free IP pool. The technique enables efficient use of available flow table space, which is critical in SDN architecture due to the small flow table size. The algorithm's efficiency is assessed using performance metrics such as network path stretch, throughput, latency delay, and so on, compared to traditional SDN controllers such as OpenDayLight, NOX, and POX. According to the experimental results, the suggested approach outperforms specific similar state-of-the-art techniques in the hybrid SDN domain.

**INDEX TERMS** Control plane, legacy forwarding devices, openflow, software-defined network, traffic engineering.

## I. INTRODUCTION

The legacy network architecture consists of networking devices, which are very complex in their functionality. The main reason behind the complex working is the tight integration of control and forwarding functionality. Because of this, these architectures become inflexible from the configuration point of view of the network administrator. In recent years, the current networking scenario has been moving towards a network architecture where the control logic functions of forwarding devices are decoupled from the forwarding functions. This new network scenario is called a Software-Defined Network (SDN). SDN provides a simplified and flexible network design where the complex control functionalities are detached from the hardware devices and placed in the centralized location known as the Controller. The Controller directs the data plane devices to work according to the various configured policies.

Fig. 1 provides the layered architecture of SDN. It consists of mainly three layers: Infrastructure Layer (Data Plane), Control Layer (Control Plane), and Application Layer (Management Plane). The infrastructure layer consists of the underlying physical devices. Similarly, the Control layer contains the control logic, which directs the underlying devices for packet processing. Finally, the Application layer has various modules which the network administrator defines for network functioning. For example, OpenFlow [34] is the southbound API that provides an interface between the control plane and the data plane.

Nowadays, organizations are interested in upgrading their existing network architecture into SDN-based design. However, the full up-gradation of infrastructure is associated with a massive deployment cost. This deployment cost will be significantly higher than a homogeneous environment in a heterogeneous environment. At this point, the incremental

The associate editor coordinating the review of this manuscript and approving it for publication was Paulo Mendes.

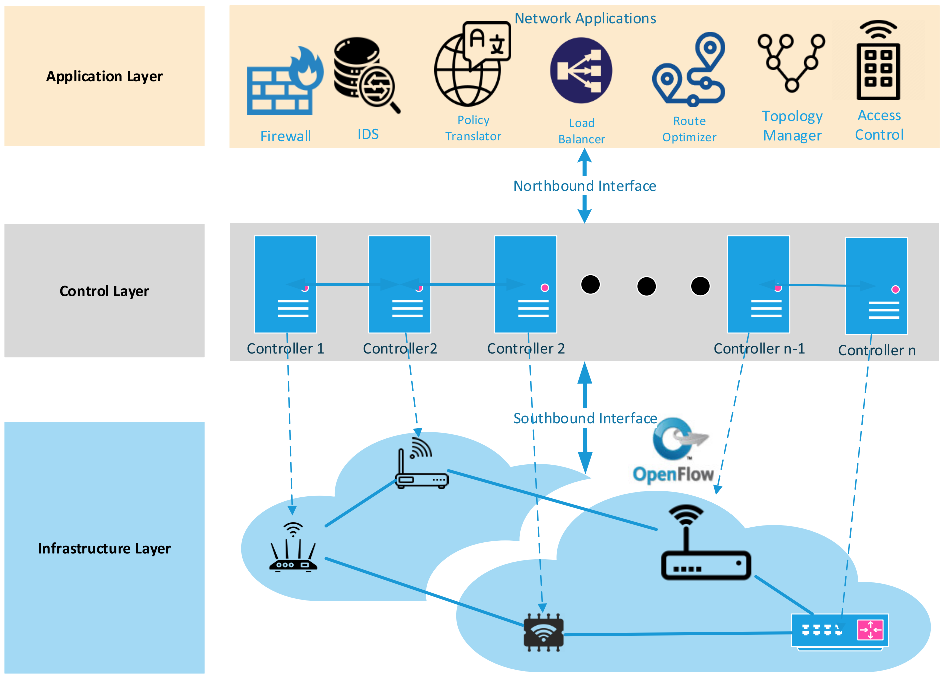

**FIGURE 1.** Layered architecture of SDN.

deployment of SDN over the traditional network would be a cost-effective solution. This hybrid network architecture will be capable of providing SDN-like features in an optimized way. Apart from it, a typical SDN device supports a flow table of size 500k to 1M flow entries. This limited flow table space sometimes bottlenecks the architecture's scalability. So, we should define the routing policies effectively to minimize the flow table entry consumption in the devices.

The article contributes to designing a policy-based routing approach in hybrid SDN architecture that systematically uses the free IP addresses available in the destination subnet. Here, the proposed policy-based routing [42], [47] defined the optimal policy path that a packet follows during its journey. The incremental deployment starts with replacing edge devices in the service provider network. The selection of edge devices is made because of their additional roles and responsibilities (e.g., bridging activity between customer and service provider networks). Besides the edge devices, all the other devices in the core network work in a legacy manner. The edge SDN devices are configured and operated from the centralized Controller using OpenFlow version 1.5.0 southbound protocol. The Controller consists of various modules, e.g., Topology Manager, Route Discovery, etc., which guide the underlying devices for their activity. Similarly, each customer network is connected to at least one SDN edge device for appropriate communication in the underlying data plane.

The paper is organized into the following different sections. Section I discuss the introduction of SDN technology and an overview of Hybrid SDN. Section II provides insights into the previous works related to hybrid SDN routing and flow table space management. Section III defines the architectural design of the proposed approach. Section IV demonstrates the methodology of the proposed design. Section V deals with the experimental results and discussions on the proposed architecture. Finally, in section VI, the paper is concluded with future work that can be achieved in the upcoming years.

## II. PREVIOUS WORKS

Routing is one of the widely studied topics in computer networking. The introduction of SDN has brought significant changes in the network routing paradigm. Gushchin *et al.* focused on middlebox activity and presented a MultiPoint-To-Point Trees (MPTPT) for SDN networks to reduce the flow table usage in the switches [22]. In [30], the authors presented a multi-AS-based routing control platform that uses the logical centralized architecture based on SDN. Kumbhare *et al.* given route segregation and prioritization approach for switch flow table utilization in SDN [31]. In ref. [28], the authors emphasized the low latency aspects of the flow table and suggested a TCAM based technique for reducing the flow table entries. In [21], the authors proposed an MPLS label-based forwarding scheme for the SDN called jump flow which reduces the bandwidth usage. It uses the VLAN identifier for carrying the routing information. Paliwal and Shrimankar suggested a practical resource management approach that depends upon the active data centre traffic and uses the hybrid SDN concept [37]. Cohen *et al.* presented a method that intends to introduce the global network objective in the limited forwarding table environment [15]. Similarly, In ref. [10], A Tag-In-Tag approach is suggested by the authors to optimize the flow table space. It is a two-layering tagging system to reduce the number of bits required for the flow.

In ref. [41], the authors suggested a hybrid SDN model with an objective of minimization of maximum link utilization by considering traffic engineering aware distributed routing. In [7], The author presented a decision tree and K-partite graph-based network policy implementation for hybrid SDN design. In [23], The author discussed the control plane operational challenge in hybrid SDN design and suggested a lightweight control plane called Hybridflow. Similarly, in ref. [6], The authors provide a brief survey about the hybrid SDN, which comprises issues. It includes network deployment strategies, controllers for hybrid SDN networks, protocols for hybrid SDN network management, traffic engineering mechanisms for hybrid SDN networks, and testing, verification, and security mechanisms for hybrid SDN networks. In ref. [38], the authors suggested a comprehensive study based on the various classical state of art SDN controllers. Finally, in ref. [49], the authors presented a hybrid SDN design mainly focused on two aspects- incremental deployment of SDN and throughput-maximization routing strategy.

Ref. [11], presented a Robust Routing Architecture for Hybrid Software-Defined and Wireless Mesh Networks (Soft-Mesh), which provide a systematic and effective transition of wireless mesh network into SDN. In ref. [20] an SDNMesh architecture is presented, which addresses the significant issues of deployment, such as resource optimization and scalability in a hybrid SDN-WMN network. Finally, in ref. [45], the authors emphasized 4G and Non-Standalone 5G support for mobile networks using Hybrid SDN Mobile Core Network design.

When using OpenFlow, network forwarding rules are translated into rule entries. A table-overflow problem might occur if the flow table's capacity is inefficiently used. A control-plane adaptive flow table management strategy (AFTM) can manage flow table resources better by combining dynamic timeout with proactive eviction [44].

According to Khorsandroo *et al.* [29] extensive network and cloud providers, including Tech Giants, have embraced Software-Defined Networking (SDN) as an evolutionary networking paradigm. SDN functions may be used in a hybrid networking environment, where legacy network infrastructures are also recognised as a compromise solution. Many enterprises and organisations are now considering hSDN as an effective networking option. It's been a long journey for HSDN, but a few areas still require additional study. They explore the use cases for high-speed distributed networking in 5G, 5G mobile networks, the cloud/data centre, and IoT. They also investigate traffic engineering, including methods for measuring and managing traffic flow and routing to maximise the quality of service. Additionally, standards and future research initiatives are addressed.

In research proposed by Osman and Mangues-Bafalluy [36], they examine the influence of unreliable controller-to-node communication channels on network performance. their hSDN system is a mix of centralised and distributed SDN. When control channel packet loss rates rose, the findings reveal that the suggested method significantly enhanced the aggregated throughput of the test system. This makes it possible to run the network even when the circumstances are difficult, while a conventional SDN control would render the network unusable. A significant aggregated throughput gain was achieved (e.g., roughly 28% for CPLR = 20% and considerably more for higher CPLR) at the expense of a modest increase in average latency in the situations examined.

