Quality of Service Optimization in Hybrid Software-Defined Network

A thesis submitted to

DELHI TECHNOLOGICAL UNIVERSITY

For the Award of the degree of

DOCTOR OF PHILOSOPHY

In

Computer Science and Engineering

By

Samiullah Mehraban

2k20/PHDCO/14

Under the supervision of

Dr. Rajesh Kumar Yadav

Associate Professor

Department of Computer Science and Engineering

Delhi Technological University

Department of Computer Science and Engineering

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042, India, 2025

DELHI TECHNOLOGICAL UNIVERSITY

(Govt. of National Capital Territory of Delhi)

BAWANA ROAD, DELHI-110042

Declaration

I, Samiullah Mehraban (2k20/PHDCO/14), hereby affirm that the research work presented in my thesis titled "**Quality of Services Optimization in Hybrid Software Defined Network**" is an original contribution, conducted under the guidance of Dr. Rajesh Kumar Yadav from the Department of Computer Science and Engineering at Delhi Technological University. This thesis has not been previously submitted to any other academic institution to obtain a degree or diploma. Throughout the writing process, I have adhered to the prescribed Ph.D. rules and regulations set forth by the Institute.

I confirm that the thesis does not contain any classified information. Whenever I have utilized external sources, proper acknowledgment has been provided by citing them within the text and including them in the reference list. Direct quotations from external sources have been explicitly identified by quotation marks and have been appropriately referenced both in the text and the reference list.

Date:

Samiullah Mehraban

2k20/PHDCO/14

DELHI TECHNOLOGICAL UNIVERSITY

(Govt. of National Capital Territory of Delhi)

BAWANA ROAD, DELHI-110042

CERTIFICATE

This is to certify that the research work presented in this thesis titled "**Quality of Services Optimization in Hybrid Software Defined Network**" by Samiullah Mehraban (2k20/PHDCO/14) is an original contribution conducted under the guidance of Dr. Rajesh Kumar Yadav from the Department of Computer Science and Engineering at Delhi Technological University.

This thesis has not been previously submitted for any other degree or diploma. Samiullah Mehraban has followed the prescribed Ph.D. rules and regulations throughout the writing process.

The thesis does not contain any classified information. Proper acknowledgment has been given for external sources through citations within the text and inclusion in the reference list. Direct quotations have been appropriately identified and referenced.

Date:

Dr. Rajesh Kumar Yadav

Associate Professor (Supervisor)

Department of Computer Science and Engineering

Delhi Technological University

ACKNOWLEDGEMENT

I want to express my sincere gratitude to Dr. Rajesh Kumar Yadav, Associate Professor, Department of Computer Science and Engineering, Delhi Technological University, Delhi, for providing valuable guidance and constant encouragement throughout the work. His immense knowledge, motivation, expertise, and insightful comments have helped me immensely at every stage of the preparation of this research plan.

I would also like to extend my heartfelt thanks to the DRC and SRC committees for their valuable insights, suggestions, and critical evaluation of my research work. Their expertise and scholarly advice have played a crucial role in improving the quality of this thesis.

I would like to sincerely thank the esteemed members of the Department of Computer Science & Engineering at Delhi Technological University. Their unwavering support and provision of invaluable research resources have been instrumental in the successful completion of this work. Their guidance and assistance have been a constant source of motivation throughout my academic journey.

Moreover, I am profoundly grateful to my parents (Haji sahib and Bibi Haji), my brothers (Engineer Hamidullah Mohmand and Ahmad Wali Mehraban), my sisters, my wife, my sons (Mr.Safiullah Mehraban and Mr Soban Mehraban), and my daughters (Ms. Sadaf Mehraban and Ms Somayye Mehraban), whose unconditional support and blessings have been a cornerstone of my achievement.

Their unwavering faith in my abilities, endless lessons, moral support, and care have played an integral role in shaping my academic pursuits. I hold the utmost respect and affection for my elders, whose wisdom and encouragement have guided every step of my journey.

I recognize that this accomplishment would not have been possible without the collective efforts and contributions of all those mentioned above. Their presence in my life has been truly invaluable, and I am forever indebted to them for their unwavering support and belief in my abilities.

Samiullah Mehraban

2k20/PHDCO/14

ABSTRACT

Traditional network infrastructures offer numerous features but exhibit limitations, particularly in providing advanced network control for implementing novel concepts. For future network design, different architectures worldwide have been proposed; among them, one of the finest network designs is Software-Defined Networking (SDN), which separates the network control and data layers, offering enhanced agility, programmability, flexibility, and advanced traffic engineering capabilities. However, because of economic and technical challenges, directly upgrading the entire conventional network to fully SDN is challenging for many organizations. The optimal solution to get the advantages of both networks is to incrementally upgrade conventional network devices to SDN nodes, called the Hybrid SDN network.

The amalgamation of SDN principles with conventional networking techniques has led to the emergence of hybrid SDN; as the current network infrastructure evolves, it combines the programmability of SDN with the traditional protocols of conventional networking systems. A practical approach involves adopting a hybrid SDN model, wherein conventional and SDN components are integrated seamlessly. This paradigm change presents new challenges and opportunities in Traffic Engineering (TE), demanding novel approaches to enhance network performance, optimize resource usage, and increase overall efficiency.

This thesis aimed to investigate the existing literature on the migration sequence to a hybrid SDN and traffic engineering optimization in the hybrid SDN environment. Specifically, we examined the focus of most authors, which primarily revolved around minimizing Maximum Link Utilization (MLU) in the hybrid SDN environment. Therefore, we introduced more precise methods to determine the optimal migration sequence toward a hybrid SDN architecture, and further refined both the OSPF weight configuration and the SDN nodes' traffic splitting ratios by incorporating a wider range of parameters in the prediction process.

To address the research gaps, we defined four primary objectives. The first objective was to review and compare the existing methods implemented for the migration from a conventional network to a hybrid SDN network, and then traffic engineering optimization in the hybrid SDN environment.

The second objective was to propose a model for the migration of the conventional legacy network to the hybrid SDN network. We introduced the FCM (Flow Control Method) approach, which investigates the gradual deployment of SDN nodes in conventional networks using a simple greedy algorithm, presenting a potential solution for the evolution of network architecture. We investigated the optimal migration sequence from a conventional network to a hybrid SDN to address the optimization impact on controllable traffic. The proposed technique efficiently integrates SDN features while minimizing network disruption and optimizing link utilization. We evaluate the optimal deployment of SDN nodes

through simulations, considering network topologies, available resources, and traffic patterns. Simulation experiments conducted on real network topologies demonstrate that by upgrading 17% of the nodes to SDN, near-optimal performance can be achieved while requiring substantially less investment in resources and effort for network upgrades. This result suggests that the greedy method is cost-effective and highly effective in practical scenarios. Achieving near-optimal load balancing with minimal upgrades could offer significant advantages for network operators, making it a valuable strategy for managing network performance in real-world applications.

The third objective entailed a mechanism for routing optimization in the migrated hybrid SDN network. In a hybrid SDN network, where traditional and SDN devices operate simultaneously, the transition introduces new challenges and opportunities for traffic engineering. This demands innovative approaches to enhance network performance, resource utilization, and overall efficiency. We explored routing optimization for traffic engineering within this evolving hybrid environment and addressed the problem as a mixed-integer nonlinear programming (MINLP) framework.

We introduced a heuristic technique, H-STE, to improve traffic engineering in the hybrid SDN; we concentrated on minimizing the MLU by optimizing two critical aspects: optimizing the OSPF weight settings across the whole network to balance the flows originating from conventional devices and optimizing the traffic splitting ratio of SDN nodes. The H-STE method helps bridge the gap between the two by optimizing the whole network performance and adhering to the limitations imposed by legacy routing infrastructure. We have conducted several experiments on real network datasets; the finding shows that with a 30% deployment ratio of SDN nodes, we can reduce the MLU and get close to the optimal result to get maximum benefits from the concept of the hybrid SDN. This research contributes to the evolution of communication infrastructure by promoting efficiency, flexibility, and robustness in the pursuit of better network management and design.

The fourth objective was to develop a framework for the interconnection of SDN nodes alongside conventional network devices. We investigated the interconnection of SDN nodes with conventional legacy devices in a hybrid SDN network. We studied the integration of SDN nodes alongside conventional legacy networks and explored how to peer and integrate SDN nodes with conventional legacy networks using the SDN-IP application located on the application layer of the ONOS controller. As multiple SDN controllers exist, selecting the appropriate controller based on the network topology is the key to achieving optimal network performance. We have studied and examined different controllers for the hybrid SDN environment, as multiple controllers exist. Notable controllers are NOX, Floodlight, POX, ODL, Ryu, and ONOS. Finally, the ONOS (Open Network Operating System) controller was chosen as the SDN controller for our hybrid SDN setup due to its compatibility with hybrid SDN networks and its support for SDN-IP applications. Nevertheless, none of the earlier studies interconnect SDN nodes with conventional legacy networks using ONOS controllers in a real network scenario. We

have considered this issue as the interconnection of SDN nodes into conventional legacy networks with ONOS controllers using Mininet tools. Through extensive simulation, we found that hybrid SDN is the optimal solution for organizations seeking a modern network infrastructure without network disruption and service downtime, a hybrid SDN can provide a network environment that is more scalable, adaptable, and programmable.

By addressing these objectives, our research offers valuable contributions to network performance enhancements in the hybrid SDN environment. The scalability of our approach ensures that the method maintains its efficiency as the topology expands, demonstrating its resilience in controlling diverse network topologies. This is a significant improvement as it reduces resource utilization while enhancing network efficiency.

CONTENT

List of Figures

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviations	**Expanded Form**
SDN	Software Defined Networking
hSDN	Hybrid Software-Defined Networking
TE	Traffic Engineering
MLU	Maximum Link Utilization
QoS	Quality of Service
OSPF	Open Shortest Path First
IoT	Internet of Things
IoE	Internet of Everything
OF	OpenFlow
RTT	Round Trip Time
ECMP	Equal Cost Multiple Path
ONOS	Open Network Operating System
BGP	Border Gateway Protocol
AS	Autonomous System
ICMP	Internet Control Message Protocol
CPLEX	Constraint Programming Linear Programming with Extensions
FCM	Flow Control Method
FRR	Free Range Routing
SDN-IP	Software-Defined Network-Internet Protocol
ODL	OpenDayLight
H-STE	Hybrid Software-Defined Network Traffic Engineering
IETE	Internet Engineering Task Force
TCP	Transmission Control Protocol
ACK	Acknowledgment
UDP	User Datagram Protocol
RL	Reinforcement Learning
GPRS	General Packet Radio Service
BDDP	Broadcast Domain Discovery Protocols

HTTP	Hypertext Transfer Protocol
RFC	Request for comments
WSN	Wireless Sensor Network
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
MAC	Medium Access Control
ML	Machine Learning
ANN	Artificial Neural Network
LLDP	Link Layer Discovery Protocols
ISP	Internet Service Provider
CDPI	Controller Data Plane interface
MPLS	Multi-Protocol Label Switching

CHAPTER-1

INTRODUCTION

In recent decades, Internet Service Providers (ISPs') networks have experienced a notable surge in traffic due to the rapid expansion of Internet applications, including games, chat tools, and videos. ISPs have been investing more in constructing their network infrastructure to guarantee a network with maximum bandwidth and minimum latency; however, this surge in traffic has resulted in frequent congestion, causing a lack of guaranteed user experience and satisfaction. Due to the exponential improvement of the Internet of Everything (IoE) and the IoT, emerging Internet applications have explosively brought network traffic [1]. Consequently, worldwide attention is being paid to finding solutions to avoid network congestion and achieve traffic load balance; simultaneously, there is an immediate need to advance network management technologies.

Besides the many advantages that traditional legacy networks provide, there are some limitations that traditional networks cannot offer, such as not providing an excellent level of network control, which makes the network challenging for implementing novel ideas such as network virtualization and traffic Engineering optimization. Network management is one of the main challenges in today's network, and professionals and businesses need help with network management; several concepts and structures have been suggested globally for future network design. To address these challenges, the design of the Software Defined Network started. The basic idea behind an SDN network is to separate the control and data planes. In recent years, SDN networks have been among the most discussed topics in the telecommunications systems disciplines. It is a novel and critical network design idea that arises with the advancement of Virtualization, Mobility, the Internet of Things, and other technologies; it is one of the hottest topics in telecommunications. SDN networks are designed to decouple the control and data planes [2]. SDN provides reliable and flexible network management. Due to its global perspective and centralized network control, it allows intelligent flow scheduling systems to improve network throughput and link utilization and supports Advanced Traffic Engineering (TE).

An SDN is an emerging design that manages complex network infrastructure by separating the data and control layers. With its worldwide perspective and centralized management, SDN enables flexible and reliable network management, increasing network throughput and link Utilization. Furthermore, it provides a unique flow scheduling approach to enhance Traffic Engineering (TE). In recent years, SDN networks have become a prominent subject of discussion in telecommunications. It is an essential and novel network architecture concept that emerged with the advancement of technologies such as mobility, Virtualization, and the Internet of Things (IoT).

Traffic Engineering is a highly effective network administrative tool that balances network load and enhances network performance by optimizing traffic routing methods [3]. TE is an efficient approach that balances the flow, prevents congestion, and improves network performance; it enables control over how traffic is routed over the network; it has gained substantial interest from both business and academia in the past few decades. However, due to the network's complex structure, the optimization of traffic engineering remains an open area for research. Figure 1.1 shows the basic architecture of a Conventional legacy network, where the data and control layers exist inside a single device. The fundamental shortcoming of today's network is the need for Traffic Engineering and QoS guarantees. Software-defined networks were introduced to solve this problem by separating the data and control layers. It provides the dynamicity and flexibility to define QoS policies that effectively push them to the data layer devices to enhance network throughput and link utilization.

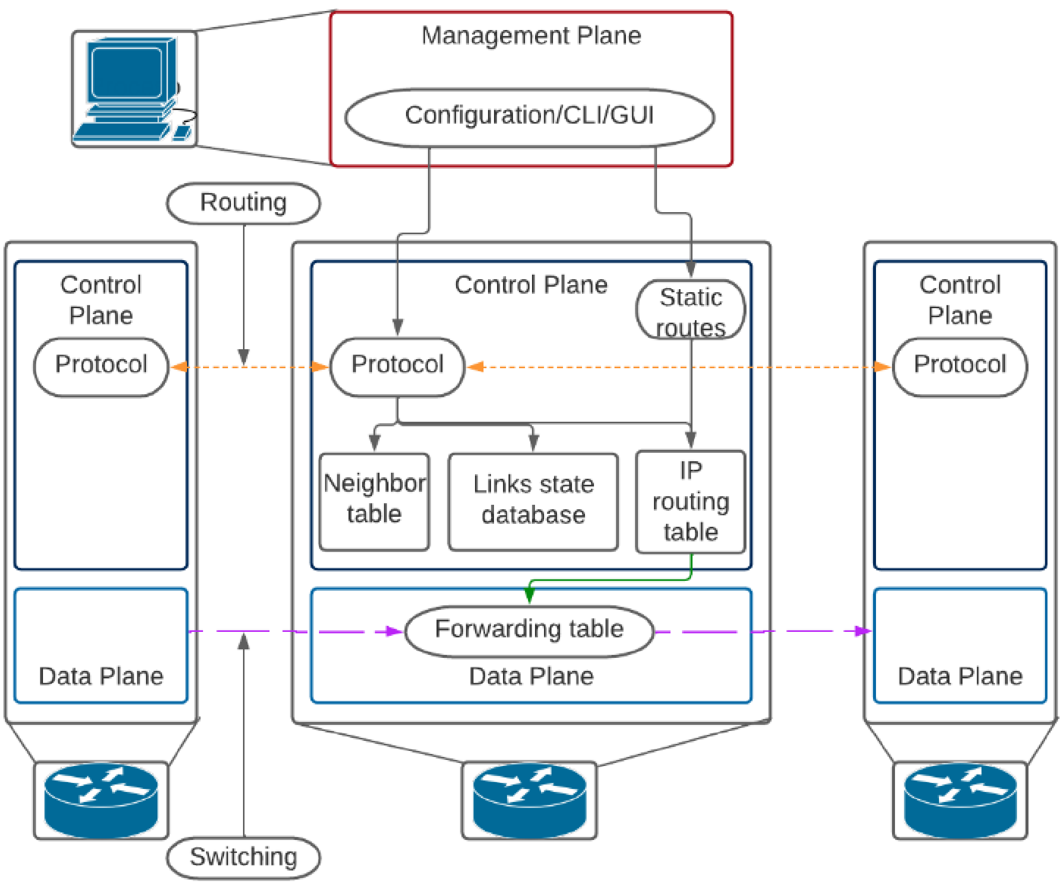


Figure 1.1: Conventional legacy network architecture

SDN is a well-established Network design that separates the network's control and data planes. It offers an efficient and flexible approach to optimizing traffic routing from a TE perspective. In an SDN network, policies or routing algorithms can be implemented in the control plane to provide flexible control over the distribution of traffic to any outgoing links in SDN switches, and the SDN switches can distribute traffic to various paths in a flexible and unrestricted manner in TE [4, 5]. SDN networks can achieve more user-friendly, flexible, and intelligent network control by tailoring the forwarding rules for individual flows [6]. The SDN network design is composed of three main layers: the first layer is the infrastructure layer, which consists of forwarding network devices that route data packets. The middle layer is the control layer, which is the leading layer of the SDN network architecture; the control layer is mainly responsible for controlling the network and translating SDN application layer requests to the SDN data layer, and the last layer is the application layer, which is composed of applications. With a centralized controller, this network can control all the networks with programmable devices; centralized control can significantly enhance overall network performance much better than traditional networks regarding traffic engineering, link utilization, and network throughput. It has gained significant worldwide attention in recent years, and the adoption of SDN has significantly increased among network operators; network operators now can dynamically control the routing flows inside the network and can immediately change the routing choice as needed; for instance, Microsoft [4], and Google [5], have been pushing towards achieving nearly 100% utilization of the links. The SDN network provides significant advantages in traffic engineering (TE) by allowing dynamic and timely network route adjustments, effectively addressing unexpected issues such as traffic variations, link failures, etc. SDN optimizes TE flexibly, which makes the SDN network a unique and attractive technology among other technologies.

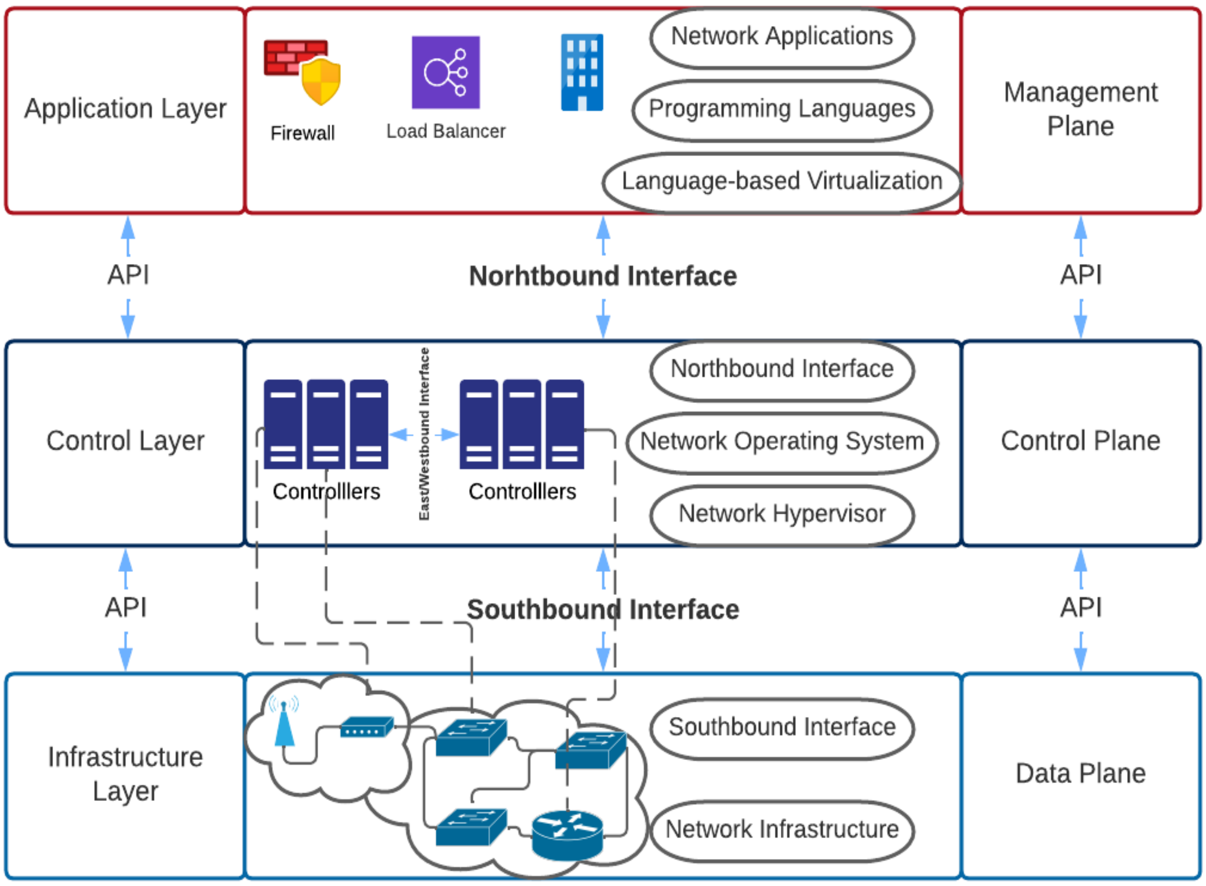
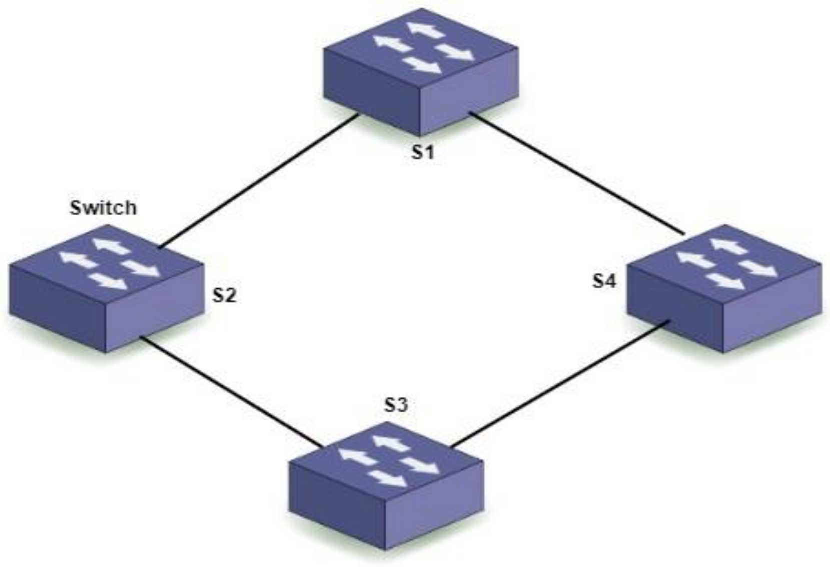


Figure. 1.2: SDN network design

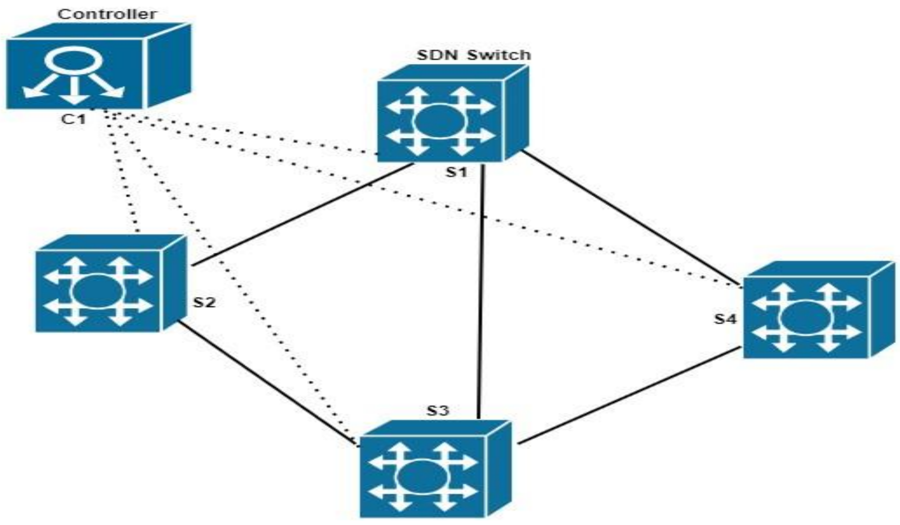
The SDN network is a novel and vital network design that has evolved because of the growth of technologies such as virtualization, the Internet of Things, and mobility, which allow reliable and flexible network management. With centralized network management and a worldwide view, SDN allows intelligent flow scheduling ideas to increase overall network performance and link utilization while facilitating advanced traffic engineering (TE). Traffic engineering is a handy tool for network managers; it focuses on improving network efficiency and balancing the network load through traffic routing strategies. TE is an effective method for balancing network flows, avoiding network congestion, and improving overall network performance. In recent years, academia and industry have shown a strong interest in improving and optimizing TE; however, due to the network's complexity, research on traffic engineering optimization remains an open area for research. According to TE, the well-known SDN network design provides a practical and adaptable method for optimizing traffic routing. By implementing policies or routing algorithms in the control plane, an SDN network can freely control the distribution of traffic to any outgoing link. This allows the SDN switches to distribute traffic to multiple paths in an adaptable and unrestricted way in TE. Customizing the forwarding rules for each flow in SDN networks can enhance their intuitiveness, flexibility, and intelligence. SDN networks are employed in nearly every type of networking, including cloud networks, data centers, wide-area networks, and wireless networks.

SDNs are employed in almost every environment to facilitate efficient communication among computer networks, including wireless networks, cloud computing, mobile telephones, data centers, and the Internet of Things. SDN has many features that provide agility and flexibility to the network. The basic idea behind SDN is to separate network layers to make the network more agile and flexible. Figure 1.2 shows the basic architecture of an SDN network, where the data and control layers are separated, while the data layer devices are used only for data forwarding. The primary advantage of SDN is the ability to manage the data flow dynamically and efficiently. After a flow passes through an SDN switch, its forwarding direction may be easily adjusted; we refer to this type of traffic as programmable. For large-scale networks, it is necessary to develop a network that can be programmed to meet the need for innovation in network access; SDN technology simplifies network administration and control operations through programmable devices and automation. Software-defined network is widely used in different scenarios and states, such as Google [5], public cloud [7], NTT gateway [8], and optical (IP/WDM) networks [9]. Software-Defined Network is used in almost every field of networking, from cloud networks [10,11] to Data Centers [12-14], wide Area Networks [5,16] to wireless networks [17,18]. According to the survey, the software-defined network market is expected to grow considerably from 8.8 billion USD in 2018 to 28.9 billion USD in 2023 at a Compound Annual Growth Rate (CAGR) of 26.8% during this period [19].

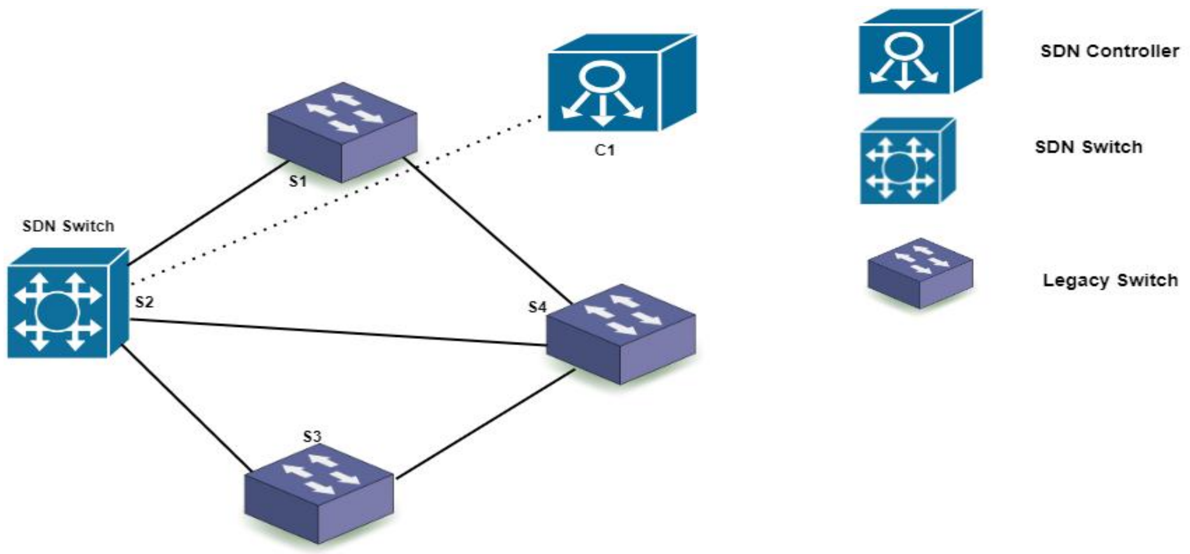
In the control layer of the SDN network, there should be at least one controller. One of the main elements of an SDN network is the controller; therefore, the controller must present features that ensure service continuity during failures and improve the overall network performance. We can have multiple controllers to avoid a single point of failure. For the Wide-area network in different locations, various controllers can be deployed physically to accomplish the purpose of a centralized control plane. Figure 1.3 shows three different network architectures: conventional legacy network design, SDN network design, and hybrid SDN network design.



(a) Conventional network design



(b) Full-SDN network design



(c) Hybrid SDN network design

Figure. 1.3: Different Network Designs

Recently, SDN has been widely researched and progressively implemented in several types of networks, including wide area networks, campus networks, enterprise networks, data center networks, and internet exchange points.

With all the significant advantages and futures an SDN network provides, directly migrating to a Fully SDN network is challenging it can be risky and expensive to upgrade an entire conventional network to SDN, as the OpenFlow protocol is not developed sufficiently [20], and Commercial controllers and Switches of SDN are not entirely reliable and stable [21]. For medium- and large-scale networks with thousands of traditional network devices, migrating directly from a conventional network to a pure SDN network is complicated. There are some limitations and potential challenges, such as financial challenges, technical difficulties for a seamless migration, lack of standards, security issues, downtime fears, and Quality of Service degradation. Many problems must be solved as a technological challenge, including how resilient, robust, and scalable the centralized Controller can be without becoming a single point of failure. One significant reason that prevents the adoption of pure SDN is the necessity to implement most standardized network functions in SDN software to get the same functionality as a traditional network provides; this implementation significantly increases the transition cost to a pure SDN network.

These challenges hold up the deployment of pure Software-Defined Networks and cause the birth of a Hybrid Software-Defined Network (hSDN), which allows regular IP network devices and SDN devices to work together in one environment. Therefore, hybrid SDN and the incremental deployment of SDN nodes would be the best way to address these challenges. A hybrid SDN is a network design where decentralized and centralized networks coexist and communicate. A hybrid SDN combines the programmability and flexibility of the SDN network with the stability and familiarity of traditional network architecture.

A hybrid SDN is a network architecture in which communication and coexistence between centralized and decentralized networks are present to take advantage of both networks. It is unnecessary to upgrade all the legacy network devices to the SDN network; we can incrementally transform them into a hybrid SDN network and then subsequently to a fully SDN network. In recent years, incremental deployment of SDN nodes has received lots of attention; previous research [22–24] indicated that programmable traffic (which involves at least one SDN node) can enhance flexibility in performing TE. Internet Service Providers are incrementally upgrading traditional network devices to programmable switches that support SDN to improve traffic management ability.

The literature provides a distinct definition of hybrid SDN networks. A hybrid SDN network is a logical step in migrating from a conventional network to an SDN network in which SDN and traditional devices coexist. In a Hybrid Software-Defined Network, both traditional network devices and SDN nodes work and coexist. There is no need to upgrade all of our network devices to SDN; we can upgrade our devices to SDN according to our organization's needs and requirements. Hybrid SDN has many benefits as it takes advantage of both SDN and traditional networks; it takes conventional network robustness and familiarity, which traditional network devices provide, robust routing, and programmability of SDN, as SDN nodes will emphasize network optimization. To provide traffic load balance, SDN nodes can randomly divide traffic flows and divide these flows via multiple paths, so network operators can manage the network and balance the load more efficiently.

In a hybrid SDN network, traditional network devices route traffic along the shortest path using distributed routing protocols such as OSPF; in contrast, SDN devices work in hybrid mode, supporting OpenFlow and OSPF protocols. By doing this, SDN and conventional network devices can consider one another legacy devices and share connection information; traditional network switches can forward link information with SDN nodes and treat each other as conventional devices. The SDN controller can also get information by supporting Link Layer Discovery Protocols (LLDP) and Broadcast Domain Discovery Protocols (BDDP) to get information on the network topology and link status.

When it comes to communication, it indicates how, when, and which devices will communicate with each other to ensure network functions work correctly [21]. Many integration levels are described in detail in the reference [25], including integration only at the data layer, integration at the control layer [26], and integration in both layers [27]. The Network's Controller instructs SDN devices to forward packets in hybrid SDN networks. Traditional network devices will continue to employ classic shortest-path routing protocols such as Equal Cost Multipath (ECMP) and OSPF. Once an SDN device is introduced to a conventional network, it is necessary to develop an effective routing mechanism for the newly deployed network to increase network performance and link usage.

Recently, hybrid SDN has gained popularity as a network design. Additionally, the investigation of traffic engineering with the hybrid SDN has attracted considerable interest from industry and academia [25]. In conventional IP networks, OSPF (Open Shortest Path First) is one of the most popular interior gateway protocols [28]. In a traditional IP network, a weight (or a cost) is assigned to every link, and the traffic is directed along the shortest paths (least cost) from the source to the destination nodes using routing protocols that prioritize shortest path routing, like OSPF. Traffic is uniformly distributed among Equal Cost Multiple Paths (ECMP) when multiple shortest paths are encountered. Therefore, the effectiveness of TE in a conventional IP network is directly influenced by the weights assigned to the network. In hybrid SDN, traditional network devices can only support the OSPF protocol; in contrast, the SDN network devices work in hybrid mode; they can support both the OSPF protocol to communicate with legacy devices and the OpenFlow protocol to communicate with SDN controllers and devices.

This coexistence of SDN alongside traditional networks gives rise to the hybrid SDN. Current hybrid SDNs fail to address the possible performance drawbacks caused by a centralized SDN controller. Deploying a proper controller is another issue that may cause significant delays in processing flow requests. Because of these factors, many researchers have proposed the incremental deployment of SDN nodes alongside traditional network devices, such as traffic engineering [29], flexible routing [30], link failure recovery [31], power saving [32], and safe updates [33].

1.1. SDN Architecture

In recent years, Software Defined Networking has been one of the most researched concepts in telecommunications [34]. SDN arose in reaction to improving virtualization, mobility, the Internet of Things, and many other technologies. It is a novel and innovative network design idea that separates control layers and data layers and passes network traffic based on commands of a control layer. The network architecture of SDN Networks is divided into two main parts: a logically centralized control layer and a programmable data layer. The split architecture's control layer contains most network control logic (defined by software programming), simplifying the data layer. As a result, the data layer is controlled exclusively by the installation forwarding decision of the control layer [35-36]. Overall, a software-defined network is considered to have seven crucial characteristics: the decoupling of the data layer and control layer, simple devices, central control, automation and virtualization of the network, independent vendor, programmability, and openness. SDN simplifies network administration and delivers robust networking programmability. SDN meets and continuously changes the demands of network end users for network resources. For instance, in cloud computing [37], the Internet of Things [38,39], and NFV (network function virtualization) [40].

SDN network with OpenFlow protocols allows network operators to give flows at a finer granularity than a traditional network with controllers. In classic networks, packets or flows are shared based on a single or few attributes such as destination MAC address, IP prefixes with the most extended destination, or combination of IP address and TCP or UDP port numbers and so on; SDN enables us to treat and manage flows base on more packet header attributes such as OpenFlow Protocol via CDPI (Controller Data Plane interface).

The OpenFlow protocol connects OpenFlow controllers and forwarding planes through communication. It is the first standard communication protocol for SDN that allows packets to traverse programmable networks. OpenFlow Protocol provides a way to route the flow, and many versions of OpenFlow are available. The fundamental advantage of the OpenFlow protocol is that it allows multiple manufacturers' switches to be set with controllers in the SDN environments. The Controller manages the OpenFlow protocol and instructs switches on managing data packets that arrive. The OpenFlow protocols keep two switch elements: The Flow table and the Secure communication channel, which are both maintained by the protocol and encompass numerous features such as encrypted channels, traffic monitoring, and processing of inbound packets produced by various controllers, among others.

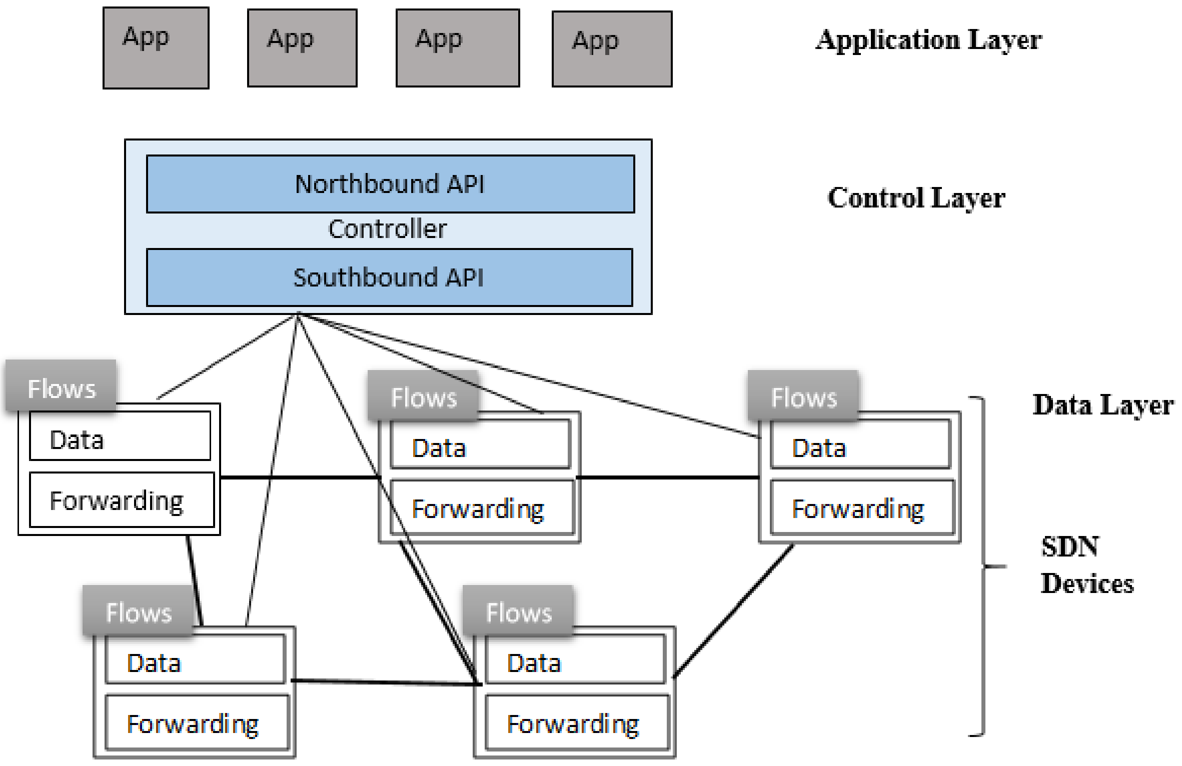


Figure. 1.4: SDN architecture

Figure 1.4 shows the SDN essential architecture consisting of three different layers. The data or infrastructure layer contains data forwarding network elements; the control layer contains the controller for controlling the network; and the application layer, where the applications are situated.

SDN architectures consist of three main layers:

- Application Layer
- Control Layer

- Data layer

1.1.1. Data Layer

In an SDN network design, the data or infrastructure layer is the initial or bottom layer and comprises network components such as access points, routers, virtual switches, physical switches, etc. A data layer controls data transmission, flow statistics collection, and network monitoring. The OpenFlow protocol is the most extensively used CDPI standard for interfacing between control and data layers' devices [41]. A secure channel is an interface that connects remote controllers to the data layer devices, allowing a controller to install and manage devices and switches safely.

1.1.2. Control layer

The second layer of the SDN network is the Control layer or Core layer; we can call it the network brain. This layer consists of one or more centralized software-based controllers that set up and administer the data plane and carry network management capabilities. It controls the flow tables and passes the logic table to the data layer. A southbound interface connects the control land data layers [2]. Controllers have two major parts: Control logic and functional components. Controllers can run multiple function mechanisms, such as a coordinator, visualizer, etc. In the control layer of the SDN network, there should be at least one controller. One of the main elements of an SDN network is the controller; therefore, the controller must present features that ensure service continuity during failures and improve the overall network performance.

1.1.3. Application layer

The last layer of the SDN network is the Application layer, which assists the control layer in configuring and managing networks following the application's needs. This layer contains one or more network applications according to our network needs for adding new network functionalities, such as Security applications, Routing Protocols, Visualization, Traffic Engineering, etc. The northbound interface connects the application and control layers [42].

Table 1.1: From conventional network to SDN

Conventional Networks- The limitations	**SDN- The Game Changer**
▓ **Rigid infrastructure**	▓ **Programmable infrastructure**
Hardware-bound configuration, hard to scale or reconfigure.	Software-defined behavior, easily reconfigured.
▓ **Manual Configuration**	▓ **Automated configuration**
Device-by-device setup leads to human errors and slow deployment	Centralized control enables faster and safer network setup
▓ **Distributed Control Plane**	▓ **Centralized Control Plane**
Lack of centralized visibility, each device has isolated intelligence	Complete visibility and real-time decision-making
▓ **Limited Traffic Awareness**	▓ **Intelligent Traffic Analysis**
Inability to adapt dynamically to varying network traffic loads.	Dynamic path selection and traffic shaping based on real-time data.

1.2. SDN Benefits

The advantages of SDN are apparent: (1) network programmability, which promotes network automation; (2) network administration, which reduces operating costs; and (3) network virtualization. These advantages motivate organizations and network operators to upgrade their conventional networks with SDN-enabled devices and servers. SDN benefits are simply the capacity to regulate flows flexibly and dynamically; once flows pass through an SDN device, the forwarding path might be flexibly controlled; such traffic is referred to as programmable traffic. SDN is utilized for multiple purposes, including traffic engineering [29], flexible routing [30], link failure recovery [31], power savings [32], and safe updates [33]. SDN has been extensively researched and used for campus networks [21], data center networks [43], wide area networks [5], and internet exchange points [44]. To avoid single-point-of-failure problems, backup controllers can be used [45-46]. Popular SDN controllers (ONOS and OpenDaylight) often employ three different controllers to offer full service at a single location, and all the controllers interact with each other (through Raft [47]) to ensure network state consistency [48-49].

1.3. SDN Deployment Challenges

The deployment of Pure SDN has many limitations and challenges. One significant reason that prevents the adoption of pure SDN is the necessity to implement most standardized network functions in SDN software to get the same functionality as a traditional network; this implementation significantly increases the transition cost to a pure SDN network. For medium- and large-scale networks with many standard devices, migrating directly from a conventional network to a pure SDN network is complicated. There are some limitations and potential challenges, such as financial challenges, technical difficulties for a seamless migration, lack of standards, security issues, downtime fears, and Quality of Service degradation. These issues stall the implementation of pure Software Defined Networks and give rise to the birth of Hybrid Software Defined Networks (hSDN), in which SDN and conventional devices coexist [50].

Many problems need to be solved as technical challenges, including how resilient, robust, and scalable the centralized Controller may be without being the only source of failure. As the Internet and networks expand, more than one SDN controller may be required. A controller in the SDN network is an attractive proposition for an attack in terms of security [51]. In the absence of a secure and robust controller, there are possibilities and opportunities for attackers to edit the controller code and modify the underlying network's behavior. As a result, a strong emphasis on SDN Security is essential to make SDN powerful and more beneficial. Until now, potential vulnerabilities exist across SDN platforms, and there are limited discussions on SDN security in the industry and research community. For instance, the systems for authentication and authorization that allow different companies to use network resources without ensuring proper resource protection have been called into doubt [52]. In terms of financial challenges, full deployment of the SDN network requires a significant budget expenditure that most enterprises can only pay some at a time. The new module requires the network administrator to develop, apply regulations, and implement modules in new hardware devices. No well-tested, production-grade techniques are available for medium/large-scale network deployment.

Regarding business challenges, migrating to Full SDN may create service downtime for end customers. As a result, network operators must build trust in implementing a new module over having well-established, time-tested old modules. Purchasing and installing SDN devices to replace the traditional network infrastructure would require a significant capital investment and an increased operational load for the internet service provider. ISPs may hesitate to perform a one-step migration from conventional networks to SDN networks.

Traditional Networks avoid implementation, as mentioned earlier, due to costs, stability, and security difficulties. We need to concentrate on building an efficient and flexible network module that combines both traditional and SDN devices to effectively take advantage of and benefit from both, and fulfill the limitations and challenges mentioned above.

1.4. Hybrid SDN Network

A hybrid Software-defined Network is a networking system that combines conventional and SDN network equipment. This type of network interacts to variable degrees to control, alter, configure, and manage network behaviors to optimize the user experience and performance. There is no need to upgrade all conventional network devices to Software Defined Networks because both SDN and conventional devices work together in Hybrid SDN Networks to benefit from both the programmability of SDN networks and the robustness of traditional networks. In this type of network, an SDN controller manages some parts of the network, while conventional network devices control the other parts. In a hybrid SDN architecture, the controller controls and manages specific network components, such as virtualized environments or the data center; conversely, conventional networking protocols will control other network segments, such as legacy systems or access layers. Hybrid SDN networks balance the benefits of SDN's programmability and centralized management with traditional networks' familiarity, robustness, and stability. Legacy network devices provide reliable routing, whereas SDN devices concentrate on network optimization. In contrast, SDN controllers route network traffic from a worldwide perspective.

Traditional network devices can only support the OSPF protocol when legacy and SDN devices coexist in a hybrid SDN environment. In contrast, the SDN network devices work in hybrid mode. They can support the OSPF in communicating with legacy devices and the OpenFlow protocol in communicating with SDN nodes and controllers. Legacy routers are limited to passing network traffic using the shortest path listed in routing tables. At the same time, SDN controllers can split the specific flows into multipath forwarding entries on SDN devices. As a result, the hybrid SDN architecture blends the robustness of a conventional network with the flexibility of a centralized network.

In hSDN, network traffic is routed according to the decisions made regarding distributed routing and switching technologies. Traffic Control in Hybrid Software-defined networks depend on deployment strategies (service-based, class-based, Island-based, controller-based, Agent-based, Hal-based, and overlay-based). SDN Network will control and transfer some traffic, while others will be maintained using traditional mechanisms—a logically centralized controller judges how to process and forward network traffic. Conventional network devices employ protocols such as ECMP (Equal Cost Multipath) and OSPF to forward packets in a Hybrid Software Defined Network. A centralized controller will control the only Software Defined Network device. These advantages make a hybrid software-defined network much more attractive and exciting for the network.

In contrast, controllers in Software Defined Networks route network traffic from a worldwide perspective. Legacy switches in conventional networks employ distribution algorithms such as IGP to govern traffic routing. However, in the case of a Hybrid Software-Defined Network design, legacy devices will manage some traffic while SDN controllers will handle the rest. There is no need to replace the existing infrastructure completely; this incremental deployment of the SDN method allows organizations to migrate to a hybrid SDN gradually and then to a complete SDN network. It offers flexibility in controlling different types of network traffic and optimizing TE and network performance based on specific policies or requirements.

The essential component in the control layer is the SDN controller, which controls all data plane operations; the capabilities and performance of the controller are crucial in achieving optimal network performance. As there are multiple SDN controllers, selecting the appropriate SDN controller based on the network topology is the key to achieving optimal network performance. We have studied and examined different controllers for the hybrid SDN environment, as multiple controllers exist. Notable controllers are NOX, Floodlight, POX, ODL, Ryu, and ONOS. Finally, we have selected the ONOS (Open Network Operating System) controller as the SDN controller for our hybrid SDN environment, as it is compatible with the hybrid SDN network and supports SDN-IP applications. SDN-IP is a program that integrates legacy and SDN networks; it runs on top of the ONOS controller and allows SDN devices to integrate and communicate with the legacy routers through BGP ASes.

Open Networking Operating System (ONOS) and OpenDaylight controllers usually use three controllers to provide resilient service at one location and to guarantee network consistency. All the controllers communicate with each other. ONOS is an open-source controller with an application for integrating SDN networks with conventional legacy networks via BGP, that is, SDN-IP, located at the top of the application layer of the ONOS controller.

According to different analyses in recent surveys of hybrid SDN networks, the coexistence of different control planes can be the source of security issues, as a Hybrid software-defined network is also open to a wide range of security issues inherited from SDN networks and traditional networks. In real-time, security is one of the leading and vital topics in any type of network, especially in SDN and hSDN networks, as it is an open and centralized network, and the chance of attack might be increased. Research and survey work in this field still needs a lot of work, and many challenges must be solved. More practical research is required to solve these challenges and make hybrid SDN better and perfect for every networking field.

1.5. Hybrid SDN models

When users want to implement a hybrid SDN network, they must keep some considerations and limits in mind. The current legacy network can provide multiple services, and many users may want to keep all these traditional systems; otherwise, giving up these legacy devices would be wasteful. Because SDN is a new technology, the associated Software and hardware may have yet to be thoroughly tested and thus may be unreliable. Users with a minor or limited budget may request a small number of OpenFlow hybrid Switches to be deployed into the traditional network. There are two hybrid deployment methods: incremental and replacement deployment for hybrid networks. Several factors must be considered when building hybrid SDN networks, including traffic, load balancing, network performance, and quick failure recovery. To reap the benefits of both ideas simultaneously, one recommended technique is to integrate SDN controllers with conventional networking incrementally to get the benefits of both concepts while maintaining network performance and reducing networking service disruptions from a traffic engineering point of view.

An appropriate model of a hybrid SDN network is needed to effectively, correctly, and efficiently deploy a hybrid SDN network. It is required to answer two critical problems: first, how to unite SDN and legacy devices in the presence of a controller, and second, how to get a corrected network status so that the Controller of the Network makes forwarding decisions. There are different classifications and approaches for the deployment of hybrid SDN networks. Categories are based on components and architectures on coexistence in the control layer, coexistence in the data layer only, or coexistence in both layers. It depends on the network administrator and network requirements to define a Hybrid SDN deployment technique suited to the network situation to allow the smooth coexistence of SDN and legacy devices.

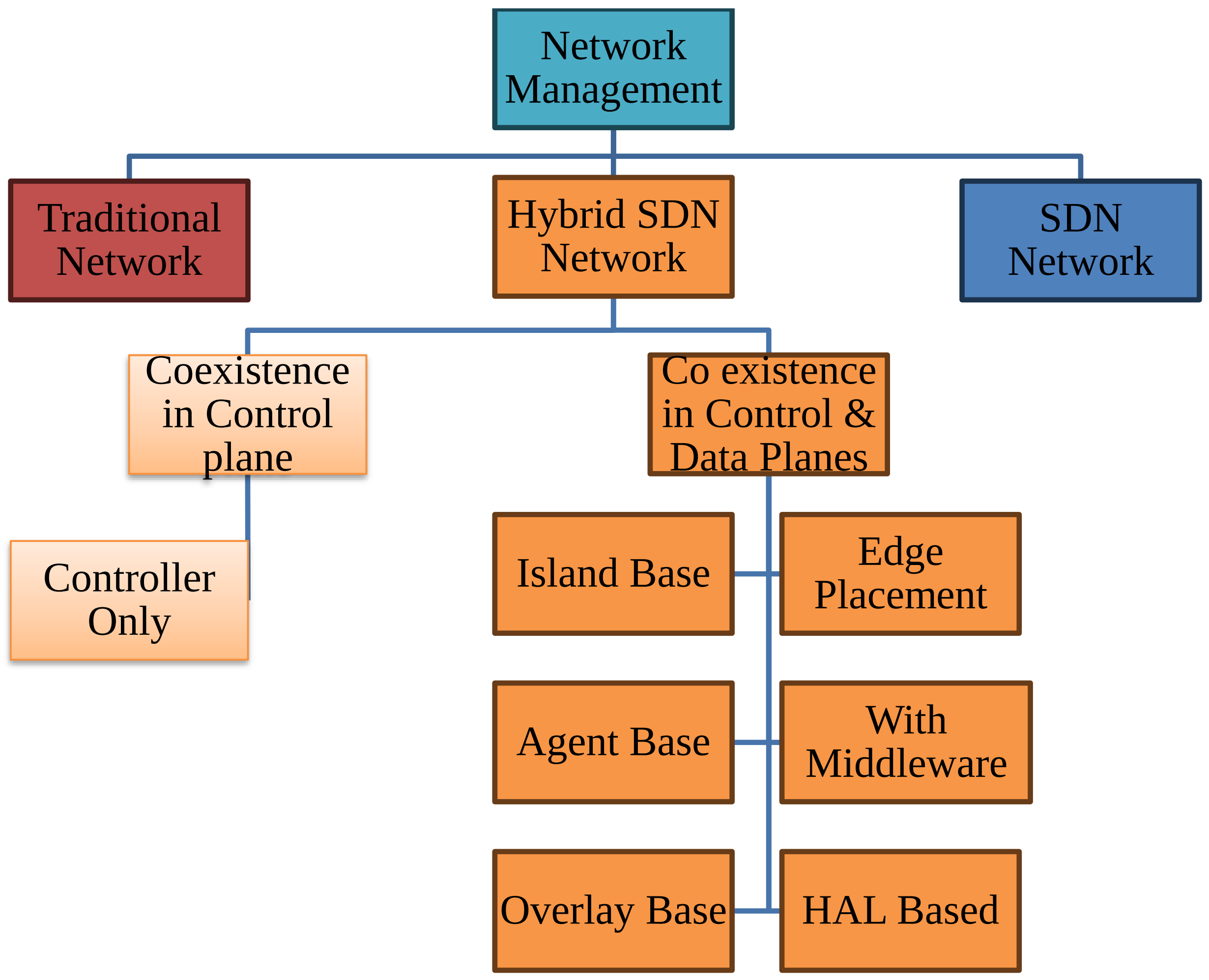


Figure 1.5. shows different categories of the Hybrid SDN network based on the coexistence in the control plane only or in both planes.

There are different strategies for deploying a Hybrid SDN network. The decision on which approach to deploy relies on many aspects, including network type, network services, capital budgets, operational budgets, and performance requirements.

- Agent Base
- Service Base
- Hal Base
- Class Base
- Island Base
- Controller Base
- Overlay Base

These are deployment strategies for hybrid SDN networks, which, in previous surveys [25,53-54], have thoroughly explored and documented the above-mentioned techniques, benefits, limitations, and so on. Readers can read these survey studies for further information on the challenges and benefits of various deployment strategies.

1.6. Motivation

Among the various challenges and issues while migrating from a conventional network to a hybrid SDN network and integrating SDN nodes with conventional network devices, traffic engineering optimization stands out as a critical problem that significantly impacts the effectiveness of quality of service and optimal network performance in a hybrid SDN environment. The motivation for selecting the problem of migration sequence to a hybrid SDN and then optimizing the routing in a migrated hybrid SDN network for this thesis is rooted in the following key factors:

Smarter migration strategies

Among the various challenges and issues faced by integrating SDN nodes with conventional network devices, there is a need for simple and smarter transition strategies for the migration of conventional network devices to SDN.

The growing complexity of modern networks

As user demands grow (Video streaming, 5G, IOT), conventional IP networks struggle to provide all the facilities. Incremental deployment of SDN nodes offers a scalable and cost-effective solution, but lacks well-researched methods for the deployment of SDN nodes in the conventional IP network.

Need for cost-effective Network Evolution

With great motivation and all the advantages that a fully SDN network provides, for many ISPs and organizations, it is not easy to fully upgrade the conventional network to a fully SDN due to economic and technical challenges.

Adaptive Traffic Engineering

Most existing hybrid SDN routing approaches are offline or static. To handle link failures and dynamic user behaviour, real-time optimization is needed, especially in a hybrid SDN environment where both SDN and legacy devices coexist.

Preparation for the future network

Future research could concentrate on integrating hybrid SDN with edge computing, in which hybrid SDN models route data intelligently to the local edge nodes for real-time processing while only sending critical data to the data centers or central cloud.

The selection of the problem of migration sequence and routing optimization in the hybrid SDN networks for this thesis is motivated by the significant impact traffic engineering has on network applications, the need to enhance the quality of service, improve link utilization and efficiency, address scalability challenges and ensure security and reliability. By investigating effective routing optimization mechanisms, this research aims to contribute to the seamless operation, performance optimization, and sustainable growth of hybrid SDN networks, ultimately benefiting a wide range of hybrid SDN applications and their users.

1.7. Problem Statement and Objectives:

This thesis focused on the problem of the migration sequence, incremental deployment, and the routing optimization in the hybrid Software Defined network, in which both issues are interconnected with each other, such as how to migrate to the hybrid SDN, and then how to optimize the routing. To address migration sequence to a hybrid SDN and routing optimization in the hybrid SDN, methods are proposed in this thesis. As hybrid SDN has gained substantial interest from both business and academia in the past few decades, however, due to the network's complex structure, the optimization of traffic engineering remains an open area for research.

The main topic of this thesis is the hybrid SDN architecture, where the SDN nodes are incrementally deployed; we focus on the effective deployment of the hybrid SDN. We investigate and address both the SDN migration and minimizing MLU problems, since they are closely related and intricately entwined. In order to maximise controlled traffic and minimise the network's MLU in a hybrid SDN environment, we first tackle the SDN migration challenge, which involves switching from a conventional network to an SDN network or successfully implementing a hybrid SDN. Second, how can traffic routing be optimised in a hybrid SDN environment, or how can MLU be decreased in the hybrid SDN once SDN nodes are deployed? We presented H-STE (Hybrid SDN Traffic Engineering), a heuristic algorithm for routing optimization that optimizes both SDN nodes splitting ratios with the OSPF weight setting. This improves routing flexibility, reduces MLU, and boosts network performance in the hybrid SDN environment. The H-STE method helps bridge the gap between both networks by optimizing the whole network performance while adhering to the limitations imposed by traditional routing protocols. This method adjusts the splitting ratio of SDN nodes to reduce network congestion while minimizing MLU, and considers other network performance metrics.

We proposed a different method in this thesis. The primary objective of this thesis is to investigate and propose a novel migration sequence for the hybrid SDN and then optimize traffic engineering techniques that have effectively addressed the challenges posed by the minimization of maximum link utilization in the hybrid SDN environment. These techniques should aim to minimize maximum link utilization, delays, packet loss, and network performance degradation while optimizing link utilization and scalability. Through the development of such effective routing optimization mechanisms, this research seeks to provide quality of service optimization methods at different stages and provide performance enhancement of the hybrid SDN network by minimizing MLU, packet loss, and delay.

Based on the literature survey and the research gaps, we have considered the following objectives:

1. To review and compare methods implemented for optimizing Traffic Engineering and Quality of Service in a Hybrid Software-defined network.
2. To propose a model for the migration of conventional networks to Hybrid SDN networks.
3. To propose a mechanism for Routing optimization in a migrated hybrid SDN network.
4. To develop a framework for the interconnection of SDN nodes, alongside conventional network devices.

1.8 Simulation platform

In this thesis, we performed all the simulation experiments on a high-performance computer with a 3.20GHz Intel Core i7 with 16 GB of RAM, and the Mininet Simulator simulates the network environments [55]; for configuring legacy routers, we have used FRR (Free Range Routing) using VTY Shel to configure network protocols. The simulation experiments are coded with the C++ program to model the traffic routing procedure and to determine the splitting ratio of SDN nodes; we employ the CPLEX program in our C++ program. To connect different networks in the conventional network and to integrate the legacy network with SDN AS (Autonomous Systems), we have configured BGP (Border Gateway Protocol) inside all the legacy routers, as it is necessary to have a peering system that makes the BGP protocol work. and the integration between the legacy network and the SDN network is carried out through the SDN-IP. This program operates on top of the ONOS controller. The SDN-IP application allows SDN nodes to communicate with legacy routers using BGP protocols.

Selecting the controller in the hybrid SDN is another issue that needs to be considered; in this simulation experiment, the ONOS [56] controller is chosen to be the SDN controller, and the integration between the legacy network and the SDN network is carried out through the SDN-IP. This program operates on top of the ONOS controller. The SDN-IP application allows SDN nodes to communicate with legacy routers using BGP protocols. To illustrate the capability of our suggested approach, we analyzed our method on different real network topologies, like Abilene, Cernet, Nobel-Germany, and Geant, in our experiment.

Table 1.2: Simulation software specification

Software/Tool	Version	Category	Purpose	Role
Mininet	v2.3.1B4	Network Emulator	Creation of Custom topologies	Emulated scalable Hybrid SDN network.
ONOS	v2.2.0	SDN Controller	Traffic monitoring and Flow control via OF	Implemented to control traffic flows dynamically.
GNS3	v2.2.43	Legacy network Simulation	Configuration of conventional devices	Creating the conventional network topology
VBOX	v7.1.6	Network Virtualization	Creating multiple OS	Used for the virtualization of the network devices
iPerf	v3.13	Traffic Generator	Performance testing	Simulated various traffic loads
Wireshark	V4.2.1	Packet analyzer	Monitoring and analyzing packet-level data	Verified packet flow and latency during simulations
Ubuntu OS	22.04 LTS	Operating System	Hosting the entire Hybrid SDN	Stable platform for Mininet & ONOS

Table 1.3: Hardware specification

Component	Specification	Purpose
Processor (CPU)	Intel Core I7-12700 @ 2.1 GHz	Supports parallel virtual node simulation and SDN process
RAM	16 GB	Smooth handling of multiple virtual hosts and controller load
Storage	512 GB SSD	For fast boot and R/W of packet capturing/logging
Virtualization Tool	VBox 7.1	Used to test and isolate different topologies in a virtual lab environment.
Network Adapter	Intel Gigabit Ethernet (Virtual NIC)	To enable realistic emulation of traffic flow between nodes.

1.9. Organization of Thesis

The thesis presents comprehensive data on Traffic Engineering and Quality of Service optimization in a hybrid Software Defined Network. The study is structured into distinct chapters, each offering unique contributions to the overarching thesis. The following outlines the contributions of each chapter.

Chapter 1: Introduction

In this chapter, we have introduced the SDN, Hybrid SDN, Traffic Engineering, and migration sequence from a conventional network to a hybrid Software-defined network, offering insight into its fundamental concepts and principles, followed by the motivation and problem statement, and also a brief overview of various chapters of the thesis.

Chapter 2: Literature survey

This chapter provides a comprehensive and in-depth survey of the existing literature on the diverse approaches and techniques utilized for the migration sequence and traffic engineering optimization in a hybrid SDN network.

Chapter 3: Migration sequence to a hybrid SDN network

This chapter explains the proposed approach for the incremental deployment of SDN nodes into the conventional legacy network, using a simple greedy algorithm considering load balancing. We have proposed a simple and efficient algorithm for the migration of conventional network devices to the SDN to maximize controllable flow while minimizing the maximum link utilization.

Chapter 4: Routing optimization in a migrated hybrid SDN network

This chapter explains the approach for optimizing routing in a migrated hybrid software-defined network by jointly optimizing the OSPF weight setting and the splitting ratio of the SDN node.

Chapter 5: Interconnection of SDN nodes alongside conventional networks

This chapter explains the method for interconnecting SDN nodes alongside a conventional legacy network using the ONOS controller.

Chapter 6: Conclusion and Future Scope

This chapter encapsulates the findings and outcomes stemming from the approaches and algorithms put forth in this study. It also delves into an extensive discourse on prospective avenues for future research and development.

CHAPTER-2

LITERATURE SURVEY

In this chapter, a survey of various methods and techniques from the state-of-the-art is provided that work in the field of hybrid SDN Traffic engineering and quality of service optimization. This survey is structured into three main sections. The first Section covers and describes approaches for the migration sequence from a pure conventional network to a hybrid SDN network. The second section delves into optimizing different network characteristics in order to optimize Traffic Engineering in the Hybrid SDN network.

2.1. Migration sequence to a hybrid SDN

We describe the related works on the migration sequence and incremental deployment of SDN nodes in the hybrid SDN environment, traffic engineering problems, and routing optimization in different network architectures in a hybrid SDN environment. In recent years, researchers and industries have drawn lots of attention to the hybrid SDN, with the many aspects that can be optimized while migrating to a hybrid SDN, and different algorithms have recently been deployed to optimize different network characteristics in the hybrid SDN environment. The incremental deployment of SDN nodes has gotten much attention recently. Many components of a hybrid SDN may be optimized during the migration process.

Guo et al. [57] investigate the incremental deployment method for the hybrid SDN by presenting it as a traffic engineering optimization problem, identifying the location and the number of migrated devices to minimize MLU in the network. They proposed greedy and genetic searching algorithms as a solution to this challenge. Their findings demonstrated that the genetic algorithm outperformed other techniques in identifying optimal migration sequences. With the deployment of 40% of the nodes, we can get optimal MLU. Kar et al. [50] studied the transition of traditional network devices to the hybrid SDN by framing an improvement in hop/path coverage while reducing cost; they proved that the optimization issue in this transition is an NP-hard problem. They proposed two different heuristic algorithms to model the problem of migration sequence with path/hop coverage. Their result showed that their algorithms achieve more hops and path coverage with the same investment as previous algorithms.

Jia et al. [58] proposed a heuristic algorithm for the incremental deployment of a hybrid SDN to maximize network control ability while minimizing upgrade costs. The simulation result showed that with a 10% upgrading cost, it is possible to control 95% of network flow through SDN devices. Yang et al. [59] concentrate on the problem of selecting suitable SDN candidates, which involves identifying the least number of legacy routers to upgrade to SDN controllers to defend against any single point of failure within a specified network. The technique they created intends to incrementally convert traditional network devices to a hybrid SDN by limiting the repair path of damaged links while reducing delay and improving network bandwidth utilization. Xu et al. [22] proposed a heuristic algorithm for the incremental deployment of hybrid SDN under budget constraints, and they used randomized rounding and depth-first search techniques to solve multi-commodity issues in a hybrid SDN to maximize performance.

Chue et al. [31] proposed a method for preventing congestion in a hybrid SDN network while promptly addressing circumstances involving the failure of single-link breakdown scenarios. Caria et al. [60] separated the OSPF domain into subdomains and implemented SDN at the border routers to enable fine-grained traffic control across subdomains. He et al. [61] investigate TE in two hybrid network modes: barrier and hybrid. They proposed a two-hybrid technique of approximating polynomial time for traffic engineering problems using an approximation of (1+w). Poularakis et al. [62] tried to upgrade the ISP networks across many phases; they prioritized upgrading specific nodes in each phase to optimize programmable traffic.

The primary focus of routing optimization algorithms in conventional distributed networks is often on improving the OSPF link weight setting. Fortz et al. [63] presented an IP-based approach for optimizing intra-domain routing. They used a sophisticated tabu search heuristic technique for the best OSPF weight configuration. Similarly, Ericsson et al. [64] present a genetic method to optimize OSPF weight setting, as traffic in conventional networks is restricted to the shortest path, resulting in a lack of flexibility in routing possibilities. Srivastava et al. [65] provide a Lagrange relaxation-based strategy to enhance traditional network routing. Prior research efforts have primarily focused on optimizing routing in pure conventional or fully SDN networks, and these techniques cannot be applied directly to a hybrid SDN environment.

This is the first study that addresses and elucidates TE concerns in hybrid SDN [29]. In this study, Agarwal et al. define TE as a linear programming problem. They presented the Fully Polynomial Time Approximation approach to balance the flow and enhance routing in a hybrid SDN. A module for the incremental implementation of hybrid SDN is presented by Hong et al. [66], who also offer a variety of heuristic approaches to lower MLU from a traffic engineering standpoint. To enable fine-grained traffic control across subdomains, Caria et al. separated the OSPF domain into subdomains and implemented SDN at the border routers in reference number [60]. He et al. [61] studied and examined TE in two distinct hybrid network modes, namely barrier mode and hybrid mode. The two-hybrid technique provides a polynomial-time approximation strategy for traffic engineering problems with an approximation of (1+w). Guo et al. suggest SOTE in [67] to optimize the OSPF weight setting and SDN node splitting ratio simultaneously in a hybrid SDN; in their case, they jointly optimized the OSPF weight setting and SDN node traffic splitting ratio. Xu et al. [68] examined the classical integration of SDN switching with all devices in a fully SDN environment.

D. Levin et al. [69] presented an optimization approach for determining the partial deployment of SDN, assuming that every flow in the network passes through at least one SDN switch. H. Lu et al. [70] developed a configuration technique to autonomously convert network settings from traditional legacy networks to Software-defined networks. C. Jin et al. [27] manipulated the forwarding entries in the conventional switches by implementing customized flow control methods. This allowed the user to direct the forwarding flow down a specific path chosen by the user rather than following the path determined by the spanning tree protocol.

Jia et al. [58] suggested a method to gradually enhance SDN to optimize the amount of traffic that may be programmed while adhering to a specified budget restriction for the upgrades. Xu et al. [22] compared a hybrid SDN by substituting existing conventional devices with SDN nodes or including new SDN nodes. Caria et al. [60] utilized the characteristics of network topologies and considered the network's centrality to modify switches in a hybrid SDN environment. Agarwal et al. [71] initially developed an optimization problem to achieve traffic engineering in SDN by partially deploying SDN nodes. S.Vissicchio et al.[30] Implemented a centralized control mechanism for distributed OP routing by injecting carefully designed routing messages into OSPF; this approach improved the adaptability, variety, and dependability of Layer 3 routing.

Chu et al. [59] developed a strategy to quickly recover from a single link breakdown while preserving load-balancing performance in the network after recovery. Hong et al. [66] suggested a method to achieve many traffic objectives in the hybrid SDN environment. Guo et al. [67] suggested modifying the connections and flow split ratio weights at the SDN nodes to perform load balancing in a specific hybrid SDN environment.

Much research has been conducted recently to solve TE challenges in the network using ML and DL. Yinga et al. [72] proposed an RL-based (Reinforcement Learning) approach to handle the traffic that is constantly changing to achieve link load balancing in the hybrid SDN network. Xu et al. [73] developed a learning-based traffic engineering approach, TEAL, to optimize WAN traffic engineering by utilizing GPUs' parallel processing capabilities. Liu et al. [74] present DRL-OR, a multi-agent deep reinforcement learning-based online routing algorithm, to enhance the Quality of Service (QoS) by optimizing traffic flow routing with diverse QoS requirements. Lin et al. [75] proposed a model to integrate the transformer model with deep reinforcement learning to enhance traffic engineering performance in a hybrid SDN with fluctuating traffic demands. Guo et al [76] proposed a multi-agent RL-based traffic engineering method that promptly chooses the network flow path in hybrid SDNs.

2.2. Traffic Engineering in Hybrid SDN

Traffic volume has grown dramatically during the last few decades, much like the Internet's fast expansion. Traffic Engineering (TE) has received a lot of interest in both business and academia as it is an effective method to minimize network congestion and increase network performance. Traffic engineering is a powerful technique that aims to measure and analyze real-time network traffic and design routing mechanisms to improve traffic routing and balance network flows. Traffic Engineering is a method that enhances network performance by studying data transmission behavior over communication networks.

The primary purpose of Traffic Engineering (TE) is to optimize network performance to enhance network reliability [77]. It may be accomplished by making the network fault-resilient, eliminating network congestion by balancing the links' load, and so on. Offering TE in a hybrid SDN is challenging because traditional devices are not fully controlled through an SDN controller. The restrictions enforced by the SDN controller to enable TE are only applicable to SDN devices, generally leaving legacy device behavior unchanged. The most prevalent traffic engineering strategies in classical networks are MPLS [78], which employs labels, and GMPLS [79], which expands MPLS to manage new switching technologies and interfaces. However, in an SDN network, a centrally controlled Controller communicates with forwarding components to maintain a worldwide view, traffic demand, network topology, and link state information. Direct routing paths make the SDN network unique and a leading candidate for many Traffic Engineering solutions; different methods for SDN networks have been proposed [80-82].

With the evolution of SDN, the division of control and data layers, and the logical centralization of the control layer, SDN has a practical and flexible way of optimizing routing. As the Controller Has Control of flow routing in the control plane, traffic may be dynamically separated and sent to multiple pathways using SDN switches. Compared to the ECMP used in the standard OSPF protocol, this traffic routing gives a more flexible approach to optimizing routing.

In a pure SDN network, programmability and global centralized network activity control are efficient aspects that allow traffic engineering. Because the flow splitting ratio on SDN switches is variable, using specific complete TE methods in a hybrid SDN network is feasible. To improve the profitability of centralized control, researchers must select a suitable migration sequence, which includes numerous optimization strategies. These approaches may be applied not just to hybrid network deployment but also to other network optimization techniques. For example, migration solutions may expand to include (1)- identifying a suitable placement for the middlebox in the legacy network to get perfect all-right network control [83], (2)- for energy requirements, wireless networks must address the issue of AP location. [84]. (3)- Enhance data center utilization by resolving the VM replacement problem [85].

This part will review the Traffic Engineering technique suggested for the Hybrid SDN network. Load balancing is a Traffic Engineering strategy that aims to optimize the allocation of traffic loads across different resources based on a given performance requirement. In the hybrid SDN network, the TE goal of maximizing link utilization may be met by traffic flow load balance to make network congestion accessible, since load balancing seeks to determine the average link rate usage for all the network links.

In traditional networks, the OSPF protocol is one of the most used IGP protocols; each link is given a cost or weight, and uses the shortest path with minimum cost forwarding. Traffic is routed between the source and destination address and distributed equally when multiple shortest paths are encountered using equal-cost multipath forwarding. Because all nodes in a hybrid SDN network do not support OpenFlow, flow abstraction, complete packet matching, or packet filtering techniques, TE applies a variety of optimization tactics to get the optimum network performance characteristics. In the hybrid SDN network, it is feasible to use particularly extensive Traffic Engineering methods because the flow splitting ratio on the SDN device is arbitrary. To optimize the profitability of centralized control, researchers must select an appropriate migration sequence comprising numerous optimization strategies. Traffic Engineering is more challenging in a Hybrid SDN network; solutions based on static routes or ACLS (access control lists) provide restricted functionality [61].

Traffic measurement entails gathering, measuring, and monitoring network status data. Traffic measurement is difficult in the hybrid SDN network because only a small percentage of devices are SDN-enabled. Traffic management is the study of controlling and arranging network traffic based on network status data provided by traffic measurement.

In Hybrid SDN networks, measuring all flows is unnecessary and too costly. Cheng et al. [86] proposed a method for collecting the load of certain specified connections and estimating the remainder with a 5% margin of error. They present a complete traffic monitoring approach that collects real-time load data for all the links in the network. In this strategy, the controller only has to gather the load of a small selection of relevant links before estimating the load of the other links. The suggested method is better adapted to dynamic traffic changes than existing methods and can minimize MLU by up to 40%. Polverini et al. [87] define and evaluate a valuable criterion for identifying the flows to be monitored that minimize the estimated error dependent on the flow spread parameter. The suggested approach may also assign measurement tasks equitably between network devices while taking forwarding table space into account.

To alleviate the effect of network disruption, one intriguing Traffic Engineering approach is to use ECMP to route the majority of traffic flows and SDN to selectively divert a few flows to balance the MLU of the network. However, critical flow rerouting is difficult due to the ample solution space for essential flow selection. Furthermore, building a heuristic solution for this situation is impossible because rule-based heuristics cannot adjust to changes in the traffic matrix and network dynamics.

Zhang et al. [88] explored Hybrid routing to achieve load balancing while supporting several traffic classes. They suggested a hybrid routing system that combines explicit routing with destination-driven routing to provide load balancing with decreased complexity and high scalability improvements.

Guo et al. [72] suggested an RL (Reinforcement Learning)-based strategy for learning traffic-splitting agents to manage constantly changing traffic to achieve and balance link load in a Hybrid SDN network. They provided an acceptable simulation environment to be built when traffic splitting policies are used to eliminate routing loops. They assess their proposed work on real traffic with different topologies, demonstrating that this strategy provides equivalent network performance and quickly creates good routing strategies.

Wang et al. [89] investigate the efficiency of Traffic Engineering in Hybrid SDN networks depending on traffic distribution and forwarding graphs structure. They defined a coherent forwarding graph and built forwarding graphs with high throughput potential while preserving consistency. Their finding reveals that their forwarding graph creation method outperforms other graph development methods regarding throughput and load balancing.

Yash Sinha et al. [90] created a hybrid SDN network paradigm that used SDN and MPLS-based capabilities and was empirically tested using a prototype implementation. Deployment of this Hybrid network model provides proof of the coexistence of SDN and MPLS-based network devices in the network. This architecture provides a better alternative for a more seamless transition from conventional to SDN networks.

Tu et al. [91] present a tunnel splicing technique for MPLS and OpenFlow router-based heterogeneous networks. The suggested paradigm was created on a Linux system and tested on test networks. The result of emulation is also deployed in several applications and demonstrates its practicality and efficiency.

Nakahado et al. [92] proposed a Hybrid SDN approach, where SDN devices can update only edge devices, while all other devices use classic OSPF routing. The OSPF network's incoming traffic is distributed via edge nodes; this method can decrease the congestion ratio in a network. Sharma et al. [93] suggest that conventional network management approaches and controller-based mechanisms in future networks should coexist and operate together to enable progress in deploying SDN devices in traditional networks.

We describe relevant Traffic Engineering (TE) work in three areas: IP Traffic Engineering, MPLS Traffic Engineering, and SDN Traffic Engineering.

2.1.1. IP Traffic Engineering

Fortz et al[63] present an IP-based intra-domain Traffic Engineering method to achieve load balance using Equal Cost Multipath; they optimized the weight of OSPF links. They demonstrate that the OSPF weight setting is an N-P complex problem, suggesting a tabu search heuristic algorithm to find the best OSPF weight setting. Their studies show that this heuristic algorithm can discover the best OSPF weight setting. Experimental results show that this approach outperforms previous techniques and algorithms, as traffic is routed based on the shortest path; this is a restriction of route optimization in an IP network.

Sridharan et al. [94] overcome the limitation of equally dividing Traffic over ECMP. For each routing address, a subset of the following hops, Multiple paths, is chosen for this strategy to transfer traffic more efficiently, simulating unequal traffic splitting to get a near-optimum solution. However, this technique necessitates a complicated configuration for each route prefix. Xu et al. proposed DEFT [95] and PEFT [96] to obtain optimum Traffic Engineering and proposed a novel link-state routing protocol. As they provide ideal performance in their strategy, traffic is not restricted to the shortest path but may route to the longest path with an increase in the penalty; the limitation of this strategy is that multiple commodity devices do not support it.

Using objective genetic functions, Xu et al. [97] present optimum intra-domain traffic engineering as a multi-commodity flow maximization issue. Adding a second weight to each link expands the OSPF routing system, allowing for flexible Traffic splitting among all shortest paths. In general, to attain optimal routing performance in most existing IP-based intradomain TE solutions, removing constraints for the shortest path routing requires setting up or updating the present link-state routing protocols, which are sophisticated and error-prone. Conventional switches must still implement these new routing protocols on an extensive-scale network.

Table 2.1: IP Traffic Engineering

Reference	**Aspect**	**Objective**	**Limitation**
[63]	IP-TE	To accomplish load balancing, maximize the weight of OSPF links.	The shortest path is used to route traffic, which is a drawback of IP network routing optimization.
[94]	IP-TE	They address the limitation of equitably splitting Traffic on equal-cost multipath to achieve a nearly optimal solution.	Every route Prefix necessitates a complex configuration.
[95,96]	IP-TE	Designed unique link-state routing protocols to maximize traffic engineering efficiency.	It can operate at peak efficiency, but most common devices cannot support it.
[97]	IP-TE	Solve the multi-commodity flow maximization problem to represent the ideal intradomain TE.	They add a second weight to each complex and error-prone link in OSPF routing to allow flexible traffic splitting over the shortest path.

2.1.2. MPLS Traffic Engineering

MPLS tunnels are built between source and destination nodes in a Multi-Protocol Label Switching (MPLS) network. Labels added in packet headers allow flows to be transmitted freely among these established tunnels. Elwalid et al. [98] and Kandula et al. [99], for achieving link load optimum routing, proposed two online routing protocols for Traffic Engineering (TE) in MPLS networks that divide flows between pre-established label switching paths (LSPs). Zhang et al. [100] propose a hybrid MPLS/OSPF network. The traffic engineering problem is formulated as a multi-commodity flow problem using linear programming that overcomes the disadvantages of tunnel construction in MPLS traffic engineering and limits the shortest route routing in IP networks to enable a suitable traffic engineering solution. MPLS Traffic Engineering can dynamically distribute traffic over many tunnels, but lacks a global perspective. Furthermore, to forward packets, MPLS TE systems need the formation of MPLS tunnels; in other words, LSPs enhance TE complexity and limit scalability.

2.1.3. SDN Traffic Engineering

Recently, traffic engineering has become a prominent issue in SDN networks. With SDN network development, we can now manage Network flows flexibly and modify their routing in a real-time network. Microsoft [4] and Google [5] have previously built their small-scale SDN-enabled Inter Data Centers network and are capable of nearly 100% network utilization. Quan et al. discuss a novel SINET-based future internet infrastructure and suggest a SINET-specific solution for crowd cooperation in Software-defined vehicle networks [101]. In [102], Quan et al. describe software-defined adaptive Transmission control systems for choosing different transmission control techniques in a time-varying vehicular environment. In [103], Agarwal et al. control co-flows by suggesting an architecture with near-optimal performance achievement.

[29] This paper is the first research paper to discuss and explain Traffic Engineering issues in hybrid SDN networks. In this paper, Agarwal et al. define TE as linear programming issues. To improve routing and flow balancing in a Hybrid SDN network, they proposed FPTAS (Fully Polynomial Time Approximation System). Likewise, in paper no [29], Hu et al. [104] and Wang et al. [105] proposed a solution for a Hybrid SDN network to maximize Traffic Flows. Hong et al. [66] present a module for the gradual deployment of hybrid SDN networks and suggest numerous heuristic strategies to reduce Maximum Link Utilization from a Traffic Engineering perspective. In paper number [60], Caria et al. deployed SDN at the border routers and divided the OSPF domain into subdomains to provide fine-grained traffic management among subdomains. He et al. [61] investigate TE in two hybrid network modes: hybrid mode and barrier mode. They present a polynomial-time approximation approach for Traffic Engineering issues with an approximation of (1+w) in the two-hybrid method.

Xu et al. investigate a comprehensive SDN network in which traditional switching combines all devices with SDN switching [68]. Chue et al. [38] describe a method for quickly responding to single link breakdown scenarios while avoiding congestion in Hybrid Networks. Vassicchio et al. [107] offer Fibbing, which introduces false nodes to provide more flexible routing by centrally controlling the link state routing protocols. Huin et al. [109] focus on the traffic engineering problem to create a smooth, energy-conscious routing system to minimize energy usage. Tajiki et al. present a unique Traffic Engineering concept for an SDN/MPLS Network in a Hybrid SDN network [110]. Guo et al. propose SOTE in [67] to improve the SDN node splitting ratio concurrently with the OSPF weight setting in a Hybrid SDN network; in their situation, they jointly optimized the SDN node traffic splitting ratio and OSPF weight setting. They optimize traffic splitting for SDN devices to increase flexibility and improve OSPF weight setting to guarantee network primary routing optimization efficiency.

Table 2.2: MPLS TE and SDN TE

Reference	Aspect	Proposed Work
98-99	MPLS-TE	For MPLS network Traffic Engineering, they proposed two online Routing Protocols.
100	MPLS-TE	Using a linear programming module, they create a Hybrid with MPLS/OSPF network and formulate a multi-commodity flow problem, the traffic engineering problem.
101	SDN TE	present a SINET-tailored solution for crowd cooperation in SDN vehicular networks.
102	SDN TE	To deal with changing vehicular environments and selecting multiple transmission approaches, they present software-defined adaptive transmission control protocols
103	SDN TE	Manage co-flows by providing a network architecture with near-optimal performance
106	SDN TE	Develop a resource reallocating mechanism that is efficient for software-defined data centers
33	hSDN TE	They present FPTAS and describe the TE Issues as a linear programming problem to balance Flows and improve routing in a Hybrid SDN network.
104	hSDN TE	Create a hybrid SDN system to maximize traffic flows
66	hSDN TE	develop a framework for incremental hybrid network deployment and suggest numerous heuristic techniques for minimizing maximum link utilization in TE
60	hSDN TE	They deployed SDN at the border routers and divided OSPF into different subdomains to provide fine-grained traffic management among subdomains.
61	hSDN TE	present polynomial time approximation techniques for TE issues with an approximation of (1+w)
68	hSDN TE	Investigate a comprehensive SDN network in which legacy switching combines all devices with SDN switching.
44	hSDN TE	Present a strategy for swiftly responding to a single link loss situation and reducing congestion in a hybrid SDN network
107	hSDN TE	Present Fibbing is a strategy for centrally controlling the routing protocol link state by generating false nodes to provide more efficient routing.
108	hSDN TE	Suggest a heuristic approach for improving routing in a hybrid SDN network using variable traffic.
109	hSDN TE	To decrease energy consumption, present a smooth, energy-conscious routing method.

110	SDN/MPLS	They suggest a unique TE design for SDN/MPLS hybrid networks, focusing on an SDN/OSPF hybrid network situation.
67	hSDN TE	SOTE proposed jointly improving the splitting ratio of SDN devices and OSPF weight settings.

Table 2.3: Traffic Engineering and Quality of Service in Hybrid SDN

Refere nce	**Title /year**	**Method**	**Analyze and Result**	**Conclusion**
[111]	Intelligent Quality of Services-aware traffic forwarding for SDN/OSPF industrial internet 2020	K-Path partition algorithm Single-path minimum cos t forwarding	They proposed a scheme that not only focuses on the guarantees of Quality of Services of industrial services, but also focuses on efficient resource bandwidth utilization through traffic load balancing in SDN/OSPF industrial internet.	They proposed a Quality of Services Aware Forwarding strategy by adopting the K-path partition algorithm and the SPMC (single path minimum cost) algorithm for the improvement of industrial application Quality of Services. Their result shows that their scheme guarantees Quality of Services requirements for Hybrid Industrial Internet services, and they efficiently used Open Shortest Path First link bandwidth and distribution of load-balanced traffic in SDN/OSPF industrial internet.
[112]	Intelligent traffic control for Quality-of-Service optimization in hybrid SDN 2021	Deep Reinforcemen t Learning (DRL)	To optimize the Quality of Services in a hybrid Software Defined Network, they propose a near-optimal traffic control method. They implemented the Flow split ratio approach by locating the Open Flow group bucket restrictions.	With open-source traffic datasets, they evaluate their proposed method, and their simulation result shows that through this method, we can achieve an important improvement in network Quality of Service performance.
[22]	Incremental deployment and throughput Maximizatio n routing for hybrid SDN 2017	Heuristic Algorithm, a depth-first search method	To enable the deployment of hybrid Software Defined Networking within economic limitations, the authors introduce a heuristic algorithm and demonstrate its approximation factor of l-le. They utilize a randomized rounding technique along with a depth-first search approach to maximize network throughput	They show that both simulation algorithms and analysis can obtain an important performance improvement and perform much better than other theoretical cases. Their results show that it helps to improve throughput around 40% compared to other schemes, and their proposed routing algorithm is able to improve the throughput of the network around 31% compared to ECMP (Equal cost multipath) in a Hybrid software-defined network.
[113]	On the trade-off B/W load balancing and energy efficiency in a hybrid network 2021	Hybrid Spreading Load Algorithm (HSLA)	They solve two optimization problems: load balance of traffic and network power consumption reduction.	Their simulation result shows that HSLA (Hybrid Spreading Load Algorithm) overtakes other state-of-the-art methods, which challenge one objective only, either saving of network power consumption or traffic load balancing.

[114]	QoS-aware routing in a Hybrid SDN network 2017	Simulated annealing-based QoS-aware routing algorithm	They propose Hybrid Software Define network architecture by using spanning tree protocol as they propose simulated annealing bas Quality of Service aware routing (SAQR) algorithm that has the ability to change weight of bandwidth and loss rate of delay to find best path for Quality-of-Service requirements, as they calculate their projected algorithm in a hybrid SDN Network that runs different application with multiple Quality of Service requirements.	Their findings indicate that the algorithm significantly outperforms existing approaches, such as MINA, in terms of delay, packet loss, and bandwidth efficiency. Specifically, 88% of flows meet delay requirements, 90.8% meet loss criteria, and 86.5% satisfy bandwidth demands, demonstrating strong adherence to Quality-of-Service standards.
[115]	Hybrid Incremental Deployment hSDN Devices 2021	Mixed Integer Linear Programming (MILP) Genetic Algorithm (GA)	They developed the MILP (Mixed Integer Linear Programming) optimization Model, and their aim was to minimize routing and deployment cost and to maximize controlled traffic in the Software Defined network, as they focus on deployments of Software Defined Network devices in a hybrid Software Defined Network incrementally.	There result shows that this method improves the Quality of traffic and rate that managed by Software Define network model, as they proposed Genetic Algorithm as an execution to achieve comparable performance to the propose MILP (Mixed Integer Linear Programming) model and they showed that Genetic Algorithm is much more scalable and stronger than the MILP model as traffic volume rises though improving the percentage of Software Define Network Flows up to 37%.
[72]	Hybrid SDN deployment using Machine learning 2021	RL (Reinforcement Learning)	They propose an RL (Reinforcement Learning) based approach in a hybrid Software Defined network to achieve load balancing link. They address traffic changing dynamically and learns a traffic splitting agent to create a direct link between traffic splitting and traffic demand, as traffic splitting agents are designed to learn offline by manipulating Reinforcement Learning.	As they emulate their work in different topologies and real traffic so their evaluation shows that this method can achieve similar network performance and performs dominance in a fast generation routing scheme.
[116]	Hybrid SDN deployment using ML 2021	Reinforcement learning Techniques	First, in the networking domain, they summarize a generic workflow for Machine Learning. Then, from the perspective of Machine Learning, they study the problem of implementing a software-defined network in a traditional network. They proposed Markov Decision Process and Reinforcement learning techniques for the	They propose a basic workflow of Machine Learning in the networking area to formulate a Hybrid software-defined network deployment using Markov Decision Process in a legacy network. Further, for model construction and validation, learning or SARSA techniques of reinforcement learning can be used.

			deployment problem in a software-defined network.	
[57]	Incremental Deployment for traffic engineering in hybrid SDN 2015	Heuristic algorithm	Their aim was to find the best migration system for the traditional routers to migrate to Software-defined network-enabled routers. They proposed a heuristic algorithm for migration structure for the routers that gets maximum benefit from a Traffic Engineering perspective.	They evaluate their algorithm by comparing it to the Static migration algorithm and the greedy migration algorithm, and conducting simulation experiments. Their result shows that the genetic algorithm is performing very well compared to other techniques in the case of properly deployed migration, where 40% of routers can obtain most of the benefits.

Table 2.3 shows the literature of the most significant related work, Quality of Services and Traffic Engineering (TE) in Hybrid Software-Defined Networking.

2.3. Quality of Services in Hybrid SDN

Suppose we define quality of service for a network as user satisfaction with certain services. It is a composite indicator that assesses user satisfaction. Quality of Service (QoS) pertains to any technology that controls data traffic within a network to minimize packet loss, latency, and jitter. We can say that Quality of service means providing specified data transmission in a particular aspect provided by the network. Quality of Service considers bandwidth, packet loss, service level, data correctness, and other factors. For example, QoS guarantees bandwidth if the network offers excellent bandwidth for bandwidth-sensitive data delivery; we do not need QoS if we have a perfect link. QoS indicates low delay if the network delivers low-latency transmission service for time-sensitive data delivery. Differentiating application flows is required to achieve QoS since they compete for available network resources. Network resources should be allocated to make the right decision concerning packet forwarding; sometimes, this procedure requires knowledge of current network states.

The Quality-of-service provisioning is mainly determined by SLAs (Service Level Agreements) between service providers and end customers. This method does not give finer-grained traffic management, yet it performs excellently for best-effort service. However, different application types, such as video conferencing, online gaming, VoIP, and many more, have Flows sensitive to latency, bandwidth, and jitter, necessitating particular handling techniques. There is no established method for imposing constraints on the depth of traffic distinction and setting high-level traffic management strategies. SDN networks divide the network's data layer and control layer. This division increases network controller control over networks. Controllers can receive a worldwide view of network status. Network applications under the SDN concept are supplied, and the network controller provides a conceptual network perspective. They are not required to deal with the low-level setup of data plane devices.

Quality of service in a network refers to enough bandwidth, minimum latency, a minimum rate of packet loss, and balancing network link loads. As we know, balancing the link load of the network can cause an increase in bandwidth utilization and a decrease in network latency. As a result, balancing the link load requirement of QoS can cover the QoS requirement of latency and bandwidth.

Quality of Service is applied in hard and Soft QoS. An accurate and efficient classification method is required to efficiently categorize data flows and meet QoS criteria in SDN and Hybrid SDN networks.

Further research has been conducted to optimize QoS in hybrid SDN networks. They proposed various schemes to optimize single QoS metrics or multiple QoS metrics for hybrid SDN networks; they used different techniques to manage traffic to deliver the best possible Quality of Service for their created topologies.

Bi et al. [111] suggested a QoS-aware forwarding approach using the SPMC (Single Path Minimum Cost) and K-path partition Algorithms to enhance the QoS of industrial applications. Their simulation findings demonstrate that their suggested strategy meets QoS criteria and effectively balances traffic load using OSPF link bandwidth resources in an SDN/OSPF hybrid industrial internet.

Huang et al. [112] present a comparative optimum traffic control approach for hybrid SDN QoS optimization. They tested their strategy with open-source traffic datasets. Their findings indicated that this strategy might significantly enhance network Quality of Service performance metrics like jitter, latency, and link utilization.

Xu et al. [22] present a heuristic approach for incremental deployment with a limited budget. For throughput maximization routing, they use a randomized rounding mechanism and the depth-first search approach. They demonstrated that both algorithms and analysis could outperform theoretical instances. Their incremental deployment improves performance by roughly 40% over the prior deployment strategy, while their suggested routing algorithm improves throughput by 31% over ECMP in a hybrid network.

Kelkawi et al. [117] try to describe an effective method for the incremental deployment of legacy network switches to enable SDN, aiming to minimize MLU of the Network. They use a mixture of two meta-heuristic algorithms (Particle Swarm Optimization and Ant Colony Optimization) to find the best sequence. Their simulation findings demonstrated that their suggested technique outperforms previous static migration algorithms, minimizing maximum link utilization by up to 8.5 percent.

Salman et al. [118] provide a technique for ensuring QoS for time-critical applications across hybrid SDN networks. With a 0.1 link usage level, the method obtains a 60% improvement. Their way of searching for the shortest path does not degrade in performance.

Hong et al. [66] designed a system with an SDN controller to answer these questions: which conventional device to upgrade, and how SDN and conventional devices interoperate in the hybrid environment. Their result shows that with 20% of devices upgraded to SDN, their system can reduce 32 % of Maximum link utilization compared to a traditional network, and compared to a pure SDN network, only an average of 41 % flow table capacity is required.

Guo et al. [67] proposed the SOTE algorithm of a Heuristic scheme to optimize the traffic splitting proportion of SDN devices and OSPF weight setting. They developed MINLP (Mixed Integer Non-linear Programming Problem) to investigate the optimization of Traffic Engineering in a Hybrid SDN network. Their finding shows that SOTE can reduce MLU by 24.19 % on average and 11.72% in Hybrid SDN compared to the Traffic splitting ratio and OSPF weight setting. They also show that with only 10 % of SDN switch deployment, we may benefit from a hybrid SDN network.