SDN offers several advantages, including programme network traffic, flexibility, and automation. However, load distribution on servers serving these services in network contexts presents a distinct problem. Continuous monitoring of server load indicators and implementations of multi-parameter metrics (CPU load, I/O Read, I/O Write, Link Upload, Link Download) for connection scheduling are the foundation of the new load balancing method [33].

Key to the timely delivery of a flow is the availability of the Software-Defined Network's flow rule entry in the Flow Table. When no rules are in place to govern the flow of information, controllers are contacted for guidance. Unfortunately, no rule has yet been established at Controller by Application Plane. A technique for self-flow rule creation is introduced into the controller to minimise bottlenecks caused by a lack of rules. The study investigated an issue with the flow rules in a typical software-defined network. In the event of an attack or any other scenario involving missing rules, flow rules keep network traffic flowing. Reinforcement learning may be used to generate flow rules automatically. Automated rule creation may eventually be accomplished via different types of supervised learning approaches [26].

To calculate the average speed and find the optimal path to the target, research presents a hybrid optimization strategy that incorporates modified Ant Colony and Firefly optimization approaches (MAF). Advances in the Internet of Vehicle (IoV) are directed toward the Intelligent Transportation System (ITS) to enhance road safety and traffic flow. For the IoV environment, this paper presents some of the newest bioinspired routing algorithms. The most important future research directions in this area are underlined. To prevent traffic difficulties, vehicle routing should also consider speed restrictions, pollution inspections, and emergency responses to traffic incidents. There are several ways children's wearable sensors may be utilized to gather data, including the Internet. Emotional impulses from these activities will go via the Delta, Theta, and Alpha bands in our brains. All of these discussed are a few examples of the application of SDN [8], [17], [18], [35].

## III. ARCHITECTURAL DESIGN

Inspired by the previous work in SDN, we propose a hybrid architecture that provides SDN-like control up to a certain extent. The suggested framework supports network services like VPN [39], [46], Traffic Engineering [4], [32] run over it. Also, The framework provides a user-defined policy to

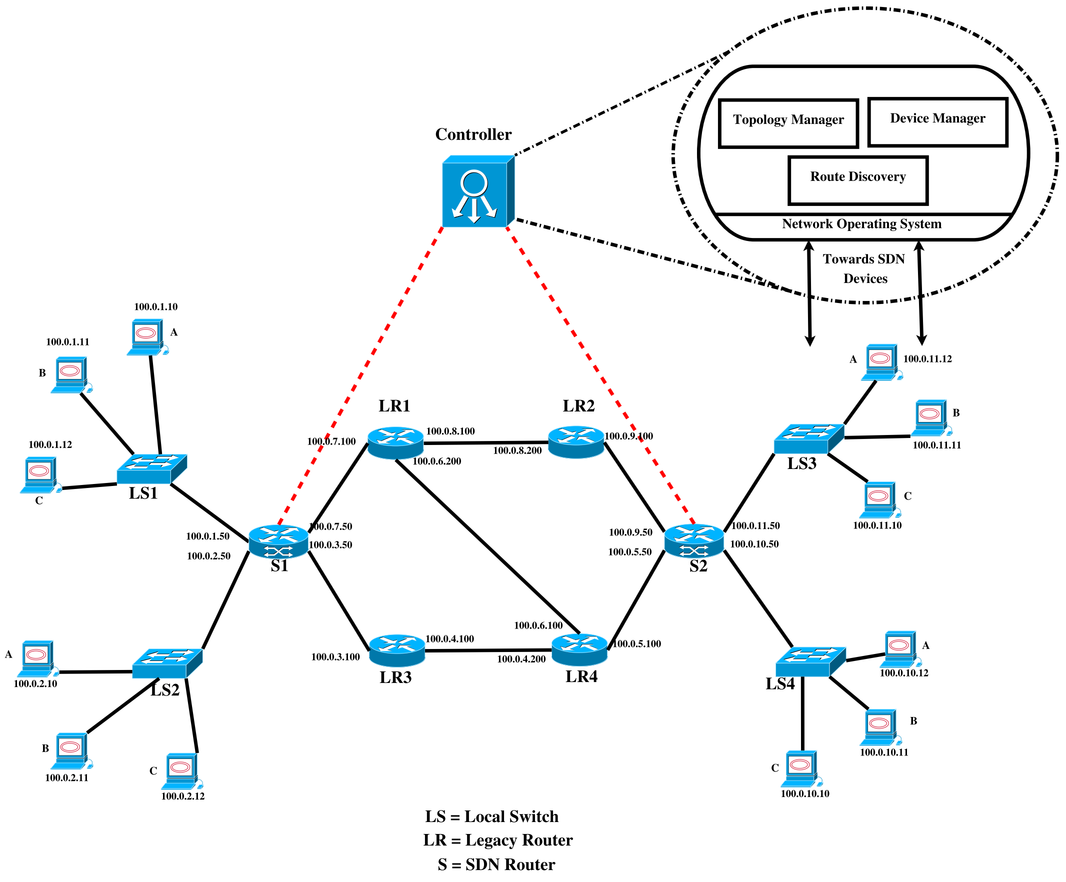

**FIGURE 2.** Architectural view of network topology.

implement it, So that the user can get the benefit of proposing their routing policy in architecture.

While observing the working of traditional routing protocol (e.g. OSPF, RIP etc.), it has been noticed that the selection of the routing path is carried out by inspecting the destination IP address field in the IP header [19]. However, this approach looks very simple, but it is not efficient always. In such a case, if all the packets have the same destination, they will follow the same path in the network. It will cause overloading to the active link while underutilizing other links in the case of multi-path architecture. Similarly, if the destination subnets connected to egress devices are independent, we need to create a separate entry in the flow table for each destination subnet. A large number of destination subnets will cause a substantial space consumption of the flow table. Moreover, this traditional but fixed policy scenario doesn't allow users to forward their packets differently. These two routing issues are the building block of the proposed approach.

The proposed network architecture is designed to send the data packet to the destination subnet differently using the free IP address. This design provides flexibility to the network administrator in network management as they can utilize the free IPs in the given destination subnet. The centralized controller, which holds the control logic of the external SDN router, performs the task of free IP selection for the packet. It maintains a central repository of Free IP for each subnet. Free IP management service, which runs inside the controller, allocates and deallocates IPs. The external SDN routers with flow tables only act as forwarding devices. The centralized controller monitors their functioning. The internal routers work according to traditional legacy fashion and are termed as legacy forwarding devices [27]. They discover their neighbouring devices independently and prepare their routing tables themselves. To map/unmap destination IP to corresponding free IP, the controller simultaneously creates entries at ingress and egress SDN router. The ingress SDN router performs mapping at the sending side, and the egress router performs reverse mapping at the receiving side.

Fig. 2 demonstrates the architectural view of network topology. As mentioned in Fig. 2, we have S1 and S2 as the SDN capable ingress and egress routers. The core network is composed of legacy routers that perform traditional routing tasks. These SDN routers are connected to various subnets, varying in size. For each subnet, the controller maintains a free IP pool that informs regarding the available free IP addresses.

The controller is a high-capacity server available at the centralized location connected to the SDN router. Various network functionalities are available in the controller implemented using network applications. The network operating system interfaces the controller applications and SDN router devices. OpenFlow is the

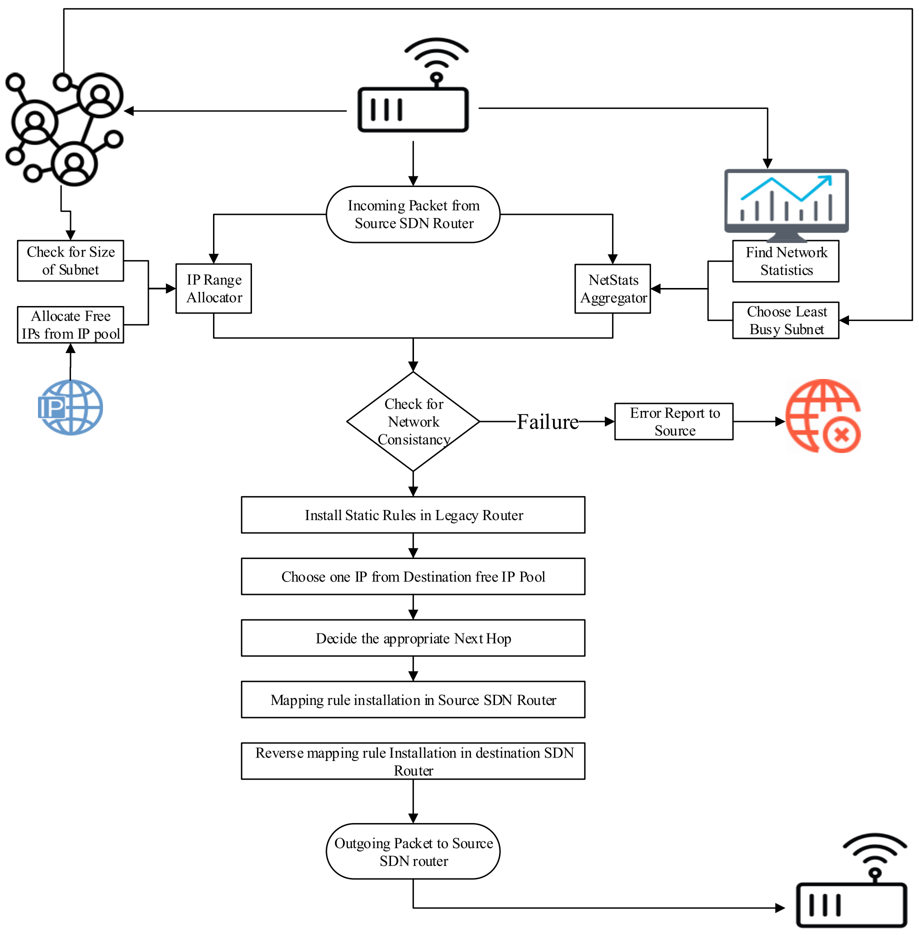
Incoming Packet from Source SDN Router
Check for Size of Subnet
Allocate Free IPs from IP pool
IP Range Allocator
NetStats Aggregator
Find Network Statistics
Choose Least Busy Subnet
Check for Network Consistancy
Failure
Error Report to Source
Install Static Rules in Legacy Router
Choose one IP from Destination free IP Pool
Decide the appropriate Next Hop
Mapping rule installation in Source SDN Router
Reverse mapping rule Installation in destination SDN Router
Outgoing Packet to Source SDN router

**FIGURE 3.** **Flowgraph of proposed approach.**

operational protocol between the controller and SDN routers.