From the viewpoint of machine learning, Siew et al. [116] address the topic of SDN implementation in traditional networks. They introduced the Reinforcement Learning and Markov Decision Process approaches for SDN deployment. They outlined the generic machine learning process in networking and proposed a hybrid SDN implementation in traditional networks using the Markov Decision Process. Model creation and validation can be accomplished using RL techniques such as Q learning or SARSA. Ren et al. [119], for TE performance, present the FRS (Flow Routing and Splitting) algorithm. They simulate their algorithm with different SDN deployment rates, and their findings show that 20% of SDN device deployment FRS methods can achieve minimum link utilization compared to previous works.

Guo et al. [57] present a heuristic technique, known as a genetic algorithm, to locate migration sequences that will benefit most notably from the viewpoint of traffic engineering. They assess and compare their approach to greedy and static migration strategies. Their findings show that the genetic algorithm outperforms alternative migration sequences, proving that 40% of the adequately deployed routers can reap the most benefits.

Jia et al. [58] present a heuristic strategy for deploying SDN devices in hybrid SDN networks. Their plan works in two separate scenarios. 1-Increasing network control capability while keeping the upgrade budget unchanged. 2-reducing upgrade expenses to gain the finest network control capability. The results demonstrated that this approach could accomplish 95 percent of the controlled flows with only a 10% upgrading cost. This technique can regulate 95 percent of a network's flow for approximately 10% of the cost of upgrading.

Guo et al. [108] construct and verify the NP-hard problem of TE across Multiple Traffic Matrices. They present a heuristic approach that combines online splitting ratio and offline weight setting optimization to optimize routing over many TM. On measured traffic datasets, they test their technique with three network topologies. Their findings show that the MLU of the Network may be significantly improved, and their suggested framework outperforms competing approaches.

Hu et al. [104] investigated methods to maximize flow in hybrid SDN networks. To solve the issue, they created FPTS (Fully Polynomial Time Approximation Scheme). When 50 percent of SDN devices are installed, simulation results using real network topologies show that a hybrid SDN network outperforms conventional networks and can attain near-ideal network performance.

Hung Chen et al. [120] describe the problem of hybrid SDN deployment as an optimization problem with resource restrictions. Because this is NP-hard, they apply greedy-based heuristic algorithms and propose the Hybrid Score switch deployment technique. In a hybrid SDN network, Hybrid Score seeks to deliver the highest control capacity given a set number of switches to upgrade. They evaluated their work with a real-world enterprise network topology. They found that their Hybrid Score improved the result by 11.28 percent compared to existing strategies. Their Hybrid Score provides comparable control capacity to a fully SDN environment with only 13 percent of SDN switches.

Table 2.4: Migration and Quality of Service in Hybrid SDN Network

Reference	**Proposed scheme**	**Result**
111	To enhance industrial applications' Quality of Service, they presented a QoS-aware forwarding approach based on SPMC and K path partition algorithms.	Their simulation findings demonstrate that their suggested strategy ensures QoS standards and traffic load balancing and efficiently leverages OSPF connection bandwidth in a hybrid SDN/OSPF for the industrial Internet.
112	For QoS optimization in a hybrid SDN network, they present a near-optimum traffic control approach	as they test their strategy with open-source traffic datasets. Their findings indicated that this strategy might significantly enhance network QoS performance metrics such as jitter, delay, and link utilization.
58	present a heuristic approach for incremental deployment with a limited budget.	They demonstrated that both algorithms and analysis could outperform theoretical instances. Their incremental deployment improves performance by roughly 40% over the prior deployment strategy, while their suggested routing algorithm improves throughput by 31% over ECMP in a hybrid network.
117	They describe an effective gradual deployment sequence of conventional devices to SDN switches based on TE to restrict the network's maximum link utilization.	Their simulation findings demonstrate that their suggested technique beats previous static migration algorithms to minimize maximum link utilization by up to 8.5 percent.
118	They propose a solution to ensure the quality of services for time-sensitive applications on a hybrid SDN network.	With a 0.1 link usage level, their approach produces a 60% improvement. Their method seeks the shortest, most comprehensive path with no reduction in performance.

114	They introduced the SAQR (Simulated Annealing-based QoS-aware Routing) method to discover the best-suited path based on QoS criteria.	They test their proposed algorithm in a hybrid SDN network. Simulation results show that the SAQR algorithm outperforms similar work, MINA, regarding latency, loss, and bandwidth ratio, with 88 percent, 90.80 percent, and 86.5 percent of Flows fulfilling their respective QoS standards.
67	They proposed the SOTE algorithm of the Heuristic scheme to optimize the traffic splitting proportion of SDN devices and OSPF weight setting. They developed MINLP to investigate the optimization of Traffic Engineering in a Hybrid SDN network.	Their finding shows that SOTE can decrease MLU by 24.19 % on average and 11.72 % in Hybrid SDN compared to the Traffic splitting ratio and OSPF weight setting. They also suggest that with only 10 % of SDN switches deployed, we may get most of them to benefit in a hybrid SDN network.
89	They investigate how the efficiency of Traffic Engineering is affected by both forwarding graph layout and traffic distribution in hybrid SDN Networks.	Their results reveal that their forwarding graph creation method outperforms other graph development methods regarding throughput and load balancing.
72	They suggested an RL (Reinforcement Learning) based strategy for teaching traffic splitting agents to dynamically address changing traffic and balance link load in a hybrid SDN network.	They recommended building a suitable simulation environment where traffic-splitting techniques are used to minimize routing loops. They tested their proposed work on real traffic on different topologies and showed that it achieves equivalent network performance and succeeds at quickly constructing good routing strategies.
119	They suggest the Flow Routing and Splitting (FRS) method regarding TE performance.	They ran simulations with varied SDN deployment rates, and their finding shows that with the only deployment of 20% of SDN switches, the FRS algorithm can achieve a Minimum MLU than previous works.
57	They suggest a heuristic approach, known as a genetic algorithm, to determine the optimal migration sequence from the viewpoint of traffic engineering.	They evaluate their algorithm and compare it to greedy and static migration algorithms. Their result shows that the genetic algorithm surpasses other migration sequences, proving that 40% of adequately deployed routers can reap the most benefits.
108	They formulate and verify the NP-hard issue of TE across Multiple Traffic Matrices. They present a heuristic approach that combines online splitting ratio and offline weight setting optimization to optimize routing over many TM.	On measured traffic datasets, they test their technique with three network typologies. Their findings show that the Network's MLU may significantly improve, and their suggested framework outperforms existing approaches.
104	They investigated methods to improve flow in hybrid SDN networks. They created FPTS to address it.	With 50 percent of SDN devices deployed, simulation findings using actual topologies demonstrate that hybrid SDN outperforms classical networks and can achieve near-ideal network performance.

2.4. Hybrid SDN Optimization Strategies

We will look at prior work in the field of optimization techniques for migrating Networks from pure traditional Networks to Hybrid SDN networks.

Guo et al. [57] present a Traffic Engineering optimization issue and investigate incremental deployment methods for migrating sequences to IP/SDN networks; they estimate the number and location of routers to upgrade to minimize Maximum Link Utilization. The author of this research claims that finding an optimal migrating sequence is much more complex and challenging than the traveling salesman problem, as MLU is based on previously visited nodes, and the cost of edges is not constant. To solve this problem, they propose greedy and genetic searching algorithms and compare their algorithms with static migration techniques. Their finding demonstrated that the proposed evolutionary algorithm could discover a better and more effective migration sequence. With the migration of 40 % of the devices, we can achieve a minimized MLU of the Network.

In reference number [50], Kar et al. studied migrating from legacy and pure conventional Networks to hybrid SDN networks using a different combination of optimizing hop and path coverage while reducing cost. They proposed a Heuristic algorithm and demonstrated that this problem is NP-hard. They offer MUcPF and MHcPF, written in the MATLAB program, to model route and hop coverage migration sequences. They tested their algorithm against three other algorithms (Accidental coverage, weighted coverage, and Degree coverage algorithms). The simulation results demonstrate that the created algorithms provided higher route and hop coverage than the current algorithms for the same expense percentage. MUcPF requires just 15% expenditure to obtain 100% route coverage, unlike other algorithms requiring 20-30%.

Jia et al. [58] present a heuristic strategy for deploying SDN devices in hybrid SDN networks. Their plan works in two separate scenarios. 1-Increasing network control capability while keeping the upgrade budget unchanged. 2-Reducing upgrade expenses to gain the finest network control capability. The results demonstrated that this approach could accomplish 95 percent of the controlled flows with only a 10% upgrading cost. This technique can regulate 95 percent of a network's flow for approximately 10% of the cost of upgrading.

Hui Yang et al. [121] proposed a distributed control architecture for SDON (Software-defined optical networking) using the blockchain technique (BLockCtrl), aiming to detect a single point of failure within the SDON control plane to address the security and efficiency issues faced by SDON. Their simulation result showed that their proposed architecture could significantly decrease the impact of DoS attacks and fault-tolerant control in the SDON network.

Yang et al. [59] concentrate on selecting SDN candidate issues to preserve all possible single points of failure in a network, which involves selecting a small portion of traditional routers to migrate to SDN controllers. This method intends to gradually update conventional network devices to hybrid SDN networks by limiting and repairing the route length of broken links, decreasing packet delay, and increasing network bandwidth utilization. Performance assessment revealed that the proposed approach reduced the number of SDN switches required while obtaining a relatively short path length. The authors' second strategy is discovering the repair path that minimizes maximum link utilization following a connection failure. Because of the higher number of repair path candidates evaluated, the author's technique outperforms the greedy algorithm (2.9% to 15% decrease) with fewer SDN switches.

2.4.1. Routing optimization.

Huang et al. [111] present DRL-TC, a novel DRL-based architecture optimizing network QoS performance in hybrid SDN networks. Based on historical traffic records, they first look for an appropriate SDN migration sequence to maximize total controlled traffic. In contrast to the prior heuristic solution for TE in hybrid SDN networks, DRL-TC focuses on self-learning routing algorithms and adaptive algorithms in various traffic situations. The results reveal that their suggested strategy achieves optimum performance in optimizing maximum link utilization with rising ratios in the deployment of SDN and a significant improvement in lowering network latency and jitter compared to alternative solutions. Routing optimization in conventional distributed networks is primarily concerned with improving OSPF link weights.

As with SDN network development, network operators can manage the network and the traffic flow in a much more flexible way, and in a real-time network, they can change their route selection. Prior research studies focused on the routing optimization of pure SDN networks, which differs from Hybrid SDN networks. Those routing optimizations cannot directly apply to hybrid SDN networks.

Caria et al. [60] implement SDN at the border routers and divide OSPF into sub-domains to provide fine-grain traffic management across sub-domains. Xu et al. [122] optimize routing in a particular Hybrid SDN situation; they collaborated to enhance flow routing and gradual deployment in a hybrid SDN network. XU et al. [101] investigate all devices with SDN and classical switching. Vassicchio et al. [107] offer Fibbing, which introduces false nodes to provide more flexible routing by centrally controlling the link state routing protocols.

2.4.1.1. Legacy optimization mechanisms

The study [54] thoroughly examined the Deployment of Hybrid SDN technologies and discussed traffic optimization methodologies. Agarwal et al. [66] present the FPTAS method to minimize maximum link utilization (MLU). The authors employ a greedy technique to select SDN sites based on optimal throughput gain. The analysis assumes a particular traffic matrix, and actual traffic will diverge from this traffic matrix. However, information on obtaining precise traffic patterns needs to be provided. Xu et al. [22] suggest maximizing throughput in hybrid SDNs.

Guo et al. [67] maximize the splitting ratio of OSPF weight setting and SDN devices. They developed SOTE (SDN/OSPF Traffic Engineering) to reduce the MLU of the Network. However, getting an ideal OSPF link weight configuration in a dynamic network is complex.

[108] Investigate Traffic Engineering across multiple TM (Traffic matrices). To improve MLU, they present a heuristic technique. Authors believe multiple traffic matrices can better describe MLU optimization; they employ the K method to cluster traffic matrices and believe numerous TM can better depict dynamic traffic characteristics.

Poularaki et al. [62] look at a broad upgrading approach for hybrid SDN implementation. They are primarily concerned with the impact of upgrading time. Guo et al. [24] investigate a hybrid SDN deployment challenge, including controller and SDN switches. Jia et al. [123] offer a heuristic strategy for hybrid SDN deployment that maximizes network control ability while keeping to upgrade budget constraints. [24,123] mainly cover hybrid SDN deployment strategies.

2.4.1.2. AI-based optimization mechanisms

The deployment challenge is different from the traffic control problem. Heuristic methods may be effective in some cases, but may not be effectively adaptable to changes in network state [124]. Even when utilizing various Traffic mattresses to show network traffic, there may be some scenarios that are not taken into account. Several initiatives have been made, and as a result, they have taken advantage of artificial intelligence (AI) techniques such as DRL for adaptive routing. Xu et al. [125] describe a unique DRL-based approach to increase end-to-end communication network throughput and discuss DRL's advantage over previous dynamic control strategies. Yu et al. [126] implement DDPG to improve network link weights in an SDN network based on various incentive goals, and Huang et al. [127] deploy DDPG in the SDN network to optimize the Quality of Experience (QoE).

Zhang et al. [128-129] present a unique Reinforcement learning method for TE in an SDN network. A key strategy of this method is automatically determining which significant flows should be rerouted; it is suited for each traffic matrix but is not ideal for hybrid SDN applications. When deployed, the flows that SDN devices may manage are fixed, and the possibilities for selection are restricted. Sun et al. [130] propose SINET for SDN routing based on DDPG. It picks critical nodes as control nodes. It uses its link weights to build routing pathways.

2.5. Quality of Service (QoS) Routing

Because it provides an open control interface by the SDN network, it facilitates a flexible traffic scheduling scheme for the network; SDN is the appropriate architecture that meets the QoS needs of numerous applications, including video and audio. Egilmez et al. [131] propose an Open QoS concept on an Open Flow Controller with support of end-to-end QoS for multimedia distribution, as it is based on Quality of Service, routing traffic paths are optimized dynamically to meet the needed QoS. They evaluate their open QoS across real-world test networks, and the findings of their experiments reveal that Open QoS guarantees transmission, and Users report little or no visual artifacts when watching the seamless video. Furthermore, compared to existing QoS systems, guaranteed services are handled in open QoS without affecting any other traffic types in the network.

Yan et al. [132] suggest Quality of Service assurances through SDN. Furthermore, they employ several pathways among source, destination, and queueing methods that provide QoS for various Traffic types. The experimental findings suggest that the proposed HiQoS method may minimize latency while increasing throughput to ensure QoS.

Silver et al. [133] present a Network Service Abstraction layer (NSAL) implementation above the management and control layers. The authors present a uniform data model for traditional and SDN devices, allowing for unified management and configuration of both networks to provide Quality of Service for time-critical jobs.

Lin et al. [125] present a QAR (QoS-aware adaptive routing) scheme for an SDN network with multi-layer distributed hierarchical control plan architectures. Mainly, it is utilized to decrease signaling latency in large-scale SDN networks through three tiers of controller design. This QAR technique suggests providing adequate time adaptive QoS packet forwarding using an (RL) Reinforcement learning technique and a QoS-aware reward function. Simulation shows that this QAR technique outperforms the traditional Q-learning method, making it realistic for large-scale SDN networks.

Bi et al. [111] investigate the challenge of managing QoS-aware data flows in the context of multiple flows. They present a game theory based on numerous data flow management strategies in the situation of optimum data flow management being an N-P complex issue and capable of minimizing network delay up to (77-98) percent while increasing network throughput by (24-47) percent.

Chienhung et al. [114] present the design of a hybrid SDN network that may use the Spanning Tree Protocol to identify the existence of traditional switches to gain a global perspective on the hybrid SDN network. Authors use the Learning Bridge Protocol to let Open Flow switches communicate with legacy switches without requiring any changes to the legacy switches. SDN applications may dynamically discover routing pathways based on pre-defined QoS criteria and current network conditions by using SDN features.

Alouche et al. [134] present a geographical routing protocol for automotive networks based on hSDN and a clustering technique. It considers three distinct characteristics when deciding which relay to transfer the data: 1- contact length among vehicles, 2- the load of each vehicle, and 3- a communication fault log integrated with each cluster head. The simulation finding reveals that the suggested approach performs well regarding throughput, packet top rate, and average routing overhead.

Hui Yang et al. [135] present a novel SUDOI architecture in ubiquitous data center optical interconnection to meet the QoS requirement of extensive user access. Their work performance is verified on an OaaS (Optimization as a Service) testbed for data center services. Their result shows that SUDOI can effectively utilize optical networks and application resources in ubiquitous data center optical interconnections without increasing the probability of blocking.

2.5.1. QoS-Aware Routing

Some applications require different QoS metrics, such as packet loss rate, latency, bandwidth, etc. To overcome this issue, we may create an SLA for each application to record QoS criteria and assign routing paths accordingly to match QIS (Quality Information System) needs, referred to as QoS-aware Routing. This routing uses GA (Genetic algorithm) to determine the best suitable route, which meets QoS requirements incrementally. Assessment results demonstrated that this algorithm performs well in networks with different QoS needs.

In Data Centers, several big data applications must send vast volumes of data from one place to another; thus, how these applications flow throughout the network to enhance network usage is critical. The authors of Hedera [136] presented an SA-based algorithm to route flows to alternative pathways based on the current network status. This study outperforms the typical ECMP in terms of performance. The inventors of Long [137] suggested a dynamic rerouting method. When the network controller detects traffic load, it utilizes a multi-hop or a single-hop algorithm to shift traffic to other paths. This method can enhance connection utilization compared to the conventional Round Robin Algorithm.

The inventors of LARAC [138] suggested a LARAC QoS-aware routing algorithm that finds near-optimum pathways in the Delay Constrained Least Cost issue using a Lagrange relaxation-based aggregated cost technique. Authors in MINA [139] suggested a QoS-aware flow scheduling technique for IoT devices; this approach uses GA (Genetic algorithm) to repeatedly generate the best-suited path to fulfil QoS criteria. Assessment results demonstrated that this method performs well in networks with different QoS requirements. Only MINA addressed various restrictions in comparable works. MINA's key challenges include static weight in the cost functions, QoS-aware Routing, and being locked in a local optimum. To address these issues, [114] introduced a QoS-aware routing (SAQR) technique based on the simulated annealing (SA) technique to identify the best-suited path that fulfils various QoS standards.

2.5.2. Interworking and inter-domain routing

The most well-known applications in the early stages of SDN are Google B4 network [5] and Microsoft WAN [4]. Google chose SDN to convert the linked WAN across data centers, and it is now wholly deployed using the OpenFlow Network. Before fully transitioning to the SDN network, the Google B4 network went to a Hybrid deployment in two different phases; the first was completed in 2010, and the second was completed in 2011 when Google started to administer the network through the Controller and doubled the network capacity. Google began to promote agile development by running both traditional and SDN routing systems in parallel. Hence, SDN routing systems have much higher priority than the classic ones. SDN is deployed incrementally to different Data Centers to allow additional traffic from the old routing system to the SDN framework. If there is any issue with the SDN, B4 provides the option to turn off the SDN framework and return to conventional routing strategies. During this stage, SDN devices can interfere with legacy routing protocols, and Google deployed consistent routing protocols for SDN applications. In any case, SDN networks and traditional Networks must communicate with each other even if a complete SDN architecture is ultimately deployed. Data centers must exchange data with traditional extermination networks.

An Internet Exchange Point (IXP) is a physical access point of the network in which several ISPs may join their networks and exchange BGP (Border Gateway Protocol) routes. SDX-L3 stands for Software-defined Internet Exchange Point Layer 3 [125]. IP prefixes are used in traditional and physical IXP routing and traffic forwarding. SDX enables those to match many header fields; each AS can use remote traffic control. Aside from SDX, the abstraction of virtual switches means that AS cannot influence or overserve Inter-Domain Routing outside their scope. Some other optimization models update rules when BGP routes or policies change. SDX improves the dependability and flexibility of IXPs.

2.5.3. QoS Policy Management

SLA or Service Level Agreement is essential to create QoS criteria for traffic management of different QoS applications like video conferencing, video streaming, and online interactive gaming. Each SLA comprises a set of Objectives to generate network-level policies translated into device-level primitives (e.g., packet dropping rate, queue configurations, traffic shaping policies, and forwarding rules). Managing QoS settings in conventional network designs, such as MPLS and DiffServ, is complex. Conversely, SDN provides a worldwide network view and has robust northbound API capability. It enables network operators to launch services quickly and execute various network regulations. The PolicyCop [140] project intends to provide controllable, adaptable, and vendor-independent administration of QoS rules in SDN/OpenFlow networks using the Northbound SDN API. PolicyCop benefits from OpenFlow features, making it a suitable management framework. It offers on-demand aggregation as well as per-flow control. Policy Cop makes traffic definition easier than MPLS and DiffServ since it does not need a network device shutdown. Because of its vendor independence, it is simple to implement in a network and lowers operational overhead. The Policy Cop design has three planes: management, control, and data plane. The data and control planes follow the conventional SDN architecture, while the management plane serves as the core of Policy Cop. This management plane consists of two main components: the policy validator and the policy enforcer. The policy validator detects and monitors any policy violations, while the enforcer takes action in response to these breaches and ensures compliance with network policy rules.

2.5.4. QoS signalling overhead

Due to the logically centralized nature of the SDN design, all overhead (QoS-related signalling messages) is gathered at the network's control mechanism (Controller). [141] OpenFlow allows network operators to collect information at various flow levels, such as aggregated and individual flows. Each of these methods of collection has a cost. The per-flow process provides better granularity in terms of QoS-related states. It has a scalability problem. A controller can pool switches in an SDN or OpenFlow network to collect data on active flows. When a flow timeout occurs, it might direct the switch to report flow statistics at a predetermined interval. Another critical consideration is how frequently QoS information should be transmitted from one network device to the Controller. Although getting statistics from the data plan often benefits the Controller in keeping an up-to-date global picture of network conditions, this procedure is a trade-off between measurement precision, speed, signalling overhead, and timeliness, and results in the scalability issue of a control plane for a controller [142]. For flow statistics, the PayLess [143] technology supports many degrees of flow aggregation through a RESTful API. It uses an adaptive statistics-gathering technique to provide accurate real-time information while incurring minimum network costs. This approach achieves accuracy extremely near that of the constant periodic polling method while reducing massaging overhead by up to 50% compared to the regular polling strategy.

CHAPTER-3

MIGRATION SEQUENCE TO A HYBRID SDN

Traditional network infrastructures offer numerous features but exhibit limitations, particularly in providing advanced network control for implementing novel concepts. SDN network addresses these limitations by separating control and data layers, offering enhanced agility, flexibility, and advanced traffic engineering capabilities. However, transitioning from traditional networks to a fully SDN-based network can be challenging due to financial and operational constraints.

A practical approach involves adopting a hybrid SDN model, wherein conventional and SDN components are integrated seamlessly. In modern networking infrastructure, integrating SDN ideas with conventional networking methods has given rise to Hybrid SDN; this type of network leverages the benefits of both conventional and SDN networks, the flexibility and programmability of SDN, with the familiarity and stability of conventional network architectures.

This research explores the incremental deployment of SDN nodes using efficient greedy algorithms to enhance existing network infrastructure. We investigated the optimal migration sequence from a conventional network to a hybrid SDN to address the optimization impact on controllable traffic. We aimed to minimize network disruptions and optimize Maximum Link Utilization while incrementally migrating towards a fully SDN-enabled environment. We evaluate the optimal deployment of SDN nodes through simulations, considering network topologies, available resources, and traffic patterns. Our simulation analysis shows that deploying SDN nodes at a rate of 30% can control over 70% of network traffic, leveraging the benefits of a hybrid SDN and achieving near-optimal network performance.

3.1. Introduction

Despite the numerous advantages of traditional legacy networks, certain limitations persist, particularly in providing optimal network control. This deficiency poses challenges for implementing innovative concepts such as network virtualization and traffic engineering optimization. Network management remains one of the primary challenges in contemporary networks, with academia and industry experiencing difficulties in this domain. SDN network was developed to address these challenges by decoupling the control and data planes, and has emerged as a prominent topic in telecommunications.

The SDN network is a critical network design that has emerged due to technology advancements such as virtualization, the Internet of Things, and mobility, which facilitate reliable and flexible network management. Through centralized network management and a global perspective, SDN enables intelligent flow scheduling concepts to enhance overall network performance and link utilization while facilitating advanced Traffic Engineering (TE).

Traffic engineering is essential for network administrators; it improves efficiency and balances network load through traffic routing strategies. TE is an effective method for balancing network flows, mitigating network congestion, and enhancing overall network performance. In recent years, academia and industry have demonstrated significant interest in improving and optimizing TE; however, due to network complexity, research on traffic engineering optimization remains an active area of investigation. According to TE principles, the established SDN design provides a practical and adaptable approach for optimizing traffic routing. By implementing policies or routing algorithms in the control plane, an SDN network can dynamically control the distribution of traffic to any outgoing link; this allows SDN switches to distribute traffic across multiple paths flexibly and unrestrictedly. Customizing the forwarding rules for each flow in SDN can enhance their intuitiveness, flexibility, and intelligence [6].

Despite the significant advantages and numerous motivations, directly transitioning to a fully SDN-enabled network presents considerable challenges, such as financial, technical, security, and upgrading issues; the process can be risky and costly, particularly for large-scale networks [3]. To address these challenges, a viable approach is the incremental deployment of SDN nodes and the implementation of a hybrid SDN concept, which enables the coexistence and interoperability of conventional IP network devices and SDN nodes within a single environment. The literature presents various definitions of hybrid SDN; it can be conceptualized as a logical stage in transitioning from a traditional legacy network to an SDN environment.

A hybrid SDN integrates the programmability and flexibility of SDN with the stability and familiarity of traditional network architecture [28]. It represents a network architecture wherein centralized and decentralized networks communicate and coexist to capitalize on the advantages of both paradigms. This method allows the organization to migrate incrementally to a hybrid SDN and then to a fully SDN-enabled environment. In recent years, the incremental deployment of SDN nodes has garnered significant global attention. One of the most popular goals for incremental SDN node deployment alongside conventional networks is improving network performance from the Traffic engineering perspective. Previous research [22–24] has indicated that programmable traffic (involving at least one SDN node) can enhance flexibility in performing Traffic Engineering (TE). In a hybrid SDN environment, efficient load balancing remains a significant challenge, particularly when SDN nodes are incrementally deployed. Existing methods fail to adjust dynamically to changing network conditions, leading to poor performance and inefficient resource utilization. In the context of a hybrid SDN, maintaining network load balance refers to the equitable distribution of traffic across all network paths to ensure efficient utilization of network resources and minimize congestion on individual links. This balance helps prevent scenarios where some links exceed their capacity (overloaded), leading to congestion and inefficient traffic handling, while other links remain unutilized or underutilized. Load balancing contributes to consistent network performance, minimized latency, and improved reliability.

Previous studies primarily focused on heuristic algorithms, which could not guarantee theoretical network performance. Our simulation experiments using real network topologies revealed that the heuristic algorithms do not behave appropriately for the load balance. Considering these limitations, an efficient and straightforward technique is required to address the incremental SDN node deployment problem while maintaining network load balance.

3.2. Proposed approach

We studied the incremental deployment of SDN nodes and addressed the problem of how to deploy SDN nodes incrementally into a pure conventional network step by step. The objective of the incremental deployment problem is to maximize controllable traffic, which the SDN controller controls. The traffic fluctuates over time in the real network, leading to varying update lists. Short-term traffic spikes can be unpredictable, while long-term traffic patterns are often consistent [144]. Based on this observation, we want to identify and use stable traffic patterns to address the incremental SDN node deployment problem.

In this chapter, we have examined the incremental deployment of SDN nodes to improve the efficiency of optimization. The problem is determining the best SDN migration sequence for migrating legacy devices to SDN-enabled. Each migration phase aims to maximize controllable traffic while minimizing MLU; the target should be updated carefully while considering current and previous deployment ratios. In each deployment phase, a new network topology will be generated. The selection of a controller in the hybrid SDN environment is another critical consideration, given the presence of multiple controllers. This research utilized the ONOS controller in the hybrid SDN environment. The proposed approach is compared and analyzed across various real network topologies; simulation analysis demonstrates that with 30% of SDN node deployment, we can achieve nearly equivalent performance to a fully SDN environment.

In contrast to the previously mentioned studies, this research directly addresses the complexity of the incremental deployment problem for load balancing in hybrid SDN through complexity and inapproximability analysis and develops a tight approximation algorithm to resolve it. We have examined the incremental deployment of SDN nodes to improve the efficiency of optimization. The problem is determining the best migration sequence for migrating legacy devices to SDN-enabled. Each migration phase aims to maximize controllable traffic while minimizing MLU and considering current and previous deployment ratios. This study provides a comprehensive solution that advances the field and offers valuable insights for future research and application in incremental deployment scenarios by emphasizing theoretical and practical implications. The investigation explores the incremental deployment of SDN nodes using simple greedy algorithms to integrate SDN capabilities into existing network infrastructure more efficiently. The conventional network environment is constructed using the Mininet and MiniEdit tools, followed by the incremental deployment of SDN nodes to create hybrid SDN environments. Multiple controllers were studied and evaluated, and finally, the ONOS controller was selected as the SDN controller for the experiment.

3.3. Hybrid SDN Network

There are different approaches for upgrading a conventional network to an SDN-capable network. First, vendors can install additional software modules on conventional devices to make them programmable. The second method is accessing edge control through virtualization, which requires installing and upgrading new networking software on all end devices in the network. The third method is to replace all the conventional devices in the SDN network fully and create SDN islands. Lastly, the optimal method is to incrementally deploy SDN nodes alongside conventional networks to take advantage of both while creating a hybrid SDN environment.

A hybrid SDN is a networking architecture comprising both SDN and conventional network devices. This type of network interacts in varying degrees to maximize user experience and performance, which regulates, modifies, configures, and manages network characteristics. This type of networking allows both SDN and conventional network devices to function together, as shown in Fig.1, the scenario in which both network devices function together. In this network design, R1-R5 present traditional legacy routers that route the traffic based on the shortest path from source to destination, and S1 presents an SDN node. One controller is there, which manages the SDN switch.

In recent years, hybrid SDN has become more prevalent in network design, and both businesses and academics have been interested in studying traffic engineering using hybrid SDN [25]. One of the most commonly used protocols in legacy networks for traffic routing is OSPF (Open Shortest Path First), which assigns a cost or weight to each link. In contrast, SDN nodes work in hybrid mode, supporting both the OSPF protocol to communicate with legacy devices and the OpenFlow protocol to communicate with SDN controllers and nodes.

In the hybrid SDN context, only traffic passing through the SDN node can be managed by the SDN controller, while the rest packets will be forwarded along the shortest path. The flow in this study refers to traffic identifiable by a source-destination pair, which combines the source and destination addresses. The controller splits the flows, and the sub-flows can reach the same destination through different paths, which avoids traffic congestion and can balance the link load; in some cases, there are some constraints when splitting the flows, such as unsplittable flows [145-146]; we ignored these constraints because the problem remains hard to find approximations when removing the constraints, and we can get the optimal traffic distribution as a reference for the incremental deployment of SDN nodes in the removed version. The splitting could result in packet reordering issues, a common issue that can be solved through different techniques [147-149].

In the hybrid SDN context, multiple paths distribute traffic across SDN nodes during the migration process; this can lead to packets belonging to the same flow following different paths and arriving out of order. Some approaches can handle out-of-order packets. The first method for handling out-of-order packets is buffering; we can implement a packet buffering mechanism. In this method, each packet is assigned a number, and out-of-order packets are temporarily stored in a buffer; once all the packets of a flow arrive, they are reordered before being sent to the destination node. Another method is sequence numbering: in this method, again, each packet will be assigned a sequence number in the flow header; this method allows the receiver node to reorder any out-of-sequence packets when they arrive through multiple paths during the migration sequence. Another method is the congestion control method: in this method, we can use a load-balancing approach that ensures packets from the same flow take the same path during the migration process; this method minimizes the chances of out-of-order delivery. The last method is the delay compensation method: in this method, when multiple paths are used, the system can use a delay compensation technique that temporarily holds packets from faster paths to synchronize their delivery to prevent out-of-order arrival. We can use a mix of these techniques by buffering and rerouting during the migration process to handle out-of-order packets when the flow is transferred across multiple paths. This method ensures that even if packets are routed across different paths during the migration process, they are delivered in the correct order, minimizing disruption to the flow.

Figure 3.1. shows the hybrid IP/SDN network architecture, as illustrated in the Figure, R1-R5 are conventional legacy routers that route traffic based on the shortest path from source to destination, while S1 represents an SDN node controlled by an SDN controller. In a hybrid SDN architecture, the controller manages particular network components, such as virtualized environments or the data center; conversely, conventional networking protocols will control other network segments, such as legacy systems or access layers. Conventional legacy devices provide reliable routing, whereas SDN nodes concentrate on network optimization. In contrast, SDN controllers route network traffic from a worldwide perspective.

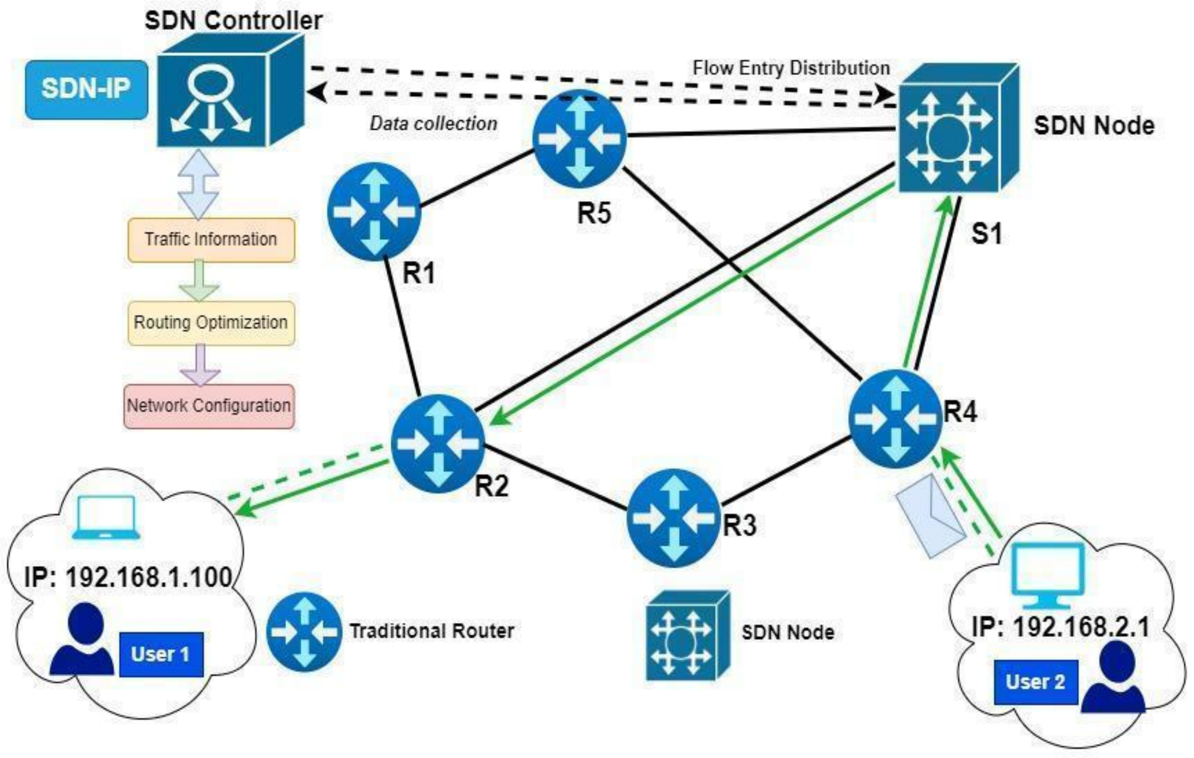


FIGURE 3.1. Hybrid IP/SDN network architecture

The essential component in the control layer is the SDN controller, which controls all data plane operations; the capabilities and performance of the controller are crucial in achieving optimal network performance. As multiple SDN controllers exist, selecting the appropriate controller based on the network topology is the key to achieving optimal network performance. We have studied and examined different controllers for the hybrid SDN environment, as multiple controllers exist. Notable controllers are NOX, Floodlight, POX, ODL, Ryu, and ONOS. Finally, we have selected the ONOS (Open Network Operating System) as the SDN controller for our hybrid SDN environment; it is compatible with the hybrid SDN and supports the SDN-IP application. SDN-IP is a program that integrates legacy and SDN networks; it runs on top of the ONOS controller and allows SDN nodes to integrate and communicate with the legacy routers through BGP (Border Gateway Protocol) ASes (Autonomous Systems).

Table 3.1 Parameter Definition

Variable	Explanation
G = (V, E)	Undirected graph
V	A set of all network nodes
E	A set of links in the network
C (e)	link capacity of the edge e
L	legacy node set
Vs	SDN node set
Vc	a subset of the migrated SDN nodes.
D=(Di)	Set of Traffic Matrices
ri	Weight coefficient of Di
Uί	The value of MLU under Di
T	Total periods
ω	OSPF weight configuration
W_t	Traffic weight
τ	controllable flow

3.3.1. Migrating from a conventional network to a hybrid SDN

As we will discuss the incremental deployment of SDN nodes to the conventional network later, it is necessary to consider the challenges mentioned in that section. Determining the best migration sequence is challenging; selecting the precise number of SDN nodes to deploy and their appropriate placement can significantly influence the available routes for controllable data flows. The migration sequence's main objective is to optimize controllable or programmable traffic, which the SDN controller controls. Traffic volumes fluctuate in the real network over time, potentially resulting in updated versions with different priorities at different times. Once we upgrade a conventional device to an SDN node in a real network deployment scenario, it does not revert to its initial configuration.

During the migration of SDN nodes, one key challenge is managing the arrival of new flows during the migration process. To address this issue, there are different techniques, such as flow buffering, rerouting new flows until the deployment completion, and flow scheduling through different paths. We provide a technique for adaptive flow management in which the new flows reaching during the migration process are rerouted to available SDN nodes, not under the migration process. This method ensures uninterrupted services for incoming traffic while the migration process continues. Once the new SDN node is fully integrated, these flows are transitioned to the appropriate nodes to ensure balanced load distribution. In some cases where flow rerouting is not feasible, the flows are buffered temporarily and processed once the migration process is completed.

Long-term observations often indicate relatively steady traffic flows, in contrast to short-term observations that may reveal surprising and unexpected surges in traffic [144]. In light of this discovery, we aim to investigate a traffic pattern that could depict steady traffic characteristics to resolve the SDN migration sequence issue. We can manage unexpected traffic surges by implementing a traffic control plan. Because of this, the traffic control technique can handle unpredictable traffic changes, whereas the deployment strategy can handle most flow distributions.

Considering those challenges, we provide the variables needed to solve this problem and find the best migration sequence. First, we relate the hybrid SDN network model, formulate the problem as an optimization problem, and analyze the problem's complexity.

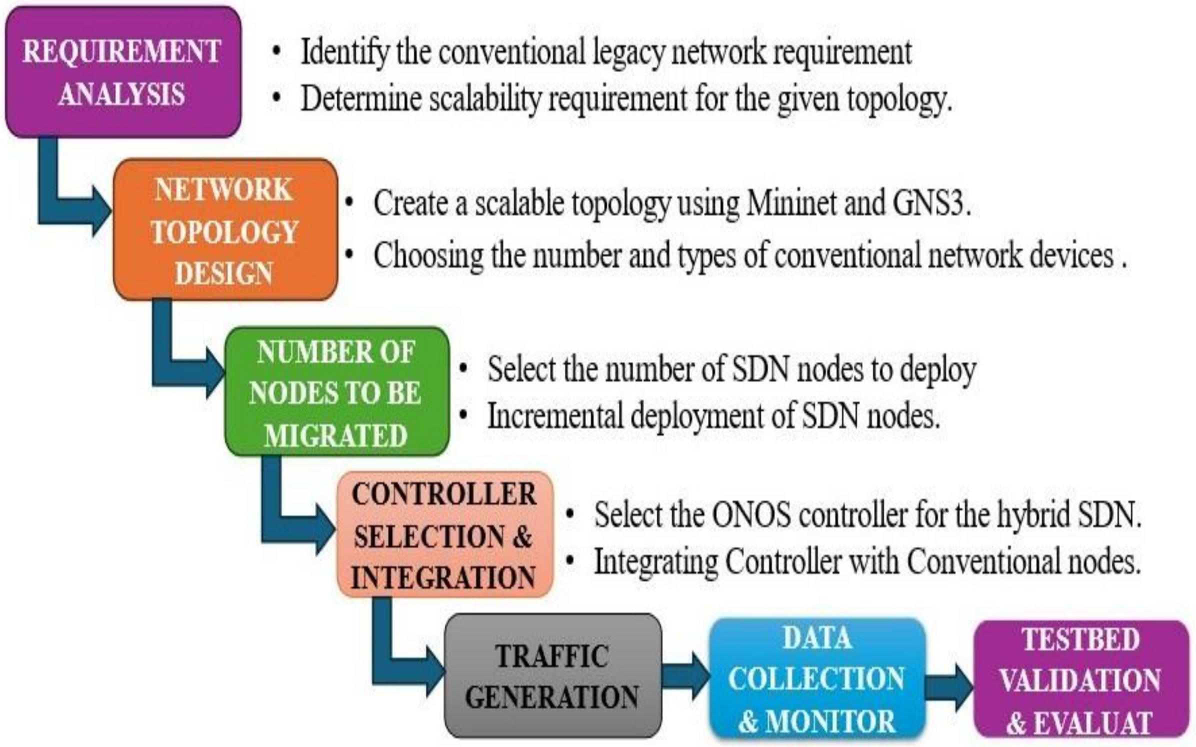

REQUIREMENT ANALYSIS
• Identify the conventional legacy network requirement
• Determine scalability requirement for the given topology.
NETWORK TOPOLOGY DESIGN
• Create a scalable topology using Mininet and GNS3.
• Choosing the number and types of conventional network devices .
NUMBER OF NODES TO BE MIGRATED
• Select the number of SDN nodes to deploy
• Incremental deployment of SDN nodes.
CONTROLLER SELECTION & INTEGRATION
• Select the ONOS controller for the hybrid SDN.
• Integrating Controller with Conventional nodes.
TRAFFIC GENERATION
DATA COLLECTION & MONITOR
TESTBED VALIDATION & EVALUAT

FIGURE 3.2. Workflow of migration sequence

3.4. Network model and problem formulation

3.4.1. Network Model and Variable Declaration

We modelled the network as a graph G = (V, E), where V represents the set of all network nodes and E represents the set of links connecting devices to the network. Each node can be a conventional network device (Vl) or an SDN device (Vs), where Vl∩VS =∅. For ∀ υ1, υ2 ∈$\boldsymbol{V}$, **υ1** ≠**υ2**, for **υ1** to **υ2**, there should be at least one path to reach. C (e) represents the link capacity of e ∈ E. D represents the set of Traffic Matrices (TMs) where D = (D1, D2, D3..., Dn), and ri represents a positive weight coefficient for each TM Di. Vc represents the subset of migrated

SDN nodes. ω represents the link weight setting of the OSPF protocol for legacy devices, and the variable f represents the traffic splitting ratio of SDN nodes. To better understand, Table 3.1 summarizes all the sets of notations used in this chapter.

In this work, we assume that all the legacy devices use conventional IP routing, which passes the packets according to the shortest path; we configure all the nodes as conventional network devices that implement the OSPF protocol. We want to incrementally upgrade legacy nodes to SDN nodes, wherein the splitting ratios are assumed to be determined by the SDN controller to split the flows to increase controllable traffic that can be efficiently managed. For this experiment, both the network topology and traffic matrix are assumed to remain constant. During each phase, traffic is routed in two distinct ways: traditional devices direct traffic along the shortest path using their routing tables, whereas SDN nodes route traffic based on the optimal splitting ratio calculated at that specific stage. We aim to upgrade most legacy devices to SDN while maintaining the overall network load balance and minimizing MLU. The most common method to keep the overall load balance is to minimize MLU, which we will consider in this chapter.

Definition: Given a directed graph G = (V, E), a traffic demand matrix TM Di, and a specified number of legacy devices, the objective is to minimize the Maximum Link Utilization (MLU) by upgrading the majority of legacy devices to SDN nodes capable of flexible flow splitting.

3.4.2. Problem formulation

We formulate this problem as a mixed integer program based on the given network model and definition. We consider the MLU for each device's migration to measure the performance of migration sequences. The primary goal is to minimize the total MLU after the entire migration process (1)-(7). In the first step, we determine the best migration sequence for the migration of each SDN node. We use an optimization algorithm to analyze different migration processes; this algorithm will calculate the MLU for each migration and, ultimately, will determine and select the nodes with the minimum MLU to improve the overall network performance.

The network performance will improve by minimizing the MLU for each traffic matrix Di; the value of MLU is specified by the traffic splitting ratio of SDN nodes f and the link weight setting of conventional devices ω. As a result, we can formulate the problem as follows:

$$\phi = \text{minimize} \sum_{s=1}^{N} \mathrm{Ui}_{\{\upsilon 1, \upsilon_2, \ldots\ldots \upsilon s\}} \tag{1}$$

$$\sum_{e \in In(c)} f_{e,D_i}^{t}(\omega) + Di(c,t) = \sum_{e \in Out(c)} f_{e,D_i}^{t}(\omega) \quad c \in Vs, t \in V, i \in [1, n] \tag{2}$$

$$\sum_{i=1}^{n} r_i = 1, \quad 0 \leq r_i \leq 1 \tag{3}$$

$$\omega(e) \in N^{+}, \quad \forall e \in E \tag{4}$$

$$0 \leq \mathrm{Ui}_{\{\upsilon_1, \upsilon_2, \ldots\ldots \upsilon s\}} \leq 1, \forall D_i \in D, \quad i \in [1,n] \tag{5}$$

$$f_{e,D_i}^{t}(\omega) \geq 0, \ \forall e \in E, \forall t \in V, \forall D_i \in D, i \in [1, n] \tag{6}$$

$$\sum_{t \in V} f_{e,D_i}^{t}(\omega) \leq \mathrm{Ui}_{\{\upsilon_1, \upsilon_2, \ldots\ldots \upsilon s\}} C(e), \forall e \in E, \forall D_i \in D, i \in [1, n] \tag{7}$$

To determine the best migration sequence, the primary objective is to minimize the total Maximum Link Utilization after the entire migration process (1)-(7); the aim is to identify the best migration sequence S = $(\upsilon_1, \upsilon_2, \ldots \ldots \upsilon s)$ with a minimum total of MLU. Equation (1) represents the flow conservation mechanism in the network at each node v, indicating that the total flow of a device, combined with the flows it generates (traffic demand), should equal the flows that exit this device, which is flow conservation; this equation implies that the traffic demands should be met equivalently. Equation (2) stipulates that the weight ri assigned to each traffic matrix should be an integer within the range [0,1]; link weights are frequently utilized in routing algorithms to define the cost or preference of a particular link. Ensuring these weights are integers facilitates the routing calculations for specific network protocols requiring integer weights. Equation (3) mandates that the link weights are positive integers, essential for practical networks. For proper network operation, the link weights must be non-negative. Equation (4) constrains the MLU values between 0 and 1, which is realistic, where one signifies full link utilization, while 0 represents no utilization. Equation (5) ensures that all traffic flows on edge e are non-negative; negative flows are not physically meaningful in network flow problems. This constraint guarantees that all flow values are realistic and feasible within the network. The constraints ensure proper flow conservation, respect link capacities, and enforce non-negative traffic flows while meeting traffic demands and integer weight requirements. Equation (6) represents the primary objective of the optimization, aiming to minimize where represents the weighted sum of MLU over the entire migration process. Equation (7) represents the capacity constraint, which imposes a limitation on link capacity; this equation ensures that the total flow on any edge node should not exceed its capacity, and the flow on each device should remain within its capacity limits, preventing overload that could result in failures or congestion. This optimization problem balances network traffic across multiple traffic matrices, ensuring efficient utilization of network resources while adhering to network constraints.