The SDN routers have a specific size (vendor-specific) of the flow table. These routers communicate with the controller in a bidirectional way. The request packet is sent to the controller from the router. Similarly, the reply packet is sent to routers from the controller. Legacy switches act as an intermediator between the subnet and SDN router. The core network architecture can have a partial mesh (as mentioned in Fig. 2) or full mesh architecture. Consider some examples of packet forwarding in the above network architecture.

Case 1: The TELNET application traffic, which has subnet 1 as a source, and subnet 2 as a destination, will follow the path P1 as S1 → LR1→ LR2 → S2. The policy configuration for this path is given as follows-

**Mapping Process**

+ Flow entries for S1
    - src = 100.0.1.0/24, dst = 100.0.11.0/24, tcp_port = 23, action: mod_dst = LR1, new_dst = 1.0.0.5

**Static Routes**

+ Static routes in LR1
    - match 1.0.0.0/24 to 100.0.8.200 via LR1-eth3
+ Static routes in LR2
    - match 1.0.0.0/24 to 100.0.9.50 via LR2-eth2

**Reverse Mapping Process**

+ Flow entries for S2
    - dst = 0.0.0.5/0.0.0.255, tcp_port = 23, action: new_dst = 100.0.11.12

Case 2: The FTP application traffic (TCP port number 20), which has subnet 1 as a source, subnet 3 as a destination, will follow the path P2 as S1→LR3→LR4→S2. The policy configuration for this path is given as follow- **Mapping Process**

+ Flow entries for S1
    - src = 100.0.1.0/24, dst = 100.0.11.0/24, tcp_port = 20, action = mod_dst:LR3, new_dst = 1.0.0.6

**Static Routes**

+ Static routes in LR3
    - match 1.0.0.0/24 to 100.0.4.200 via LR3-eth2
+ Static routes in LR4
    - match 1.0.0.0/24 to 100.0.5.50 via LR4-eth3

**Reverse Mapping Process**

+ Flow entries for S2
    - dst = 0.0.0.6/0.0.0.255, tcp_port = 20, action: new_dst = 100.0.11.13

**TABLE 1. Address count statistics of Free-IPs in global internet.**

| SN | Reserved Block Address | Address Count | References |
|---|---|---|---|
| 1 | 10.0.0.0/8 | 16,777,216 | RFC 1918 [40] |
| 2 | 100.64.0.0/10 | 4,194,304 | RFC 6598 [48] |
| 3 | 127.0.0.0/8 | 16,777,216 | RFC 990 [43] |
| 4 | 169.254.0.0/16 | 65,536 | RFC 3927 [13] |
| 5 | 172.16.0.0/12 | 1,048,576 | RFC 1918 [40] |
| 6 | 192.0.0.0/24 | 256 | RFC 5736 [25] |
| 7 | 192.0.2.0/24 | 256 | RFC 5737 [9] |
| 8 | 192.88.99.0/24 | 256 | RFC 3068 [24] |
| 9 | 192.168.0.0/16 | 65,536 | RFC 1918 [40] |
| 10 | 192.18.0.0/15 | 131,072 | RFC 2544 [12] |
| 11 | 198.51.100.0/24 | 256 | RFC 5737 [9] |
| 12 | 203.0.113.0/24 | 256 | RFC 5737 [9] |
| 13 | 224.0.0.0/40 | 268,435,456 | RFC 5771 [5] |
| 14 | 240.0.0.0/4 | 268,435,456 | RFC 6890 [16] |
| 15 | 255.255.255.255/32 | 1 | RFC 6890 [16] |
| | Total Reserved | | 592,702,864 |
| | Class A | | 16,777,216 |
| | Total unavailable | | 609,486,070 |
| | Available for Re-MAP use | | 3,685,481,226 |

## IV. PROPOSED METHOD

The proposed routing approach is based on the concept of utilization of free IPs of the subnet. It has been seen that not all the IP addresses are used inside a subnet. So, we can maintain a pool of free IP addresses at the controller. Whenever we need to forward the packet, the controller assigns one of the free IPs from the pool to the packet. After this, the destination address field in the packet header gets modified and decided by the controller previously. The controller simultaneously installs the flow entries for mapping and reverse mapping in the ingress and egress SDN routers. In our architecture, only the ingress and egress routers are replaced with the SDN router, while the other routers remain legacy routers. Fig. 3 represents the flow graph of the proposed approach. Certain assumptions are made for the efficient functioning of the proposed routing methods. They are as follows-

1) Every subnet in the network should be connected to legacy routers via at least one SDN switch.
2) All the details regarding the network should be available with the controller.
3) Only Ingress and Egress routers are enabled with the SDN functionality.
4) Topology should be in the form of a full mesh or partial mesh.

Besides these, as mentioned in Fig. 2, the links work in a bidirectional (duplex) fashion, and each router is capable of performing the task of ingress and egress router (depending upon the flow direction of the packet). The mesh topology allows a pair of subnets to have multiple paths for communication and packet transmission. For example, in Fig. 2, A packet can have paths S1-LR1-LR2-S2, S1-LR3-LR4-S2 & S1-LR1-LR4-S2 from source LS1 (subnet 1) to destination LS4 (subnet 4). The same paths are available in reverse order when the packets make the reverse journey. However, selecting the most efficient path depends on the controller.

Now, the total count of the free IPs available in the whole world needs to be decided. The different RFCs available at IETF and counting the free IPs for each class are used. Table 1 gives a brief idea about the count of free IPs.

The Flow table of SDN routers and the routing table of Legacy Routers have limited memory. It might be possible that some of the flow entries may overflow in case of many requests. Another thing to consider is that we need to track the total number of policies inside the system. To deal with these conditions, we impose certain restrictions on the system architecture, which are discussed as follows- We add the policies inside the subnet and make it count for this. We repeatedly apply this procedure to all subnets and finally make a total count by adding each subnet's counts. Eq. 1 represents the given criteria.

$$\sum_{i=1}^{N} \mathrm{PP}^{i} = \mathrm{PP}_{\mathrm{total}} \tag{1}$$

where,

$\mathrm{PP}^{i}$ = Number of policy path for i SDN switch

$\mathrm{PP}_{\mathrm{total}}$ = Total number of policy paths in the system from any source to any destination, and

$N$ = Number of SDN Switches satisfying the criteria of subnets as mentioned in the assumption 1.

The flow table size of the SDN router depends upon three factors. First, we define flow entries for the host already present in the subnet. To convert destination IP into host-specific IP, these entries are required at reverse mapping. Second, we add the entries for those destination subnets which are directly or indirectly connected by this SDN router. Third, we check for the entries that are not used for IP mapping in the SDN router and mark them as non-mapped entries. Finally, we sum all these three parameters and get the final count for total flows. This flow count should be less than the full flow capacity of the SDN router. Eq. 2 specifies the criteria as mentioned above.

$$H_{\mathrm{src}}^{n} + \mathrm{FE}_{\text{non-map}}^{n} + \sum_{j=1}^{p_k} H_{\mathrm{dst}}^{j} \leq FT_{\max} \forall k \in S \tag{2}$$

where-

$H_{\mathrm{src}}^{n}$ = Count for IPs of subnet src which are directly connected to nth SDN switch

$\mathrm{FE}_{\text{non-map}}^{n}$ = Count for flow entries non-mapped to nth SDN switch

$H_{\mathrm{dst}}^{j}$ = Count for IPs in destination subnet for $j^{th}$ policy

$FT_{\max}$ = Size of flow table for an SDN switch

$P_k$ = Count on number of polices at SDN switch behaves as source

$S$ = Set of SDN switches satisfying the assumptions

The routing table of the legacy router has finite memory to store the routing entries. So, again, it will restrict the number of routing entries so that the overflow condition does not affect the legacy routers. First, we calculate the routing entries necessary for the legacy functioning of the router. Second, we calculate the routing entries, which are created by identifying the flow path by the controller, pass through the legacy router. Third, we calculate the routing entries required for IP clash handling when a packet is destined for another destination available in the extranet. Finally, we sum up these counts and make sure that this should be less than the size of the routing table. Eq. 3 defines the above condition.

$$RP^i_{\text{def}} + RP^i_{\text{clashes}} + PP^j \leq RT_{\max} \forall j \in L \tag{3}$$

where-

$RP^i_{\text{def}}$ = Count for routing paths for $i^{th}$ legacy router to perform legacy functioning

$RP^i_{\text{clashes}}$ = Count for routing paths which can handle IP clash issue for $i^{th}$ legacy router

$PP^j$ = Count for policy paths through $j^{th}$ legacy router

$RT_{\max}$ = Size of routing table for a legacy router

$L$ = Set of legacy routers/forwarding devices, where manual policies have to define

We have to define the total IPs which can exist in the system at a particular time. This count cannot be more than the total IP available in the universe. If we are using IPv4 architecture, then this count cannot be more than 4 billion(approximately). Eq. 4 specifies this condition.

$$\sum_{k=1}^{M} H^k_{\text{dst}} \leq FIP_{\max} \tag{4}$$

where-

$H^k_{\text{dst}}$ = Count for the IPs in the destination subnet for $k^{th}$ path

$FIP_{\max}$ = Total number of free IPs in the free ip pool

$M$ = Count on total number of paths for any combination of source and destination

From Table 1, it is clear that the total number of free IPs for class A is 16,777,216. This count becomes sufficient to prepare a subnet when we deal with a practical scenario. On the other hand, the SDN router has a capacity from 500K to 1M flows. So even if we think that the policies occupied 50% storage of the legacy routing table, the system will perform efficiently.