3.4.3. Problem Complexity

Several variables are involved in determining the best migration sequence, including the number of devices to upgrade and identifying the most suitable location for the upgrade, SDN node splitting ratio, and weight settings, which results in a complex set of optimization problems. Realizing that the weight setting is necessary to evaluate each combination of splitting ratios. With a fixed weight setting, this problem can translate to a multi-commodity flow problem that can be solved in polynomial time [150]; however, the weight setting is undetermined, and it is integers that add up more complexity to the problem, so it has been proved that in this case optimization of weight setting is an NP-complete problem [151]. In this problem, all the traffic is directed to the shortest path, while none is sent to the longest route. Therefore, it is simple to demonstrate that the total complexity of the problem is NP-complete.

Table 3.2 Comparison of SDN Controller

Controller	**API/Language**	**Traffic Monitoring**	**Scalability in the Environment**	**Topology &adaptability**	**Simulation tool**	**Custom traffic framework**
ONOS	Java APIs-High	Native support via OF	High (Mininet)	Dynamic Topology reaction	Excellent	Excellent
RYU	Python API-Moderate	Plugin-based Support	High	Good but static adaptation	Good	Good
ODL	Java-Complex	Limited native traffic view	Very high	Limited dynamic	Good	Fair, not suited for prototypes
POX	python-legacy	Very limited	Low	Poor adaptability	Good	Very limited capability
NOX	C++-obsolete	Very limited	Low – unsupported	No real adaptability	No toolchain	Not applicable
Floodlight	Java-lightweight	Limited	Moderate	Weak	Basic Mininet support	Limited

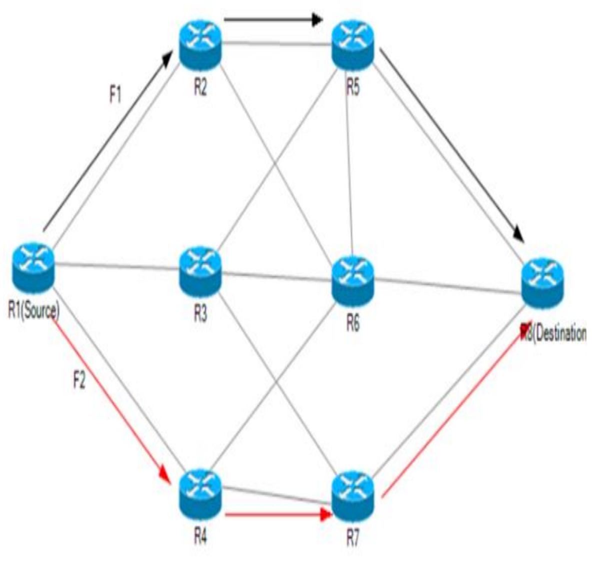
3.4.4. Incremental SDN deployments

While SDN is an attractive networking platform, it has made tremendous progress in recent years; however, with technical and Significant financial challenges, many network providers hesitate to migrate directly to a fully SDN network. Instead, they are considering incrementally deploying SDN nodes to a hybrid SDN to improve network performance from a traffic engineering perspective. Knowing the importance of SDN node deployment in the conventional legacy network is essential while considering optimal network performance in a hybrid SDN. Identifying the number of devices to migrate and selecting the appropriate node for the upgrade is necessary to achieve optimal network performance from the perspective of TE in the hybrid SDN.

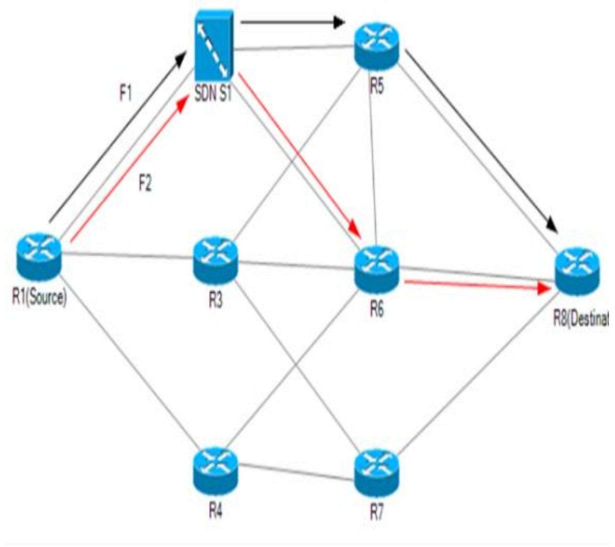
While deploying SDN nodes in the conventional network, some key challenges must be considered to achieve the ideal TE performance in a hybrid SDN. Defining the number of SDN nodes to deploy, or the deployment rate is the first challenge because it balances network performance and deployment costs. The second challenge entails identifying and selecting the most suitable node for the upgrade, considering the balance of each source-to-destination combination to maximize TE performance. The third challenge is choosing the node with the maximum number of routes for passing the flows through the SDN nodes. Figure 3.3 describes how these challenges substantially impact TE performance in the hybrid SDN environment. In the given topology, there are two different flows (F1 and F2) from source R1 to destination R8, with traffic rates of 300 Mbps, as depicted in Figure 3.3. We assumed the links had an identical capacity of 256 Mbps.

Figure 3.3 (a) shows the pure conventional network with the shortest path routing using the OSPF protocol for F1 and F2, with 75% MLU. To minimize the MLU in our topology and improve TE performance, we will incrementally upgrade this network to the hybrid SDN by deploying SDN nodes. Figure 3.3(b) illustrates that initially, we tried to upgrade only one device to the SDN node; we selected the R2 to be upgraded to S1. Following this upgrade, we rerouted both flows (F1, F2) through S1 to implement waypoint enforcement. As we can see, S1 has only two outgoing links that are not able to use all of the forwarding entry capacity; we see the MLU is still 75%, so compared to the first conventional network, there is not that much difference, as we still have not minimized the network's MLU. For the second time, as shown in Figure 3.3 (c), we tried to change the location of the migrated SDN nodes. This time, we attempted to upgrade R3 to the SDN nodes instead of R2, and again, we directed the flows (F1, F2) through S2, as S2 cannot split the flows to achieve the necessary load balance. We observed no difference in minimizing MLU, as it remains the same.

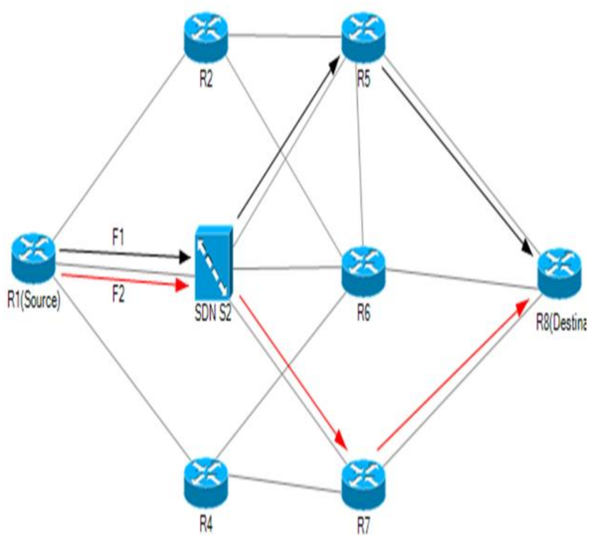
There are now two methods to enhance the performance of TEs: the first is to increase the total number of SDN nodes, and the second is to eliminate the limit on the number of forwarding elements in SDN nodes. Figure 3.3 (d) illustrates that increasing forwarding entries to 4 minimizes the MLU to 61.66%. Lastly, when we raised the number of SDN nodes to 2, as shown in Figure 3.3 (e), we upgraded R2 and R3, and we removed the limit on the number of forwarding entries; we made F1 and F2 pick S1 and S2 as each other's destinations, which is an ideal selection method. As a result, the MLU drops to 48.34%, showing the importance of considering the challenges listed above when deploying the SDN nodes into a traditional network architecture to get the most out of hybrid SDN.



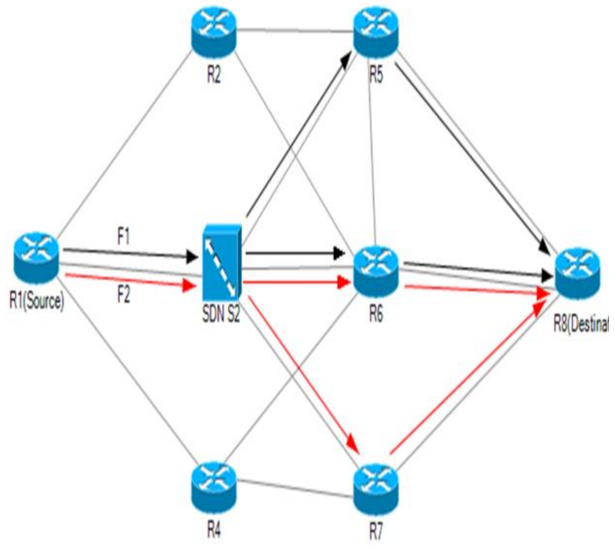
(a) Pure conventional network using shortest path routing, with MLU of 75%



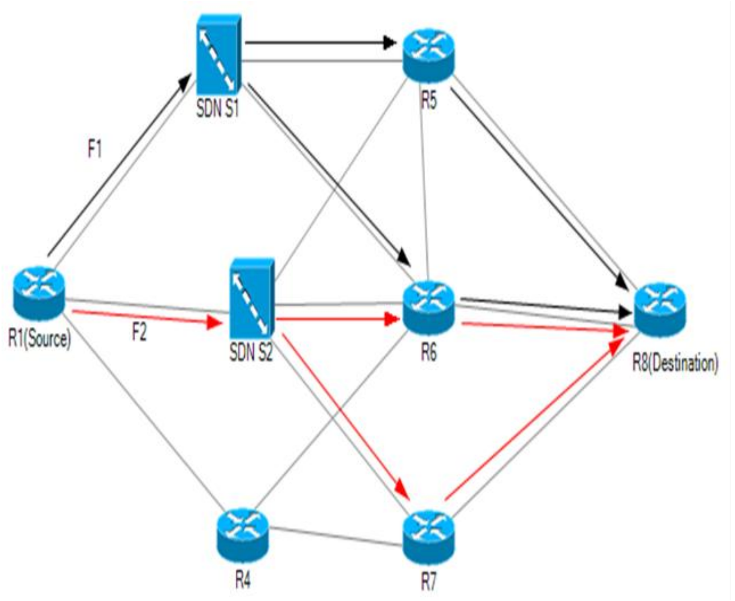
(b) Incrementally upgraded R2 to SDN; now, it is a hybrid SDN with an MLU of 75%.



(c) Changed the location of the SDN node. This time, instead of R2, we upgraded R3 to SDN, and the MLU is still 75%.



(d) This time, removed the limit on the number of forwarding elements in SDN devices. As there are multiple waypoints, MLU has been minimized to 61.67%.



(e) This time, considering all the challenges, increasing the number of SDN nodes, and removing the limit on the number of SDN nodes forwarding entries, the MLU has been minimized to 48.34%.

FIGURE 3.3. An example of SDN node deployment

3.5. Migration to the Hybrid SDN algorithm

This section describes the algorithm used to determine the best migration sequence. We introduce a simple greedy algorithm to identify an ideal migration sequence from a conventional network to a hybrid SDN and achieve manageable traffic flows in a hybrid SDN environment.

Finding the best migration sequence is crucial to a successful SDN transition. As discussed in the incremental section, different locations of SDN nodes can yield different results when minimizing MLU. The migration problem aims to increase the amount of traffic the SDN controller controls. We provide a detailed presentation of the algorithm and analyze its complexity.

Algorithm I: Greedy Algorithm

Algorithm I: incremental deployment of SDN nodes

```
        Input: $G = (V, E), T, k, traffic weight$
        Output: $X$
1       Algorithm initialization
2       $L \leftarrow V, C \leftarrow \emptyset, X \leftarrow \emptyset$
3       Compute initial network metrics (MLU)
4       Iteration process
5       For $i = 1$ to $k$:
6           Set $f_m \leftarrow 0, x_i \leftarrow None$
7           For each node $\alpha \in L$:
```

```
8           Simulate converting α to SDN ($H(\alpha) = 1$)
9           Identify controllable flows $T(\alpha)$
10          Compute $f_\alpha = \sum_{t\in T} \sum_{t\in T(\alpha)} W_t$
11          Evaluate network metrics (MLU)
12          If $f_\alpha > f_m$ and metrics improve:
13              -$f_m \leftarrow f\alpha$ $x_i \leftarrow \alpha$
14          If $x_i$ = None or improvement is negligible, break
15          end if
16      end for
15  Add $x_i$ to X, remove $x_i$ from L, and update network state
16  Return X
```

Algorithm I is a simple greedy algorithm for solving the best migration sequence problem. It might take time to determine the ideal migration order. The precise deployment and placement of SDN nodes can influence the data flow control routes. The migration sequence's primary goal is to maximize programmable or controllable traffic under the SDN controller's direction. Searching for an optimized migration sequence is computationally demanding because the solution space is N! We can solve the problem using the enumeration technique or linear programming optimizers like Gurobi [152] or CPLEX [153], but these methods may consume excessive time.

This algorithm uses a simple greedy approach that selects the node from the legacy network to maximize the amount of controllable traffic in SDN. After completing k iterations, the nodes with the maximum controllable traffic will be selected for the migration to the SDN node in the set X. This method will loop every time to find an optimal node to be upgraded; in each loop, it explores all the nodes in the traditional node-set L, then identifies a node with the most controlled traffic and adds it to the SDN node set Vs. As shown in algorithm 1, it is a simple greedy algorithm, and the complexity of this algorithm is O(N), where N = |V|, which is the number of nodes in the network; as the algorithm runs k iterations, the overall complexity is O (k. N), which is considered efficient for the problem. The selection process persists, yielding an updated list of nodes carefully selected for their capacity to direct and manage network traffic efficiently.

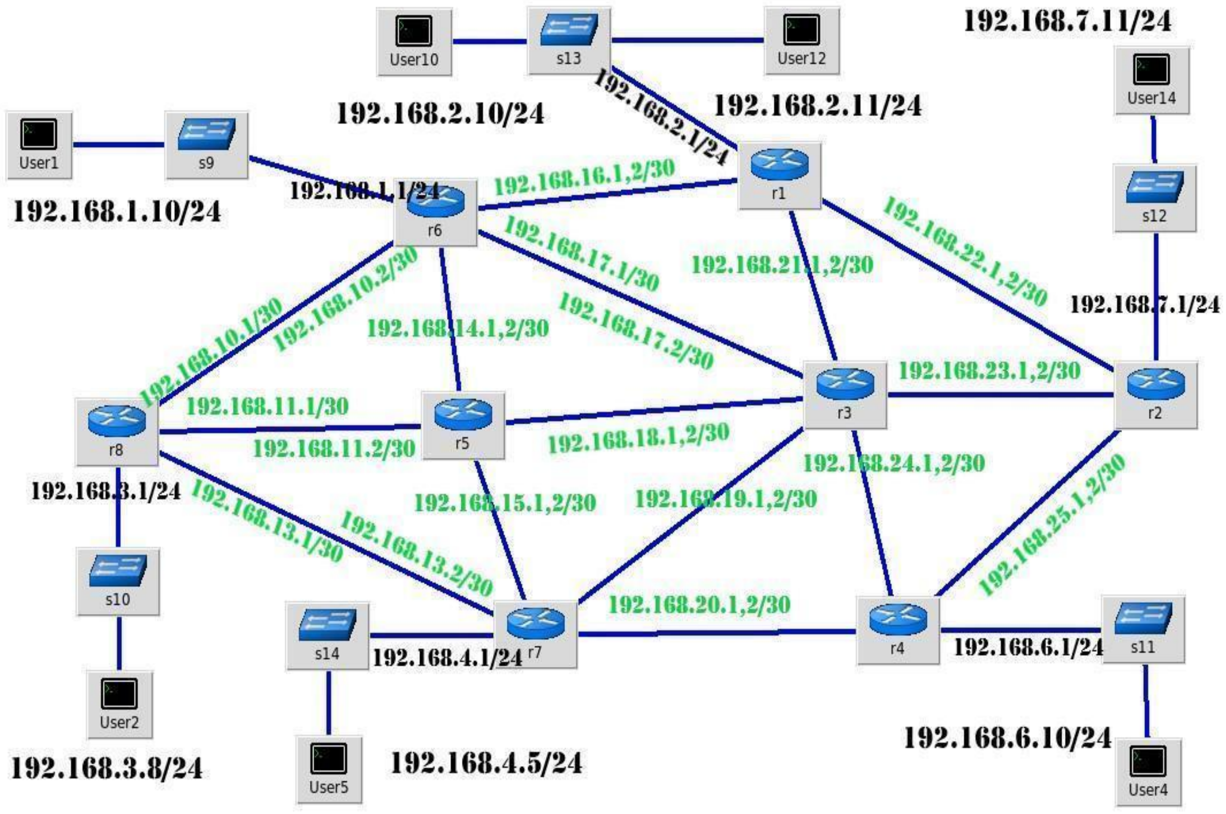


FIGURE 3.4. Traditional network topology

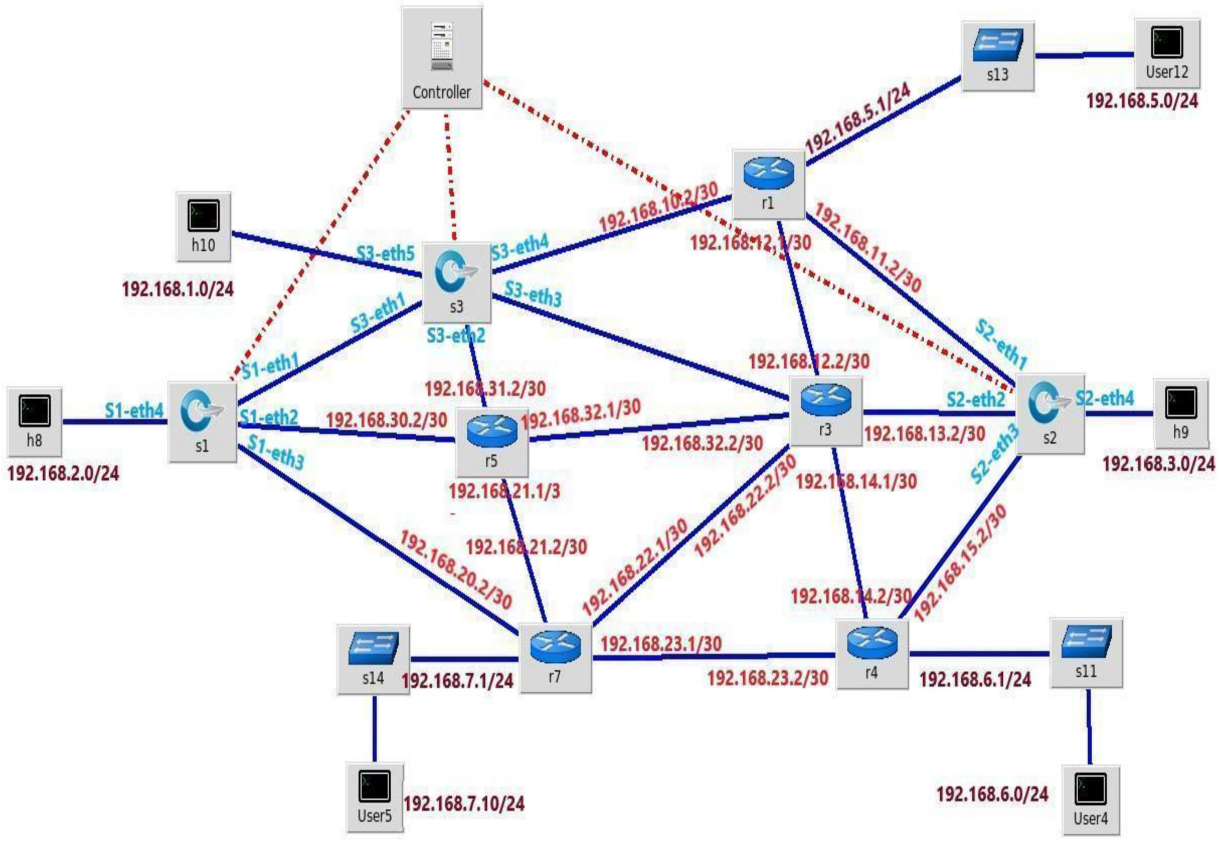


FIGURE 3.5. Incrementally deployed hybrid SDN topology

3.6. Experiments and evaluations

In this section, we present the performance of our proposed method on various network datasets and topologies utilized in our experimental analysis, the simulation of the algorithms delineated in the algorithm section, and an analysis of their performance on diverse real network topologies. Initially, we will examine the environmental parameters, simulation topologies, and traffic datasets. Subsequently, we will present visual representations of the migration sequence algorithm, called the FCM (Flow Control Method), with a controllable traffic weight pattern.

Table 3.3: Network Topologies Information

Network	Nodes	Links	Time	Time-range
Abilene	12	30	5 min	Six months
Cernet	14	32	5 min	One month
Nobel-Germany	17	26	5 min	One day
Geant	23	74	15 min	Four months

Table 3.4: Software specification

Software/Tool	Version	Category	Purpose	Role
Mininet	v2.3.1B4	Network Emulator	Creation of Custom topologies	Emulated a scalable Hybrid SDN network.
ONOS	v2.2.0	SDN Controller	Traffic monitoring and Flow control via OF	Implemented to control traffic flows dynamically.
GNS3	v2.2.43	Legacy network Simulation	Configuration of conventional devices	Creating the conventional network topology
VBOX	v7.1.6	Network Virtualization	Creating multiple OS	Used for the virtualization of the network devices
iPerf	v3.13	Traffic Generator	Performance testing	Simulated various traffic loads
Wireshark	V4.2.1	Packet analyzer	Monitoring and analyzing packet-level data	Verified packet flow and latency during simulations
Ubuntu OS	22.04 LTS	Operating System	Hosting the entire Hybrid SDN	Stable platform for Mininet & ONOS

Table 3.5: Hardware specification

Component	Specification	Purpose
Processor (CPU)	Intel Core I7-12700 @ 2.1 GHz	Supports parallel virtual node simulation and SDN process
RAM	16 GB	Smooth handling of multiple virtual hosts and controller load
Storage	512 GB SSD	For fast boot and R/W of packet capturing/logging
Virtualization Tool	VBox 7.1	Used to test and isolate different topologies in a virtual lab environment.
Network Adapter	Intel Gigabit Ethernet (Virtual NIC)	To enable realistic emulation of traffic flow between nodes.

3.6.1. Environment Setting

The simulation experiments were conducted on a high-performance personal computer with a 3.20GHz Intel Core i7 processor and 16 GB of RAM. The network environments were simulated using the Mininet Simulator [55]. Initially, a traditional network topology was created, followed by the incremental deployment of SDN nodes utilizing the Mininet tool. FRR (Free Range Routing) and VTY Shell were employed to configure legacy routers and network protocols. To establish connections between different networks in the conventional network and to integrate the legacy network with SDN AS (Autonomous Systems), BGP (Border Gateway Protocols) were configured within all legacy routers, as the implementation of a peering system is essential for the functionality of the BGP protocol.

In this simulation experiment, we select the ONOS [56] controller as the SDN controller and use SDN-IP to integrate the legacy network with the SDN network. This program operates on the ONOS controller, allowing SDN nodes to communicate with legacy routers using BGP protocols. To illustrate the capability of our suggested approach, we analyzed our method on different real network topologies, such as Abilene, Cernet, Nobel-Germany, and Geant; these are the network topologies we used in our experiment analysis.

The dataset and network topologies used in this experiment are available publicly on SDNlib [154]. The Abilene network topology includes 12 nodes connected by 30 links. The CERNET topology features 14 nodes and 32 links. The Noble-Germany network comprises 17 nodes with 26 links, while the GEANT topology consists of 23 nodes and 74 links. The traffic datasets required in the experiment analysis are measured or created from the relevant topologies; all network traffic datasets are actual network traffic datasets measured in the designated topologies every five or fifteen minutes. Details regarding topologies and traffic measurements are provided in Table 3.3.

Simulation experiment on real network topologies shows the effectiveness of our method in solving a real-world problem; the performance of our method can be examined in the future on large-scale ISP networks and synthetic topologies with more connectivity. In the initial network design, we built a conventional legacy network using legacy routers, and the traffic was transferred using the OSPF protocol; then, a specific portion of the legacy nodes was selected to be upgraded using various algorithms. To demonstrate the capability of our proposed algorithm on minimizing MLU while considering load balance, we have evaluated it on different real network topologies and compared it to the following methods:

Conventional Network: The conventional legacy network uses legacy devices, in which the OSPF routing protocol is used to transfer traffic among all devices in the network.

DEG [66]: A heuristic approach that prioritizes nodes according to their degree, focusing on the node with the highest degree and the most connections in the network. This strategy seeks to improve the network backbone by upgrading the most critical nodes for network connectivity.

VOL [69]: A heuristic method that takes a distinct approach by concentrating on nodes with the most significant traffic volume. This method enhances high-traffic nodes to boost performance and avoid network congestion inside the SDN framework.

Modified Greedy [62]: a method that prioritizes the migration of the nodes based on the migration cost and highest traffic volume; it balances the cost of migrated high-traffic nodes while minimizing overall migration costs. This method seeks to enhance network performance, ensure efficient upgrades, and optimize resource allocation by considering migration costs.

Local search [62]: A method that uses different approaches; it prioritizes controllable paths or routes across network infrastructure over migration costs or traffic volumes, and its goal is to increase the number of controllable paths in the network. This strategy aims to improve network flexibility and manageability by optimizing controllable paths for effective traffic steering and policy enforcement through the SDN environment.

FCM: Our method uses a simple greedy algorithm that selects the node with the maximum amount of controllable traffic to SDN and aims to minimize MLU while considering network load balance in the hybrid SDN environment. Compared to other alternative approaches, our method performs well in reducing MLU while considering load balance in the hybrid SDN environment. The FCM method, which considers previous deployment ratios, outperforms the VOL and DEG methods and achieves comparable performance to the modified and local search methods, which do not rely on previous deployment ratios.

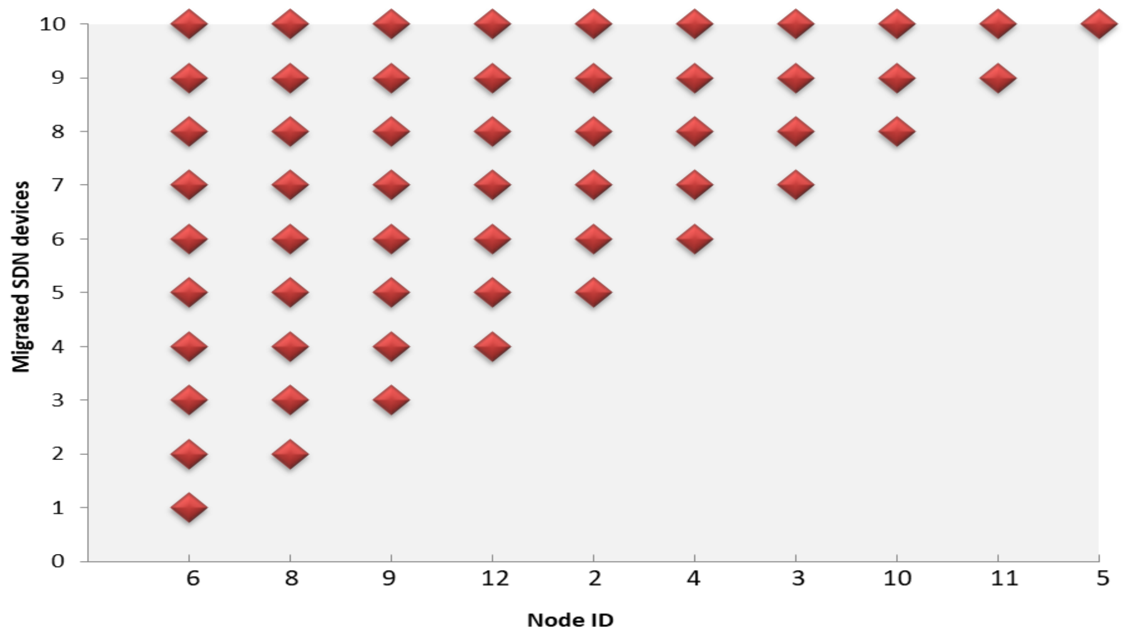


FIGURE 3.6. SDN node upgrades using the FCM method.

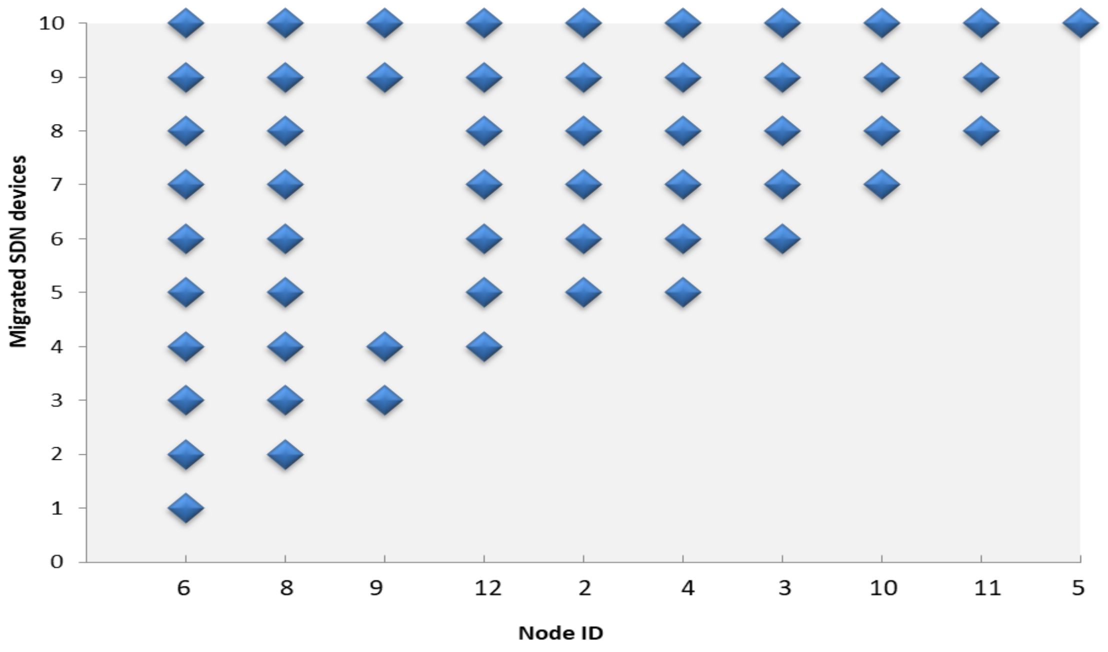


FIGURE 3.7. SDN nodes upgrade using the Local Search Method

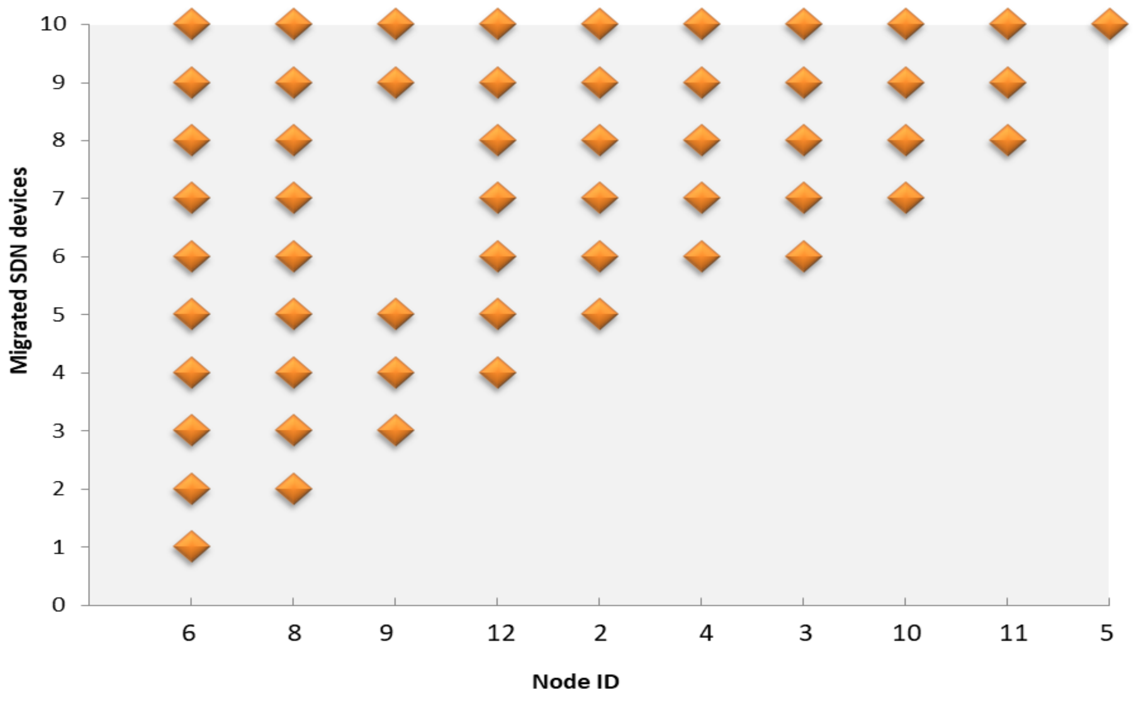


FIGURE 3.8. SDN nodes upgrade using the modified-greedy method.

3.6.2. Results and discussion

As previously discussed, Software-Defined Networking is an innovative design with numerous features that can address traditional networking challenges. It has garnered significant attention globally in recent years. However, transitioning directly to full SDN implementation remains challenging, particularly for large-scale networks with numerous traditional network devices, as upgrading the entire network to SDN presents various obstacles. The optimal approach is to incrementally deploy SDN nodes alongside the conventional networking infrastructure to leverage the capabilities of both networks. In hybrid SDN, it is not necessary to upgrade all network devices to SDN; organizations can incrementally deploy SDN nodes according to their specific requirements and subsequently upgrade the entire network to SDN. The primary objective of incremental deployment of SDN nodes is to enhance overall network performance from a traffic engineering perspective. In operational networks, traffic volumes fluctuate over time, potentially resulting in updated traffic patterns with varying priorities at different intervals.

In this simulation, the Mininet simulator was utilized to establish the network environment; the Mininet simulator is one of the most frequently employed platforms for the simulation of SDN and hybrid SDN environments. Figure 3.4 illustrates the initial network topology, which is a conventional network. Legacy routers and switches were employed and configured with different network addresses; these routers route traffic based on the shortest path from the source to the destination using OSPF protocols. BGP was configured within these legacy routers to connect different network addresses. To enhance network performance and leverage the advantages of the SDN network, it is necessary to migrate to a hybrid SDN network to deploy SDN nodes alongside these legacy routers incrementally, thus benefiting from both SDN and legacy networks. Three SDN nodes were incrementally deployed to the network topologies, as shown in Figure 3.5. R8 was upgraded to S1, R2 to S2, and R6 to S3 as SDN nodes; a controller was added to manage the SDN nodes. To configure and integrate these devices, FRR routing using the VTY shell was employed to configure the network protocols. BGP was configured to incorporate the conventional network into the SDN-ASes.

In the hybrid SDN, it is necessary to select a controller for managing the SDN nodes; the capabilities and performance of the controller are crucial in achieving optimal network performance. As multiple SDN controllers exist, selecting the appropriate SDN controller based on the network topology is essential for achieving optimal network performance. We have conducted a comprehensive study and examination of different controllers for the hybrid SDN environment. Notable controllers include NOX, Floodlight, POX, ODL, Ryu, and ONOS. Subsequently, we have selected the ONOS [56] (Open Network Operating System) controller as the SDN controller for our hybrid environment, as it is compatible with the hybrid SDN and supports SDN-IP applications. SDN-IP is a program that integrates legacy and SDN networks; it operates on top of the ONOS controller and enables SDN nodes to integrate and communicate with the legacy routers through BGP ASes. The results demonstrate that with 30% appropriate SDN node deployment, we can get near-optimal results and can achieve the maximum benefits from the concept of hybrid SDN.

For the evaluation of incremental deployment and migration sequence to a hybrid SDN, there is a substantial body of research employing different approaches and strategies, particularly migration sequence, concerning traffic control method, as it is an essential aspect of SDN upgrading strategies. In this aspect, we have extensively compared different methodologies, such as DEG [66], VOL [69], Modified Greedy, and Local Search [62]. In the context of SDN upgrading strategies, as presented in Figures 3.6,3.7, and 3.8, our method, known as the FCM, is recommended as a migration sequence solution, as it aims to maximize controllable traffic weight with the incremental deployment of each SDN node. The objective of the FCM method is to maximize the controllable traffic that can be effectively managed with the incremental deployment of SDN nodes. Through systematic analysis and empirical investigations, we seek to ascertain the relative benefits, constraints, and suitability of the FCM method compared to alternative contemporary traffic control methods within the dynamic context of SDN deployment solutions. These kinds of efforts make a big difference in improving SDN migration frameworks and making network infrastructure more flexible and efficient in responding to changing needs and obstacles. DEG prioritizes nodes with the highest degree and the most connections. This strategy seeks to improve the network backbone by upgrading the most critical nodes for network connectivity. Conversely, VOL takes a distinct approach by concentrating on nodes with the most significant traffic volume. This method enhances high-traffic nodes to boost performance and avoid network congestion inside the SDN framework.

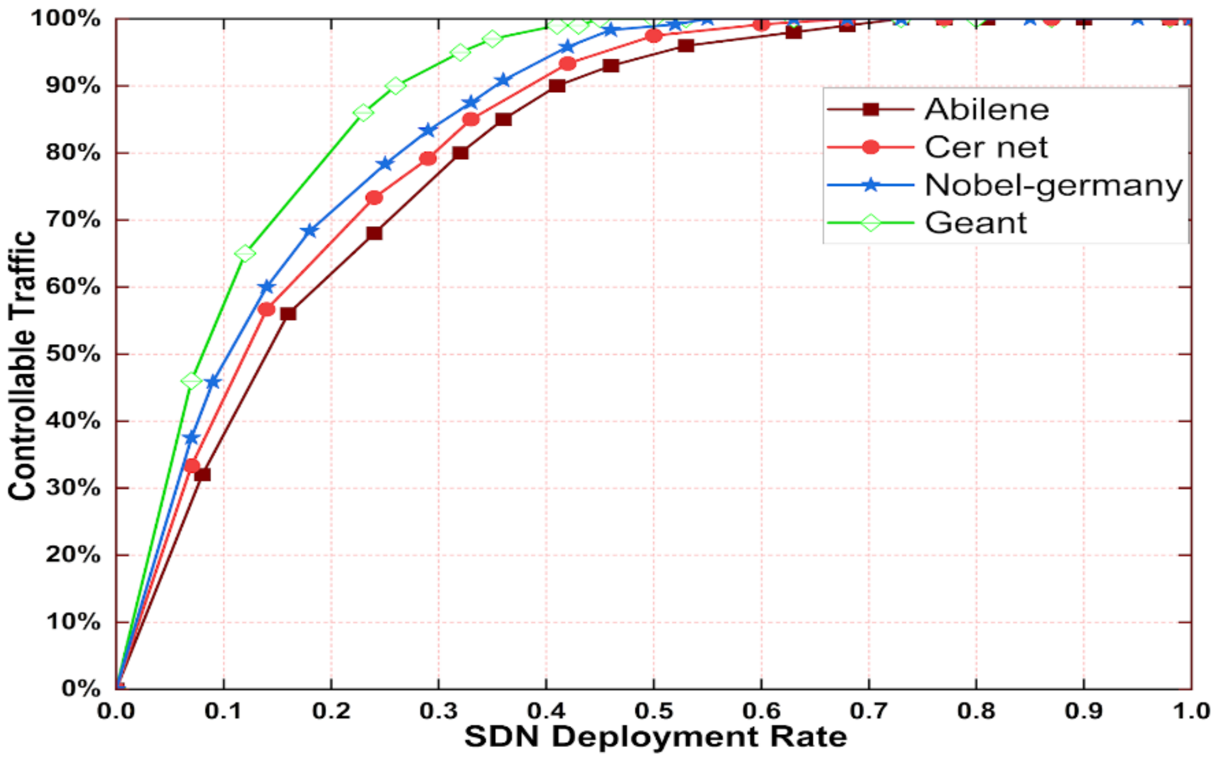
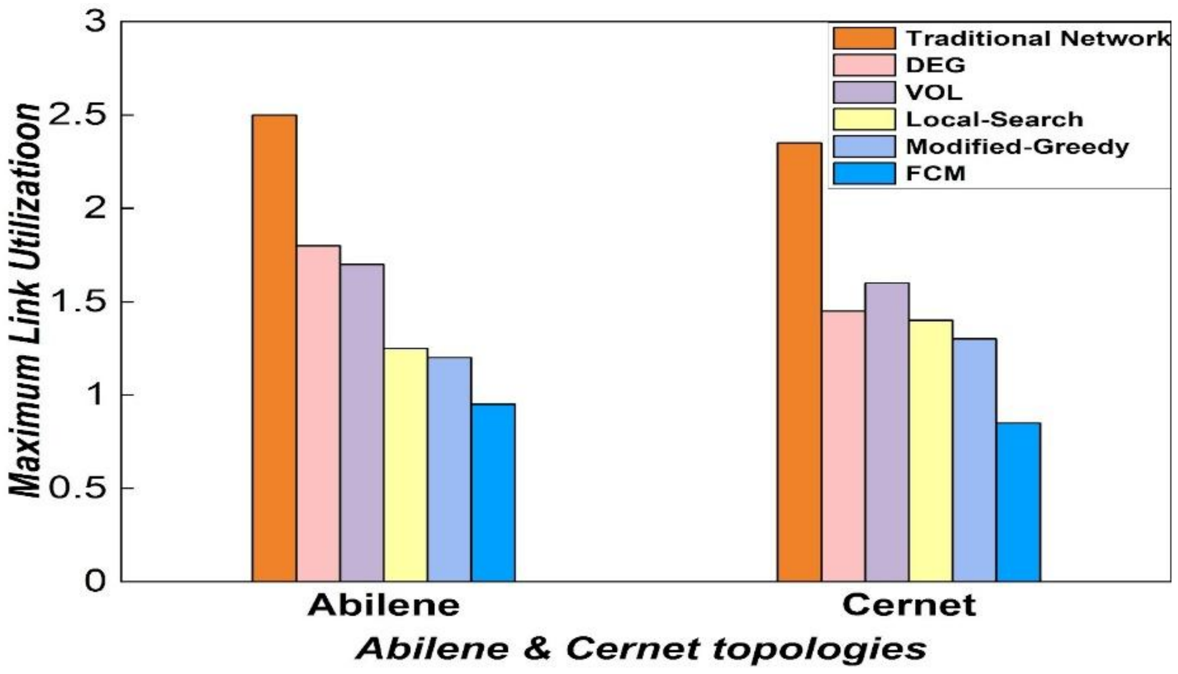
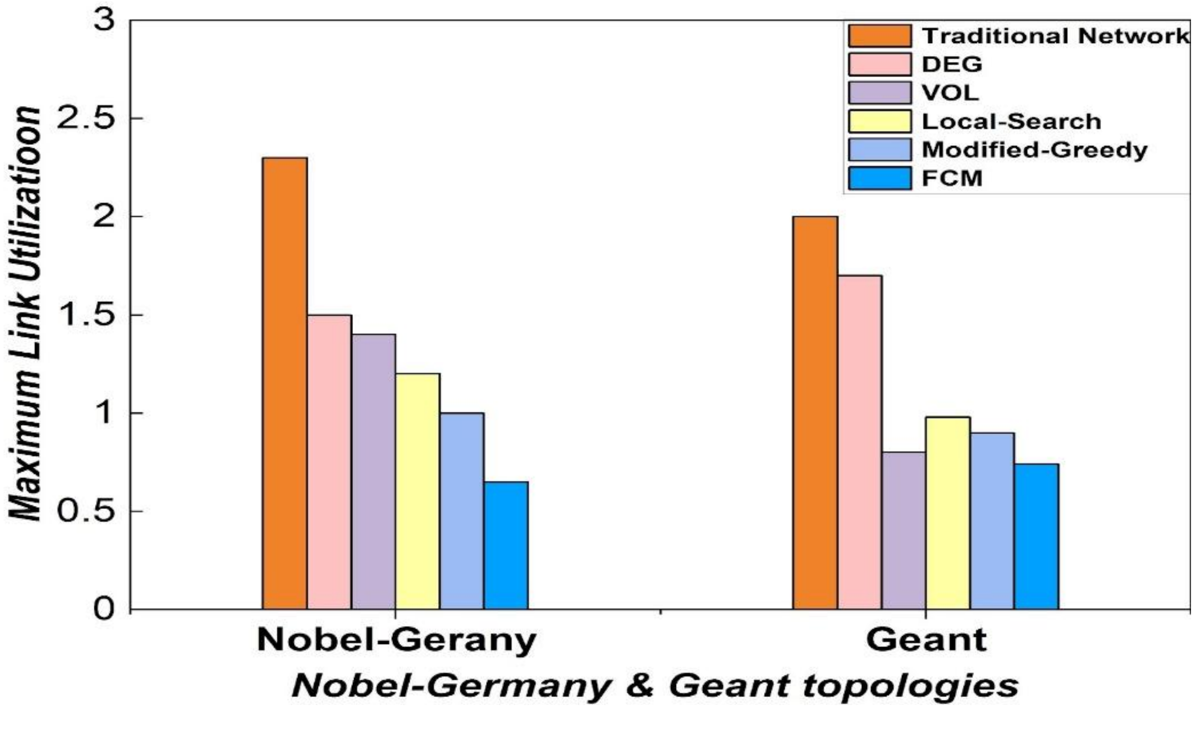


FIGURE 3.9 SDN nodes deployment ratios with controllable traffic

Some other strategies do not consider previous deployment ratios, such as the modified greedy approach, which prioritizes the migration of the nodes based on the migration cost and highest traffic volume; it balances the cost of migrated high-traffic nodes while minimizing overall migration costs. This method seeks to enhance network performance, ensure efficient upgrades, and optimize resource allocation by considering migration costs. In contrast, the local search method uses different approaches; it prioritizes controllable paths or routes across network infrastructure over migration costs or traffic volumes, and its goal is to increase the number of controllable paths in the network. This strategy aims to improve network flexibility and manageability by optimizing controllable paths for effective traffic steering and policy enforcement through the SDN environment. The FCM method, which considers previous deployment ratios, outperforms the VOL and DEG methods and achieves comparable performance to the modified and local search methods, which do not rely on previous deployment ratios.



a. Abilene and Cernet topologies



b. Nobel-Germany and Geant topologies

FIGURE 3.10 Performance analysis on minimizing MLU

We have conducted numerous experiments on various network topologies, and the results show that the FCM approach performs similarly to the local search and modified-greedy techniques while outperforming the VOL and DEG methods. Furthermore, to determine the complete migration solution, the computation complexity of these methods follows different orders, as the complexity of the FCM method is O(N), for local search, it is O(N^2), and for modified-greedy, it is O (N^3logN.), where N represents the number of nodes in the network. As a result, the FCM method achieves almost equal performance to local search and modified-greedy with lower computation complexity. The finding highlights the competitiveness and effectiveness of the FCM method, showcasing its ability to provide competitive outcomes in managing network traffic and optimizing network functionality, even compared to strategies that do not rely on previous deployment solutions.

Figures 3.5, 3.6, and 3.7 offer a detailed visual depiction of selecting nodes for migration to SDN nodes in each migration round, utilizing FCM, Modified-Greedy, and local-search mechanisms. The horizontal axis lists the node ID, while the vertical axis displays the total number of nodes upgraded to SDN nodes. Unlike the prior deployment strategies for the deployment ratios of SDN nodes, the FCM method takes the previous deployment ratios. Figure 3.6 illustrates the FCM method for SDN node selection, this method considers previous deployment ratios; we can see when the number of devices to be migrated is three, nodes 4, 8, and 12 are selected, and when the number of devices to be upgraded to SDN is five, nodes 4, 8, 12, 16, and 20 will be chosen, as this method considers previous deployment ratios. Meanwhile, local search and modified greedy methods do not consider previous deployment ratios. Figure 3.7 illustrates the local search method for SDN node selection; we can see that when the number of devices to be migrated is 6, nodes 4, 8, 16, 20, 24, and 28 will be selected, which are not based on the previous deployment ratios. Figure 3.8 illustrates the modified greedy node selection; we can see that when the number of devices to be migrated is four, nodes 4, 8, 12, and 16 will be selected, and when the number of devices to be migrated is five, then nodes 4,8,16,20 and 24 will be selected, in this method it will not select the node of 12 as this method consider highest traffic volume which not considering previous deployment ratios.

As we aim to increase controllable traffic weight, the traffic that is controlled by the SDN controller or programmable traffic, Figure 3.9 shows the relationship between the number of migration sequences or SDN deployment ratios with the controllable traffic weight for different real network topologies. There is a direct link between increasing the number of SDN nodes and the increase in controllable traffic; with more SDN nodes, we experience significant improvements in traffic management and overall network performance. As shown in Figure 3.9, the findings show that with a proper 30% SDN deployment ratio, we can control over 70% of the network traffic and get the maximum benefit in managing controllable traffic, which improves the overall network performance from a traffic engineering perspective. This empirical data highlights the importance of deploying SDN nodes in shaping and optimizing network traffic management strategies.

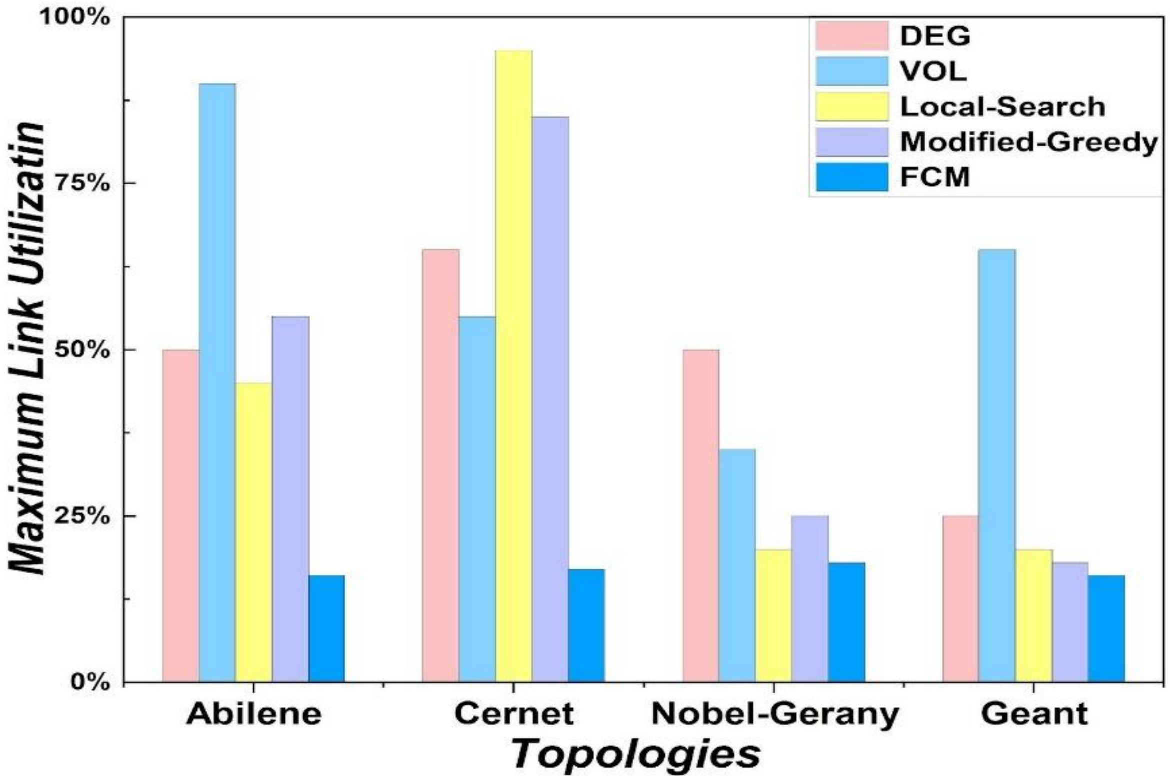


FIGURE 3.11 Comparison results on network load balance

To evaluate the performance of the migration sequence in the concept of a hybrid SDN network, we add up the MLU (Maximum Link Utilization) with the migration of each device. The main goal for many networks is to reduce the MLU to enhance network performance and avoid congestion. We will determine how much we can reduce a network's MLU after upgrading devices to SDN nodes. The MLU on various network topologies using various algorithms is calculated and shown in Figure 3.10 (a, b). We observe that utilization of more than one indicates the link exceeds its capacity, leading to congestion and an inability to handle the traffic efficiently. We find that in comparison to the traditional network approach, all of the other methods perform well in reducing network MLU. Still, compared to other techniques, our method, FCM, minimizes the MLU very fast and performs very well in lowering MLU in the hybrid SDN environment. Minimizing MLU in a hybrid SDN directly depends on the number of legacy devices to be upgraded to SDN nodes; with more SDN nodes, we can get a minimum MLU. As in the FCM method, we used a simple greedy algorithm; it selects the node fast and selects the most efficient device to be upgraded to SDN; in this case, we can minimize MLU with a small number of SDN nodes. Compared to other methods, our method can get the optimal result by deploying 30% of devices to SDN nodes in all the given topologies. In comparison, other methods for some topologies require 70% of devices to be upgraded to SDN nodes to get the optimal result.

The MLU is calculated using the linear programming solver CPLEX [153] optimization method. The analysis findings demonstrate a direct correlation between minimizing MLUs and increasing the number of deployment ratios; a higher number of SDN nodes corresponds to lower MLUs in the hybrid SDN network. As depicted in the graphs, the MLU of the FCM method decreases as the SDN deployment ratio increases; when the deployment ratio reaches 30%, the MLU stabilizes. This stability indicates that the network has achieved an appropriate balance between the distribution of SDN nodes and the utilization of network links, resulting in more sustainable and stable network performance. Increasing the number of SDN nodes can enhance the flexibility to forward and split the traffic flow in the hybrid SDN. Therefore, based on the network size, upgrading 30% of the nodes to SDN is recommended to maximize the benefits of the hybrid SDN concept. Consequently, with an appropriate 30% SDN node deployment, maximum benefits can be derived from a hybrid SDN network, leading to optimal performance.

In the context of a hybrid SDN, maintaining network load balance refers to the equitable distribution of traffic across all network paths to ensure efficient utilization of network resources and minimize congestion on individual links. This balance helps prevent scenarios where some links exceed their capacity (overloaded), leading to congestion and inefficient traffic handling, while other links remain unutilized or underutilized. Load balancing contributes to consistent network performance, minimized latency, and improved reliability.

Different metrics and approaches are available to assess load balancing in the hybrid SDN environment, including link utilization distribution, load variation, path diversity, failover capacity, and MLU. The most important metric for evaluating load balance in the hybrid context is MLU; in this study, we evaluate the load balance using MLU. It displays the fraction of the network links with the highest capacity or overloaded links, and those with zero capacity or underutilized. A high MLU implies that a specific link has reached its maximum capacity, resulting in network congestion and the inability to control traffic effectively. Minimizing MLU is crucial for load balancing since it helps to prevent overload on any link; with minimum MLU, we can distribute the traffic load more efficiently across the network.

Regarding network load balance, we find that even with the deployment of 17% of nodes to SDN, we can get the optimal result; the load balancing problems can be effectively solved by the deployment of a small portion of nodes to SDN, and with the upgrading of 17% we can get the optimal result. As shown in Figure 3.11, considering Abilene, Cernet, Nobel, and Geant topologies, upgrading approximately 16 % of the nodes gets nearly optimal performance while requiring much less investment in resources and effort for network upgrades. This result suggests that our greedy approach is cost-effective and highly efficient in practical scenarios. Complete network upgrades might be too expensive or impractical. Achieving near-optimal load balancing with minimum upgrades could offer significant advantages for network operators, making it a valuable strategy for managing network performance in real-world applications. The result demonstrates that our method has the potential to deliver significant advantages with a minimum deployment ratio, enhancing the efficiency and scalability of network management strategies.

This finding is informative and serves as a benchmark for balancing the benefits of SDN control with resource cost in a typical hybrid SDN; it offers a balance point where minimum SDN control is used to regulate traffic without relying excessively on SDN resources. However, the generalizability of this finding varies depending on network setup and circumstances. Changing the SDN deployment ratio may increase performance in more specialized or severe cases, such as networks with high dynamic traffic, stringent resilience requirements, or limited route redundancy. This ratio can be used as a starting point, but it may need to be changed based on network characteristics and operational requirements.

3.7. Summary

In recent years, the incremental deployment of SDN nodes into traditional network architecture has received significant attention globally. The primary objective of incrementally deploying SDN nodes into existing legacy architectures is to leverage the advantages of both networks and enhance overall network performance from a traffic engineering perspective. This chapter investigates the incremental deployment of SDN nodes in conventional networks using a simple greedy algorithm, presenting a potential solution for the evolution of network architecture. The proposed technique efficiently integrates SDN features while minimizing network disruption and optimizing link utilization. Simulation experiments conducted on real network topologies demonstrate that by upgrading 17% of the nodes to SDN, near-optimal performance can be achieved while requiring substantially less investment in resources and effort for network upgrades. This result suggests that the greedy approach is cost-effective and highly efficient in practical scenarios. Achieving near-optimal load balancing with minimal upgrades could offer significant advantages for network operators, making it a valuable strategy for managing network performance in real-world applications. This study contributes to the knowledge of SDN deployment methodologies, providing valuable insights for organizations seeking to transition to a more flexible and agile network infrastructure. Future research will involve implementing our method on large-scale networks, synthetic topologies with diverse connectivity, and a real testbed network environment.

CHAPTER-4

Routing Optimization in Migrated Hybrid Software-Defined Network

The amalgamation of Software Defined Networking (SDN) principles with conventional networking techniques has led to the emergence of hybrid SDN; as current network infrastructure evolves, it integrates SDN's programmability with the proven protocols of conventional networking systems. A practical approach involves adopting a hybrid SDN model, wherein conventional and SDN components are integrated seamlessly. This shift in paradigm introduces both new opportunities and challenges for Traffic Engineering (TE), demanding creative approaches to enhance network performance, resource efficiency, and overall effectiveness. In this study, we explored routing optimization for TE within a hybrid network undergoing migration and modelled the problem as a mixed-integer non-linear programming framework. Additionally, we examined the specific challenges faced in TE within a hybrid SDN environment, where conventional network devices operate alongside SDN-enabled nodes. We proposed a heuristic approach (H-STE) to simultaneously optimize OSPF weight configurations and the traffic splitting ratios of SDN nodes within the hybrid network environment. To assess the effectiveness of our method, we performed comprehensive evaluations using real-world network topologies, and the results indicate that with a 30% SDN deployment ratio, we can achieve superior TE performance as the Maximum Link Utilization (MLU) stabilizes, leading to a near-optimal result. This study provides valuable insights for researchers, practitioners, and network architects working with SDN, hybrid SDN environments, and traffic engineering optimization.

4.1. Introduction

In the current network design, the primary challenge is network management, wherein both industry and professionals require assistance in managing contemporary networks; several concepts and structures have been proposed globally for future network design. To address these challenges, the software-defined networking (SDN) design was initiated; SDN's fundamental concept is the separation of data and control layers. With its global perspective and centralized management, SDN enables flexible and reliable network management, increasing network throughput and link utilization. Furthermore, it provides a novel flow scheduling approach to enhance Traffic Engineering (TE). In recent years, SDN networks have become a prominent topic of discussion in telecommunications. It is a crucial and innovative network architecture concept that emerged with the advancement of technologies such as mobility, virtualization, and the Internet of Things (IoT).

In recent decades, Internet Service Providers (ISPs) networks have experienced a significant increase in traffic due to the rapid expansion of Internet applications, including games, communication tools, and videos. ISPs have been investing substantially in constructing their network infrastructure to ensure a network with maximum bandwidth and minimum latency; however, this increase in traffic has resulted in frequent congestion, leading to a lack of guaranteed user experience and satisfaction. Due to the exponential advancement of the IoT and the Internet of Everything (IoE), emerging Internet applications have substantially increased network traffic [1]. Consequently, global attention is being directed towards finding solutions to mitigate network congestion and achieve traffic load balance; simultaneously, there is an urgent need to advance network management technologies. Traffic Engineering is a highly effective network administrative tool that balances network load and enhances network performance by optimizing traffic routing methods [3]. TE is an efficient approach that balances the flow, prevents congestion, and improves network performance; it enables control over how traffic is routed over the network; it has gained substantial interest from both business and academia in the past few decades. However, due to the network's complex structure, the optimization of traffic engineering remains an open area for research.

SDN network design is a well-established network design that decouples the network's control and data planes. It offers an efficient and flexible approach to optimizing traffic routing from a TE perspective. In an SDN network, policies or routing algorithms can be implemented in the control plane to provide flexible control over the distribution of traffic to any outgoing links in SDN switches, and the SDN switches can distribute traffic to various paths in a flexible and unrestricted manner in TE [4-5]. SDN networks can achieve more user-oriented, flexible, and intelligent network control by tailoring the forwarding rules for individual flows [6].

The SDN network architecture is structured into three core layers. The first is the data layer, made up of forwarding devices that handle the routing of data packets. The second is the control layer, which serves as the central component of the SDN framework. Its main role is to oversee network management and convert application-layer requests into instructions for the data layer. The last layer is the application layer, which is composed of applications. With a centralized controller, this network can control all the networks with programmable devices; compared to traditional networks, centralized control can significantly enhance overall network performance, such as traffic engineering, link utilization, and network throughput. It has gained significant worldwide attention in recent years, and the adoption of SDN has substantially increased among network operators; network operators can now dynamically control the routing flows inside the network and immediately modify the routing choice as needed. For instance, Microsoft [4] and Google [5] have been striving to achieve nearly 100% utilization of the links. The SDN network provides significant advantages in traffic engineering by allowing dynamic and timely network route adjustments, effectively addressing unexpected issues such as traffic variations and link failures. SDN optimizes TE flexibly, which renders the SDN network a unique and advantageous technology.

Despite significant motivation and numerous advantages, the direct migration to a fully SDN infrastructure presents considerable challenges. Upgrading an entire conventional network to SDN appears risky and costly, particularly for large-scale networks comprising thousands of conventional network devices. This transition faces multiple obstacles, including economic, technical, and security concerns, and potential service disruptions. Consequently, implementing hybrid SDN and the incremental deployment of SDN nodes emerges as the most viable approach to address these challenges. A hybrid SDN integrates the flexibility and programmability of SDN networks with the established reliability of traditional network infrastructure, allowing conventional IP network devices to coexist with SDN nodes. This hybrid architecture represents a network design in which decentralized and centralized networks operate concurrently and maintain communication. There is no need to upgrade all the traditional network devices to the SDN network; we can incrementally migrate to a hybrid SDN and then to a fully SDN network.

In recent years, hybrid SDN has emerged as a prominent network design paradigm. Furthermore, exploring traffic engineering within hybrid SDN environments has garnered significant attention from industry and academia [25]. In conventional Internet Protocol (IP) networks, Open Shortest Path First (OSPF) remains one of the most widely utilized interior gateway protocols [28]. Within a traditional IP network, a weight (or cost) is assigned to each link, and traffic is routed along the shortest paths (least cost) from source to destination nodes using routing protocols that prioritize shortest path routing, such as OSPF. When multiple shortest paths are available, traffic is uniformly distributed among Equal Cost Multiple Paths (ECMP). Consequently, the efficiency of traffic engineering in a conventional IP network is directly influenced by the weights assigned to the network. In hybrid SDN environments, conventional network devices are limited to supporting the OSPF protocol. In contrast, SDN network nodes operate in hybrid mode, enabling them to support the OSPF protocol for communication with legacy devices and the OpenFlow protocol for interaction with SDN controllers and programmable nodes.

4.2. Proposed Approach

This study primarily focuses on a hybrid Software-Defined Networking architecture in which the SDN nodes are incrementally deployed. We concentrate on minimizing the MLU of a hybrid SDN by optimizing two critical aspects: optimizing the OSPF weight settings across the entire network to balance the flows originating from conventional devices and optimizing the traffic splitting ratio of SDN nodes. Optimizing both aspects presents significant challenges as these devices are closely interconnected. The OSPF weight settings are crucial in determining the shortest paths for conventional devices to their destination. When the weights are adjusted, they alter the routes that data packets traverse through the network, which affects the volume of traffic flowing into SDN nodes, subsequently influencing how the traffic is divided among different output paths. As both network devices are interconnected, optimizing both is critical for effective traffic management and ensuring a balanced load across the network.

To address this challenge, we introduce a heuristic algorithm, H-STE (Hybrid SDN Traffic Engineering), which aims to minimize MLU more effectively in a hybrid SDN environment. Considering the cost and advantages of SDN nodes, we found that with a 30% deployment ratio of SDN nodes, we can achieve the optimal result compared to a fully SDN environment. Furthermore, the experimental analysis indicates that upgrading only 30% of SDN nodes can yield the most benefits and achieve nearly optimal performance. This chapter addresses the routing optimization in a hybrid SDN environment; we proposed a heuristic algorithm to minimize MLU by optimizing the traffic splitting ratio of SDN nodes and the OSPF weight setting of conventional networks. Lastly, we evaluate the efficiency of our suggested method on various topologies and real network traffic datasets. The key contributions of our study are summed up as follows:

These hybrid network scenarios addressed in the studies differ from the hybrid SDN scenario investigated in our context. In this chapter, we formulated a mixed integer non-linear programming problem for routing optimization in a hybrid SDN environment. We proposed an effective heuristic algorithm that optimizes both the traffic splitting ratio of the SDN nodes and the OSPF weight setting of conventional network devices and minimizes MLU more efficiently than the previous methods. We have built the conventional network setup using Mininet and MiniEdit tools, then incrementally deployed SDN nodes into our topology. In this simulation, we have studied and tested multiple SDN controllers; finally, the ONOS controller is chosen as the SDN controller, and the integration between the legacy network and the SDN network is carried out through the SDN-IP, which operates on top of the ONOS controller, allowing SDN nodes to communicate with legacy routers using BGP protocols.

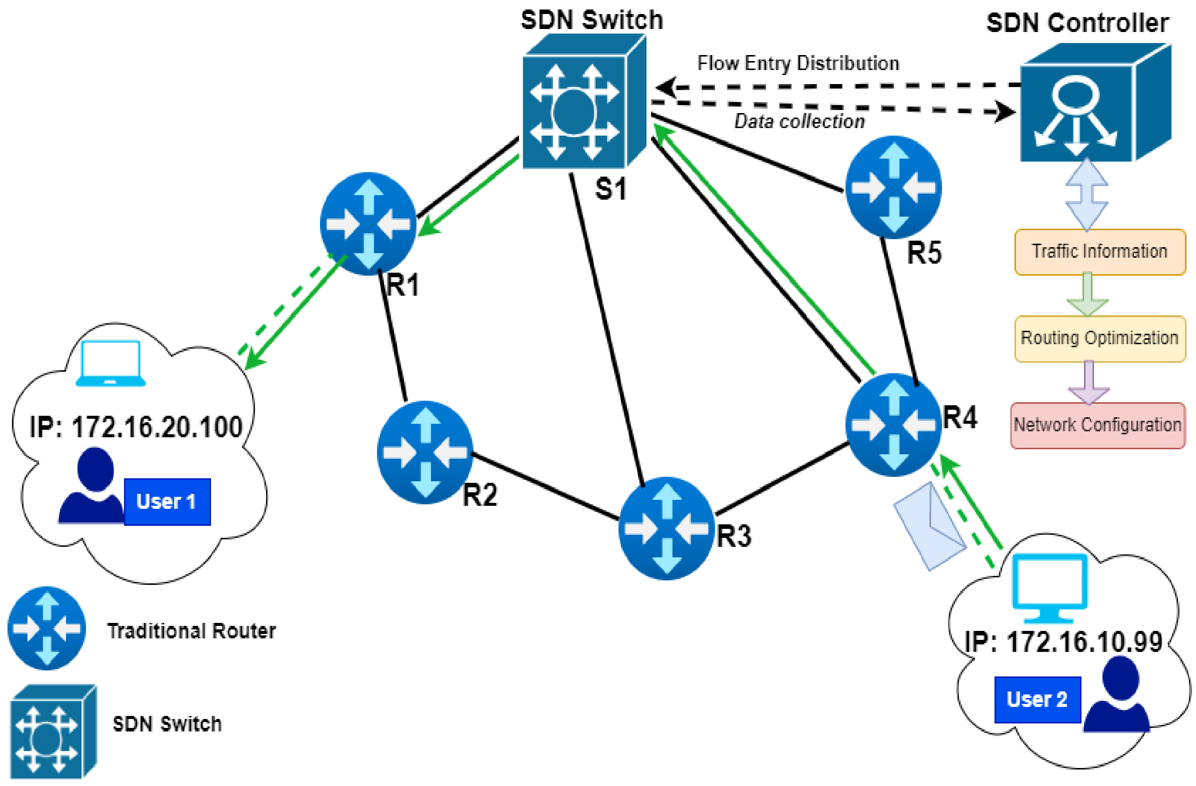


Figure 4.1. Hybrid SDN design

4.3. Hybrid SDN Network

This section describes the hybrid software-defined network discussed in this chapter. A hybrid SDN integrates traditional network equipment with SDN to enhance user experience and performance by regulating, modifying, configuring, and managing network behaviors. It combines SDN's centralized management and programmability with conventional networks' reliability and stability; traditional devices ensure reliable routing, while SDN nodes focus on network optimization. Hybrid SDN comprises three main components: conventional network devices, SDN nodes, and controllers. In this architecture, traditional devices manage certain network parts, such as access layers or legacy systems, while centralized controllers oversee other areas, like data centers or virtualized environments. As illustrated in Figure 4.1, R1-R5 are conventional legacy routers that route traffic based on the shortest path from source to destination, while S1 represents an SDN node controlled by an SDN controller.

Upgrading all conventional network devices to SDN is unnecessary at once; instead, we can incrementally deploy SDN nodes to migrate to a hybrid SDN before fully integrating into a fully SDN network. Traditional devices utilize distributed routing protocols like IGP and OSPF to forward packets along the shortest path defined by OSPF. In contrast, SDN nodes operate in hybrid mode, supporting OSPF and OpenFlow protocols. The SDN controller oversees the forwarding operations of these nodes while managing network traffic from a worldwide perspective. It provides flow entries to the flow tables of SDN nodes, and then the SDN nodes forward packets according to the established rules.

The SDN controller is a critical network component responsible for managing all data plane operations. The controller's effectiveness directly impacts overall network performance. Selecting the appropriate SDN controller based on network topology is essential, as various options are available. We have thoroughly analyzed several controllers suitable for the hybrid SDN environment, including NoX, POX, Floodlight, Ryu, ODL, and ONOS. After careful consideration, we chose the ONOS controller for our hybrid SDN topology due to its compatibility with hybrid networks and support for SDN-IP applications. SDN-IP integrates traditional and SDN nodes, enabling them to connect and exchange information using Border Gateway Protocol (BGP) and Autonomous Systems (AS).

Several factors must be considered when constructing an SDN network: load balancing, traffic engineering, network performance, security, and link failure recovery. A practical strategy to leverage these aspects is incrementally deploying SDN nodes and controllers alongside conventional networking. This approach enhances overall network performance while minimizing service disruption.

Table 4.1. Key notations.

Variable	Explanation
G= (V, E)	Undirected graph
V	set of all network nodes
E	set of links in the network
C (e)	link capacity of the edge e
Vl	legacy node set
Vs	SDN node set
e_{ij}	A link with nodes for the head (i) and tail (j).
Vc	a subset of the migrated SDN nodes.
D=(Di)	Set of Traffic Matrices
f^t	Traffic assigned destined to (t)
ri	Weight coefficient of Di
Ui	The value of MLU under Di
T	Total periods
ω	OSPF weight configuration
ϕ	The weighted MLU

τ	controllable flow

4.4. Network model and problem formulation

4.4.1. Network Model and Variable Declaration

We modeled the network as a graph G = (V, E), where V represents the set of all network nodes and E represents the set of links connecting devices to the network. Each node can be a conventional network device (Vl) or an SDN node (Vs), where Vl∩VS =∅. For $\forall$ υ1, υ2 $\in \boldsymbol{V}$, $\boldsymbol{\upsilon 1} \neq \boldsymbol{\upsilon 2}$, for $\boldsymbol{\upsilon 1}$ to $\boldsymbol{\upsilon 2}$, there should be at least one path to reach. C (e) represents the link capacity of e∈ E. D represents the Traffic Matrices (TMs) set where D = (D1, D2, D3, ...,Dn), and ri represents a positive weight coefficient for each TM Di. Vc represents the subset of migrated SDN nodes. ω represents the link weight setting of the OSPF protocol for legacy devices, and the variable f represents the traffic splitting ratio of SDN nodes. **Table 4.1** summarizes all the sets of notations utilized in this chapter for a better understanding.

In this study, we posit that all legacy devices employ conventional IP routing, such as OSPF, which transmits packets according to the shortest path. However, we configure all nodes as conventional network devices that implement the OSPF protocol. Our objective is to incrementally upgrade legacy nodes to SDN nodes, wherein the splitting ratios are presumed to be determined by the SDN controller to split the flows and increase controllable traffic that can be efficiently managed. For this experiment, both the network topology and traffic matrix are assumed to be static. During each phase, traffic is routed in two distinct ways: traditional devices forward traffic along the shortest path as defined by their routing tables, while SDN nodes distribute traffic based on the optimal splitting ratio determined at that particular stage. We aim to upgrade most legacy devices to SDN while maintaining the overall network load balance and minimizing MLU. The most crucial metric for evaluating load balance in the hybrid SDN is to minimize MLU; in this study, we evaluate the network load balance by considering the MLU. It represents the utilization percentage of the link with the most capacity or overloaded and with zero capacity or underutilized links in the network.

1. ***Definition:*** Given a directed graph G (V, E), a traffic demand matrix TM Di, and a specified number of legacy devices, the objective is to minimize the Maximum Link Utilization (MLU) by upgrading the majority of legacy devices to SDN nodes that support flexible flow-splitting capabilities.

4.4.2. Problem formulation

We formulate this problem as a mixed integer program based on the given network model and definition. We consider the MLU for each device's migration to measure the performance of migration sequences. The primary goal is to minimize the total MLU after the entire migration process (1)-(7). In the first step, we determine the best migration sequence for the migration of each SDN node. We use an optimization algorithm to analyze different migration processes; this algorithm will calculate the MLU for each migration and, ultimately, will determine and select the nodes with a minimum MLU to improve the overall network performance and minimize the overall network load during the migration process. For conventional network devices, the flows are restricted using the ECMP (Equal cost multi-path) rule, which routes the packet based on the shortest paths. To achieve flow balancing, we optimize the OSPF weight setting. As the traffic is splitting along their outgoing links for the SDN nodes, we optimize the splitting ratio of the SDN nodes to achieve load balancing.

The network performance improved by minimizing the MLU for each traffic matrix Di. In the hybrid SDN environment, the value of MLU is specified by the traffic splitting ratio of SDN nodes f and the link weight setting of conventional devices ω. As a result, we can formulate the problem as follows:

$$\phi = \text{minimize} \sum_{s=1}^{N} \text{Ui}_{\{\upsilon_1,\upsilon_2,\ldots\ldots \text{Vs}\}} \tag{1}$$

$$\sum_{e\in \text{In}(c)} f^{t}_{e,D_i}(\omega) + \text{Di}(c,t) = \sum_{e\in \text{Out}(c)} f^{t}_{e,D_i}(\omega) \quad c \in \text{Vs}, t \in V, i \in [1, n] \tag{2}$$

$$\sum_{i=1}^{n} r_i = 1, \quad 0 \leq r_i \leq 1 \qquad (3)$$

$$\omega(e) \in N^{+}, \quad \forall e \in E \qquad (4)$$

$$0 \leq Ui_{\{\upsilon_1,\upsilon_2,\ldots\ldots Vs\}} \leq 1, \forall D_i \in D, \quad i \in [1,n] \qquad (5)$$

$$f_{e,D_i}^{t}(\omega) \geq 0, \ \forall e \in E, \forall t \in V, \forall D_i \in D, i \in [1,n] \qquad (6)$$

$$\sum_{t \in V} f_{e,D_i}^{t}(\omega) \leq Ui_{\{\upsilon_1,\upsilon_2,\ldots\ldots Vs\}} C(e), \forall e \in E, \forall D_i \in D, i \in [1,n] \qquad (7)$$

To find the best migration sequence, the primary goal is to minimize the total MLU after the entire migration process (1)-(7); our objective is to identify the best migration sequence S = (υ1, υ2, υ3,…, Vn) with a minimum total MLU.

Equation (1) represents the primary goal of the optimization, aiming to minimize where represents the weighted sum of MLU over the whole migration process. Equation (2) signifies the flow conservation mechanism in the network at each node v, implying that the total flow of a device, along with the flows it generates (traffic demand), should be equal to the flows that flow out of this device, which is flow conservation; this equation suggests that the traffic demands should be met equally. Equation (3) represents the capacity constraint, which imposes a constraint on link capacity; this equation ensures that the total flow on any edge node should not exceed its capacity, and the flow on each device should remain within its capacity limits, preventing overload that could lead to failures or congestion.

Equation (4) ensures that the weight ri assigned to each traffic matrix should be an integer within the range [0,1]; link weights are often used in routing algorithms to define the cost or preference of a particular link. Ensuring these weights are integers presents the routing calculations necessary for specific network protocols requiring integer weights. Equation (5) ensures that the link weights are positive integers, essential for practical networks. For proper network operation, the link weights must be non-negative.

Equation (6) ensures that the MLU values are between 0 and 1, which is realistic, where one means the link is fully utilized, while 0 represents no utilization. Equation (7) ensures that all traffic flows on edge e are non-negative; negative flows are not physically meaningful in network flow problems. This ensures that traffic flow values cannot be negative, guaranteeing that all flow values are realistic and feasible within the network.

The constraints ensure proper flow conservation, respect link capacities, and enforce non-negative traffic flows while meeting traffic demands and integer weight requirements. This optimization problem balances network traffic across multiple traffic matrices, ensuring efficient use of network resources while adhering to network constraints

4.4.3. Problem Complexity

In a hybrid SDN environment, minimizing MLU is an NP-complete problem. The MLU in a hybrid SDN network depends on the deployment ratio of SDN nodes; if the splitting ratio and the link weights are given, then the network MLU can be computed in polynomial time and can be compared to the threshold U, whether it is less or more than threshold U; we can check quickly if the MLU is below a certain threshold U. When the weight settings are in integer or unknown values, the problem becomes much more complex, so minimizing the MLU in a migrated hybrid SDN is shown to be more difficult than an NP-complete problem. As a result, the complexity of our problem is an NP-complete problem.

To show NP-completeness, we reduce the exact cover by the 3-Sets (X3C) problem, known as NP-Complete [155], to our problem in polynomial time. We prove that the problem of minimizing MLU in hybrid SDN can be transformed into another well-known NP-complete problem: Exact cover by 3 sets (X3C).

X3C problem definition: Given a set 3X with 3p elements and a collection of subsets 3c, the question is whether a subfamily of 3c covers all elements of 3X without overlap.

To relate MLU to X3C, we build a flow graph, as shown in Figure 4.2. In the graph, S represents the source node, and t represents the destination node; 3X represents the node element and the subsets; directed edges connect the subsets to their elements, which indicate which elements belong to which subsets. All the flows are generated from the source node S to all unit flows. The flow from source S to destination t must match the exact cover size (represented by q), and the total flow from S to t equals q. In this way, if a valid cover exists, the MLU will be 1, which means the flows are evenly distributed on the links. We achieve a q flow from source S to node t, and the MLU is 1, meaning the chosen subset covers 3x without overlap, confirming a solution to X3C.

By showing that solving the MLU problem in a hybrid SDN can be transformed into solving the X3C problem, the proof establishes that minimizing MLU is NP-complete.

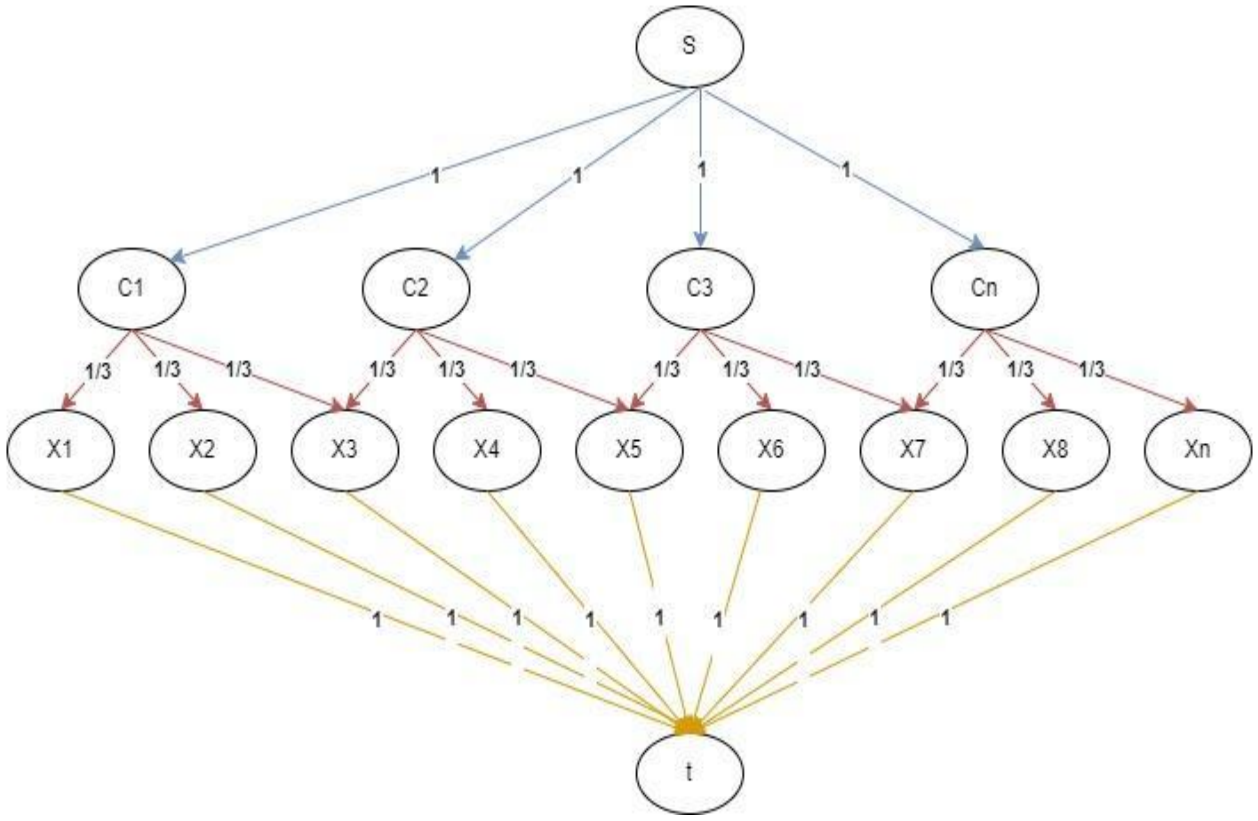


Figure 4.2. MLU Flow graph

In a hybrid SDN environment, if the link weights and flow splitting ratios are known, the Maximum Link Utilization (MLU) can be calculated in polynomial time. This allows us to efficiently check whether the MLU falls below a specified threshold UUU. We demonstrate that the MLU optimization problem can be reduced to the Exact Cover by 3-Sets (X3C) problem, which is a known NP-complete problem. The X3C problem is defined as follows:

Instance: a set X={x1,x2,…,xn} consisting of n elements (with n=3q), and a collection C of subsets, where each subset Ci contains exactly three elements from X.

Question: Can you find q pairwise disjoint subsets from C that cover all elements in X together?

We build a flow graph, as shown in Figure 4.2**,** to model the relationships between the elements and subsets in the context of the hybrid SDN. Each element x_j in X becomes a node; each subset C_i becomes an SDN node. We added the nodes s t as the source and destination nodes to manage the flows in the hybrid SDN. Directed edges from s to each subset C_i (capacity of 1), directed edges from C_i to its respective elements x_j (capacity of 1/3), and the edges from each x_j to the destination node t (capacity of 1). For flow analysis, if a valid cover exists in X3C, we assign unit flows from node *s* to the selected C_i nodes. Since the flow is distributed evenly, the MLU will be 1, and the total flow from *s* to *t* will equal *q* (the number of subsets chosen). Conversely, if the flow from *s* to *t* is *q* and the

MLU is 1, the chosen subsets cover all elements without overlap, confirming a solution to the X3C problem. By showing that the problem of minimizing MLU can be transformed into solving the X3C problem, we conclude that minimizing MLU in a hybrid SDN is NP-complete.

4.5. Routing Optimization

Optimizing routing flows is essential after migrating conventional network devices to SDN and creating a hybrid SDN environment. In a hybrid SDN, traditional devices use OSPF, while SDN nodes utilize flow routing, necessitating the optimization of both OSPF weight settings and the splitting ratio for SDN nodes. To optimize the OSPF weight setting and the splitting ratio of SDN nodes, we introduce a heuristic approach, H-STE, using different algorithms to achieve this optimization offline in a hybrid network. We employed the distributed Floyd-Warshall algorithm in the H-STE approach to quickly determine optimal paths between source and destination nodes; to optimize the traffic splitting ratio of SDN nodes and to compute the shortest paths for each node, we employ the Bellman-Ford algorithm, which is particularly useful in network scenarios where edge weights may change dynamically.

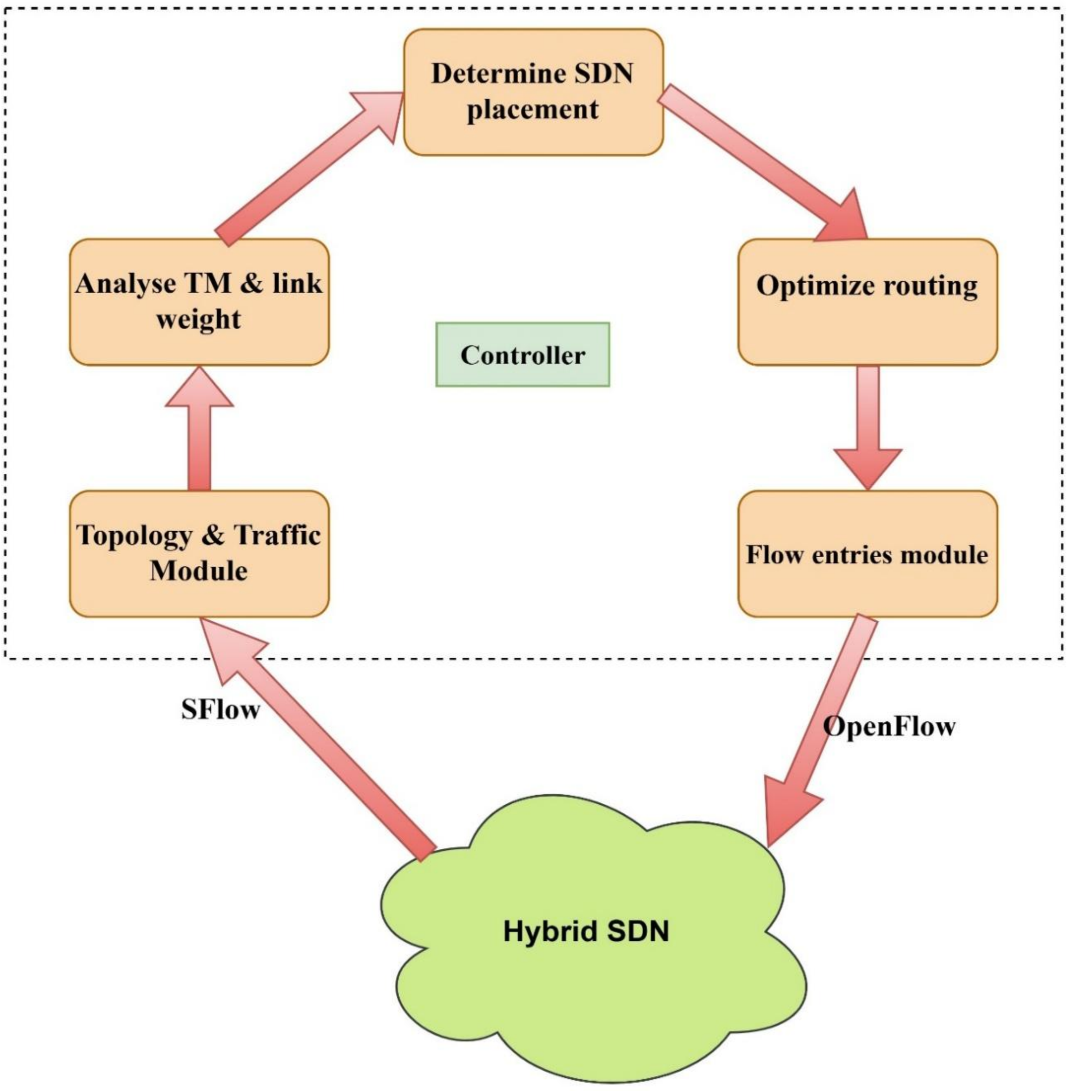


Figure 4.3. Routing Optimization framework

4.5.1. Algorithms

In this section, we outline the algorithms used in this experiment. First, we explain the local search heuristic approach designed to optimize flow routing in a hybrid SDN; next, we present the approximation ratios and analyze the complexity of the algorithms.

Flexible SDN nodes coexist with traditional routing protocols in a hybrid SDN environment. The H-STE method helps bridge the gap between the two by optimizing the whole network performance and adhering to the limitations imposed by legacy routing infrastructure. It iteratively adjusts the SDN node's traffic splitting ratios to reduce network congestion, and it has a multi-objective focus, as it minimizes MLU and considers other network performance metrics. The algorithm dynamically switches between local refinement and occasional worldwide search perturbations to improve the chance of finding the global optimum.

4.5.2. Algorithm 1: H-STE

After migrating to a hybrid SDN, we introduced the heuristic algorithm H-STE, described in Algorithm 1, for routing optimization in hybrid SDN environments. The algorithm's main objective is to optimize traffic engineering in a hybrid SDN environment by optimizing the OSPF weight setting and traffic splitting ratio of SDN nodes. The goal is to minimize MLU while considering multiple objectives, such as throughput and latency. As it optimizes both, the optimization process is divided into two phases: the optimization of the OSPF setting and the optimization of the traffic Splitting ratio.

```
Algorithm 1: H-STE

Input: G= (V, E), $V_s$, C($e$), D, initial_step_size, threshold, increase_factor, decrease_factor, max_iter_time
Output: U
1   Initialize weight-setting matrix ;
2   $U_{current}$ = parallel_distributed_floyd_warshall (G, , D);
3   Set $U_{best}$ = $U_{current}$;
4   Set best= ω;
5   Set Adaptive_step_size = initial_step_size;
6   While iteration_times ≤ max_iter_time do
7       $U_{current}$ = splitting_ratio (  best, $V_{s,}$ D, C($e$), G, adaptive_step_size;
8       If Ucurrent < $U_{best}$ then
9           Update $U_{best}$ = $U_{current;}$
10          Update best= ω;
11      If random (0,1) < global_search_probability then
12           = global_perturbation(ωbest);
13      If last_improvement < threshold then
14          adaptive_step_size∗ = increase_factor × random (0.9, 1.1);
15      Else
```

```
16          adaptive_step_size∗ = decrease_factor × random (0.9, 1.1);
17      if avg_improvement < small_threshold **then**
18          max_iter_time -= decrease_factor;
19      else if avg_improvement > large_threshold **then**
20          max_iter_time += increase_factor;
21      if distributed_stopping_criterion_met () **then**
22          break;
23      Return U = $U_{best}$;
```

This local search heuristic approach is designed to optimize the splitting ratio of SDN nodes and the OSPF weight setting to minimize MLU in a hybrid SDN environment. In the initialization phase (lines 1-4), the algorithm begins to set the initial weight matrix. The starting point is based on historical traffic patterns, ensuring a better initial solution. To achieve minimum MLU, we initially utilized a distributed version of the Floyd-Warshal algorithm to find the most efficient routes between each combination of source and destination nodes faster (line 2), and both the current and best MLU values are set (lines 3-4). Following that, we distribute traffic demands along these optimal shortest paths.

The main loop phases (lines 6-22) iterate until the stopping criterion or a maximum iteration limit is reached. In each iteration, the splitting ratio function updates the traffic splitting ratios to split and distribute traffic more efficiently, balancing multiple objectives such as MLU and QoS (line 7). The splitting ratio plays a crucial role in the iterative process of the H-STE method. It is used to modify the traffic splitting ratios at the SDN nodes at each iteration, ensuring effective load balancing and traffic distribution over multiple paths. The splitting_ratio algorithm is designed to consider multiple objectives and uses the adaptive approach to fine-tune the traffic flows. This approach ensures that traffic is routed to minimize MLU and meet service-level requirements for critical applications. If an improvement is found, the best solution will be updated (lines 8-10). To avoid local minima, the solution is periodically modified by a hybrid global search mechanism (lines 11-13). An adaptive step size adjustment ensures that the algorithm can search the solution space more aggressively when improvements are slow and more thoroughly when close to the optimal solution (lines 14-16). The iteration count is dynamically adjusted based on the convergence rate (lines 17-20). Finally, a distributed stopping condition (lines 21-22) terminates the algorithm early if improvements stagnate across the network. The distributed approach ensures scalability for large networks. The algorithm returns the best-found solution Ubest, representing the configuration with the lowest MLU (line 23). In this algorithm, we employed the distributed Floyd-Warshall algorithm to efficiently determine the shortest path between every pair of source and destination nodes. This algorithm operates with a time complexity of $O(n^3)$, where n represents the total number of nodes in the network topology.

4.5.3. Algorithm 2: Splitting ratio with Bellman-Ford

Algorithm 2: Splitting ratio with Bellman-Ford

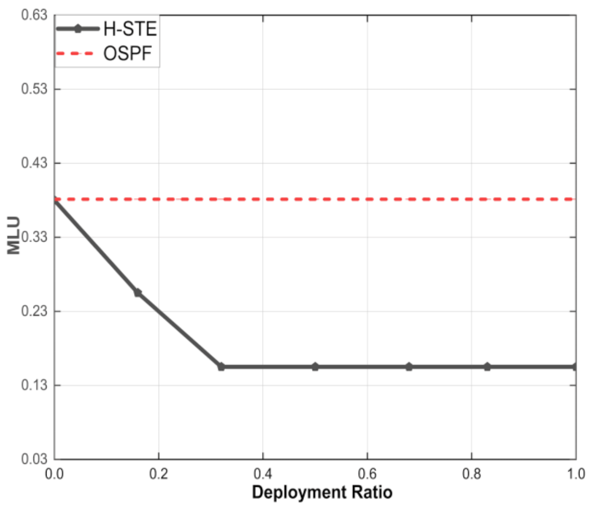
```
     Input: local optimal OSPF weight setting current, $V_s$, C($e$), D, V, E
     Output: U
1    foreach v ∈ V do in parallel:
2        DAG[v] = Bellman_Ford_Shortest_Path (ωcurrent, v, E, dynamic_weight_adjustment);
3    foreach i ∈ Vs, eij in outgoing links of i do
4        if link eij meets heuristic criteria then
          Add link eij to DAG where i ≠ j and i, j are incomparable;
5    foreach iteration in max_refinement_iterations do
6    foreach v in Topological_Sort(V) do
7        Route_Flow_Adaptively (G, v, D, utilization_data);
8    expr = Multi_Commodity_With_Adjusted_Constraints (E, V, utilization_data);
9    U_candidate = cplex_solve(expr);
10   if |U_candidate - U_previous| < tolerance then
11       break;
12   U_previous = U_candidate;
13   U = U_previous;
14   return U;
```

We present a thorough explanation of the splitting ratio in Algorithm 2. The algorithm optimizes the traffic splitting ratio of SDN nodes in a hybrid SDN environment using the Bellman-Ford shortest path computation and adaptive flow distribution. It adjusts traffic flows over multiple paths while considering real-time network conditions, such as link utilization and dynamic weight updates. The main objective of this algorithm is to optimize the traffic splitting ratio of SDN nodes when OSPF weight is given, and the time complexity of the splitting ratio function is $O(n^2)$, as n is the number of nodes in the topology. Traffic demand in a hybrid SDN can be acquired through network monitoring approaches such as sFlow [156]. The algorithm operates in several phases. The (lines 1-2) compute the shortest paths for each node using the Bellman-Ford algorithm, which is particularly useful in networks where edge weights may change dynamically. The weights are updated adaptively during the computation based on the current network condition, such as traffic demand and link utilization. The shortest paths are stored in a Directed Acyclic Graph (DAG). Initially, we form a DAG by considering each node as a destination node. In this algorithm, DAG refers to the matrix containing all the feasible routes from a node to a particular node v. In creating a DAG, we begin by picking a node and identifying all the shortest paths to the node using the Bellman-Ford Shortest Path algorithm, ensuring no loops in the shortest routes tree. We add all the other edges representing the SDN node outbound links into the DAG; after constructing the DAG, we can easily direct the traffic needs in the route flow function. The DAG is refined based on heuristic criteria (lines 3-4). For example, outgoing links with low utilization or high capacity are prioritized when constructing the DAG.