The number of policy paths that the administrator can implement in the system depends upon the size of the Flow table of SDN routers. For example, open vSwitch [2], which is a widely used SDN compatible switch, supports 500K – 1M flow entries.

The proposed approach is helpful in a complex core network environment with a continuous stream of the data packet to serve. For example, consider a scenario where several forwarding requests are served between the two specific subnets. In such a case, the traditional routing architecture works where each device needs to inspect every incoming packet. Unfortunately, this conventional approach will increase the complexity of the forwarding devices and cause an unavoidable delay in packet processing.

On the other hand, the proposed network architecture allows the device to work simplified. For a specific destination, the forwarding devices only need to send the first incoming packet towards the controller to get the flow entries. The rest of the packets will be served directly from the forwarding devices using a simple look and forward approach. However, the delay for the first packet is high, but the delay for subsequent packets can be reduced significantly. This will result in reducing the overall average delay of the subnet.

## A. CONTROLLER MODULES

There are certain modules implemented inside the controller so that system can perform as per the proposed model. The modules are discussed as follows-

1) Topology Manager: The Topology Manager module checks for the available nodes and their corresponding links to identify the type of topology. Our article uses the mesh topology, which can be full mesh or partial mesh. It first sends the request message to all nodes to identify the topology. On getting the request message, nodes prepare a reply message which contains the information e.g. neighbouring nodes, link cost, link operational mode (i.e. simplex, duplex, etc.), propagation delay, etc. Once this information is prepared, it is attached to the reply packet and forwarded to the controller. After receiving complete information from all nodes, the topology manager prepares a graph-like data structure where routers act as vertices and links act as edges. The link information is stored in the form of an adjacency matrix.
2) IP Range Allocate: This module checks for the size of the subnet, i.e. number of hosts in the subnet etc. Whenever a new host joins the subnet, the corresponding SDN switch sends a control packet to the controller to inform about the addition of a new host. After receiving the control packet, the IP range allocates module decides the size of the subnet and accordingly allocates the IP addresses in multiple factor k. We set the minimum value of k to 2 so that the fault-tolerance features can be achieved. The network administrator can assign the value of k as per their preference. One thing to be noticed here is that a higher value of k leads to a higher fault tolerance capacity of the system, but at the same time, it leads to the allocation of a large set of IPs to the subnet. After assigning specific IPs from the free IP pool, the controller updates its free IP pool information. A similar activity is carried out when subnets release IPs. The controller should carry out the Updation task at a regular time interval. The subnet and controller should equally collaborate in the process of updating.
3) Policy Translate: The Policy translate module converts the destination IP address of the packet to one of the free IPs in the destination subnet. This module works in collaboration with the Topology Manager Module. From the topology module, it gets the information on the link statistic. Based on that, it performs the mapping operation. If it identifies that the links are not functioning correctly, it temporarily stops the packet transmis-

sion and shares this information with the corresponding source subnet.

4) Flow Push: The flow push module works according to the decision given by the Policy translator module. The flow push module should perform the mapping and reverse mapping function simultaneously. These modules send the control information to the source and destination SDN routers to make appropriate packet transmission changes in their flow table. This module interacts with the Source and destination SDN routers only. An additional responsibility of this module is to modify other fields, e.g. MAC address etc. if required.
5) Legacy Route Decider: As we know, the rest of the routers in the system are legacy routers and need to be configured manually by the network administrator. So this legacy route Decider takes care of this activity. It installs appropriate routing table entries in the legacy router, which works along with policies decided by the controller. This module ensures the proper functioning of legacy routes. The Network administrator performs remote access procedures e.g. Telnet, SSH, to configure these routers.
6) NetStats Aggregator: This module is externally taken to analyze the network behaviour e.g. current traffic, multiple path information, etc. The data obtained from this module is shared with the other controller modules like the Topology manager.

The algorithmic view of the proposed approach is presented in Algorithms 1, 2, and 3.

**Algorithm 1** Push Flow Entries in the SDN Devices

```
1: procedure FP(Src, Dst, Match)
2:     for SDNR ∈ Src do
3:         Insert mapping rule in Src flow table.
4:     for SDNR ∈ Dst do
5:         Insert reverse mapping rule in Dst flow table.
6:     Return;
```

**Algorithm 2** Legacy Route Decide for Legacy Devices

```
1: procedure LRD($P$, LBS)
2:     $P_{\text{rev}}$ = reverse of P
3:     for ∀ R ∈ $P_{\text{rev}}$ do
         • Install static routes to match the prefix of LBS.
         • Add entries to map IPs of Dst to map
         • Add appropriate next hop to next router in $P_{\text{rev}}$
4:     Return;
```

## B. IP MAPPING

Whenever a packet enters for the first time at the ingress SDN router, it needs to be forwarded to the controller. As the controller prepares a complete view of network topology and keeps a clear image of all paths in the network, it checks the consistent path for the packet. After inspecting the destination IP address, it picks up one of the free IPs from the pool, which belongs to the destination subnet. Now controller installs a mapping rule inside the flow table of the ingress SDN router and simultaneously establishes the reverse mapping rule at the egress SDN router. The destination address inside the packet is replaced with the mapped IP. After a certain number of hops, the packet reaches the egress SDN router, where the router performs reverse mapping. Now The destination IP changes into a specific host id. After entering the destination subnet, the wild-card operation identifies the specific host. We can understand the mapping process by the following example.

**Algorithm 3** Policy Translate for Routing the Packets

```
1: procedure PT(IPpool, Src, Dst, NSD, Match, $P$)
2:     $P_{\text{rev}}$ = reverse of P
3:     sub procedure NSD ()
4:     for ∀ Dst ∈ DstSwitch do
5:     {
6:     Sort{Dst, NSD};
7:     Return LBS;
8:     }
9:     end sub procedure;
10:    sub procedure RangeAllocator ()
         • calculate the size of CN using NetStats aggregator.
         • Allocate the free IPs as per size of subnet.
         • Returns the free IPs to the pool, if any.
11:    end sub procedure;
12:    Call LR{P, LBS}
13:    Call FP{LBS, Src, Dst, Match}
14:    Return;
```

Let's assume, at ingress router packet has destination IP 192.168.16.10/24. Now controller maps this IP to one of the free IPs from the pool as 192.168.16.5/24. So now, the new packet will have an updated destination address as 192.168.16.5/24. Mapping and reverse mapping rules are installed simultaneously at ingress and egress routers. After specific hops of travel, it reaches the egress SDN router to which the destination subnet is attached. Now the destination IP is changed into the initial destination address as 192.168.16.10/24. The last thing we need to perform now is identifying the specific host. For this, we perform a wild-card operation of this IP with 0.0.0.255.

As we have seen, once a flow is established, there is no need to assign extra flow entries for packets destined to the same subnet. So all the packets which belong to the same destination subnet will follow the same path specified by the mapping rule. This idea helps us save space inside the flow table as all the packets require only a single entry to travel to the same destination.

The network administrator is responsible for allowing the packet to travel on a consistent path. The policy Translator module ensures consistent path selection. Consistent path identification is carried out in reverse order so that the closest consistent link to the destination subnet can be identified first. This consistency process ensures that all the policy requirements should be satisfied before assigning the flow entries to the router. If any policy fails to satisfy, the flow entries are not updated. The legacy route module decides the internal router's functionalities, assigning the routing entries to their routing table. Traditional OSPF routing protocol runs inside them, which periodically refreshes the table at the interval of 30 seconds. Address reuse can be performed at the legacy router so that we can optimize the routing process.

The controller also ensures no failure occurs during the packet transmission activity. Controller modules are designed in such a way that they recognize the SNMP events during the failure. Once the controller identifies the failure in the network, it performs a rollback operation over the SDN router and Legacy router to put the complete system at the initial stage.

### C. IP ADDRESS CLASH PROBLEM

In a real-world network scenario, it may be possible to associate free IP available for mapping with some other host in the universe. If the packet is required to deliver to such a host, our network architecture will provide them to another host. So it becomes necessary to identify such packets at the entry gate of the network. The ingress router is the entry point to any network architecture, so they must ensure it. Once it identifies such a packet, it gets forwarded to the controller. Now controller does a similar task as previously to assign one of the free IPs from the pool. The difference lies in the fact that a timestamp value is associated with such a packet. If the packet is not delivered in the given time interval, it gets discarded, and the IP gets returned to the free IP pool. Such a dynamic process can help deliver the packet to the correct destination in case of an IP clash.

Besides this, the controller should analyze the traffic activity and performs optimization if needed. For example, the controller should decide the timestamp value efficiently so that packet does not travel in the network for a large time frame. Transmission delay should be kept as minimum as possible by applying optimization techniques.

### D. PROCESSING OVERHEAD

The separation of control and data plane presents a little overhead during packet processing as the packet needs to transfer from SDN router to controller and vice versa. However, this overhead is only for the first packet, as initially, the flow table in the SDN router is empty. Once the flow entry for the packet of a specific destination is installed, subsequent packets can be forward quickly by observing the header information. These observations do not need further complex routing algorithms to run because forwarding devices are simplified now. We can also observe that not every packet needs to be forward to the controller side for processing.