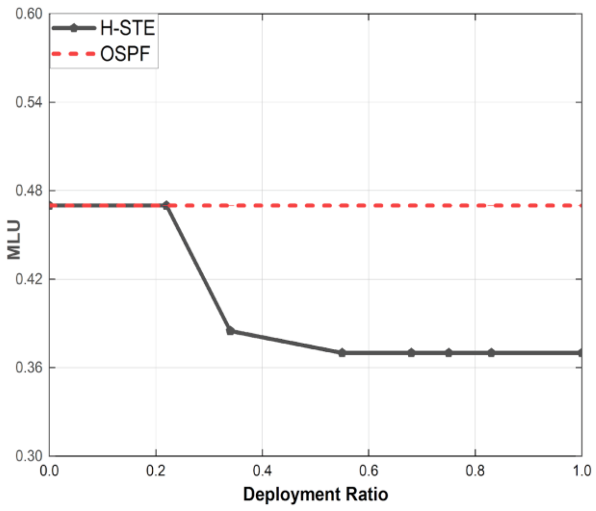
Lines (5-10) represent iterative flow distribution with adaptive adjustment: in this phase, the algorithm iteratively adjusts traffic flows across the network, routing traffic adaptively based on real-time link utilization data. We select a node based on the topological sorting sequence, such that all the flows traveling through the current node are computed before being allocated to the next hop; the flows are then routed to all outbound links in the DAG (5-6) To split the flow, we may utilize the hash functions to map the flows to distinct next hops; if the selected node is a legacy node, the flows are uniformly distributed throughout the shortest pathways and directed toward the following hops. This guarantees that all the flows from the selected node to node v are correctly routed (line 7). Each iteration solves a multi-commodity flow problem, adjusting constraints to better balance the load (line 8). We repeat the process this way and route the flow of each node in a topological sort order to obtain the linear constraints. And finally, we employ the linear programming solver CPLEX [153] to address the optimization problem (line 9). The algorithm stops iterating when improvement becomes negligible, ensuring computation efficiency. Line (14) presents the final output; once convergence is achieved, the algorithm outputs the final MLU, representing the most optimized traffic distribution during the iterative process.

4.5.4. Analysis of the algorithm

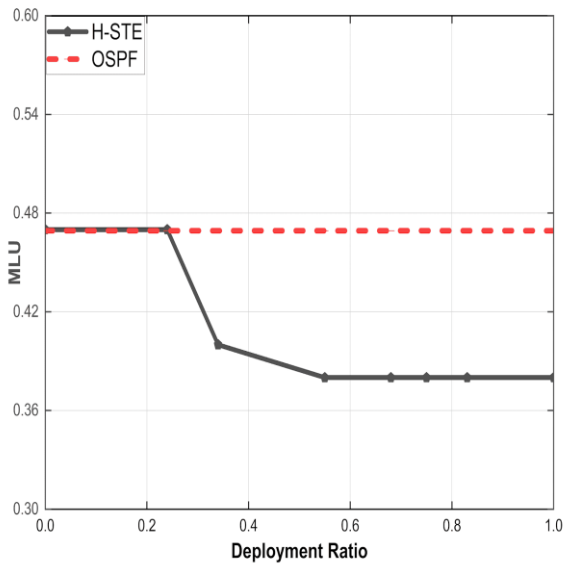
The splitting ratio with the Bellman-Ford algorithm provides a scalable and adaptive method for traffic engineering in a hybrid SDN environment. Three main factors shape the performance of this algorithm: the first one is the parallelized computation of shortest paths using the Bellman-Ford algorithm, which ensures efficiency and can handle large-scale networks. The Bellman-Ford algorithm performs very well in the network scenario with dynamically changing traffic demands, so it is the perfect option for handling dynamic changes in link weights. The second one is the heuristic-based DAG construction, which helps the algorithm focus on high capacity and minimum link utilization, ensuring the traffic is directed via the most effective paths, which improves the overall network load balance and minimizes network congestion. The last one is the multi-commodity flow optimization and iterative flow distribution, which allows the algorithm to improve the traffic distribution incrementally, and using an optimization solver (CPLEX) guarantees that the algorithm finds a globally optimized solution for the network. Solver CPLEX uses the dual simplex approach to solve linear programming problems, and its time complexity is in polynomial time. Considering the above algorithms, the overall time complexity of our algorithm is O(n^3).



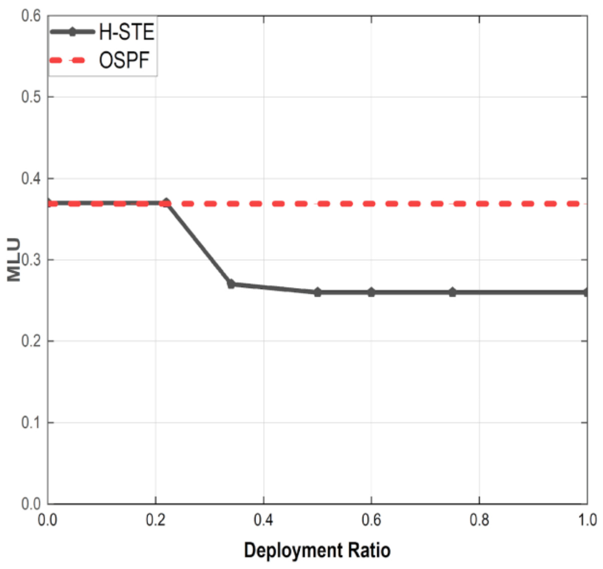
a. Abilene



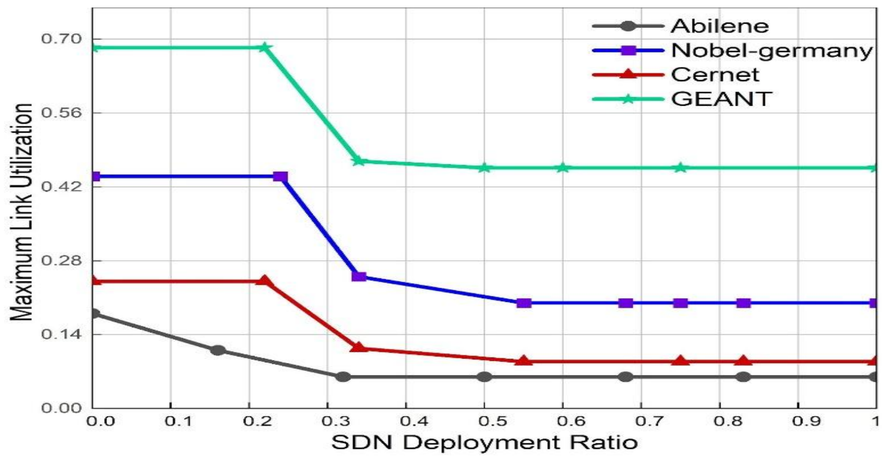
b. Cernet



c. Nobel-Germany



d. GEANT



(f) MLU using various ratios of SDN nodes

Figure 4.4. MLU with various SDN deployment ratios in real network topologies.

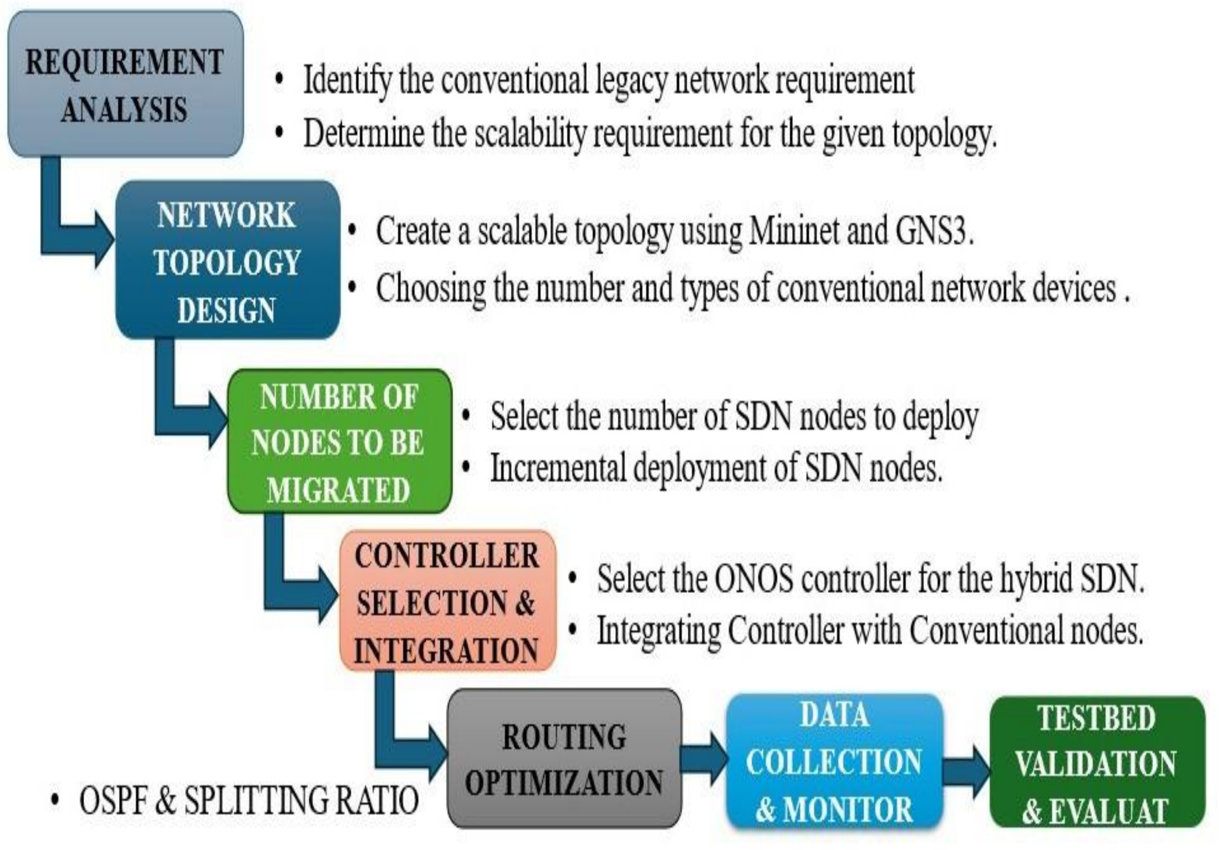

REQUIREMENT ANALYSIS
• Identify the conventional legacy network requirement
• Determine the scalability requirement for the given topology.
NETWORK TOPOLOGY DESIGN
• Create a scalable topology using Mininet and GNS3.
• Choosing the number and types of conventional network devices .
NUMBER OF NODES TO BE MIGRATED
• Select the number of SDN nodes to deploy
• Incremental deployment of SDN nodes.
CONTROLLER SELECTION & INTEGRATION
• Select the ONOS controller for the hybrid SDN.
• Integrating Controller with Conventional nodes.
• OSPF & SPLITTING RATIO
ROUTING OPTIMIZATION
DATA COLLECTION & MONITOR
TESTBED VALIDATION & EVALUAT

Figure 4.5. Workflow of routing optimization

4.6. Experiments and analysis

In this part, we demonstrate the performance of our proposed method on different real network datasets and topologies used in our experimental analysis; we execute the algorithms mentioned in the Algorithm Section and analyze their performance on different real network topologies. First, we will examine the environment setting, simulation topologies, and traffic datasets. Following that, we present the effectiveness of the proposed method H-STE on minimizing MLU and flow entries under different deployment ratios of SDN nodes.

4.6.1. Environment Setting

All the simulation experiments are simulated on a high-performance computer with a 3.20GHz Intel Core i7 with 16 GB of RAM, and the Mininet Simulator simulates the network environments [55]; for configuring legacy routers, we have used FRR (Free Range Routing) using VTY Shel to configure network protocols. The simulation experiments are coded with the C++ program to model the traffic routing procedure and to determine the splitting ratio of SDN nodes; we employ the CPLEX program in our C++ program. To connect different networks in the conventional network and to integrate the legacy network with SDN AS (Autonomous Systems), we have configured BGP (Border Gateway Protocol) inside all the legacy routers, as it is necessary to have a peering system that makes the BGP protocol work. and the integration between the legacy network and the SDN network is carried out through the SDN-IP. This program operates on top of the ONOS controller. The SDN-IP application allows SDN nodes to communicate with legacy routers using BGP protocols.

Selecting the controller in the hybrid SDN is another issue that needs to be considered; in this simulation experiment, the ONOS [56] controller is chosen to be the SDN controller, and the integration between the legacy network and the SDN network is carried out through the SDN-IP. This program operates on top of the ONOS controller. The SDN-IP application allows SDN nodes to communicate with legacy routers using BGP protocols. To illustrate the capability of our suggested approach, we analyzed our method on different real network topologies like Abilene, Cernet, Nobel-Germany, and Geant in our experiment.

To demonstrate the capability of our proposed approach for routing optimization, MLU, and flow entries minimization with minimum SDN deployment ratios, we have evaluated our experiment on different real network topologies (Abilene, Cernet, Nobel-Germany, and GEANT) and compared it to the following methodologies

OSPF [63]: The OSPF technique is a routing protocol primarily used in conventional legacy networks; it supports routing the flows based on the shortest path from the source to the destination to get load balanced. The shortest pathways are determined based on connection weights. To compare it with the heuristic algorithm H-STE, we set all the parameters the same as [63]; if there are multiple shortest paths, the flows are eventually split into the shortest paths using ECMP.

OSRO [29]: To thoroughly investigate and contrast the efficiency of SDN node optimization, we compare with the only splitting ratio of SDN nodes for TE performance; we compare and conduct experiments with static OSPF and optimization of only the splitting ratio of SDN devices.

RMC [71]: Rule multiplexing scheme studied the rule placement problem to minimize rule space occupation for multiple unicast sessions under QoS constraints. This method saves TCAM resources significantly and guarantees high QoS satisfaction. To efficiently use TCAM resources, the RMC method attempts to multiplex a set of routing rules deployed on nodes to reduce the use of flow entries

HPRS [157] formulated the routing optimization problem in a hybrid SDN with path cardinality constraints and proposed the H-permissible Path Routing Scheme, which effectively routes traffic flows under path cardinality constraints.

OORO [108] formulated the traffic engineering problem in hybrid SDN over multiple traffic matrices by mixing online splitting ratio optimization with offline weight setting optimization.

H-STE: We conducted experiments on different real network topologies to optimize the splitting ratio of SDN nodes and OSPF weight setting. All of the initial weight settings of the network are set to 1. The CDF curves of the MLU are shown in Figure 4.6. As shown in Figure 4.6, compared to OSPF and OSRO under different real network topologies, our method achieves minimum MLU; with a deployment ratio of 30% SDN nodes, we can get close to the optimal result and MLU in the network.

Table 4.2. Network Data Information

Network	**Nodes**	**Links**	**Time**	**Time-range**
Abilene	12	30	5 min	Six months
Cernet	14	32	5 min	One month
Nobel-Germany	17	26	5 min	One day
Geant	23	74	15 min	Four months

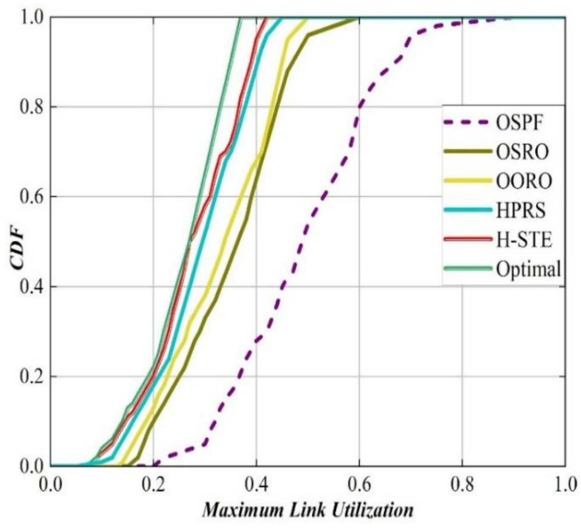
Table 4.3. Software/ tools Specification

Software/Tool	Version	Category	Purpose	Role
Mininet	v2.3.1B4	Network Emulator	Creation of Custom topologies	Emulated a scalable Hybrid SDN network.
ONOS	v2.2.0	SDN Controller	Traffic monitoring and Flow control via OF	Implemented to control traffic flows dynamically.
GNS3	v2.2.43	Legacy network Simulation	Configuration of conventional devices	Creating the conventional network topology
iPerf	v3.13	Traffic Generator	Performance testing	Simulated various traffic loads
Wireshark	V4.2.1	Packet analyzer	Monitoring and analyzing packet-level data	Verified packet flow and latency during simulations
Ubuntu OS	22.04 LTS	Operating System	Hosting the entire Hybrid SDN	Stable platform for Mininet & ONOS

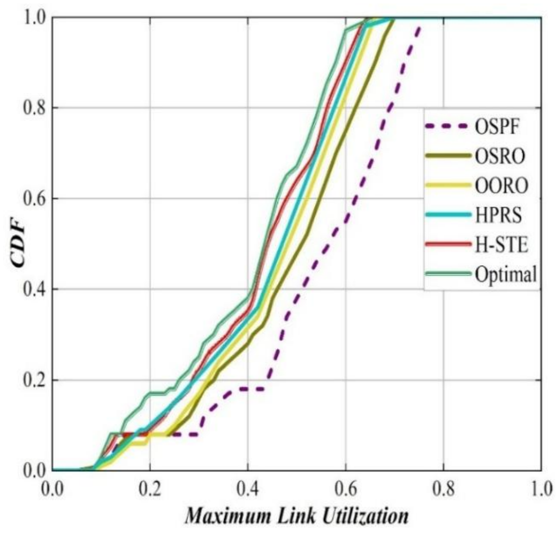
4.6.2. Datasets and Topologies

The dataset and network topologies used in this experiment are shown in Table 4.2; the network topologies datasets are available on SDNlib [154]. The Abilene topology, part of the American Research and Education Network, comprises 12 nodes and 30 links. The Cernet topology, belonging to the China education and research network, consists of 14 nodes and 32 links. The Noble-Germany network topology consists of 17 nodes and 26 links. Lastly, the GEANT topology, part of the European research and education network, consists of 23 nodes and 74 links. The traffic datasets required in the experiment analysis are real network datasets measured or created in the specific topologies every five or fifteen minutes.

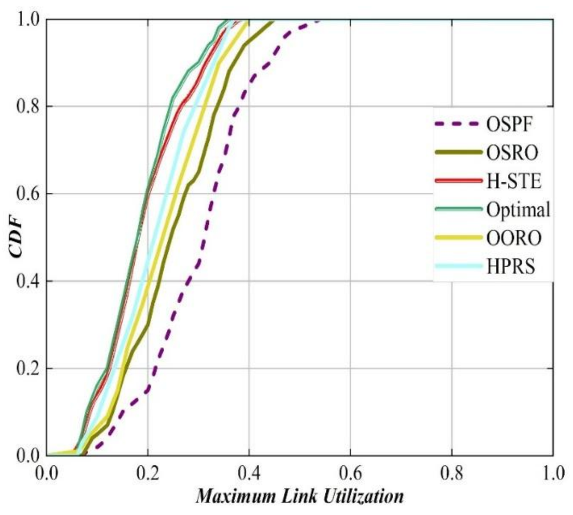
For the Abilene topology, the total traffic provided by the TOTEM project [158] was measured from the first of March 2004 to September 2004, and every 24 hours, snapshots were created at 5-minute intervals. For the Cernet topology, the datasets are provided by Zhang et al. [159]. They were measured every five minutes from February 19, 2013, to March 26, 2013. For the GEANT topology datasets provided by Uhling [160], the traffic matrices were measured every 15 minutes from 04-May-2005 to 31-Aug-2005.

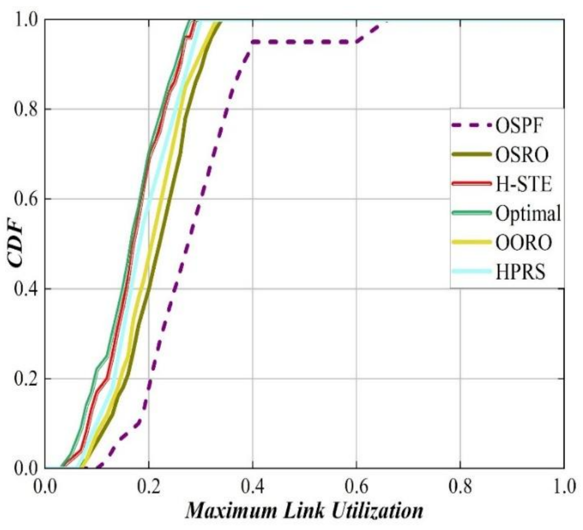


a. Abilene



b. Cernet





c. Nobel-Germany **d. Geant**

Figure 4.6. MLU CDF curve under real network topologies

4.6.3. Evaluation of Network Performance

4.6.3.1. MLU

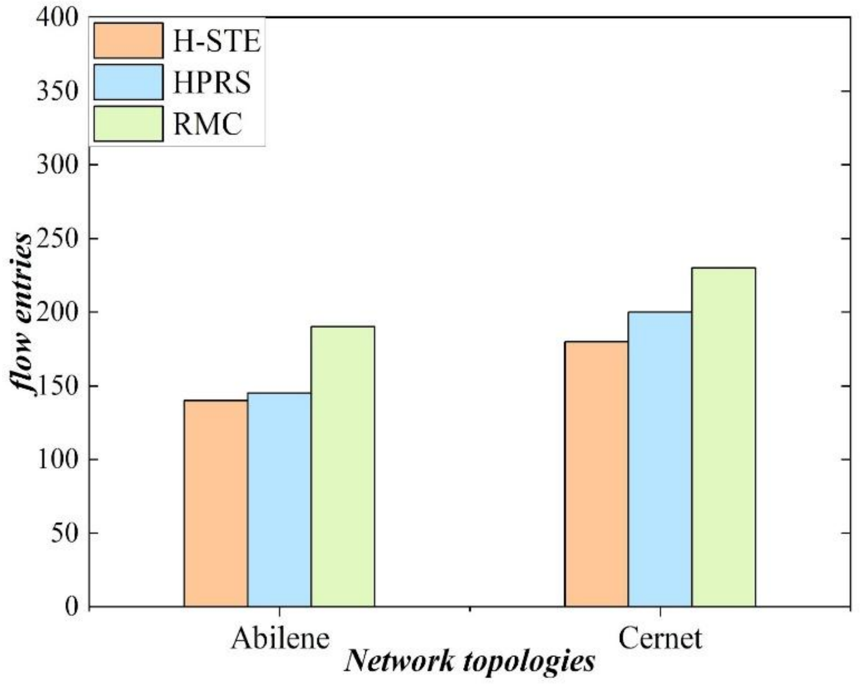
In a hybrid SDN network, the SDN deployment ratio is defined as the ratio of SDN nodes throughout the network. Minimizing the network's MLU is a critical parameter in the hybrid SDN when considering network load balance. The MLU of each link can be calculated as the maximum ratio of the link load divided by the link capacity. The primary objective is to minimize the MLU with minimum SDN nodes while maintaining the network's load balance to enhance performance. Maintaining network load balance in a hybrid SDN refers to the equitable distribution of traffic across all network paths to ensure efficient utilization of network resources and minimize congestion on any single link. This balance helps to prevent the situation in which some links exceed their capacity (overloaded), leading to congestion and inefficient traffic handling, while other links remain unutilized or underutilized. By balancing the load, the network can provide consistent performance, minimize latency, and improve reliability.

The H-STE method helps bridge the gap between both networks by optimizing the overall network performance while adhering to the limitations imposed by traditional routing protocols. This method adjusts the splitting ratio of SDN nodes to reduce network congestion while having a multi-objective focus, primarily minimizing MLU and considering other network performance metrics. Various methods and metrics can evaluate the load balance in the hybrid SDN environment, such as load variance, link utilization distribution, MLU, failover capacity, and path diversity. The most crucial metric for evaluating load balance in the hybrid SDN is MLU; in this study, the network load balance is evaluated by considering the MLU. It represents the utilization percentage of the link with the most capacity or overloaded, and with zero capacity or underutilized links in the network. A high MLU indicates that particular links are approaching their maximum capacity, which causes network congestion and inefficient traffic handling. Minimizing MLU is critical for load balancing because it helps prevent overload on any single link; with minimum MLU, traffic load can be distributed more efficiently across the network.

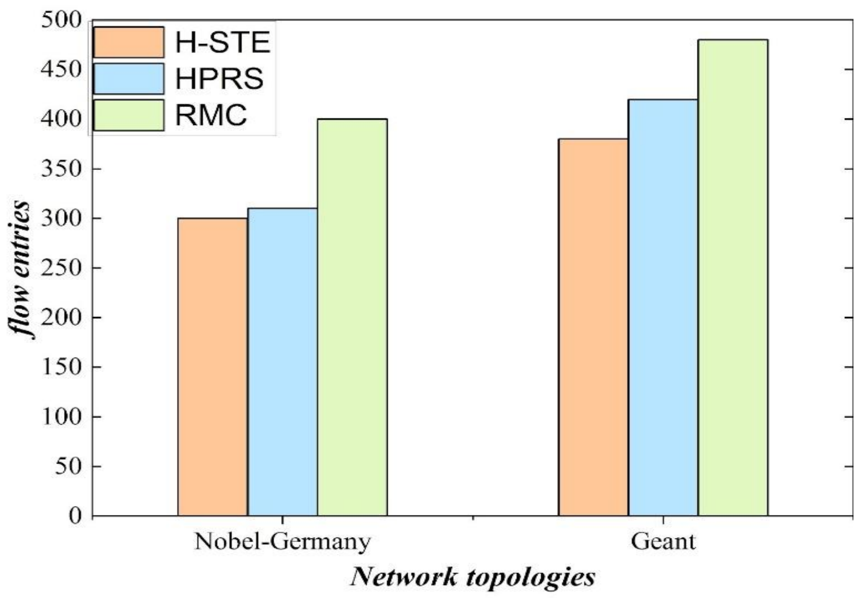
Our main objective is to minimize the MLU with the minimum number of SDN nodes while maintaining the network's load balance. To minimize the MLU, we increase the deployment ratios; when we deploy 0.3, we can see an improvement in MLU as the MLU of different topologies begins to stabilize. We find that the value of MLU varies and directly depends on the increase in SDN deployment ratios; we plot the curves to show that the value of MLU becomes smaller and decreases with the increase in the SDN node deployment ratio under different network topologies. Figure 4.4 depicts the effect of the SDN nodes deployment ratio on improving the MLU under different network topologies. The result shows that the improvement of MLU directly depends on the increase of SDN deployment ratios; we can see a rapid decrease in the value of MLU with a shift from 0 to 0.3 in the SDN deployment ratios. As soon as the deployment ratio surpasses 0.3, the MLU begins to stabilize, and the curve of H-STE becomes flat.

The findings indicate that increasing the deployment ratios of SDN can obtain near-optimal outcomes in optimizing MLU comprehensively. Increasing the number of SDN nodes can boost the ability to split and direct the traffic flows in a hybrid SDN, as the main goal is to reduce the MLU to prevent network congestion and evenly distribute the network load to enhance the overall network performance. In Figure 4.4, we can see that the OSPF curve remains steady and flat with decreasing or increasing SDN deployment ratios; the reason is that OSPF routes traffic based on the shortest path from the source to the destination. Figure 4.4 (f) shows the impact of varying SDN deployment ratios on improving MLU. As shown in the Figure, when the deployment ratio increases, the value of MLU becomes smaller. The optimal splitting ratios are obtained using the CPLEX Solver based on the optimization model. The finding indicates that as the deployment ratio increases, MLU improves correspondingly, and the MLU stabilizes as the deployment ratio reaches approximately 30% for different network topologies.

We consider and set the SDN deployment ratios to 30% based on the experimental analysis of different network topologies. As the MLU starts to stabilize, we can see that the curves are getting near-optimal results. To show the priority and the efficiency of our proposed strategy for routing optimization and enhancing TE performance, we compared our suggested H-STE strategy with different strategies under different real network topologies, as shown in Table 4.2. To compare it with the only optimization of OSPF weight setting [63], we set all the parameters the same as [63]. We compared it with the optimization of only the splitting ratio of the SDN node with a fixed link weight setting [29], performed multiple experiments, and compared it with different approaches, and drew the average MLU result in CDF graphs under different topologies in Figure 4.6.



(a) Abilene & Cernet topologies



(b) Nobel & Geant topologies

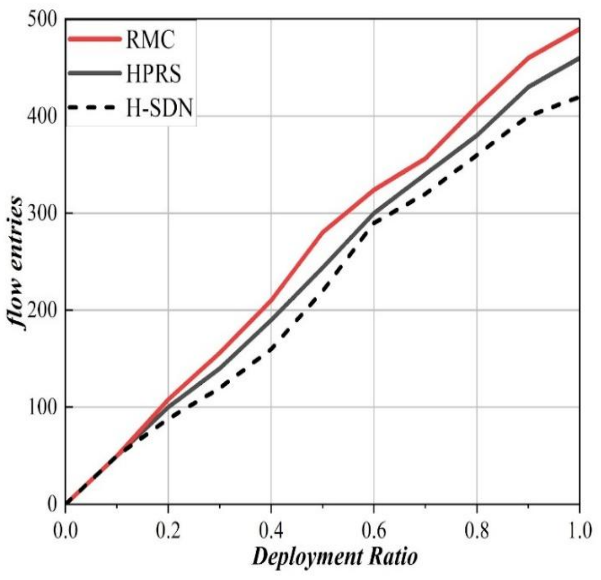
Figure 4.7. Flow entries under various network topologies

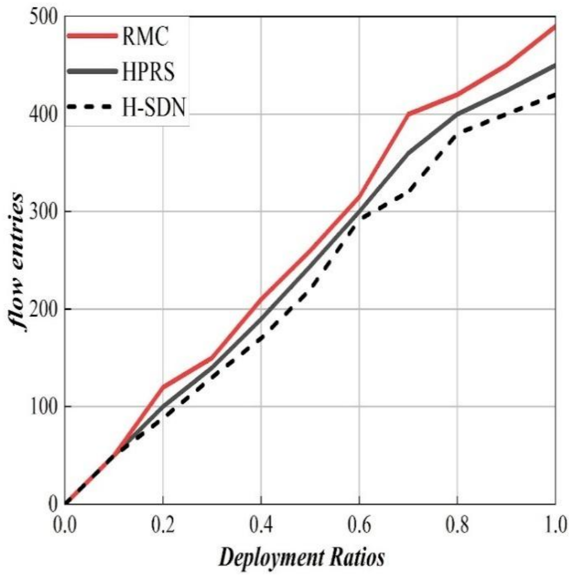
Table 4.4. Average MLU under Different Network Topologies

Name	**OSPF**	**OORO**	**H-STE**
Abilene	0.084 (15.07)	0.073	0.053 (-27.40%)
Cernet	0.287 (10.38%)	0.260	0.225 (-13.46%)
Nobel-Germany	0.214 (9.18%)	0.196	0.157 (-19.89%)
Geant	0.174 (6.10%)	0.164	0.098 (-40.24)
Name	**OSPF**	**HPRS**	**H-STE**
Abilene	0.495(15.12%)	0.430	0.422(-1.86%)
Cernet	0.605(8.03%)	0.560	0.557(-0.54%)
Nobel-Germany	0.189(43.20%)	0.132	0.120(-9.09%)
GEANT	1.970(43.80%)	0.137	0.115 (-12.88%)
Name	**OSPF**	**RMC**	**H-STE**
Abilene	0.062 (14.5%)	0.055 (3.6%)	0.053
Cernet	0.26(15.4%)	0.24(8.3%)	0.22
Nobel-Germany	0.18 (19.5%)	0.15 (10.7%)	0.11
GEANT	0.11(18.2%)	0.097(7.2%)	0.090

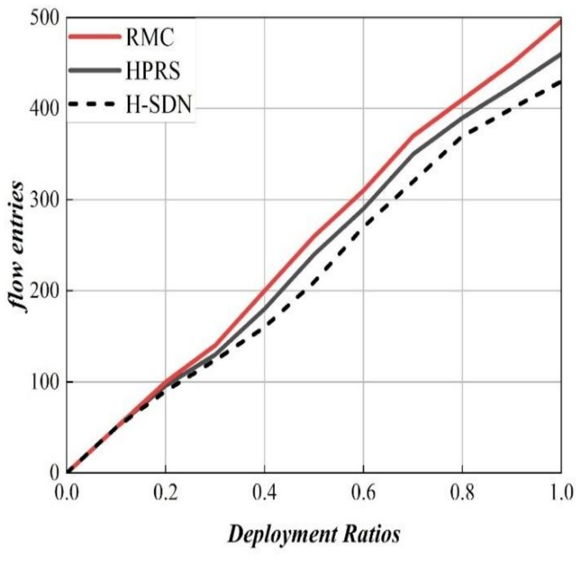
In hybrid SDN, reducing the MLU and optimizing routing present significant challenges. To minimize the MLU, we increase the deployment ratios, and we can see that with a 30% deployment ratio of SDN, the MLU of different topologies begins to stabilize. The findings indicate that increasing the deployment ratios of SDN can obtain near-optimal outcomes in optimizing MLU comprehensively. More SDN nodes can bring more flexibility to split and forward the traffic flows in a hybrid SDN, as our common goal is to minimize the MLU while balancing the network load to enhance the overall network performance. As a result, compared to other optimization methods, the H-STE method obtains a much lower MLU under different network topologies.

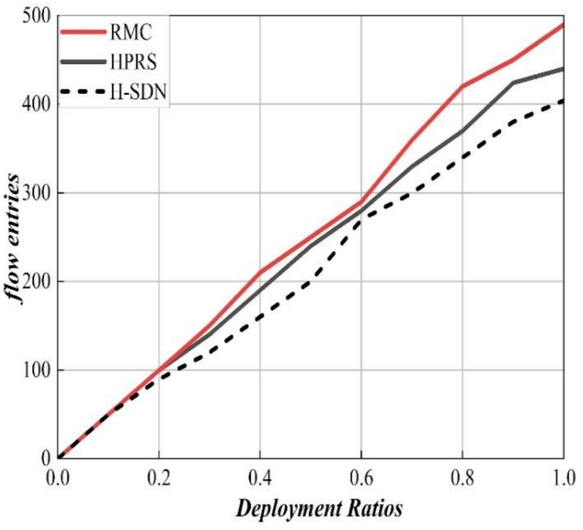
Minimizing MLU is directly dependent on the increasing deployment ratio of SDN nodes; with a 30% deployment ratio of SDN nodes, we can reduce the MLU and get close to the optimal result to get maximum benefits from the concept of a hybrid SDN network. As shown in Figure 4.6, when the deployment ratio increases, the value of MLU becomes smaller. Compared to other methods, the H-STE method is close to the optimal result, as this method optimizes both the SDN node splitting ratio and OSPF weight setting. In contrast, the OSPF method only optimizes the OSPF weight setting, and the OSRO method optimizes only the splitting ratio of SDN nodes with a fixed length weight.





a. Abilene **b. Cernet**





c. Nobel-Germany **d. Geant**

Figure 4.8. Flow entries under various SDN deployment ratios

A 30% deployment of SDN nodes in a real-world hybrid SDN network means that only a portion of the network is managed through SDN principles. At the same time, the rest operate using conventional distributed network protocols. The finding is insightful and suggests a balance point where SDN control is applied to control traffic without heavily depending on SDN resources. It brings several practical implications across different network dimensions, such as scalability, cost-effectiveness, operational complexity, and network performance; it can significantly reduce hardware replacement costs. Depending on the organization's requirements, they can incrementally migrate to a hybrid SDN and then to a fully SDN network. However, the result may vary under different network conditions and configurations. The deployment of 30% of SDN nodes offers a scalable and cost-effective migration toward full SDN; its effectiveness depends on how the SDN nodes are deployed strategically. They can offer significant advantages regarding flexibility and TE when deployed in high-impact areas. However, poor placement could lead to more complex systems with few performance advantages.

4.6.3.2. Flow entries

In the hybrid SDN environment, minimizing flow entries is crucial to maximizing resource utilization and enhancing network efficiency, particularly in situations with limited bandwidth and potential performance issues due to network congestion. Our method effectively controls flow entries, enhances the data transmission process, and decreases the costs of maintaining extensive routing tables. This capability is particularly helpful for network environments that require a flexible and dynamic management approach to accommodate different application demands and fluctuating traffic volumes. With digital resources and online platforms continuing to expand, minimizing flow entries can enhance network stability and dependability. It is essential to provide low latency and seamless connectivity. We demonstrate the stability and scalability of our method in adapting to different SDN deployment ratios while consistently outperforming alternative methods.

In this study, our objective is to minimize flow entries with the minimum deployment ratios. We have compared the proposed H-STE method with previous routing optimization approaches in the hybrid SDN environment. Our method demonstrates superior performance in minimizing the number of flow entries under different real network topologies. Figure 4.7 illustrates the total flow entries employed in different methods under various real network topologies.

As evident in Figure 4.7, as the topological scale increases, the number of flow entries for H-STE rises due to the increased potential for minimizing flow entries with larger topological scales. Compared to the previous routing algorithm, on average, the flow entries utilized in H-STE decrease by 10.9% and 25.6% for the given network topologies. This reduction can be attributed to H-STE's efficient flow management strategy, which reduces flow entry redundancy and optimizes the routing path, enabling networks to function effectively with fewer entries while maintaining performance. Furthermore, the scalability of our approach ensures that the method maintains its efficiency as the topology expands, demonstrating its resilience in controlling diverse network topologies. The comparisons indicate that our method outperforms others in minimizing flow entries under different network topologies. This is a significant improvement as it reduces resource utilization while enhancing network efficiency. Considering these factors, our method can optimize routing decisions more effectively, ensuring the network functions optimally within its capacity limitations while meeting performance requirements.

The flow entries vary with different SDN deployment ratios under different network topologies. To better understand and evaluate the necessary flow entries under various deployment ratios, we generated curves demonstrating how the number of flow entries varies with different deployment ratios of SDN nodes. As illustrated in Figure 4.8, with changing SDN deployment ratios, we can observe trends and patterns of flow entry utilization; this graphical representation can be utilized to determine the optimal deployment ratios for minimizing flow entries while maintaining efficient network performance. As the SDN deployment ratios increase, flow entries also increase for the given topologies. This trend emphasizes the dynamic nature of network traffic and the associated requirement for flow entries, considering more SDN nodes in the network.

For minimizing flow entries in the hybrid SDN, our method more effectively manages flow entries compared to previous methods. For instance, considering the Abilene network topology and taking an average across different SDN deployment ratios, H-STE minimizes the flow entries by 6.8% compared to HPRS and 16.3% compared to the RMC method. This reduction demonstrates the efficacy of our method in optimizing resource utilization and minimizing unnecessary flow entries while balancing the network load. Similarly, in the Cernet topology, the H-STE method demonstrates a notable decrease in flow entries of 6.1% compared to HPRS and 15.3% compared to RMC, which highlights the benefits of this approach on different network topologies. The same phenomenon is observed in the Nobel-Germany topology; the H-STE approach demonstrates a significant decrease in the flow entries compared to HPRS and RMC by 6.5% and 17.1%, respectively.

The ability of H-STE to modify and reduce flow entries demonstrates further potential for network scalability and efficiency. Lastly, the H-STE approach outperforms the other methods in the GEANT topology, with an average decrease in the flow entries of 10.4% compared to HPRS and a substantially larger decrease of 23.6% compared to RMC on average at various SDN deployment ratios. This significant reduction demonstrates the efficacy of H-STE in handling and routing traffic in complex network environments, improving overall network performance and reducing overhead. The findings conclusively demonstrate that H-STE performs significantly better than other approaches and efficiently controls flow entries. Consequently, H-STE is a viable option for expediting SDN deployment ratios in different network topologies, ultimately enhancing resource management and network performance. The experimental results indicated that the proposed H-STE approach outperformed the other approaches in modifying and decreasing the number of used flow entries across various SDN deployment ratios within different network topologies. Future research studies could explore the long-term implications of these findings and examine the adaptability of H-STE in other traffic patterns and network topologies with large-scale network topologies.

4.7. Summary

Traffic engineering in hybrid SDN networks is one of the open areas for researchers; with the combination of centralized SDN and decentralized traditional networks, it is much more challenging to optimize traffic flows, resource utilization, etc. This research contributes to the expanding discussion of TE and routing optimization in the hybrid SDN, necessitating innovative solutions to enhance network efficiency, performance, and resource utilization. This chapter studied traffic engineering in a migrated hybrid SDN where SDN nodes are incrementally deployed to the conventional network. We introduced a heuristic technique, H-STE, to improve traffic engineering in the hybrid SDN; we concentrated on minimizing the MLU by optimizing two critical aspects: optimizing the OSPF weight settings across the entire network to balance the flows originating from conventional devices and optimizing the traffic splitting ratio of SDN nodes. As these devices are closely interconnected, optimizing both aspects presents significant challenges for effective traffic management and ensuring a balanced load across the network. The H-STE method helps bridge the gap between the two by optimizing the whole network performance and adhering to the limitations imposed by legacy routing infrastructure. We have conducted several experiments on real network datasets; the results demonstrate that with a 30% deployment ratio of SDN nodes, we can reduce the MLU and get close to the optimal result to get maximum benefits from the concept of a hybrid SDN network. The study findings contribute to the progress and development of efficient, flexible, and robust communication infrastructures as we strive to enhance network management and design. For adaptability and scalability, future research could explore integrating hybrid SDN with the expansion of 5G networks to adjust to diverse environments, including multi-access edge computing platforms and IoT devices.

CHAPTER-5

Interconnection of SDN nodes with the conventional network using the ONOS controller

For future network design, different architectures worldwide have been proposed; among them, one of the finest network designs is Software-Defined Networking (SDN), which separates the network control and data layers and enables the network with more agility, flexibility, and programmability. However, because of economic and technical challenges, directly upgrading the entire conventional network to SDN is challenging for many organizations. The optimal solution to get the advantages of both networks is to incrementally upgrade conventional network devices to SDN nodes, called the Hybrid SDN network. In the hybrid SDN network, SDN nodes interconnect with the conventional legacy network, taking advantage of both network designs. In this study, we studied the integration of SDN networks with conventional legacy networks. We explore how to peer and integrate SDN nodes with conventional networks using the SDN-IP application located on top of the application layer of the ONOS controller. Through different experiments on both networks, IP-IP and IP-SDN, we found that compared to IP-IP networks, IP-SDN improves the overall network performance and provides more remarkable performance in the maximization of throughput and the minimization of delay.

5.1. Introduction

A software-defined network (SDN) is an emerging network design that manages complex network infrastructure by separating the data layer and control layers. SDNs are employed in almost every environment to facilitate efficient communication among computer networks, including wireless networks, cloud computing, mobile telephones, data centers, and the Internet of Things. SDN has many features that provide agility and flexibility to the network. The basic idea behind SDN is to separate network layers to make the network more agile and flexible. The primary advantage of SDN is the ability to manage the data flow dynamically and efficiently. After a flow passes through an SDN switch, its forwarding direction may be easily adjusted; we refer to this type of traffic as programmable traffic. For large-scale networks, it is necessary to develop a network that can be programmed to meet the need for innovation in network access; SDN technology simplifies network administration and control operations through the use of programmable devices and automation [7].

The SDN architecture has three main layers: the data, control, and application layers. The application layer hosts network applications designed to communicate with data layer devices. The control layer, which is the main layer of the SDN network, is mainly responsible for controlling the data layer, and the data layer, which hosts network elements responsible for deploying packet-forwarding flows. In the control layer of the SDN network, there should be at least one controller; we can have multiple controllers to avoid a single point of failure. For the Wide-area network in different locations, multiple controllers can be deployed physically to accomplish the purpose of a centralized control plane. Figure 5.1 shows the SDN basic architecture, consisting of three main layers: the Data, Control, and Application.

Recently, SDN has been widely researched and progressively implemented in several types of networks, including campus networks [20], data centers [43], wide area networks [5], enterprise networks [161], and internet exchange points [44]

With all the advantages and features that an SDN network provides, directly migrating to a fully SDN network is challenging because of technical and economic challenges. Hence, migrating to a hybrid SDN is the best solution, but there are challenges in integrating SDN capabilities with the traditional IP network. Due to cost and operational factors, SDN technology is mainly implemented incrementally alongside traditional network devices. Namely, with each network upgrade, only a certain number of existing network devices will be upgraded with programmable Switches. Internet Service Providers are incrementally upgrading traditional network devices to programmable switches that support SDN to improve traffic management ability.

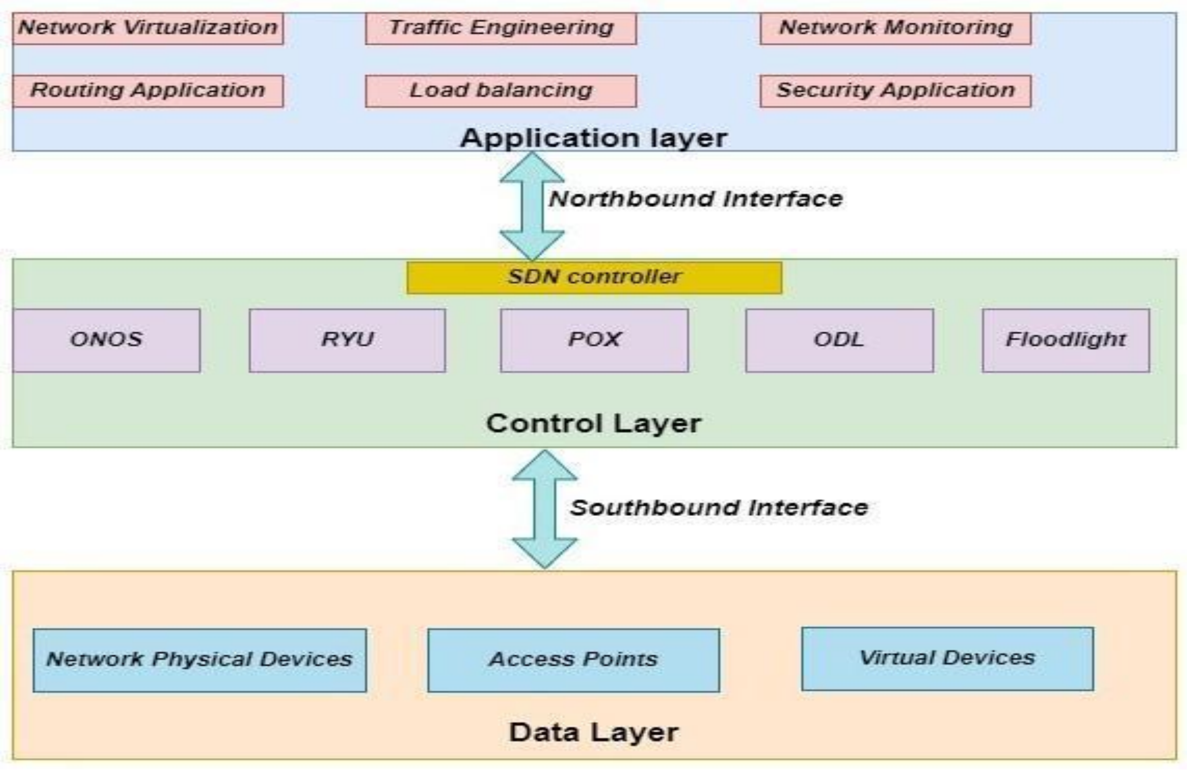


Figure 5.1. SDN architecture

This coexistence of SDN alongside traditional networks gives rise to the hybrid SDN. Current hybrid SDNs fail to address the possible performance drawbacks caused by a centralized SDN controller. Deploying a proper controller is another issue that may cause significant delays in processing flow requests. Because of these factors, many researchers have proposed the incremental deployment of SDN nodes alongside traditional network devices, such as traffic engineering [162], flexible routing [30], link failure recovery [31], power saving [163], and safe updates [33].

One of the main elements of an SDN network is the controller; therefore, the controller must present features that ensure service continuity during failures and improve the overall network performance. Open Networking Operating System (ONOS) and OpenDaylight controllers usually use three controllers to provide resilient service at one location and to guarantee network consistency. All the controllers communicate with each other. ONOS is an open-source controller with an application for integrating SDN networks with conventional legacy networks via BGP, that is, SDN-IP, located at the top of the application layer of the ONOS controller.

With the evolution of SDN and the division of control and data layers, logical centralization of the control layer, SDN has a practical and flexible way of optimizing routing. As the Controller Has Control of flow routing in the control plane, traffic may be dynamically separated and sent to multiple pathways using SDN switches. Compared to the ECMP used in the standard OSPF protocol, this traffic routing gives a more flexible approach to optimizing routing.

In a pure SDN network, programmability and global centralized network activity control are efficient aspects that allow traffic engineering. Because the flow splitting ratio on SDN switches is variable, using specific complete TE methods in a hybrid SDN network is feasible. To improve the profitability of centralized control, researchers must select a suitable migration sequence, which includes numerous optimization strategies. These approaches may be applied not just to hybrid network deployment but also to other network optimization techniques. For example, migration solutions may expand to include: (1)- identifying a suitable placement for the middlebox in the legacy network to get perfect network control, (2)- for energy requirements, wireless networks must address the issue of AP location. (3)- Enhance data center utilization by resolving the VM replacement problem.

In traditional networks, the OSPF protocol is one of the most used IGP protocols; each link is given a cost or weight, and uses the shortest path with minimum cost forwarding. Traffic is routed between the source and destination address and distributed equally when multiple shortest paths are encountered using Equal Cost Multipath forwarding. Because all nodes in a hybrid SDN network do not support OpenFlow, flow abstraction, complete packet matching, or packet filtering techniques, TE applies a variety of optimization tactics to get the optimum network performance characteristics, etc. In the hybrid SDN network, it is feasible to use particularly extensive Traffic Engineering methods because the flow splitting ratio on the SDN device is arbitrary. To optimize the profitability of centralized control, researchers must select an appropriate migration sequence comprising numerous optimization strategies. Traffic Engineering is more challenging in a Hybrid SDN network; solutions based on static routes or ACLS (access control lists) provide restricted functionality [61].

Traffic measurement entails gathering, measuring, and monitoring network status data. Traffic measurement is difficult in the hybrid SDN network because only a small percentage of devices are SDN-enabled. Traffic management is the study of controlling and arranging network traffic based on network status data provided by traffic measurement.

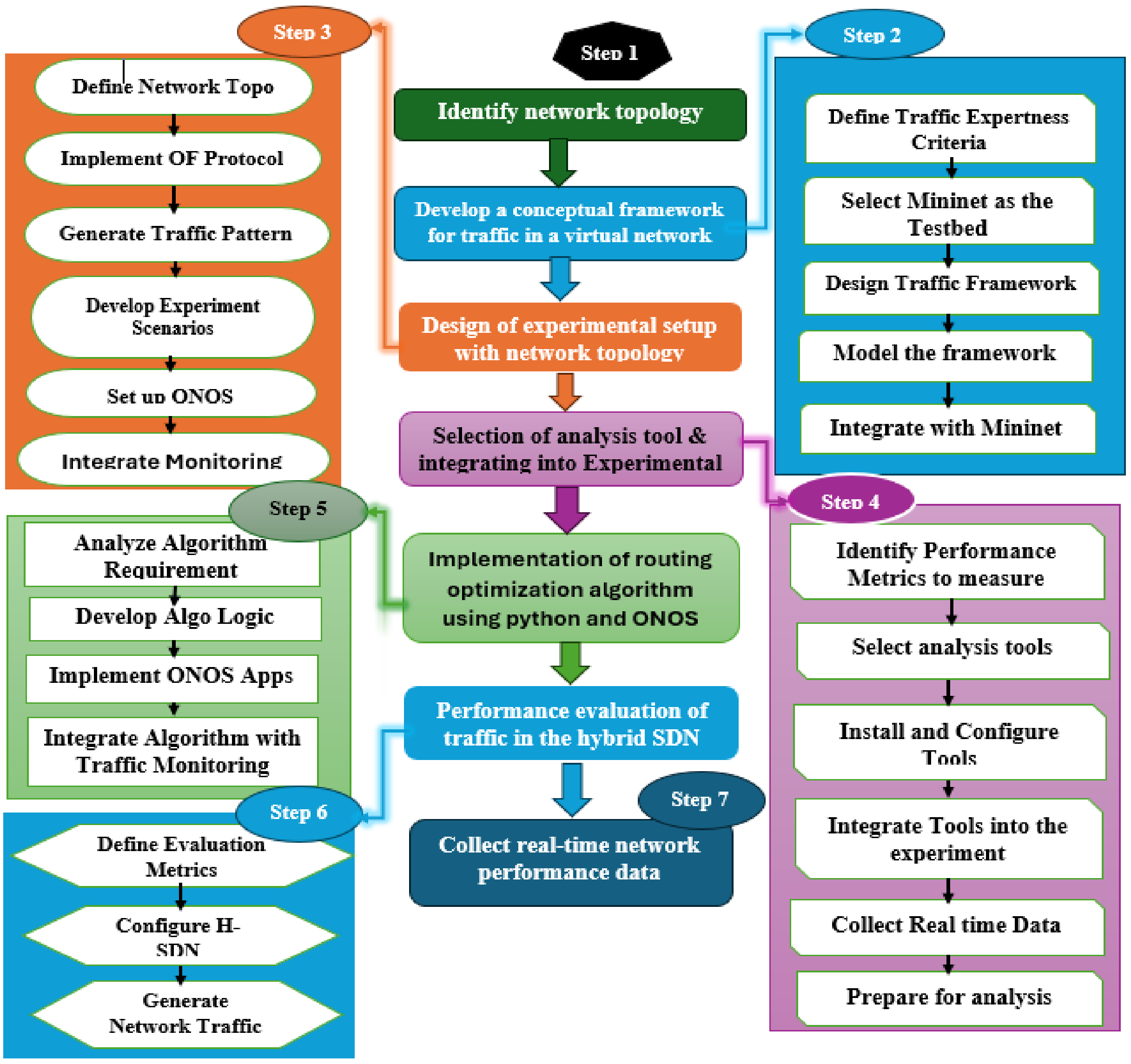

Step 1
Identify network topology
Develop a conceptual framework for traffic in a virtual network
Design of experimental setup with network topology
Selection of analysis tool & integrating into Experimental
Implementation of routing optimization algorithm using python and ONOS
Performance evaluation of traffic in the hybrid SDN
Collect real-time network performance data
Step 2
Define Traffic Expertness Criteria
Select Mininet as the Testbed
Design Traffic Framework
Model the framework
Integrate with Mininet
Step 3
Define Network Topo
Implement OF Protocol
Generate Traffic Pattern
Develop Experiment Scenarios
Set up ONOS
Integrate Monitoring
Step 4
Identify Performance Metrics to measure
Select analysis tools
Install and Configure Tools
Integrate Tools into the experiment
Collect Real time Data
Prepare for analysis
Step 5
Analyze Algorithm Requirement
Develop Algo Logic
Implement ONOS Apps
Integrate Algorithm with Traffic Monitoring
Step 6
Define Evaluation Metrics
Configure H-SDN
Generate Network Traffic
Step 7

Figure 5.2. Work Flow of hybrid SDN traffic analysis

5.2. Proposed Approach

This chapter investigates the integration of SDN networks with conventional legacy networks. We explore how to peer and integrate SDN with legacy networks using the SDN-IP application located on top of the application layer of the ONOS controller. This application allows the SDN network to communicate with conventional network devices through BGP and explains how the controller converts the BGP information into OpenFlow entries. The Internet is a collection of autonomous systems (ASs) or interconnected networks. An AS is a group of connected networks under a single domain. Conventional networks depend on routing protocols to share and interconnect routing information.

On the other hand, on the internet, BGP is the standard EGP protocol designed to exchange routing information among ASs. BGPv4 is the most generally deployed mechanism for peering ASs on the Internet. So, a peering mechanism is necessary to integrate conventional networks with SDN-driven ASs; this integration is performed through an SDN-IP application, which runs on top of the SDN controller.

Nevertheless, none of those, as mentioned in earlier studies, interconnect SDN nodes with conventional legacy networks using ONOS controllers in a real network scenario. We have considered this issue as the interconnection of SDN nodes into conventional legacy networks with ONOS controllers using Mininet tools.

5.3. Quality of Services (QoS) in Hybrid SDN

Overall, suppose we define Quality of Service for a network. In that case, user satisfaction with specific services is evaluated using a composite factor, including bandwidth, delay, loss, and jitter. All technologies that manage data flow inside a network to reduce packet loss, latency, and jitter are called quality of service (QoS) technologies. Providing specific data transmission in a particular network aspect is what we refer to as quality of service.

Considerations for quality of service include data accuracy, packet loss, bandwidth, and service level. For instance, QoS ensures bandwidth delivery for bandwidth-sensitive data if the network has exceptional bandwidth; if we have an ideal link, QoS is unnecessary. QoS denotes low delay if the network provides low-latency transmission service for time-sensitive data delivery. Since application flows compete for available network resources, they must be differentiated to achieve QoS. Network resources must be assigned to make the best decision for packet forwarding; occasionally, this process necessitates knowledge of the present network conditions. With a broad dynamic range of Quality-of-Service features, the architecture supports various communication applications, including those for people and traditional services [164].

In a network, sufficient bandwidth, low latency, low packet loss rates, and balanced network connection loads are all considered aspects of quality of service. As is well known, adjusting the network's link load can result in higher bandwidth use and lower latency. Consequently, the latency and bandwidth QoS requirements can be met by balancing the connection load demand.

Service Level Agreements (SLAs) between service providers and end users largely dictate service quality. Although this approach performs exceptionally well for best-effort service, it does not offer more precise granular traffic management. Specific applications, like VoIP (Voice over Internet Protocol), online gaming, video conferencing, and many more, have flows sensitive to jitter, latency, and bandwidth, so they require unique handling methods. Setting high-level traffic management strategies and establishing limits on the depth of traffic distinction are not well-established methods.

SDN networks separate data and control layers in a network. This section gives network controllers more authority over networks. Controllers can view the status of the network globally. The network controller offers a conceptual network perspective, providing network applications under the SDN idea. They are not needed to handle data plane device low-level configuration.

5.3.1. QoS requirements

Quality of service parameter in a network is classified into two categories: policies and metrics, as shown in Figure 5.3.

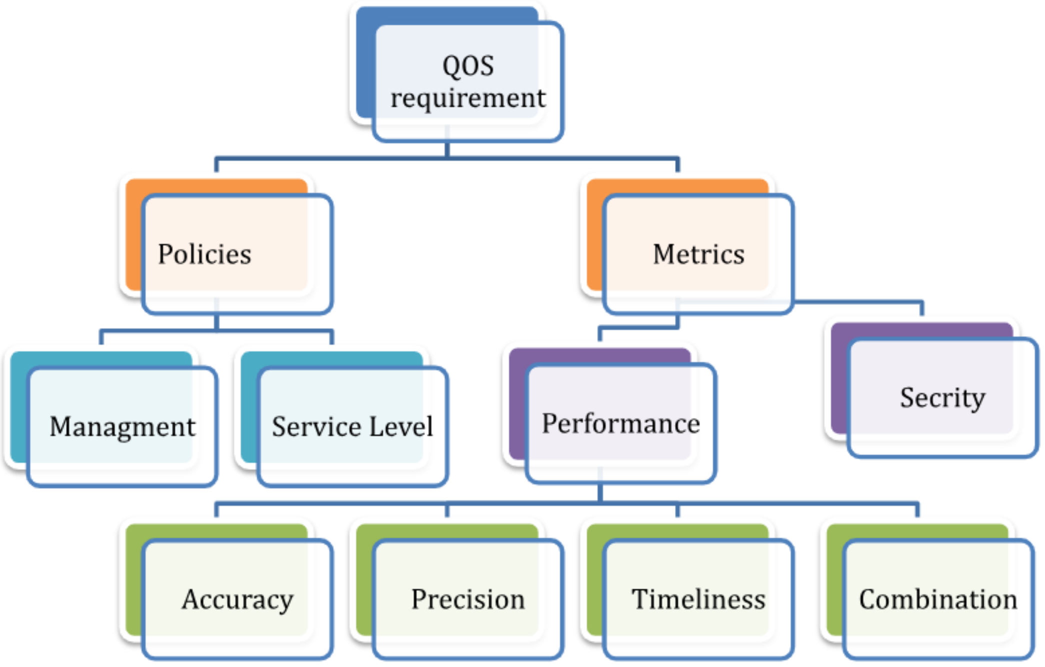


Figure 5.3. QoS requirement

Quality of service measurements include security, performance, cost, and reliability, which are used to establish QoS characteristics. The primary performance measures are latency, loss, and data rates. Rarely, as traffic increases, the network load needs to be balanced; nevertheless, controlling network loads is challenging. To provide a network management tool for managing network bandwidth, QoS regulations are essential. There are many advantages of QoS policies, including performance, security, and manageability.

The SDN network model's benefits may improve service quality in many network operations. SDN provides a global view of the network, making it possible to preserve relevant states over the whole course of flows. It can also monitor network statistics at other levels, such as port, device, and flow. Furthermore, SDN may help network operators create an automated, robust, and user-friendly QoS management framework for their networks by enabling resource reservation, packet scheduling, and queue management. Because network resources are dynamic, control methods for QoS provisioning in network applications must be well-defined.

5.3.2. QoS observation

The hybrid network is divided into domains by SDN switches, establishing connections between each domain. Various QoS metrics, including packet loss rate, traffic load, throughput, latency, and so on, must first be gathered to enable QoS-aware forwarding in a hybrid network. After that, these variables are merged to depict network performance precisely.

Traffic load monitoring: The connections to SDN devices may provide load information to the OpenFlow protocol. For a directed link, we have to figure out how many data flows are going through the switch in a certain amount of time. The traffic load on the link can then be determined by computing the number of data flows per unit of time. Flow statistics data is kept in the flow table on an SDN switch. The source/destination IP address, output port, input port, and other matching details for a given routing path are contained in each matching field of a flow entry that corresponds to that path.

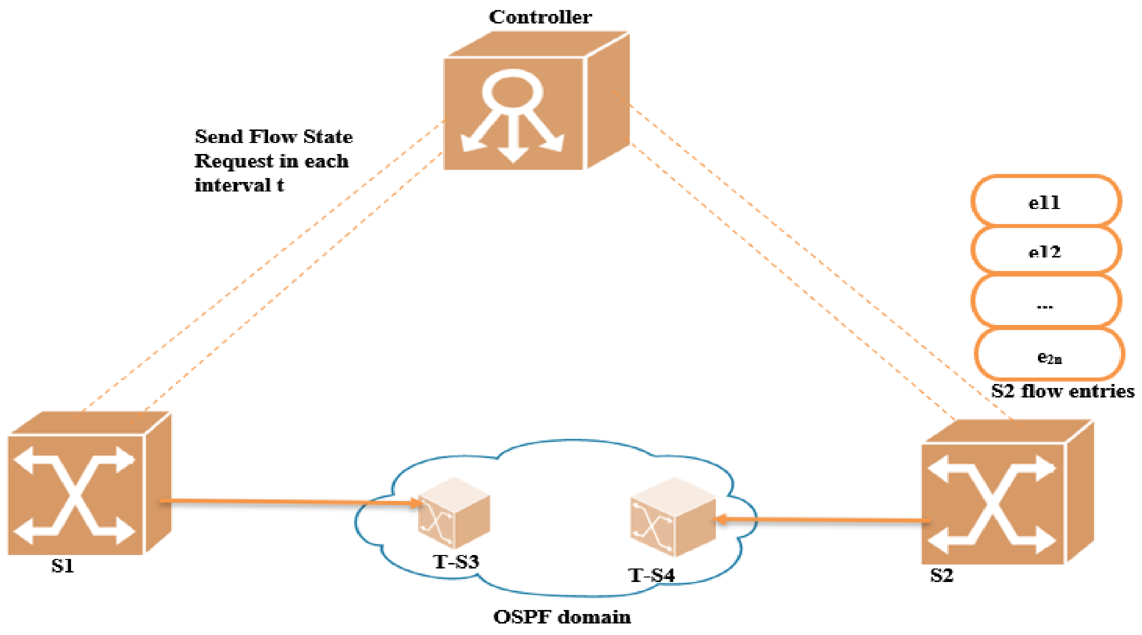


Figure 5.4. Traffic load monitoring in a hybrid network

Figure 5.4 demonstrates the hybrid network load monitoring module. In this network, the controller, at each time interval t, sends a flow state request message to the switch S1 over the OpenFlow control channel. It counts the number of data flows that reach port 1 of S2 and compares it with the required flow entries,e11, e12, e13, e2n, etc., using the feedback message it receives.

The SLA message on a link not connected to SDN switches may allow us to obtain information about the traffic load. An OSPF router receives SLA broadcasts from SDN nodes and relays the message and its status to the SDN switches. After receiving the SLA from the SDN switch, the controller resolves the message to ascertain the traffic load on that link.

Measuring every flow in a hybrid SDN network would be pointless and prohibitively expensive. The authors proposed a method for gathering the load of specifically designated connections and predicting the remaining amount with a 5% margin of error [86]. An applicable criterion for determining which flows need to be monitored to minimize the estimated error that relies on the flow spread parameter is defined and assessed by the authors in [87]. The proposed method may take forwarding table space into consideration and distribute measurement jobs among network devices in an equitable manner.

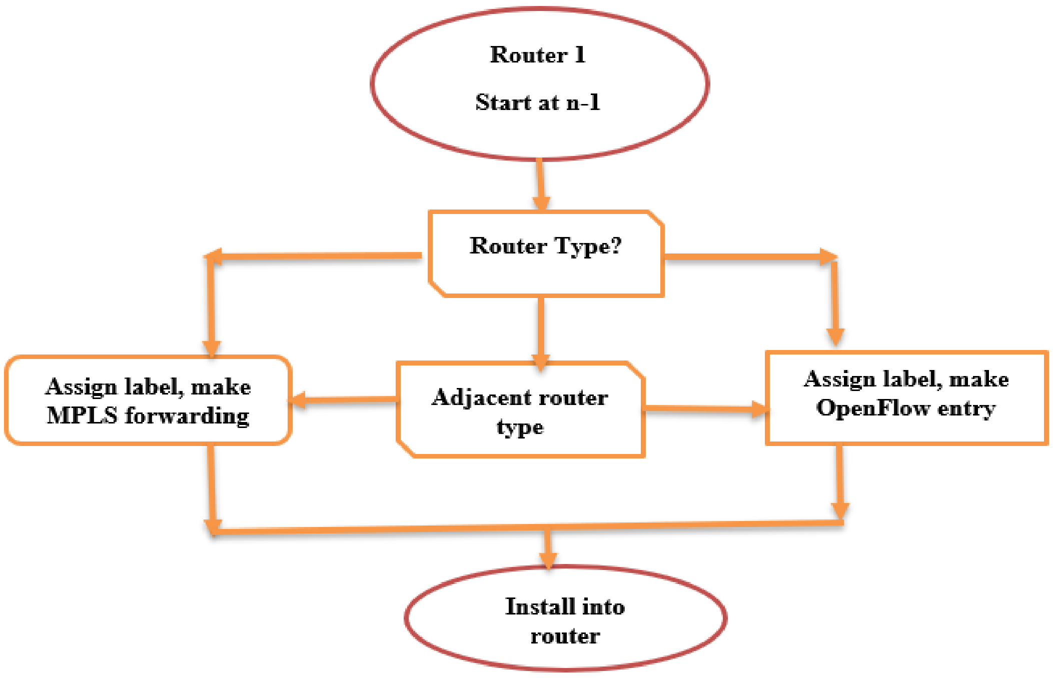


Figure 5.5 Tunnel construction procedure

Figure 5.5. demonstrates the tunnel construction method in the hybrid network using the penultimate hopping method. In tunnel splicing, the link translator configures the forwarding rule for each device from device n1 to device 1. Depending on the device types, this procedure involves adding OpenFlow entries or assigning MPLS (Multi-Protocol Label Switching) forwarding rules.

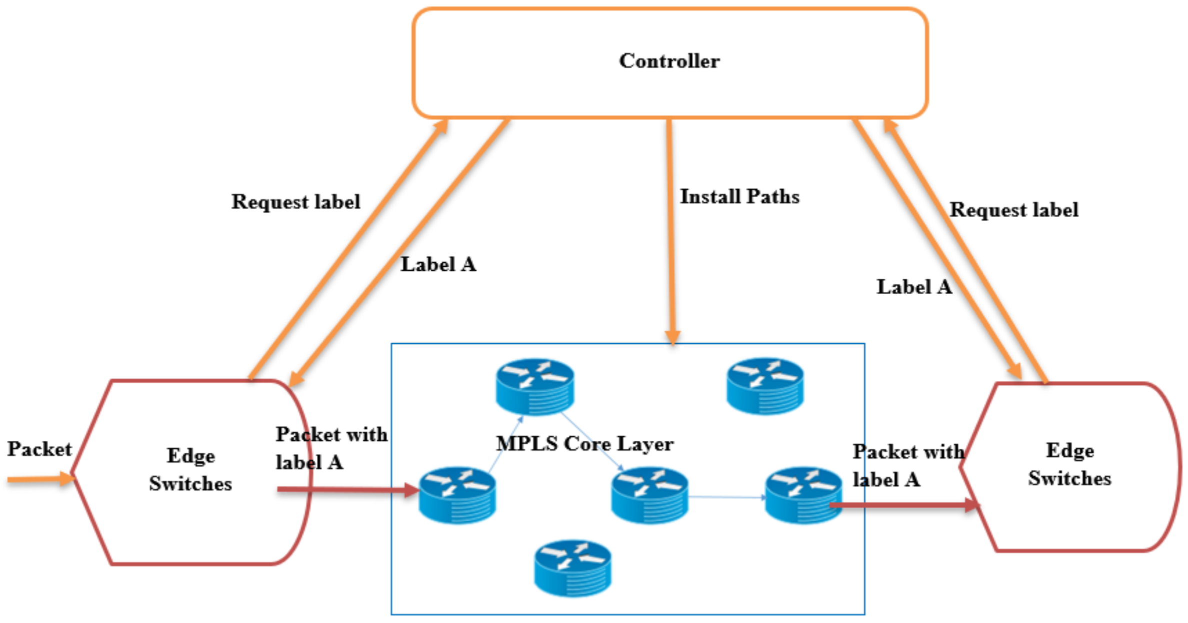


Figure 5.6 Installation of flow rules

Figure 5.6 depicts a Hybrid SDN network that combines SDN with MPLS-based networks. The network topology comprises a mesh of standard MPLS switches at the core and SDN devices at the edges. The network controller injects rules into the MPLS switch routes to push and pop the appropriate labels.

Packet loss rate and delay: SDN nodes serrates the hybrid network environment into multiple domains in a Hybrid SDN environment, frequently linked by SDN nodes. The internal domain devices function as OSPF routers; each pair of SDN switches can have several OSPF paths. Determining each link's packet loss rate and latency is problematic. On the other hand, the OpenFlow protocol can be used to calculate the packet loss rate and delay of a channel connecting SDN nodes. Although traffic within the OSPF domain does not pass through SDN switches in the hybrid SDN network, a significant portion will pass between multiple OSPF domains, necessitating its passage through one or more SDN switches. As a result, packet loss rate and latency can be estimated.

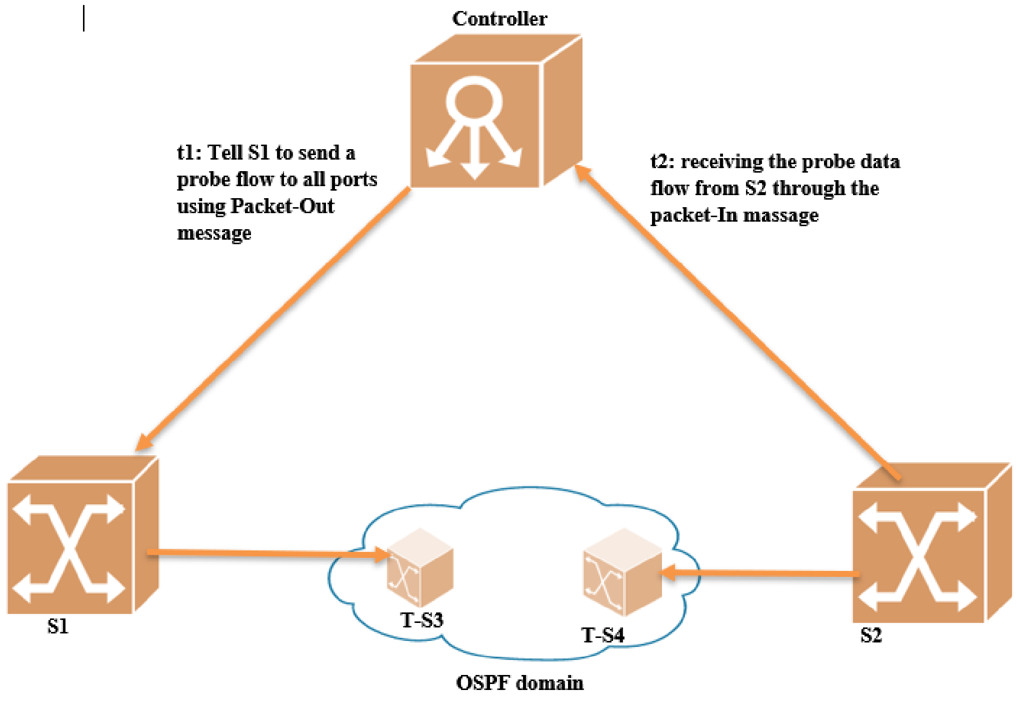

Controller
t1: Tell S1 to send a probe flow to all ports using Packet-Out message
t2: receiving the probe data flow from S2 through the packet-In massage
S1
T-S3
T-S4
S2
OSPF domain

Figure 5.7. Hybrid networks delay measurement

Figure 5.7 depicts the delay measurement in a hybrid SDN network; the delay of each link is seprated into two components (transmission delay and round-trip delay of data flows between SDN switches and controllers. The controller adds the time t1 to the probe data flow before sending the packet-out message to S1. After getting it, S1 sends the message to all ports in compliance with the regulations. This communication arrives at S2 via an OSPF path and is routed as a packet-in message to the SDN controller. The controller receives the message at t2 and uses it to get the transmitting port of S1, the receiving port of S2, and the time at t1.

More investigation has been carried out to enhance the Quality of Service in hybrid SDN networks. They used a variety of traffic management strategies to offer the highest quality of service for their designed topologies, and they proposed numerous ways to optimize single or multiple QoS metrics for hybrid SDN networks.

5.4. Methodology

To interconnect SDN nodes alongside conventional legacy devices, we have simulated all the simulations using Mininet and Mini-Edit tools. We have created the topology and the hybrid SDN environment using these tools, and as a controller of SDN nodes, we have tested multiple controllers. Finally, we have selected the ONOS controller as the SDN controller for our hybrid SDN environment.

Mininet Simulator simulates the network environments [55]; for configuring legacy routers, we have used FRR (Free Range Routing) using VTY Shel to configure network protocols. To connect different networks in the conventional network and to integrate the legacy network with SDN AS (Autonomous Systems), we have configured BGP (Border Gateway Protocol) inside all the legacy routers, as it is necessary to have a peering system that makes the BGP protocol work. And the integration between the conventional and the SDN networks are carried out through the SDN-IP. This program operates on top of the ONOS controller.

In this simulation experiment, the ONOS [56] controller is chosen to be the SDN controller, and the integration between the conventional network and the SDN network is carried out through the SDN-IP. This program operates on the application layer of the ONOS controller. The SDN-IP application enables SDN nodes to use BGP protocols to communicate with legacy routers.

First of all, we will open the Mininet tool to set up the network environment; with the MiniEdit tool, we can create and emulate network topologies. It also has the ability to configure network device interfaces such as IP addresses and gateways, and we can save the topologies. Figure 5.8 shows the graphical interface of Mininet which is called MiniEdit.

To open the MiniEdit tool, we first open the Mininet tool, and then we execute this command to open the MiniEdit tool's graphical interface.

```
$ sudo ~/mininet/examples/miniedit.py
```

Some tools are inside the MiniEdit tool, such as the host tool, legacy Switch, legacy router, switch tool, select tool, Netlink, and controller tools.

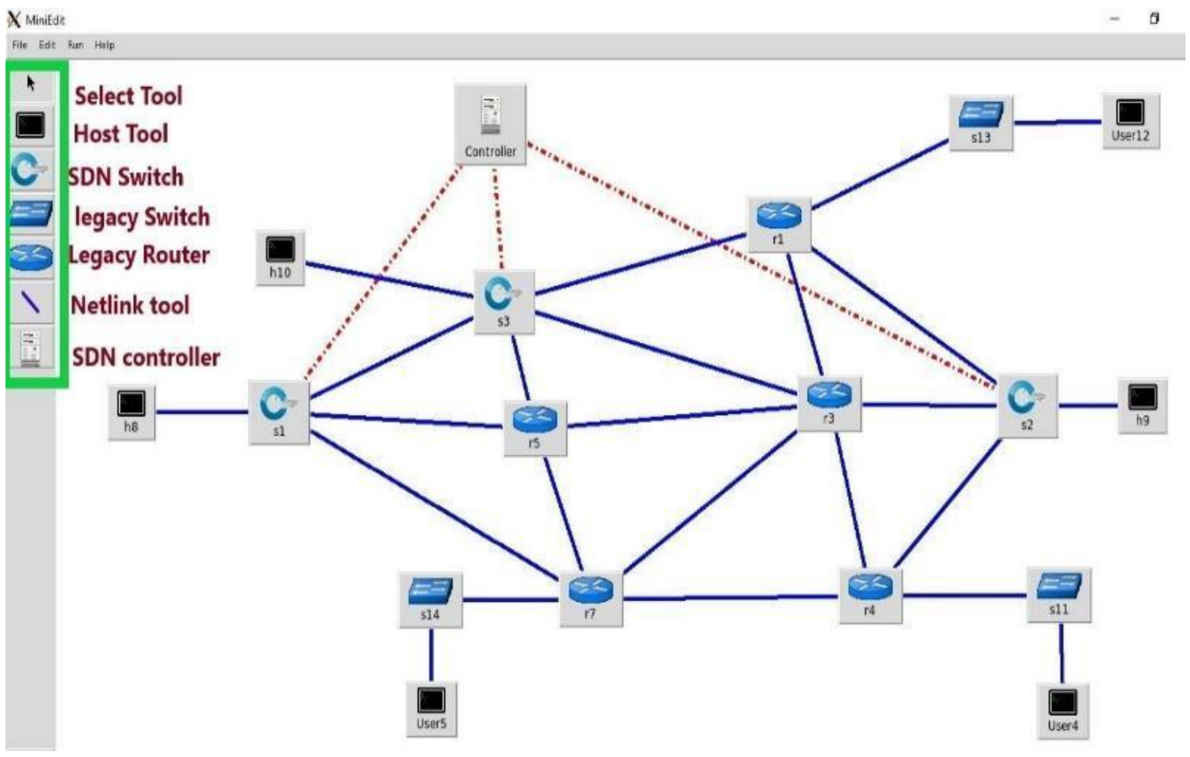


Figure 5.8 MiniEdit graphical interface

<u>Select tool</u>: used to select and move nodes from one area to another.

<u>Host tool:</u> Creates the nodes that conduct the host computer's functions.

<u>Switch tool:</u> This tool produces SDN-based OpenFlow switches that a controller manages.

<u>Legacy switch:</u> This tool creates a legacy Ethernet switch.

<u>Legacy router tool:</u> This tool generates a legacy router that does not require a controller.

NetLink tool: The NetLink tool creates linkages between nodes.

Controller tool: This tool is used to create a controller for the topology.

Once the topology is created, we will run it. To check the links on the command line, we will write links, and then we will assign the IP address according to the given topology; once we assign the IP address to the interfaces, we will check it.

To verify whether the IP is assigned properly, we can check it on each host by executing the “ifconfig” command.

We will configure and execute the commands on each router to configure each legacy router. First, we will start the Zebra domain. It is multi-server routing software that provides TCP/IP-based routing protocols.

```
R2# zebra

R2# vtysh

main# configure terminal

main(config) # interface r2-eth1

main(config-if)# ip address 192.168.10.2/30

main(config-if)# exit

main(config) # interface r2-eth2

main(config-if)# ip address 192.168.2.1/24

main(config-if)# exit
```

In this way, we have to configure all the legacy routers according to the given topologies. Once all the configuration is completed, we can check them by running this command on each legacy router

```
R2# show ip route
```

When all the interfaces have been successfully configured and assigned IP addresses, we will configure the BGP routing protocol inside all the conventional legacy routers to connect different ASes. We first have to initialize the daemon that provides BGP configuration to configure BGP. We can enable and configure the BGP daemon by running these commands on the conventional legacy router:

```
R2# zebra

R2# bgpd

R2#vtysh

main# configure terminal

main(config) # router bgp 30

main(config-router) # neighbor 192.168.10.1 remote as 20

main(config-router) #network 192.168.2.0/24

main(config-router) #end
```

By executing the following commands, we can configure BGP on different legacy routers. Once the configuration is completed, we can check it with the following command:

```
main#show ip bgp neighbors
```

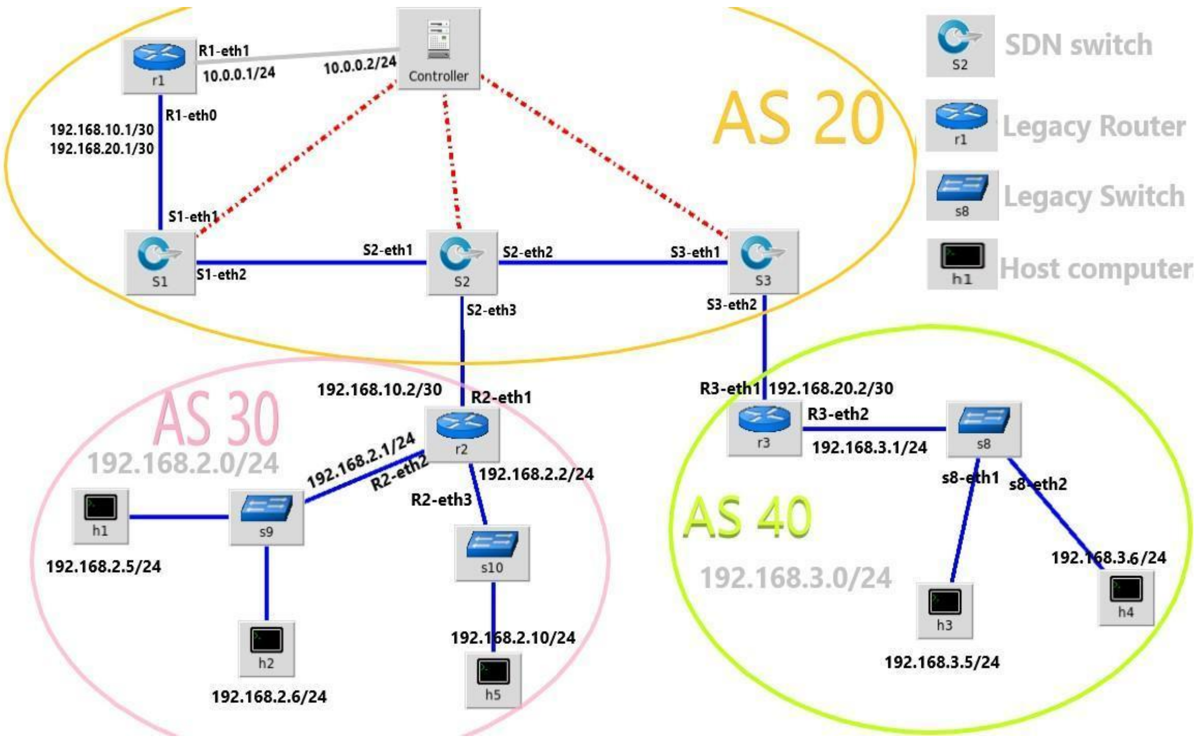


Figure 5.9. Hybrid SDN environment.

5.5. Results and discussion

To connect different networks in the conventional network and to integrate the legacy network with SDN AS (Autonomous Systems), we have configured BGP (Border Gateway Protocol) inside all the legacy routers, as it is necessary to have a peering system that makes the BGP protocol work. The integration between the conventional network and the SDN network is carried out through the SDN-IP. This program operates on the application layer of the ONOS controller. The SDN-IP application enables SDN nodes to use BGP protocols to communicate with legacy routers.

Figure 5.9 shows the topology we created using the Mininet and the MiniEdit tool; as shown we have assigned IP addresses to the interfaces of network devices, here we have three SDN nodes and three legacy routers, and we have a controller to manage SDN nodes; inside the legacy routers, we have configured BGP protocols with different AS, as shown in Figure 5.9. We have used the Zebra domain, a multi-server routing program that offers TCP/IP-based routing protocols, and the FRR (Free Range routing) domain.

Table 5.1: Topology information

Node	Interfaces	IP Addresses	Subnet	Default Gateway
Router -1	R1 eth0	192.168.10.1	/30	N0
		192.168.20.1	/30	N0

	R1 eth1	10.0.0.1	/24	N0
Router -2	R2 eth0	192.168.2.2	/24	No
	R2 eth1	192.168.10.2	30	No
	R2 eth2	192.168.2.1	/24	No
Router -3	R3 eth1	192.168.20.2	/30	No
	R3 eth2	192.168.3.1	/24	No
Host1	H1 eth0	192.168.2.5	/24	192.168.2.1
Host2	H2 eth1	192.168.2.6	/24	192.168.2.1
Host5	H5 eth0	192.168.2.10	/24	192.168.2.2
Host3	H3 eth0	192.168.3.5	/24	192.168.3.1
Host4	H4 eth1	192.168.3.6	/24	192.168.3.1

Once all the configuration is successfully done, we will save the topology and run the ONOS controller to connect it with our network topology and interconnect SDN nodes with the conventional legacy network devices.

We have already installed the ONOS controller; now, we will open it on a different terminal and run it. To integrate SDN nodes with the legacy network devices, activating IPv4 forwarding and the SDN-IP application is mandatory. To activate these applications on the ONOS controller terminal, we will run these commands:

First, we will configure and activate the configuration application by executing this command

```
>app activate org.onosproject.config
```

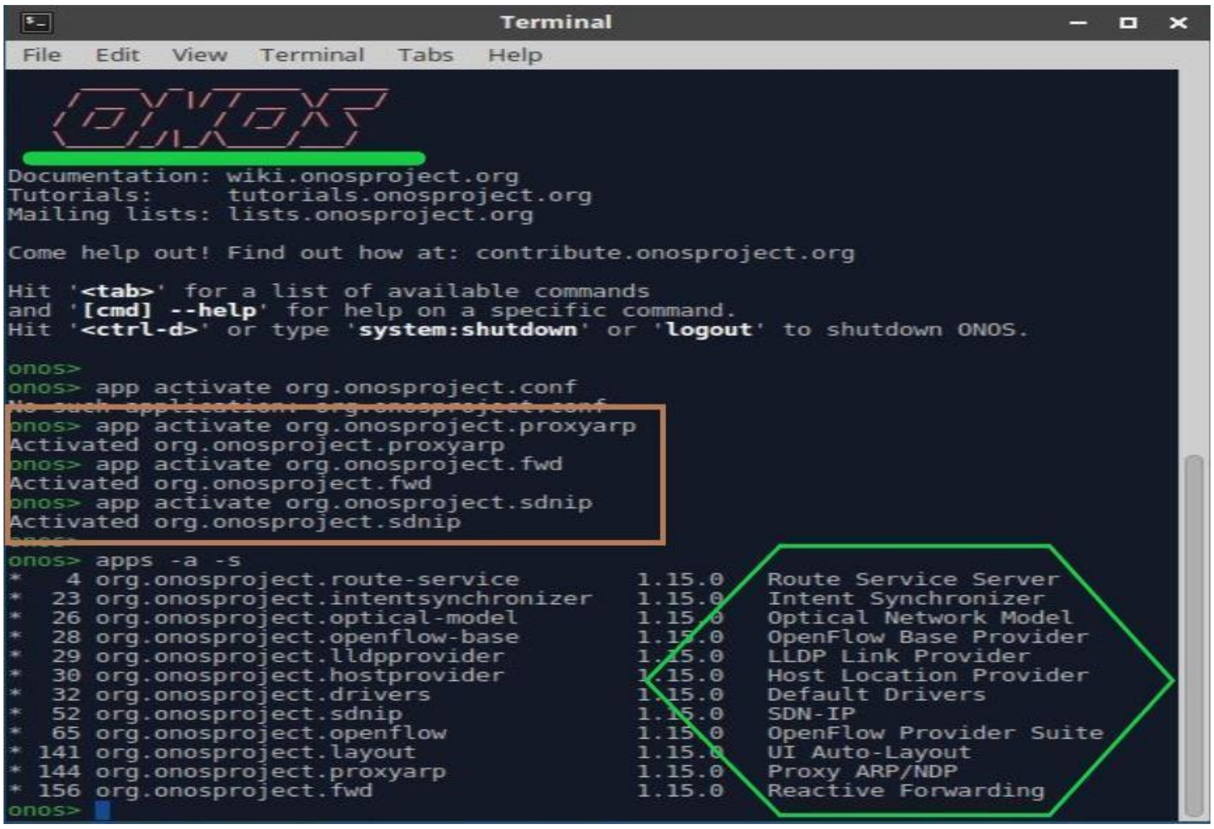
Then, we will activate the IPv4 forwarding by executing this command

```
>app activate org.onosproject.fwd
```

```
>app activate org.onosproject.proxyarp
```

For the integration, we will activate SDN-IP by executing this command

```
>app activate org.onosproject.sdnip
```



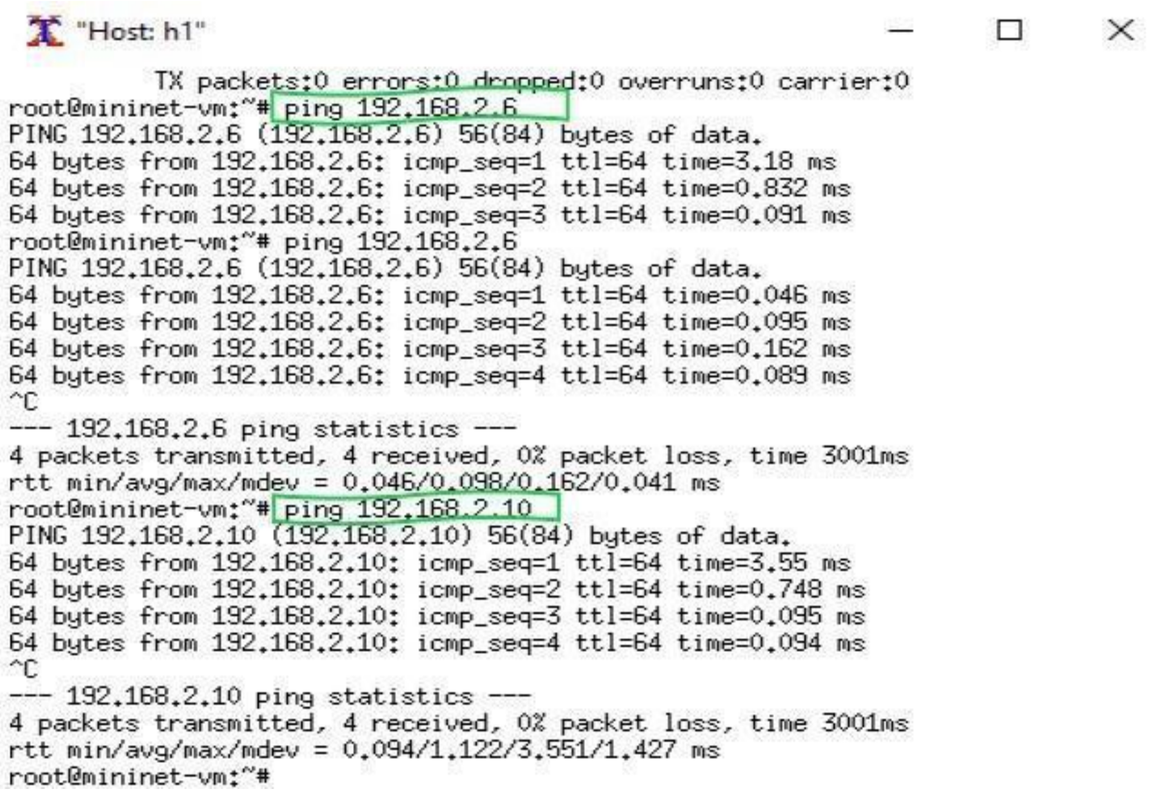
```
Documentation: wiki.onosproject.org
Tutorials:     tutorials.onosproject.org
Mailing lists: lists.onosproject.org

Come help out! Find out how at: contribute.onosproject.org

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown ONOS.

onos>
onos> app activate org.onosproject.conf
No such application: org.onosproject.conf
onos> app activate org.onosproject.proxyarp
Activated org.onosproject.proxyarp
onos> app activate org.onosproject.fwd
Activated org.onosproject.fwd
onos> app activate org.onosproject.sdnip
Activated org.onosproject.sdnip
onos>
onos> apps -a -s
*   4 org.onosproject.route-service        1.15.0   Route Service Server
*  23 org.onosproject.intentsynchronizer   1.15.0   Intent Synchronizer
*  26 org.onosproject.optical-model        1.15.0   Optical Network Model
*  28 org.onosproject.openflow-base        1.15.0   OpenFlow Base Provider
*  29 org.onosproject.lldpprovider         1.15.0   LLDP Link Provider
*  30 org.onosproject.hostprovider         1.15.0   Host Location Provider
*  32 org.onosproject.drivers              1.15.0   Default Drivers
*  52 org.onosproject.sdnip                1.15.0   SDN-IP
*  65 org.onosproject.openflow             1.15.0   OpenFlow Provider Suite
* 141 org.onosproject.layout               1.15.0   UI Auto-Layout
* 144 org.onosproject.proxyarp             1.15.0   Proxy ARP/NDP
* 156 org.onosproject.fwd                  1.15.0   Reactive Forwarding
onos>
```

Figure 5.10: ONOS terminal and applications

Figure 5.10 shows the ONOS controller terminal in which we have executed the commands for activating different applications. Once the application of ONOS is successfully activated, we will wait for some time as it will take some time, and then we will try to ping from one host to another.