This small overhead has no impact on the system when many packets are continuously transmitted from one subnet to another subnet. As the transmitted packets increase

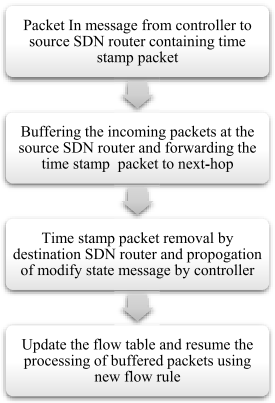
Packet In message from controller to source SDN router containing time stamp packet
Buffering the incoming packets at the source SDN router and forwarding the time stamp packet to next-hop
Time stamp packet removal by destination SDN router and propogation of modify state message by controller
Update the flow table and resume the processing of buffered packets using new flow rule

**FIGURE 4. Procedure of time stamp packet based flow table update consistency scheme.**

in quantity, the average delay for the destination reduces significantly. However, for the first packet transmission, the overhead associated with this approach is comparatively high from the traditional routing approach.

So we can claim that, however, this architecture will increase the processing delay for the first packet, but it will improve performance for subsequent packets. We can also limit the queueing delay for the first packet by incorporating the multithreaded architecture. Now, each thread can serve a specific packet from the queue and can reduce the waiting time.

### E. FLOW TABLE UPDATE APPROACH

The flow tables are updated in both proactive and reactive manners in the proposed approach. The proactive approach is adopted by the controller when some changes occur in the network state (i.e. addition or deletion of a network element). Similarly, the reactive approach is adopted for those packets whose flow rules do not match/exist in the flow table. Table 2 defines the various delays incurred during the installation of flow entries in the table.

When the flow table is updated in a reactive manner then the total delay experienced for the flow table update is given as follow-

$$T_{\text{flow\_update}} = T_{\text{lookup}} + T_{\text{pkt\_in}} + T_{\text{prop}} + T_{\text{proc}} + T_{\text{push}} \tag{5}$$

Similarly, when the flow table is updated in a proactive manner then the total delay experienced for the flow table update is given as follow-

$$T_{\text{flow\_update}} = T_{\text{proc}} + T_{\text{push}} \tag{6}$$

From Eq. 5 and Eq. 6, it can be observed that the flow table updates propagate quickly when a new node is added into the core network. However, for the new packet, the flow table updates propagate in a similar manner to the conventional update scheme. Considering both the methods (proactive & reactive), the proposed architecture performs an efficient update of the flow table when network changes occur.

**TABLE 2. Specification of various delay for flow table update.**

| Delay Parameter | Specification |
|---|---|
| $T_{\text{lookup}}$ | Time required to perform lookup for incoming packet against the existing flow rules |
| $T_{\text{pkt\_in}}$ | Time required to encapsulate the packet in message to forward the controller |
| $T_{\text{prop}}$ | Time required for the propagation of packet in message from switch to controller |
| $T_{\text{proc}}$ | Time required to process the packet against the network state and prepare of appropriate policy |
| $T_{\text{push}}$ | Time required to push the flow rules in the flow table |

### F. FLOW TABLE UPDATE CONSISTENCY

The controller adopts a timestamp packet-based scheme for the flow table update consistency in the proposed approach. In this scheme, the controller first generates a packet out message named time stamp packet, which indicates the SDN routers for the arrival of the new flow entries. Figure 4 represents the procedure of flow table update consistency. The step by step procedure for the flow table update is mentioned below-

Step1: When the update starts, the controller sends two types of messages. First, packet out message comprising the time stamp packet and the specified actions. The second is the modified state message, which indicates the modification in the flow rules. Once the source SDN router receives the time stamp packet, it buffers the incoming packet related to this old rule until the update occurs.

Step2: Once the source SDN router process the time stamp packet, it forwards the packet to the next hop where the legacy routers are available. These legacy routers forward the time stamp packet to the destination SDN router.

Step3: The time stamp packet is processed and forwarded back to the controller at the destination SDN route. Once the controller receives its time stamp packet, it starts the flow modification procedure where the modified state message is propagated to source and destination SDN routers.

Step4: Once the update has taken place completely and flow rules are installed in the SDN nodes, the buffered messages can be resumed according to the new flow rules.

### G. FLOW TABLE MANAGEMENT

Flow table management involves the addition and removal of the flow entries from time to time. Since the switches don't have the control logic, so the flow entries in the flow table are added only after getting instructions from the controller. On the other hand, the flow entries from the table can be removed in two ways. First, the controller can request to delete the specific entries. Second, a switch flow expiry mechanism can be used for the removal of flow entries. The proposed architecture supports the switch flow expiry mechanism. Each flow entry has two parameters associated with it which are-

- Idle Timeout: The number of seconds after which a flow entry is removed from the flow table because no packet matches it.
- Hard Timeout: The number of seconds after which a flow entry is removed from the flow table whether a match is found or not.

Since the proposed architecture uses the OpenvSwitch for the SDN devices, the Idle timeout and Hard timeout value is set to 10 seconds and 0 seconds, respectively. If no packet match is found for the entry in 10 seconds, it can be removed from the flow table. Similarly, no specific deadline is imposed on the hard timeout limit for the flow entry.

## V. RESULTS AND DISCUSSIONS

In this simulation study, The major performance metrics taken for the comparison are- flow table usage, network stretch, and average bandwidth utilization. The experimental setup and the performance analysis of the proposed architecture are as follows.

### A. SIMULATION ENVIRONMENT

The proposed network environment is simulated in Mininet [1] simulator with OpenFlow version 1.5.0. Wire shark [3] is taken as a packet analyzing tool and works in integration with Mininet. In mininet, the openVswitch and openVrouter are taken as forwarding devices to create the SDN portion of the environment. In each subnet the number of hosts is limited (e.g. 3 in our case). An equal number of hosts have been considered for all the subnets, but they can also have a variable number. Table 4 provides the specification of various network topologies based on the different number of SDN and legacy nodes. Below is the description of the notation of the topology.

- Hosts: A, B, and C (Available in each subnet and take IP addresses to which they belong)
- Legacy Switches: LS1, LS2, LS3 and LS4 (Represent the intermediate device between SDN switch and subnet)
- SDN Switches: S1 and S2 (Connected to legacy switches and controlled by the centralized controller)
- Legacy Router: LR1, LR2, LR3 and LR4 (Prepare the carrier network of the system)
- Controller: OpenDayLight & NOX (Centralized Server to forward the logic of the SDN devices and logically connected to them.)

### B. FLOW TABLE ENTRIES CONSUMPTION

The flow table entries consumption depends upon the size of the network topology. Figure 5 shows the graphical representation of the flow entries usage by the various topology. Observation clearly states that the proposed policy-based hybrid SDN routing approach saves flow table space in the range of 87.512% - 81.052%. The maximum saving occurs

**TABLE 3.** Advancements, drawbacks and proposed solutions for the existing state-of-the-art approaches.

| Routing strategy in SDN | Advancements | Drawbacks | Issues overcome in the proposed Hybrid SDN approach |
|---|---|---|---|
| Scalable Routing with Consolidate Middleboxes [22] | (a) Multipoint-To- Point tree-based approach with major focus on the middleboxes, (b) Efficient routing strategy with maximum number of flow support for switch and (c) Routing strategy which satisfy the constraints e.g. Middlebox processing capability, traversal constraints, and switch memory capacity. | All the networking functions (IDS, firewall etc.) are considered equal-cost which is sometimes not possible in a practical scenario. | The devices selected to be replaced with SDN devices based on their functionality |
| Compact TCAM [14] | (a) Significant reduction (upto 12 times) in the overall power requirement of the system compare to Openflow enabled switches and (b) A flow carrying 356 bits for the identification is now identified by just 16 bits. | Power requirement of TCAM is still an bottleneck for the network design | The high power consumable TCAM are only available at the edge of the core network |
| Jumpflow [21] | (a) Enhanced per-hop based forwarding scheme which uses VID for route information-carrying and (b) Uses the reactive flow entry placement strategy on switch | At present, Security and Failure recovery related features are not supported | Failure recovery feature is enabled as a subnet can connect to more than one SDN devices and vice versa. |
| Bounded path degree max flow problem [15] | A bounded path-degree optimization approaches with the goal to maximize the overall feasible traffic while satisfying the limited bandwidth capacity. | Lack of consideration of some classical data center designs (e.g. FatTree etc.) in the simulation study | The state of art data center designs are considered for the approach |
| Tag-In-Tag [10] | (a) Use of Two layer tagging (Path Tag & Flow tag) to reduce the total flow table entries, (b) Use of high speed TCAM for the implementation of flow table and (c) An optimized flow table management approach which saves around 80% power consumption and 15 times flow table space in traditional TCAM | (a) The limited size of TCAM (ranging from 1Mb to 2 Mb) can affect the large network design and (b) The Edge nodes must contain two dedicated TCAM to perform the required ingress and egress activity | The efficient flow table space management is provided using single storage at each side. |
| SDN/OSPF Traffic Engineering (SOTE) [29] | a) Hybrid SDN/OSPF design which uses the weight adjustment approach to minimize the maximum link utilization of network, (b) The 30% SDN devices in the network can lead to near optimal performance and (c) The weight adjustment and the splitting ration of SDN nodes can be changed simultaneously. | The exact criteria for the selection of SDN nodes is not clarified properly | The selection criteria for the device replacement is clearly defined |
| Outsourcing of routing control logic [50] | An outsourcing approach where the control logic of the ISP is transferred to an external trusted service contractor | The external service contractor must ensure for its trust which may be time consuming phase of the model | No third part contractor involved in the design which provides a transparent design. |

**TABLE 4.** Specification of various topology for experimentation.

| Topology | Number of Edge SDN Nodes | Number of Legacy Nodes | Number of Links |
|---|---|---|---|
| TP1 | 2 | 4 | 9 |
| TP2 | 4 | 6 | 21 |
| TP3 | 8 | 12 | 76 |
| TP4 | 16 | 15 | 112 |
| TP5 | 32 | 24 | 296 |
| TP6 | 64 | 56 | 1488 |
| TP7 | 128 | 112 | 3648 |
| TP8 | 256 | 176 | 42988 |

in the case of the OpenDayLight controller, where an average of 12488.75 flow entries are consumed for a varying range of topology. A slight variation of 0.029% of flow entries consumption has been seen in the NOX and OpenDayLight controller.

Figure 5 provides the graphical representation of the flow entries consumption by various topologies on the different controller platforms.

Apart from the controller-based comparison, the proposed approach is compared with two flow table aggregation approaches Tag-in-Tag & SDN-OSPF. Figure 6 & Figure 7 represents the comparison of the maximum number of flow rules used by the three approaches using the OpenDayLight and NOX controllers over the various topologies.

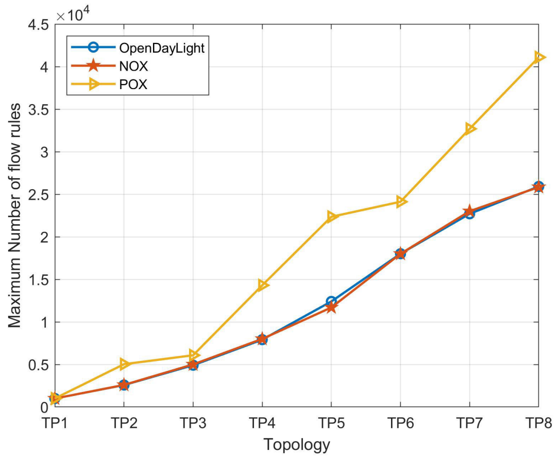

**FIGURE 5.** Average flow table entry consumption for varying size of topology.

The experimental results show that the proposed approach consumes an almost similar number of flow table entries as compared to Tag-in-Tag for some topology (i.e. TP1 to TP3).

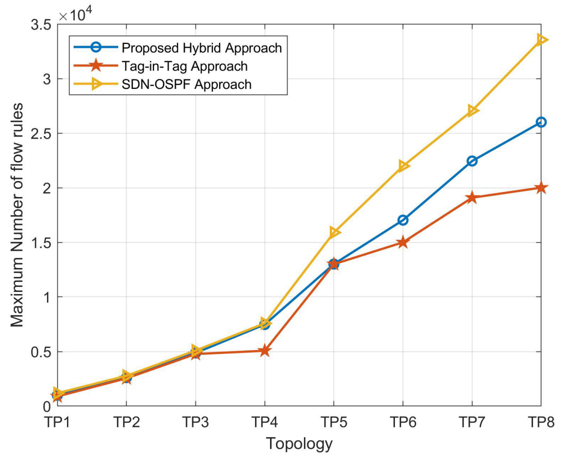

**FIGURE 6.** Comparison of average flow table entry consumption over OpenDayLight.

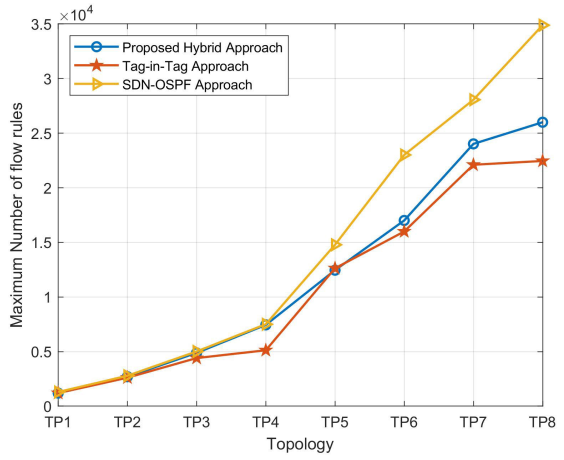

**FIGURE 7.** Comparison of average flow table entry consumption over NOX.

For some of the topology (i.e. equal number of flow entries in the flow table. The Tag-in-Tag's less flow entry consumption is based on its two dedicated flow table availability at each SDN node. On the other side, the proposed approach uses only one flow table per SDN device, providing a cost-effective solution. However, the performance of the proposed approach is always superior to the SDN-OSPF approach in terms of flow entry consumption.

### C. NETWORK PATH STRETCH

The Network path stretch can be defined as the difference between the best available path and the actual path taken by the packet during its journey in the network. The path stretch should be minimal because an increase in the path stretch denotes the sub-optimal usage of the network resources. Figure 8 provides the graphical representation of the Average Network path stretch by various topologies on the different controller platforms. The results clearly state that the Average network path stretch is minimum when using the OpenDayLight controller. However, the approach also performs well on the NOX controller where a minimum path stretch is

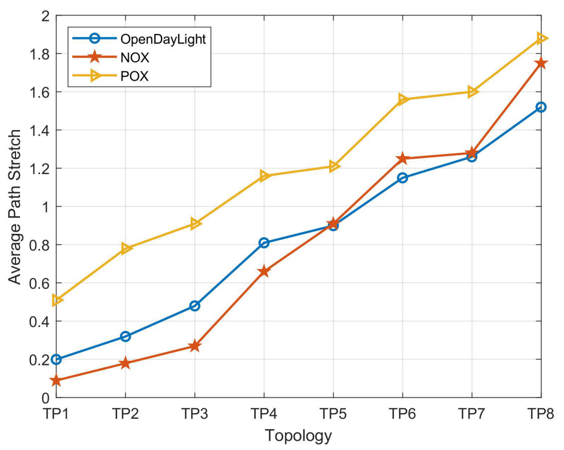

**FIGURE 8.** Average network path Stretch for varying size of topology.

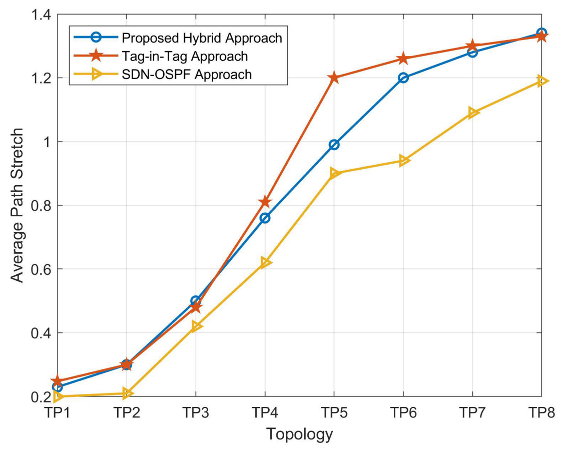

**FIGURE 9.** Comparison of average network path stretch over OpenDayLight.

computed, but the OpenDayLight outperforms the other two controllers for the more extensive network.

Similarly, on comparing the network path stretch of the proposed hybrid approach with the other two approaches, it is observed that the results of the proposed approach are promising compared to the Tag-in-Tag approach results. The Figure 9 & Figure 10 presents the graphical demonstration of average network path stretch for all the approaches over OpenDayLight and NOX controller respectively. Except for the few cases (TP6 & TP8 in Figure 10), the proposed approach is the winner in the category of the average network path stretch. Once the path is decided in the proposed hybrid approach, this is the only path that a packet follows until the network change. On the other hand, in the Tag-in-Tag approach, the alternate paths are stored proactively, increasing the probability for the higher network path stretch.

### D. THROUGHPUT ANALYSIS

The throughput analysis of the proposed approach in a multithreaded environment is represented in Figure 11. The proposed network environment is tested along with three different controllers namely OpenDayLight [15], NOX [68], POX [69]. OpenDayLight is an open-source multithreaded SDN

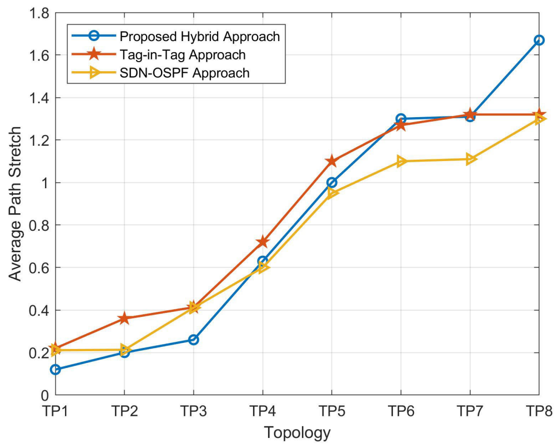

**FIGURE 10.** Comparison of average network path stretch over NOX.

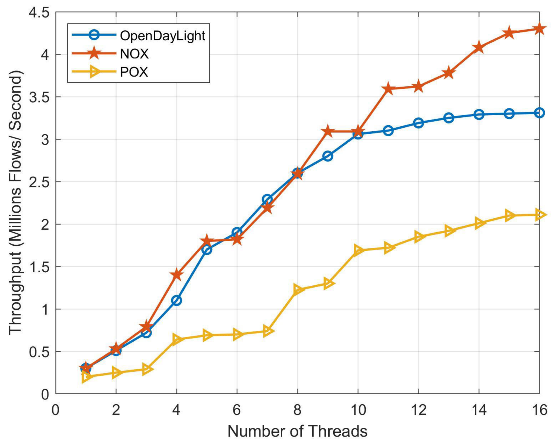

**FIGURE 11.** Throughput in multi-threaded environment.

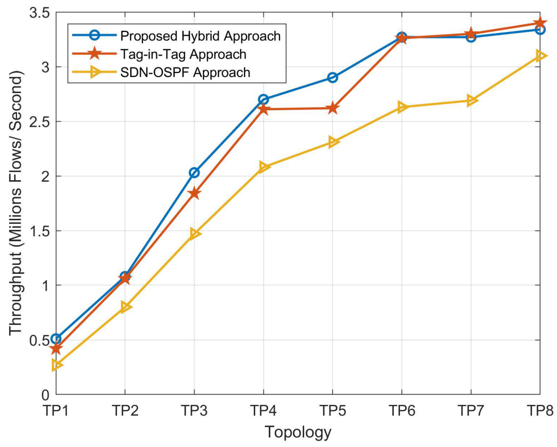

**FIGURE 12.** Comparison of throughput in multithreaded environment over ODL.

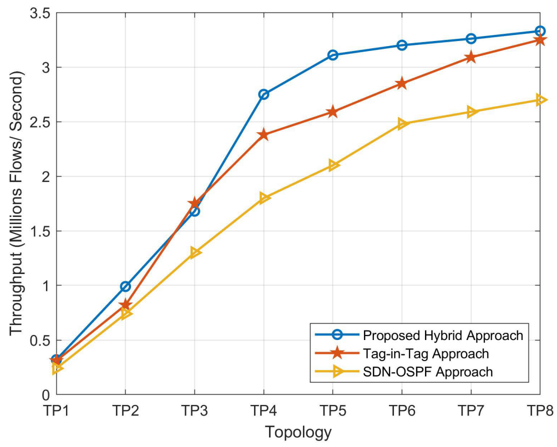

**FIGURE 13.** Comparison of throughput in multithreaded environment over NOX.

controller, while POX and NOX are highly multithreaded SDN controllers. From Figure 11, it is clear that when the number of threads is increased in the range of 1-16, a steady increase in the system performance can be seen in terms of throughput (number of requests satisfied in unit time). It can be observed from the graph that when the number of threads is limited up to 8, both the OpenDayLight and NOX controller give similar performance. Beyond 8, when the threads are increased from 9 to 16, NOX performs superior over OpenDayLight. NOX's superior performance over OpenDayLight is that NOX is highly multithreaded compared to OpenDayLight. On the other hand, The POX controller performs worst among the selected controller in terms of throughput. The average throughput for OpenDayLight is calculated at 2.31 similarly, for NOX and POX, it is calculated as 2.625 & 1.273, respectively. The experimental results indicate that NOX is the preferred choice for the multithreaded application environment.

The throughput of the proposed hybrid approach is also compared with the Tag-in-Tag & SDN-OSPF approaches using OpenDayLight and NOX controllers. Figure 12 & Figure 13 represents the throughput values for all approaches over various topologies. From both the figures, it is clear that the proposed approach utilizes the bandwidth maximum and hence has maximum throughput among all approaches. The proposed approach selects the path based on the predefined policy. So every packet belongs to the same policy forwarded through the same path, which makes the proper utilization of the link. Besides this, the alternate paths are selected only when they are needed.

### E. LATENCY ANALYSIS

The Latency comparison of the proposed architecture over the OpenDayLight, NOX and POX controller instances is presented in Figure 14. The IP header is taken in different sizes. Different IP header sizes consume different latency times for searching and forwarding operations. In OpenDayLight and POX, the packet experienced a higher latency delay than NOX. From the graph, it can be observed that there is a moderate gap in latency when the header gradually increases from 40. The average Latency time for OpenDayLight is calculated at 66.796 $\mu$s, and similarly, for NOX and POX, it is calculated as 55.782 $\mu$s & 67.501 $\mu$s respectively.

Figure 15 & Figure 16 provide the graphical representation of average latency for the three approaches. In this case, the proposed approach behaves almost like the Tag-in-Tag approach. However, for the few topologies, the Tag-in-Tag

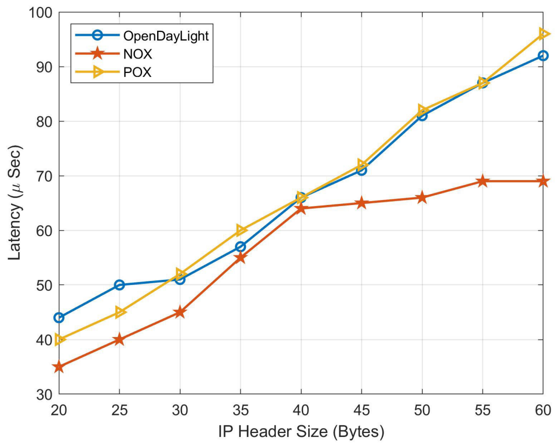

**FIGURE 14.** Latency experienced by packet with different IP header size.

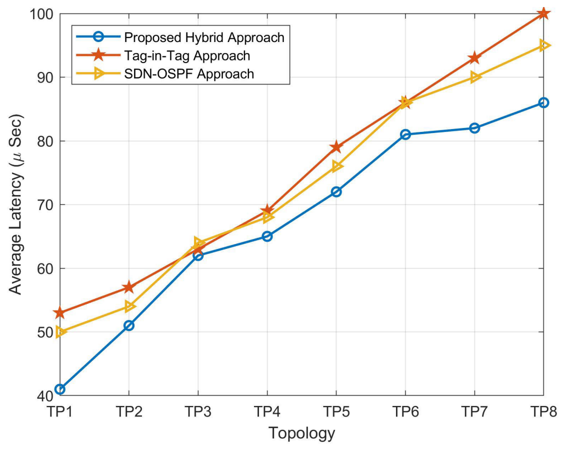

**FIGURE 15.** Comparison of average latency over OpenDayLight.

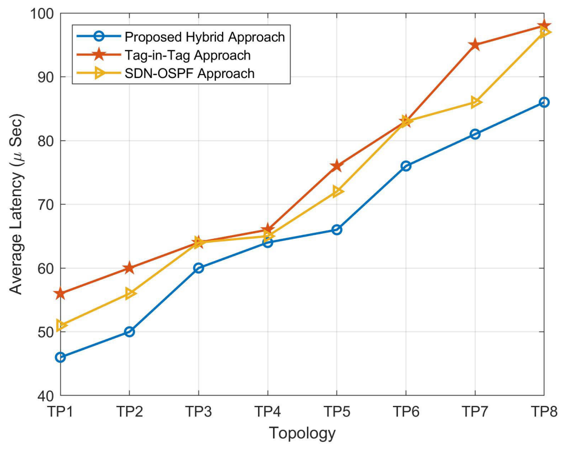

**FIGURE 16.** Comparison of average latency over NOX.

approach possesses high latency because of its multiple entries in multiple flow tables for the same destination subnets. On the other hand, The proposed approach is superior in every respect from the SDN-OSPF approach while considering the average latency for various topologies.

## VI. CONCLUSION

This paper proposes an effective network architecture that supports policy-based routing in a hybrid SDN scenario. Experimental results demonstrate the algorithm's effectiveness over the three different multithreaded controllers. Initially, the experiment is limited to up to three controller instances. Still, from the future point of view, the architecture can be simulated with the other controllers to verify its effectiveness. Similarly, Each subnet has a fixed and a similar number of end-host devices, but the idea can be extended over the variable number of end-host devices per subnet. The suggested framework supports network services like VPN [39], [46], Traffic Engineering [4], [32] run over it. This approach is a good mixing of IP and SDN concepts which provides a cost-effective solution and can benefit the organization to migrate from the traditional IP networks to SDN while keeping the overall cost of migration as low as possible.

## ACKNOWLEDGMENT

The authors wish to thank all the anonymous reviewers for their valuable suggestions.

## REFERENCES

[1] *Mininet*. Accessed: Jun. 16, 2021. [Online]. Available: http://mininet.org/
[2] *Open vSwitch*. Accessed: Jun. 16, 2021. [Online]. Available: http://openvswitch.org/
[3] *Wireshark*. Accessed: Jun. 16, 2021. [Online]. Available: https://www.wireshark.org/
[4] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "Research challenges for traffic engineering in software defined networks," *IEEE Netw.*, vol. 30, no. 3, pp. 52–58, May/Jun. 2016.
[5] Z. Albanna, K. Almeroth, D. Meyer, and M. Schipper, *IANA Guidelines for IPv4 Multicast Address Assignments*, IEF, Fremont, CA, USA, document RFC3171, 2001.
[6] R. Amin, M. Reisslein, and N. Shah, "Hybrid SDN networks: A survey of existing approaches," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3259–3306, 4th Quart., 2018.
[7] R. Amin, N. Shah, and W. Mehmood, "Enforcing optimal ACL policies using $K$-partite graph in hybrid SDN," *Electronics*, vol. 8, no. 6, p. 604, May 2019.
[8] S. Ansari and K. K. Nagwanshi, "An empirical study of soft computing approaches in wireless sensor networks," *J. Cases Inf. Technol.*, vol. 24, no. 4, pp. 1–10, Oct. 2022. [Online]. Available: https://www.igi-global.com/article/empirical-study-soft-computing-approaches/296722
[9] J. Arkko, L. Vegoda, and M. Cotton, "IPv4 address blocks reserved for documentation," 2010.
[10] S. Banerjee and K. Kannan, "Tag-in-tag: Efficient flow table management in SDN switches," in *Proc. 10th Int. Conf. Netw. Service Manage. (CNSM) Workshop*, Nov. 2014, pp. 109–117.
[11] M. Bano, A. Qayyum, R. N. B. Rais, and S. S. A. Gilani, "Soft-mesh: A robust routing architecture for hybrid SDN and wireless mesh networks," *IEEE Access*, vol. 9, pp. 87715–87730, 2021.
[12] S. Bradner and J. McQuaid, *Benchmarking Methodology for Network Interconnect Devices*, IEF, Fremont, CA, USA, document RFC1944, 1999.
[13] S. Cheshire, B. Aboba, and E. Guttman, *Dynamic Configuration of IPv4 Link-Local Addresses*, IEF, Fremont, CA, USA, document RFC3927, 2005.
[14] C.-C. Chuang, Y.-J. Yu, A.-C. Pang, and G.-Y. Chen, "Minimization of TCAM usage for SDN scalability in wireless data centers," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–7.
[15] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "On the effect of forwarding table size on SDN network utilization," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2014, pp. 1734–1742.
[16] M. Cotton, L. Vegoda, R. Bonica, and B. Haberman, *Special-Purpose Ip Address Registries*, IEF, Fremont, CA, USA, document RFC6890, 2013.
[17] R. Dhanare, K. K. Nagwanshi, and S. Varma, "Enhancing the route optimization using hybrid MAF optimization algorithm for the internet of vehicle," *Wireless Pers. Commun.*, pp. 1–21, Mar. 2022, doi: 10.1007/s11277-022-09629-7.

[18] R. Dhanare, K. K. Nagwanshi, and S. Varma, "A study to enhance the route optimization algorithm for the internet of vehicle," *Wireless Commun. Mobile Comput.*, vol. 2022, Apr. 2022, Art. no. 1453187, doi: 10.1155/2022/1453187.

[19] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Commun. Mag.*, vol. 40, no. 10, pp. 118–124, Oct. 2002.

[20] S. S. A. Gilani, A. Qayyum, R. N. B. Rais, and M. Bano, "SDNMesh: An SDN based routing architecture for wireless mesh networks," *IEEE Access*, vol. 8, pp. 136769–136781, 2020.

[21] Z. Guo, Y. Xu, M. Cello, J. Zhang, Z. Wang, M. Liu, and H. J. Chao, "JumpFlow: Reducing flow table usage in software-defined networks," *Comput. Netw.*, vol. 92, pp. 300–315, Dec. 2015.

[22] A. Gushchin, A. Walid, and A. Tang, "Scalable routing in SDN-enabled networks with consolidated middleboxes," in *Proc. ACM SIGCOMM Workshop Hot Topics Middleboxes Netw. Function Virtualization*, Aug. 2015, pp. 55–60.

[23] S. Huang, J. Zhao, and X. Wang, "HybridFlow: A lightweight control plane for hybrid SDN in enterprise networks," in *Proc. IEEE/ACM 24th Int. Symp. Quality Service (IWQoS)*, Jun. 2016, pp. 1–2.

[24] C. Huitema, *An anycast Prefix for 6to4 Relay Routers*, IEF, Fremont, CA, USA, document RFC3068, 2001.

[25] G. Huston, M. Cotton, and L. Vegoda, *IANA IPv4 Special Purpose Address Registry*, IEF, Fremont, CA, USA, document RFC5736, 2010.

[26] S. Iqbal, H. Maryam, K. N. Qureshi, I. T. Javed, and N. Crespi, "Automised flow rule formation by using machine learning in software defined networks based edge computing," *Egyptian Informat. J.*, vol. 23, no. 1, pp. 149–157, Mar. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1110866521000682

[27] C. Jin, C. Lumezanu, Q. Xu, Z.-L. Zhang, and G. Jiang, "Telekinesis: Controlling legacy switch routing with OpenFlow in hybrid networks," in *Proc. 1st ACM SIGCOMM Symp. Softw. Defined Netw. Res.*, Jun. 2015, p. 20.

[28] K. Kannan and S. Banerjee, "Compact TCAM: Flow entry compaction in TCAM for power aware SDN," in *Proc. Int. Conf. Distrib. Comput. Netw.* Berlin, Germany: Springer, 2013, pp. 439–444.

[29] S. Khorsandroo, A. G. Sánchez, A. S. Tosun, J. Arco, and R. Doriguzzi-Corin, "Hybrid SDN evolution: A comprehensive survey of the state-of-the-art," *Comput. Netw.*, vol. 192, Jun. 2021, Art. no. 107981, doi: 10.1016/j.comnet.2021.107981.

[30] V. Kotronis, X. Dimitropoulos, and B. Ager, "Outsourcing the routing control logic: Better internet routing based on SDN principles," in *Proc. 11th ACM Workshop Hot Topics Netw.*, 2012, pp. 55–60.

[31] A. P. Kumbhare, D. G. Kamath, V. A. Pandey, and N. Mukherjee, "Switch routing table utilizing software defined network (SDN) controller programmed route segregation and prioritization," U.S. Patent 9 225 635, Dec. 29, 2015.

[32] X. Li, J. Yan, and H. Ren, "Software defined traffic engineering for improving quality of service," *China Commun.*, vol. 14, no. 10, pp. 12–25, Oct. 2017.

[33] T. Malbašić, P. D. Bojović, Ž. Bojović, J. Šuh, and D. Vujošević, "Hybrid SDN networks: A multi-parameter server load balancing scheme," *J. Netw. Syst. Manage.*, vol. 30, no. 2, p. 30, Jan. 2022, doi: 10.1007/s10922-022-09642-y.

[34] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[35] K. K. Nagwanshi, A. Noonia, S. Tiwari, N. V. Doohan, V. Kumawat, T. A. Ahanger, and E. T. Amoatey, "Wearable sensors with Internet of Things (IoT) and vocabulary-based acoustic signal processing for monitoring children's health," *Comput. Intell. Neurosci.*, vol. 2022, Apr. 2022, Art. no. 9737511, doi: 10.1155/2022/9737511.

[36] M. Osman and J. Mangues-Bafalluy, "Hybrid SDN performance: Switching between centralized and distributed modes under unreliable control communication channels," *J. Sensor Actuator Netw.*, vol. 10, no. 3, p. 57, Aug. 2021. [Online]. Available: https://www.mdpi.com/2224-2708/10/3/57

[37] M. Paliwal and D. Shrimankar, "Effective resource management in SDN enabled data center network based on traffic demand," *IEEE Access*, vol. 7, pp. 69698–69706, 2019.

[38] M. Paliwal, D. Shrimankar, and O. Tembhurne, "Controllers in SDN: A review report," *IEEE Access*, vol. 6, pp. 36256–36270, 2018.

[39] H. Redžović, A. Smiljanić, and B. Savić, "Performance evaluation of software routers with VPN features," in *Proc. 24th Telecommun. Forum (TELFOR)*, Nov. 2016, pp. 1–4.

[40] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear, *Address Allocation for Private Internets*, IEF, Fremont, CA, USA, document RFC1918, 1996.

[41] C. Ren, S. Bai, Y. Wang, and Y. Li, "Achieving near-optimal traffic engineering using a distributed algorithm in hybrid SDN," *IEEE Access*, vol. 8, pp. 29111–29124, 2020.

[42] G. Rétvári, A. Gulyás, Z. Heszberger, M. Csernai, and J. J. Bíró, "Compact policy routing," *Distrib. Comput.*, vol. 26, nos. 5–6, pp. 309–320, Oct. 2013.

[43] J. Reynolds and J. Postel, *Assigned Numbers*, IEF, Fremont, CA, USA, document RFC1700, 1994.

[44] Y. Shen, C. Wu, Q. Cheng, and D. Kong, "AFTM: An adaptive flow table management scheme for OpenFlow switches," in *Proc. IEEE 22nd Int. Conf. High Perform. Comput. Commun.; IEEE 18th Int. Conf. Smart City; IEEE 6th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Dec. 2020, pp. 917–922.

[45] R. Silva, D. Santos, F. Meneses, D. Corujo, and R. L. Aguiar, "A hybrid SDN solution for mobile networks," *Comput. Netw.*, vol. 190, May 2021, Art. no. 107958.

[46] A. Skarmeta and G. Perez, "Policy-based dynamic provision of IP services in a secure VPN coalition scenario," *IEEE Commun. Mag.*, vol. 42, no. 11, pp. 118–124, Nov. 2004.

[47] B. R. Smith and J. J. Garcia-Luna-Aceves, "Efficient policy-based routing without virtual circuits," in *Proc. 1st Int. Conf. Quality Service Heterogeneous Wired/Wireless Netw. (QSHINE)*, 2004, pp. 242–251.

[48] J. Weil, V. Kuarsingh, C. Donley, C. Liljenstolpe, and M. Azinger, *IANA-Reserved IPv4 Prefix for Shared Address Space*, IEF, Fremont, CA, USA, document RFC6598, 2012.

[49] H. Xu, X. Y. Li, L. Huang, H. Deng, H. Huang, and H. Wang, "Incremental deployment and throughput maximization routing for a hybrid SDN," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1861–1875, Jun. 2017.

[50] H. Zhu, H. Qiu, J. Zhu, and D. Chen, "SMSEI-SDN: A suppression method of security incident impact for the inter-domain routing system based on software-defined networking," *Wireless Commun. Mobile Comput.*, vol. 2021, pp. 1–16, May 2021, doi: 10.1155/2021/5539790.

**MANISH PALIWAL** received the Ph.D. degree from the Visvesvaraya National Institute of Technology, Nagpur, India. He is currently working as an Assistant Professor with the Department of Computer Science Engineering, School of Technology, Pandit Deendayal Energy University, Gujarat. His research interests include software defined networks, wireless networks, and network function virtualization.

**KAPIL KUMAR NAGWANSHI** (Senior Member, IEEE) received the Ph.D. degree from Chhatisgarh Swami Vivekanand Technical University, Bhilai, India. He is currently working as an Associate Professor with ASET, Amity University Rajasthan, Jaipur, India. He has awarded five patents. He has an extensive publication record in SCI, SCOPUS, and other reputed indexed journals. He edited five proceedings of the Shaastrarth International Conference series, four books, two datasets, many book chapters, and presented his research work at national and national international conferences. He is highly motivated, dynamic, and single-minded in a fair for connecting with young minds and nurturing and fostering their development to their utmost potential. His primary domain of teaching and research interests include the Internet of Things, digital image processing, cyber forensics, data science and engineering, AI, and computer networking. He is a Senior Member of YHAI. He is a Life Member of CSI, IETE, IAENG, IACSIT, and other professional bodies. He is a Reviewer of reputed journals, such as IEEE Access, *The Imaging Science Journal*, *Journal of Real-Time Image Processing*, and *International Journal of Computer and Electrical Engineering*.