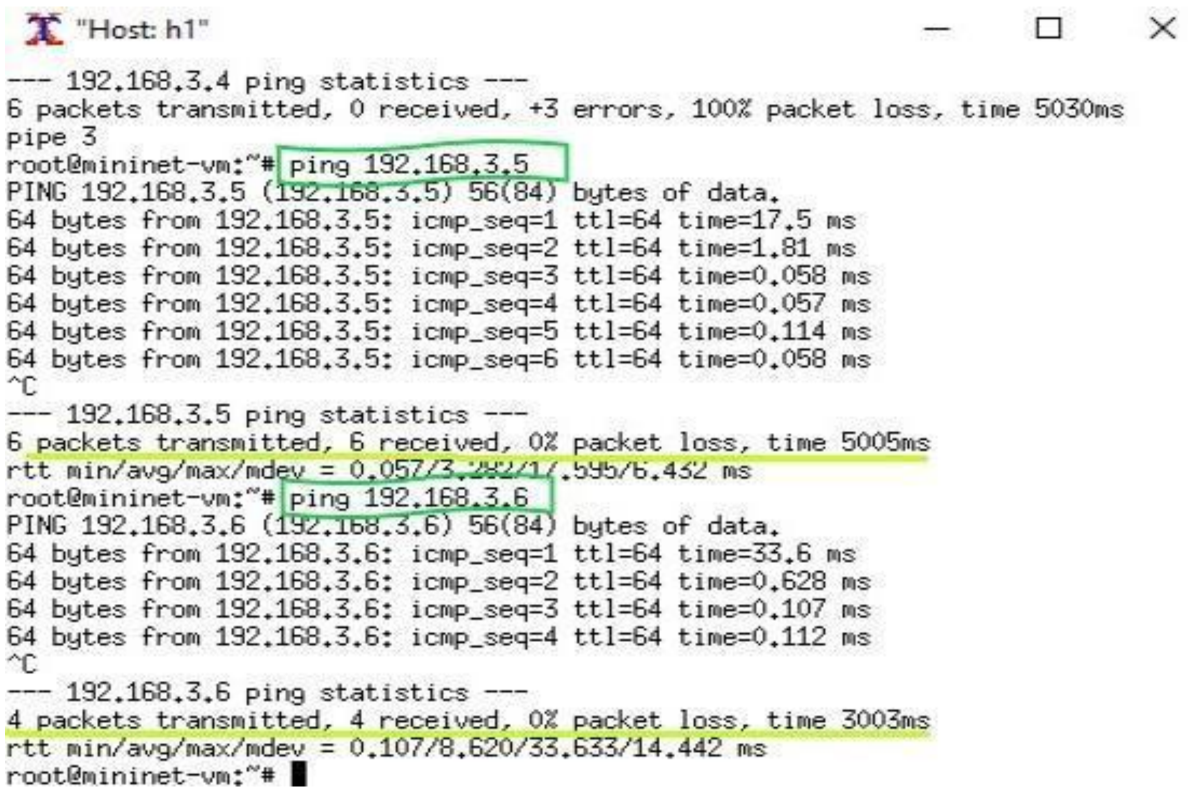
Table 5.2: Sampling for RTT and Jitter

Metric	**Traffic**	**Number of Tests**	**Number of samples**
RTT	ICMP	30	15000
Jitter	UDP	60	30,000



```
"Host: h1"
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
root@mininet-vm:~# ping 192.168.2.6
PING 192.168.2.6 (192.168.2.6) 56(84) bytes of data.
64 bytes from 192.168.2.6: icmp_seq=1 ttl=64 time=3.18 ms
64 bytes from 192.168.2.6: icmp_seq=2 ttl=64 time=0.832 ms
64 bytes from 192.168.2.6: icmp_seq=3 ttl=64 time=0.091 ms
root@mininet-vm:~# ping 192.168.2.6
PING 192.168.2.6 (192.168.2.6) 56(84) bytes of data.
64 bytes from 192.168.2.6: icmp_seq=1 ttl=64 time=0.046 ms
64 bytes from 192.168.2.6: icmp_seq=2 ttl=64 time=0.095 ms
64 bytes from 192.168.2.6: icmp_seq=3 ttl=64 time=0.162 ms
64 bytes from 192.168.2.6: icmp_seq=4 ttl=64 time=0.089 ms
^C
--- 192.168.2.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 0.046/0.098/0.162/0.041 ms
root@mininet-vm:~# ping 192.168.2.10
PING 192.168.2.10 (192.168.2.10) 56(84) bytes of data.
64 bytes from 192.168.2.10: icmp_seq=1 ttl=64 time=3.55 ms
64 bytes from 192.168.2.10: icmp_seq=2 ttl=64 time=0.748 ms
64 bytes from 192.168.2.10: icmp_seq=3 ttl=64 time=0.095 ms
64 bytes from 192.168.2.10: icmp_seq=4 ttl=64 time=0.094 ms
^C
--- 192.168.2.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 0.094/1.122/3.551/1.427 ms
root@mininet-vm:~#
```

Figure 5.11: Ping from h1 to h2 and h5 inside a single AS.



```
"Host: h1"
--- 192.168.3.4 ping statistics ---
6 packets transmitted, 0 received, +3 errors, 100% packet loss, time 5030ms
pipe 3
root@mininet-vm:~# ping 192.168.3.5
PING 192.168.3.5 (192.168.3.5) 56(84) bytes of data.
64 bytes from 192.168.3.5: icmp_seq=1 ttl=64 time=17.5 ms
64 bytes from 192.168.3.5: icmp_seq=2 ttl=64 time=1.81 ms
64 bytes from 192.168.3.5: icmp_seq=3 ttl=64 time=0.058 ms
64 bytes from 192.168.3.5: icmp_seq=4 ttl=64 time=0.057 ms
64 bytes from 192.168.3.5: icmp_seq=5 ttl=64 time=0.114 ms
64 bytes from 192.168.3.5: icmp_seq=6 ttl=64 time=0.058 ms
^C
--- 192.168.3.5 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5005ms
rtt min/avg/max/mdev = 0.057/3.282/17.595/6.432 ms
root@mininet-vm:~# ping 192.168.3.6
PING 192.168.3.6 (192.168.3.6) 56(84) bytes of data.
64 bytes from 192.168.3.6: icmp_seq=1 ttl=64 time=33.6 ms
64 bytes from 192.168.3.6: icmp_seq=2 ttl=64 time=0.628 ms
64 bytes from 192.168.3.6: icmp_seq=3 ttl=64 time=0.107 ms
64 bytes from 192.168.3.6: icmp_seq=4 ttl=64 time=0.112 ms
^C
--- 192.168.3.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.107/8.620/33.633/14.442 ms
root@mininet-vm:~# █
```

Figure 5.12 ping from h1 to h3 & h4

Figures 5.11 and 5.12 show that the hosts are successfully pinging as the integration between SDN nodes and the conventional legacy devices has been done through SDN-IP, an application of the ONOS controller. Now, we will try to retrieve the data and analyze it to take measurements of the hybrid SDN network performance.

The data is retrieved in a real-time network using TCP (Transmission Control Protocol), which is a reliable protocol based on our network environment. The data includes background traffic, which is generated using the Iperf tool to measure network performance, and we have used the Wireshark tool to analyze the data as it passes through the network.

Table 5.3: Generated data with traffic load

Protocol	**Bandwidth**	**Avg Data sent (Mb/sec)**	**Avg Date lost (Mb/sec)**	**Number of Tests**	
				IP to IP	**IP to SDN**
TCP	25	2.4	1.87	10	15
TCP	50	3.9	1.61	10	15
TCP	75	6.2	1.49	10	15
TCP	100	9.5	1.25	10	15
TCP	125	13.62	0.97	10	15
TCP	150	18.24	0.86	10	15
TCP	175	21.5	0.67	10	15
TCP	200	24.6	0.33	10	15
TCP	225	28.5	0.13	10	15
TCP	250	33.8	0	10	15

5.5.1. Performance metric test and results

To analyse the performance, twenty tests were run for each of the ten packet size variations (32 bytes, 64 bytes, 128 bytes, 256 bytes, 512 bytes, 1024 bytes, 1500 bytes, 2048 bytes, 4096 bytes, and 8192 bytes) in order to perform performance analysis in each network scenario. A total of 5000 samples were taken for each link test, with 500 samples extracted for each test.

Using the ping tool, ICMP traffic was sent between Host 1 as the sender and Host 2 and Host 3 as the receivers in order to test the RTT statistic. Using Host 2 as the sender and Host 3 as the recipient, more tests were conducted. Fifteen thousand samples were obtained from a total of thirty RTT tests. The iPerf3 program was used to send UDP traffic for jitter, and each network component was configured as a server or client to guarantee bidirectional data flow. The RTT and jitter testing configuration is shown in Table 2.

To guarantee a thorough assessment, six packet transmission scenarios were established for jitter testing, and 60 tests (10 per scenario) were carried out, yielding a total of 30,000 samples for this measure. A total of 90 tests were conducted once data collection for both measures was finished, producing 45,000 samples. The testing and sampling procedure for each measure is compiled in Table 5.2.

Table 5.4: Average Round Trip Time result

RTT average (ms)						
	H1-H2		**H1-H3**		**H2-H3**	
Test (bytes)	IP	HSDN	IP	HSDN	IP	HSDN
32	39.53	0.91	57.50	0.87	40.17	0.80
64	38.42	0.92	58.01	0.88	39.09	0.80
128	39.19	0.92	57.16	0.88	39.70	0.85
256	39.55	0.92	59.10	0.89	39.45	0.91
512	39.45	0.92	59.17	0.89	39.68	0.91
1024	40.13	0.92	59.00	0.95	41.78	0.91
1500	60.69	0.97	80.66	1.09	60.80	0.98
2048	60.76	1.24	81.55	1.11	62.40	1.10
4096	81.45	1.5	102.24	1.20	82.11	1.10
8192	143.69	1.9	166.02	1.41	145.65	1.23

Table 5.5: Average Jitter result

Jitter average (ms)						
	H1-H2		**H1-H3**		**H2-H3**	
Test (bytes)	IP	HSDN	IP	HSDN	IP	HSDN
32	20.36	0.009	20.31	0.010	15.23	0.009
64	23.36	0.009	18.93	0.009	47.50	0.008
128	23.26	0.010	18.04	0.012	23.84	0.011
256	20.84	0.15	34.83	0.017	11.87	0.010
512	29.90	0.011	29.25	0.015	12.79	0.019
1024	58.63	0.020	66.16	0.017	41.43	0.021
1500	51.34	0.035	74.57	0.023	51.41	0.029
2048	72.84	0.032	80.16	0.022	64.53	0.028
4096	204.97	0.027	139.97	0.020	147.17	0.024
8192	378.47	0.020	354.61	0.018	306.24	0.017

5.5.2. Performance Analysis of Hybrid SDN

5.5.2.1. Throughput

Throughput refers to the rate of successfully processed or transferred data through a network, device, or system, divided by the observed duration of the time in seconds. It is often measured in terms of data transmission rate and is often stated in bits per second (bps).

The throughput obtained by each device in a network may not always correspond to the channel bandwidth; the network throughput varies based on the number of devices and the data traffic in the network.

Throughput = amount of data received per unit time delivery of data (seconds) (1)

In a computer network, the term throughput describes the quality of data that can be delivered via a network in a given time frame. The factors influencing throughput in a network include latency, bandwidth, packet loss, and network congestion. With higher bandwidth, network throughput increases since it gives more capacity for the data to be transferred, while lower latency and minimum packet loss boost throughput by decreasing delays and ensuring reliable data transmission.

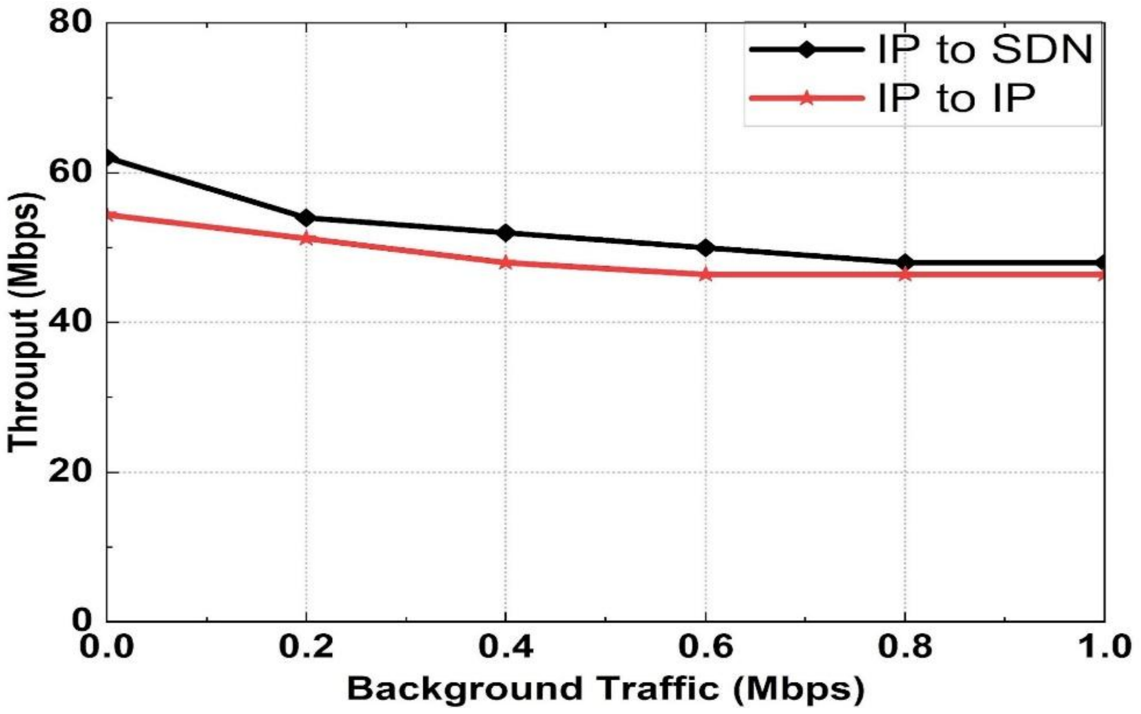


Figure 5.13: Throughput comparison

The throughput parameter measures the total number of packets that may be transferred in a given time frame in a network. Larger packets lead to faster data transmission as they improve network quality and increase network throughput. In TCP traffic communication, one host acts as a customer and another as a server, as shown in Figure 5.13; in the case of IP to IP using TCP data protocol, we first measured throughput without adding background traffic, then we added background traffic, and we increased the background traffic; as a result, with the increase in the size of the network traffic, we found 5-20% decreases in the network throughput. This indicates that the presence and growth of background traffic negatively impact the IP-IP network throughput.

In the second case, IP-SDN, we again take network throughput measurement without background traffic. We found no decrease in the network throughput, and then we increased the size of the background traffic, and the result showed that with the increase in the network traffic, the throughput value increased too, by 5-10%. This means that the presence and increase of background traffic positively impact the network throughput in the case of IP-SDN, which is a very good sign for organizations that want to migrate from a conventional legacy network to a hybrid SDN and fully SDN network.

5.5.2.2. Delay

Basically, the delay is another important parameter: the time taken to transfer data from the source to the destination. In a network, delays can occur due to different variables, such as device quality, source-to-destination distance, and congestion. In a computer network, we can measure the delay of the network by this formula:

Delay = t2-t1n (2)

Variable ‘t2’ indicates the time when a packet is received on the destination nodes. The value of ‘t2’ is calculated by subtracting the packet time sent from the sender side, represented by the variable ‘t1’, and n represents the total amount of data.

The delay value, which is the difference between “t2” and “t1,” is determined by the number of packets transmitted into the devices or network, so the volume of packets being sent influences the total delay.

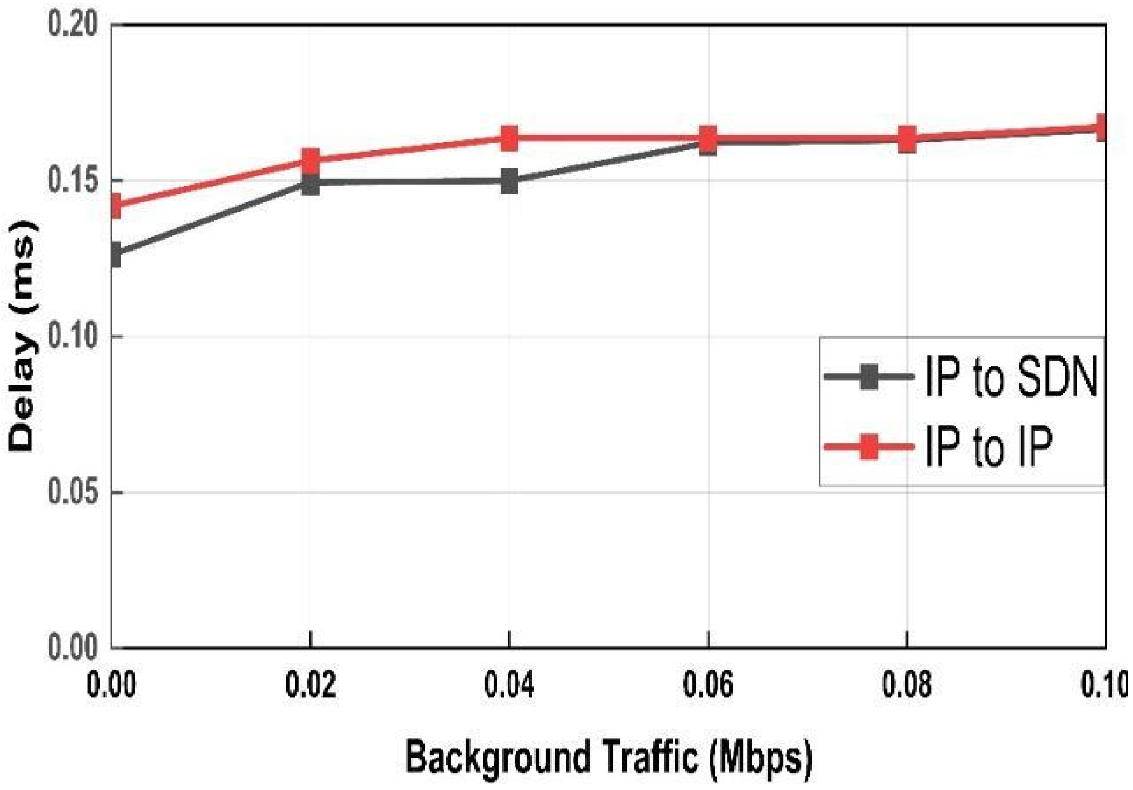


Figure 5.14. Delay comparison.

To measure the delay of the network in our environment, we have used Wireshark to collect information from each packet delivered from the source to the destination, as delay refers to the time a packet takes to be sent from the sender to the receiver.

As you can see in Figure 5.14, for the delay measurement, we took and considered both the case IP-IP and IP-SDN, the same as before, we increased the background traffic gradually, or simply we increased the size of the traffic data for both cases to take the measurement of the delay. As a result, we found that with the traffic increases, the delay value increases too in both cases, IP-to-IP and IP-SDN. More experiments and taking the average delay showed that with increasing gradual traffic, IP-SDN performs much better than IP-IP networks, as it gets a much lower delay than IP-IP networks.

For the delay measurement in both cases, we experienced an increased delay with more background traffic, but the IP-SDN network processes extra traffic more effectively, resulting in a minimum average delay compared to the IP-IP network.

5.6. Summary

It is well known that SDN is a cutting-edge network design that decouples the control and data layers to offer exceptional network control, flexibility, and efficiency. However, migrating straight from a conventional network to a fully SDN-based network appears challenging. A contemporary network design, known as hybrid SDN, is the optimal solution as it blends the programmability and adaptability of SDN with the stability and familiarity of conventional network designs. Hybrid SDN is a modern network design that combines the flexibility and programmability of SDN with the familiarity and robustness of conventional networking. In this study, we investigated the interconnection of SDN nodes with conventional legacy devices in a hybrid SDN environment; we looked at how SDN nodes interconnect with conventional network devices using the ONOS controller. Through extensive simulation, we found that hybrid SDN is the optimal solution for organizations seeking a modern network infrastructure. Our finding indicates that hybrid SDN, which provides a more scalable, flexible, and programmable network environment, is the best option for enterprises looking to upgrade their network infrastructure without experiencing service outages or network disruptions. By deploying sophisticated QoS policies in the hybrid environment, we can see significant improvements in the overall network performance, including lower latency, increased bandwidth usage, and enhanced overall user experience. Using machine learning algorithms for adaptive QoS and traffic prediction can improve the network's capacity to deal with diverse and unpredictable traffic patterns. The future of hybrid SDN networks holds enormous potential, especially when considering long-term sustainability, adaptability, and integration with fast-changing technologies such as IoT and 5G.

CHAPTER-6

CONCLUSION AND FUTURE SCOPE

In this comprehensive thesis, we have delved into the critical issue of migration to a hybrid SDN and optimization of QoS in the hybrid SDN networks, presenting three novel mechanisms across chapters 3,4, and 5. These mechanisms address different aspects of migration toward hybrid SDN and then QoS optimization in a migrated hybrid SDN and offer valuable insights into improving network performance, reducing packet loss, minimizing maximum link utilization, minimizing delays, and optimizing resource utilization.

Chapter 3 introduced the FCM (Flow Control Method) approach, which investigates the incremental deployment of SDN nodes in conventional networks using a simple greedy algorithm, presenting a potential solution for the evolution of network architecture. We investigated the optimal migration sequence from a conventional network to a hybrid SDN to address the optimization impact on controllable traffic. The proposed technique efficiently integrates SDN features while minimizing network disruption and optimizing link utilization. We evaluate the optimal deployment of SDN nodes through simulations, considering network topologies, available resources, and traffic patterns. Simulation experiments conducted on real network topologies demonstrate that by upgrading 17% of the nodes to SDN, near-optimal performance can be achieved while requiring substantially less investment in resources and effort for network upgrades. This result suggests that the greedy approach is cost-effective and highly efficient in practical scenarios. Achieving near-optimal load balancing with minimal upgrades could offer significant advantages for network operators, making it a valuable strategy for managing network performance in real-world applications. This study contributes to the knowledge of SDN deployment methodologies, providing valuable insights for organizations seeking to transition to a more flexible and agile network infrastructure.

In Chapter 4, we have proposed a method for the routing optimization in a hybrid software-defined network. In the hybrid SDN environment, two different network devices coexist and interconnect with each other; this change presents significant challenges and opportunities in traffic engineering in the hybrid SDN, which requires a creative solution to improve network efficiency, performance, and resource utilization. We studied the challenge of the routing optimization of traffic engineering in a migrated hybrid network and formulated the problem as a mixed integer nonlinear programming model.

We introduced a heuristic technique, H-STE, to improve traffic engineering in the hybrid SDN; we concentrated on minimizing the Maximum Link Utilization by optimizing two critical aspects: optimizing the OSPF weight settings across the whole network to balance the flows originating from conventional devices and optimizing the traffic splitting ratio of SDN nodes. The H-STE method helps bridge the gap between the two by optimizing the whole network performance and adhering to the limitations imposed by legacy routing infrastructure. We have conducted several experiments on real network datasets; the results demonstrate that with a 30% deployment ratio of SDN nodes, we can reduce the MLU and get close to the optimal result to get maximum benefits from the concept of a hybrid SDN network. The study findings contribute to the progress and development of efficient, flexible, and robust communication infrastructures as we strive to enhance network management and design. This research contributes to the expanding discussion of TE and routing optimization in the hybrid SDN, necessitating efficient solutions to improve network efficiency, performance, and link utilization.

In Chapter 5, we investigated the interconnection of SDN nodes with conventional legacy devices in the hybrid SDN environment. We studied the integration of SDN nodes alongside conventional legacy networks and explored how to peer and integrate SDN nodes with conventional legacy networks using the SDN-IP application located on top of the application layer of the ONOS controller.

As multiple SDN controllers exist, selecting the perfect controller based on the network topology is the key to achieving optimal network performance. We have studied and examined different controllers for the hybrid SDN environment, as multiple controllers exist. Notable controllers are NOX, Floodlight, POX, ODL, Ryu, and ONOS. Finally, we have selected the ONOS (Open Network Operating System) controller as the SDN controller for our hybrid SDN environment, as it is compatible with the hybrid SDN network and supports SDN-IP applications.

SDN-IP is a program that integrates legacy and SDN networks; it runs on the application layer of the ONOS controller and allows SDN devices to integrate and communicate with the legacy routers through BGP ASes.

Nevertheless, none of the earlier studies interconnect SDN nodes with conventional legacy networks using ONOS controllers in a real network scenario. We have considered this issue as the interconnection of SDN nodes into conventional legacy networks with ONOS controllers using Mininet tools. Through extensive simulation, we found that hybrid SDN is the optimal solution for organizations seeking a modern network infrastructure without network disruption and service downtime, as hybrid SDN can deliver a more scalable, flexible, and programmable network environment.

In conclusion, our thesis contributed comprehensive insights and novel solutions to the challenging problem of migration sequence to hybrid SDN and optimizing traffic engineering in the migrated hybrid SDN. The methods presented, including FCM and the H-STE approach, collectively offered valuable contributions to network performance enhancement, congestion mitigation, and resource optimization in the hybrid SDN environment. Furthermore, the scalability of our approach ensures that the method maintains its efficiency as the topology expands, demonstrating its resilience in controlling diverse network topologies. This is a significant improvement as it reduces resource utilization while enhancing network efficiency. Considering these factors, our method can optimize routing decisions more effectively, ensuring the network functions optimally within its capacity limitations while meeting performance requirements.

This finding is insightful and a baseline for balancing the advantage of SDN control with resource costs in a typical hybrid SDN network; it suggests a balance point where minimum SDN control is applied to control traffic without heavily depending on SDN resources. However, the generalizability of this result can vary under different network configurations and conditions. Adjusting the SDN deployment ratio may improve performance for more specialized or extreme scenarios, such as networks with high dynamic traffic, strict resiliency requirements, or limited path redundancy. This ratio can be a starting point, but may need to be adjusted to suit specific network characteristics and operational demands.

This thesis also opens up several avenues for future research and advancements in traffic engineering in a hybrid SDN environment. The future of HSDN systems holds enormous potential, especially when considering long-term sustainability, adaptability, and integration with fast-changing technologies such as IoT and 5 G. Future research could concentrate on integrating hybrid SDN with edge computing, in which hybrid SDN solutions route data intelligently to the local edge nodes for real-time processing while only sending critical data to the data centers or central cloud. As IoT and 5G networks generate massive amounts of data, this technique would improve network latency, minimize congestion, and enhance overall response in IoT and 5G applications. Another potential future scope is the exploration of more robust algorithms that may automatically adapt threshold values based on the specific characteristics and requirements of different network types. In the Future, we will try to develop an AI-driven optimization model to determine which 30% of nodes should be SDN-enabled to gain maximum performance with minimal cost, and try to compare the performance impact of 30% vs. 50% vs. fully SDN deployment in ISP and enterprise networks.

We believe that the Hybrid SDN network has come a long way; however, some missing areas still need more research and exploration, which leads to further improvements. Scalability in the Hybrid SDN network is one of the key problems caused by adding or changing the number of network nodes; for complicated hybrid SDN scenarios, more proven research is needed with scalability studies. Security is another crucial deficiency in the hybrid SDN network that must be at the essence of it and worthy of being considered as the central part of seamless hybrid SDN networks. In addition to that, researchers should focus more on privacy issues, new standards, and models of the hybrid SDN networks. By addressing these future research areas, we may continue to advance traffic engineering optimization techniques, enabling reliable, scalable, and optimized operations of hybrid SDN networks across a wide range of applications.

References

[1] Cisco, U., 2022.Cisco Annual Internet Report (2018-2023) White Paper.

[2] Xia, Wenfeng, et al. "A survey on software-defined networking." IEEE Communications Surveys & Tutorials 17.1 (2014): 27-51.

[3] Mehraban, Samiullah, and Rajesh Kumar Yadav. "Traffic engineering and quality of service in hybrid software-defined networks." China Communications 21, no. 2 (2024): 96-121.

[4] Hong, Chi-Yao, et al. "Achieving high utilization with software-driven WAN." Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM. 2013.

[5] Jain, Sushant, et al. "B4: Experience with a globally-deployed software-defined WAN." ACM SIGCOMM Computer Communication Review 43.4 (2013): 3-14.

[6] Amin, Rashid, Martin Reisslein, and Nadir Shah. "Hybrid SDN networks: A survey of existing approaches." IEEE Communications Surveys & Tutorials 20.4 (2018): 3259-3306.

[7] Open networking, "Software-Defined Networking (SDN) Definition," Open Networking Foundation. https://www.opennetworking.org/sdndefinition (accessed April 2024)

[8] Natarajan, Sriram, Anantha Ramaiah, and Mayan Mathen. "A software-defined cloud-gateway automation system using OpenFlow." 2013 IEEE 2nd International Conference on Cloud Networking (CloudNet). IEEE, 2013.

[9] Thyagaturu, Akhilesh S., et al. "Software-defined optical networks (SDONs): A comprehensive survey." IEEE Communications Surveys & Tutorials 18.4 (2016): 2738-2786.

[10] Banikazemi, Mohammad, et al. "Meridian: an SDN platform for cloud network services." IEEE Communications Magazine 51.2 (2013): 120-127.

[11] Mambretti, Joe, Jim Chen, and Fei Yeh. "Next generation clouds, the chameleon cloud testbed, and software-defined networking (Sdn)." 2015 International Conference on Cloud Computing Research and Innovation (ICCCRI). IEEE, 2015.

[12] Wang, Tao, Fangming Liu, and Hong Xu. "An efficient online algorithm for dynamic SDN controller assignment in data center networks." IEEE/ACM Transactions on Networking 25.5 (2017): 2788-2801.

[13] Velasco, L., et al. "Towards a carrier SDN: An example for elastic inter-datacenter connectivity." Optics Express 22.1 (2014): 55-61.

[14] Liu, Jing, et al. "SDN-based load balancing mechanism for elephant flow in data center networks." 2014 International Symposium on Wireless Personal Multimedia Communications (WPMC). IEEE, 2014.

[15] Jin, Xin, et al. "Optimizing bulk transfers with software-defined optical WAN." Proceedings of the 2016 ACM SIGCOMM Conference. 2016.

[16] Hong, Chi-Yao, et al. "B4 and after managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined WAN." Proceedings of the 2018 ACM Special Interest Group Conference on Data Communication. 2018.

[17] Costanzo, Salvatore, et al. "Software-defined wireless networks: Unbridling SDNs." 2012 European Workshop on Software Defined Networking. IEEE, 2012.

[18] Riggio, Roberto, et al. "Programming abstractions for software-defined wireless networks." IEEE Transactions on Network and Service Management 12.2 (2015): 146-162.

[19] MARKETSANDMARKETS, Software-defined networking market by SDN type (open SDN, SDN via overlay, and SDN via API), component (solutions and services), end user (data centers, service providers, and enterprises), and region - global forecast to 2023, 2019.

[20] McKeown, Nick, et al. "OpenFlow: enabling innovation in campus networks." ACM SIGCOMM computer communication review 38.2 (2008): 69-74.

[21] Vissicchio, Stefano, Laurent Vanbever, and Olivier Bonaventure. "Opportunities and research challenges of hybrid software-defined networks." ACM SIGCOMM Computer Communication Review 44.2 (2014): 70-75.

[22] Xu, Hongli, Xiang-Yang Li, Liusheng Huang, Hou Deng, He Huang, and Haibo Wang. "Incremental deployment and throughput maximization routing for a hybrid SDN." IEEE/ACM Transactions on Networking 25, no. 3 (2017): 1861-1875.

[23] Poularakis, Konstantinos, George Iosifidis, Georgios Smaragdakis, and Leandros Tassiulas. "Optimizing gradual SDN upgrades in ISP networks." IEEE/ACM transactions on networking 27, no. 1 (2019): 288-301.

[24] Guo, Zehua, Weikun Chen, Ya-Feng Liu, Yang Xu, and Zhi-Li Zhang. "Joint switch upgrade and controller deployment in hybrid software-defined networks." IEEE Journal on Selected Areas in Communications 37, no. 5 (2019): 1012-1028.

[25] Sinha, Yash, and K. Haribabu. "A survey: Hybrid sdn." Journal of Network and Computer Applications 100 (2017): 35-55.

[26] Caesar, Matthew, et al. "Design and implementation of a routing control platform." Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2. 2005.

[27] Jin, Cheng, et al. "Telekinesis: Controlling legacy switch routing with OpenFlow in hybrid networks." Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research. 2015.

[28] Mehraban, Samiullah, and R. K. Yadav. "Quality of services in hybrid sdn (hsdn): A review." In 2022 7th International Conference on Communication and Electronics Systems (ICCES), pp. 652-658. IEEE, 2022.

[29] Agarwal, Sugam, Murali Kodialam, and T. V. Lakshman. "Traffic engineering in software-defined networks." 2013 Proceedings IEEE INFOCOM. IEEE, 2013.

[30] Vissicchio, Stefano, et al. "Central control over distributed routing." Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication. 2015.

[31] Chu, Cing-Yu, et al. "Congestion-aware single link failure recovery in hybrid SDN networks." 2015 IEEE Conference on Computer Communications (INFOCOM). IEEE, 2015.

[32] Wang, Huandong, et al. "Saving energy in partially deployed software-defined networks." IEEE Transactions on Computers 65.5 (2015): 1578-1592.

[33] Vissicchio, Stefano, et al. "Safe update of hybrid SDN networks." IEEE/ACM Transactions on Networking 25.3 (2017): 1649-1662.

[34] Goransson, Paul, Chuck Black, and Timothy Culver. Software-defined networks: a comprehensive approach. Morgan Kaufmann, 2016.

[35] SDN Architecture, Last Accessed 15 May 2022.

[36] Software-defined networking (SDN): Layers and architecture terminology, https://tools.ietf.org/rfc/rfc7426.txt

[37] Jain, Raj, and Subharthi Paul. "Network virtualization and software-defined networking for cloud computing: a survey." IEEE Communications Magazine 51.11 (2013): 24-31.

[38] Bizanis, Nikos, and Fernando A. Kuipers. "SDN and virtualization solutions for the Internet of Things: A survey." IEEE Access 4 (2016): 5591-5606.

[39] Yin, Bo, and Xuetao Wei. "Communication-efficient data aggregation tree construction for complex queries in IoT applications." IEEE Internet of Things Journal 6.2 (2018): 3352-3363.

[40] Li, Yong, and Min Chen. "Software-defined network function virtualization: A survey." IEEE Access 3 (2015): 2542-2553.

[41] Vaughan-Nichols, Steven J. "OpenFlow: The next generation of the network?." Computer 44.08 (2011): 13-15.

[42] Kreutz, Diego, et al. "Software-defined networking: A comprehensive survey." Proceedings of the IEEE 103.1 (2014): 14-76.

[43] Bates, Adam, et al. "Let SDN be your eyes: Secure forensics in data center networks." Proceedings of the NDSS workshop on security of emerging network technologies (SENT'14). 2014.

[44] Gupta, Arpit, et al. "Sdx: A software-defined internet exchange." ACM SIGCOMM Computer Communication Review 44.4 (2014): 551-562.

[45] ONOS Controller. Accessed: May. 2022. [Online]. Available: https://onosproject.org

[46] OpenDayLight Controller. Accessed: March. 2022. [Online]. Available:
https://www.opendaylight.org

[47] Ongaro, Diego, and John Ousterhout. "In search of an understandable consensus algorithm." 2014 USENIX Annual Technical Conference (Usenix ATC 14). 2014.

[48] Setting Up Clustering. Accessed: March 2022. [Online]. Available: https://docs.opendaylight.org/en/stable-nitrogen/getting-started guide/commonfeatures/clustering.html

[49] ONOS Clustering. Accessed: July 2021. [Online]. Available:
https://wiki.onosproject.org/pages/viewpage.action?pageId=28836788

[50] Kar, Binayak, Eric Hsiao-Kuang Wu, and Ying-Dar Lin. "The budgeted maximum coverage problem in partially deployed software-defined networks." IEEE Transactions on Network and Service Management 13.3 (2016): 394-406.

[51] Sezer, Sakir, et al. "Are we ready for SDN? Implementation challenges for software-defined networks." IEEE Communications Magazine 51.7 (2013): 36-43.

[52] Hartman, S., M. Wasserman, and D. Zhang. "Security requirements in the software-defined networking model." Internet Engineering Task Force, Internet-Draft draft-hartman-sdnsec-requirements-01 (2013).

[53] Khorsandroo, Sajad, et al. "Hybrid SDN evolution: A comprehensive survey of the state-of-the-art." Computer Networks 192 (2021): 107981.

[54] Huang, Xinli, et al. "A survey of deployment solutions and optimization strategies for hybrid SDN networks." IEEE Communications Surveys & Tutorials 21.2 (2018): 1483-1507.

[55] Lantz, Bob, Brandon Heller, and Nick McKeown. "A network in a laptop: rapid prototyping for software-defined networks." Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. 2010.

[56] https://opennetworking.org/onos/ accessed 13/04/2023.

[57] Guo, Yingya, et al. "Incremental deployment for traffic engineering in hybrid SDN network." 2015 IEEE 34th international performance computing and communications conference (IPCCC). IEEE, 2015.

[58] Jia, Xuya, Yong Jiang, and Zehua Guo. "Incremental switch deployment for hybrid software-defined networks." 2016 IEEE 41st Conference on Local Computer Networks (LCN). IEEE, 2016.

[59] Yang, Ze, and Kwan L. Yeung. "SDN candidate selection in hybrid IP/SDN networks for single link failure protection." IEEE/ACM Transactions on Networking 28.1 (2020): 312-321.

[60] Caria, Marcel, et al. "Divide and conquer: Partitioning OSPF networks with SDN." 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM). IEEE, 2015.

[61] He, Jun, and Wei Song. "Achieving near-optimal traffic engineering in hybrid software-defined networks." 2015 IFIP Networking Conference (IFIP Networking). IEEE, 2015.

[62] Poularakis, Konstantinos, George Iosifidis, Georgios Smaragdakis, and Leandros Tassiulas. "One step at a time: Optimizing SDN upgrades in ISP networks." In IEEE INFOCOM 2017-IEEE Conference on Computer Communications, pp. 1-9. IEEE, 2017.

[63] Fortz, Bernard, and Mikkel Thorup. "Internet traffic engineering by optimizing OSPF weights." Proceedings IEEE INFOCOM 2000. conference on computer communications. Nineteenth annual joint conference of the IEEE computer and communications societies (Cat. No. 00CH37064). Vol. 2. IEEE, 2000.

[64] Ericsson, M., Mauricio G. C. Resende, and Panos M. Pardalos. "A genetic algorithm for the weight setting problem in OSPF routing." Journal of Combinatorial Optimization 6 (2002): 299-333.

[65] Srivastava, Shekhar, et al. "Determining link weight system under various objectives for OSPF networks using a Lagrangian relaxation-based approach." IEEE Transactions on Network and Service Management 2.1 (2005): 9-18.

[66] Hong, David Ke, et al. "Incremental deployment of SDN in hybrid enterprise and ISP networks." Proceedings of the Symposium on SDN Research. 2016.

[67] Guo, Yingya, et al. "SOTE: Traffic engineering in hybrid software-defined networks." Computer Networks 154 (2019): 60-72.

[68] Xu, Hongli, et al. "Scalable software-defined networking through hybrid switching." IEEE INFOCOM 2017-IEEE Conference on Computer Communications. IEEE, 2017.

[69] Levin, Dan, et al. "Panopticon: Reaping the Benefits of Incremental SDN Deployment in Enterprise Networks." 2014 USENIX Annual Technical Conference (USENIX ATC 14). 2014.

[70] Lu, Hui, et al. "Hybnet: Network manager for a hybrid network infrastructure." Proceedings of the Industrial Track of the 13th ACM/IFIP/USENIX International Middleware Conference. 2013.

[71] Huang, Huawei, et al. "The joint optimization of rules allocation and traffic engineering in software-defined networks." 2014 IEEE 22nd International Symposium of Quality of Service (IWQoS). IEEE, 2014.

[72] Guo, Yingya, et al. "Traffic engineering in hybrid software-defined network via reinforcement learning." Journal of Network and Computer Applications 189 (2021): 103116.

[73] Xu, Zhiying, et al. "Teal: Learning-accelerated optimization of wan traffic engineering." Proceedings of the ACM SIGCOMM 2023 Conference. 2023..

[74] Liu, Chenyi, et al. "Scalable deep reinforcement learning-based online routing for multi-type service requirements." IEEE Transactions on Parallel and Distributed Systems 34.8 (2023): 2337-2351.

[75] Lin, Bin, et al. "TITE: A transformer-based deep reinforcement learning approach for traffic engineering in hybrid SDN with dynamic traffic." Future Generation Computer Systems 161 (2024): 95-105.

[76] Guo, Yingya, et al. "MATE: A multi-agent reinforcement learning approach for Traffic Engineering in Hybrid Software Defined Networks." Journal of Network and Computer Applications 231 (2024): 103981.

[77] Awduche, Daniel, et al. Overview and principles of Internet traffic engineering. No. rfc3272. 2002.

[78] Davie, Bruce S., and Yakov Rekhter. MPLS: technology and applications. Morgan Kaufmann Publishers Inc., 2000.

[79] Mannie, Eric. Generalized multi-protocol label switching (GMPLS) architecture. No. rfc3945. 2004.

[80] Akyildiz, Ian F., et al. "Research challenges for traffic engineering in software-defined networks." IEEE Network 30.3 (2016): 52-58.

[81] Cianfrani, Antonio, Marco Listanti, and Marco Polverini. "Incremental deployment of segment routing into an ISP network: A traffic engineering perspective." IEEE/ACM Transactions on Networking 25.5 (2017): 3146-3160.

[82] McCormick, William C., Peter Ashwood-Smith, and Francis P. Kelly. "Systems and methods for traffic engineering in software-defined networks." US Patent No. 9,407,561. 2 Aug. 2016.

[83] Lukovszki, Tamas, Matthias Rost, and Stefan Schmid. "It's a match! near-optimal and incremental middlebox deployment." ACM SIGCOMM Computer Communication Review 46.1 (2016): 30-36.

[84] Zheng, Zhongming, et al. "Constrained energy-aware AP placement with rate adaptation in WLAN mesh networks." 2011 IEEE Global Telecommunications Conference-GLOBECOM 2011. IEEE, 2011.

[85] Luo, Gangyi, et al. "Improving performance by network-aware virtual machine clustering and consolidation." The Journal of Supercomputing 74.11 (2018): 5846-5864.

[86] Cheng, Tracy Yingying, and Xiaohua Jia. "Compressive traffic monitoring in hybrid SDN." IEEE Journal on Selected Areas in Communications 36.12 (2018): 2731-2743.

[87] Polverini, Marco, et al. "The power of SDN to improve the estimation of the ISP traffic matrix through the flow spread concept." IEEE Journal on Selected Areas in Communications 34.6 (2016): 1904-1913.

[88] Zhang, Junjie, et al. "Load balancing for multiple traffic matrices using SDN hybrid routing." 2014 IEEE 15th International Conference on High Performance Switching and Routing (HPSR). IEEE, 2014.

[89] Wang, Wen, Wenbo He, and Jinshu Su. "Enhancing the effectiveness of traffic engineering in hybrid SDN." 2017 IEEE international conference on communications (ICC). IEEE, 2017.

[90] Sinha, Yash, et al. "MPLS-based hybridization in SDN." 2017 Fourth International Conference on Software Defined Systems (SDS). IEEE, 2017.

[91] Tu, Xiaogang, et al. "Splicing MPLS and OpenFlow tunnels based on SDN paradigm." 2014 IEEE International Conference on Cloud Engineering. IEEE, 2014.

[92] Nakahodo, Yasunori, Takashi Naito, and Eiji Oki. "Implementation of smart-OSPF in hybrid software-defined network." 2014 4th IEEE International Conference on Network Infrastructure and Digital Content. IEEE, 2014.

[93] Sharma, Puneet, et al. "Enhancing network management frameworks with SDN-like control." 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013). IEEE, 2013.

[94] Sridharan, Ashwin, Roch Guerin, and Christophe Diot. "Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks." IEEE/ACM Transactions On Networking 13.2 (2005): 234-247.

[95] Xu, Dahai, Mung Chiang, and Jennifer Rexford. "DEFT: Distributed exponentially-weighted flow splitting." IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications. IEEE, 2007.

[96] Xu, Dahai, Mung Chiang, and Jennifer Rexford. "Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering." IEEE/ACM Transactions on Networking 19.6 (2011): 1717-1730.

[97] Xu, Ke, et al. "One more weight is enough: Toward the optimal traffic engineering with OSPF." 2011 31st International Conference on Distributed Computing Systems. IEEE, 2011.

[98] Elwalid, Anwar, et al. "MATE: MPLS adaptive traffic engineering." Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213). Vol. 3. IEEE, 2001.

[99] Kandula, Srikanth, et al. "Walking the tightrope: Responsive yet stable traffic engineering." ACM SIGCOMM Computer Communication Review 35.4 (2005): 253-264.

[100] Zhang, Mingui, Bin Liu, and Beichuan Zhang. "Multi-commodity flow traffic engineering with hybrid MPLS/OSPF routing." GLOBECOM 2009-2009 IEEE Global Telecommunications Conference. IEEE, 2009.

[101] Quan, Wei, et al. "Adaptive transmission control for software defined vehicular networks." IEEE Wireless Communications Letters 8.3 (2018): 653-656.

[102] Quan, Wei, et al. "Enhancing crowd collaborations for software-defined vehicular networks." IEEE Communications Magazine 55.8 (2017): 80-86.

[103] Agarwal, Saksham, et al. "Sincronia: Near-optimal network design for coflows." Proceedings of the 2018 ACM Special Interest Group Conference on Data Communication. 2018.

[104] Hu, Yannan, et al. "Maximizing network utilization in hybrid software-defined networks." 2015 IEEE Global Communications Conference (GLOBECOM). IEEE, 2015.

[105] Wang, Wen, Wenbo He, and Jinshu Su. "Boosting the benefits of hybrid SDN." 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2017.

[106] Tajiki, Mohammad Mahdi, et al. "CECT: computationally efficient congestion-avoidance and traffic engineering in software-defined cloud data centers." Cluster Computing 21.4 (2018): 1881-1897.

[107] Vissicchio, Stefano, Laurent Vanbever, and Jennifer Rexford. "Sweet little lies: Fake topologies for flexible routing." Proceedings of the 13th ACM Workshop on Hot Topics in Networks. 2014.

[108] Guo, Yingya, et al. "Traffic engineering in hybrid SDN networks with multiple traffic matrices." Computer Networks 126 (2017): 187-199.

[109] Huin, Nicolas, et al. "Bringing energy aware routing closer to reality with SDN hybrid networks." IEEE Transactions on Green Communications and Networking 2.4 (2018): 1128-1139.

[110] Tajiki, Mohammad Mahdi, et al. "SDN-based resource allocation in MPLS networks: A hybrid approach." Concurrency and Computation: Practice and Experience 31.8 (2019): e4728.

[111] Bi, Yuanguo, et al. "Intelligent quality of service aware traffic forwarding for software-defined networking/open shortest path first hybrid industrial internet." IEEE Transactions on Industrial Informatics 16.2 (2019)

[112] Huang, Xiaohong, Man Zeng, and Kun Xie. "Intelligent traffic control for QoS optimization in hybrid SDNs." Computer Networks 189 (2021)

[113] Galàn-Jimènez, Jaime, et al. "On the tradeoff between load balancing and energy-efficiency in hybrid IP/SDN networks." 2021 12th International Conference on Network of the Future (NoF). IEEE, 2021.

[114] Lin, Chienhung, Kuochen Wang, and Guocin Deng. "A QoS-aware routing in SDN hybrid networks." Procedia Computer Science 110 (2017): 242-249.

[115] Cespedes-Sanchez, Pedro, et al. "Hybrid Incremental Deployment of HSDN Devices." IEEE Transactions on Network and Service Management (2021).

[116] Siew, Hong Wei, Saw Chin Tan, and Ching Kwang Lee. "Hybrid SDN Deployment Using Machine Learning." Computational Science and Technology. Springer, Singapore, 2021. 215-225.

[117] Kelkawi, Ali, Ameer Mohammed, and Anwar Alyatama. "Incremental Deployment of Hybrid IP/SDN Network with Optimized Traffic Engineering." 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN). IEEE, 2020.

[118] Salman, Ola, et al. "QoS guarantee over hybrid SDN/non-SDN networks." 2017 8th International Conference on the Network of the Future (NOF). IEEE, 2017.

[119] Ren, Cheng, et al. "Enhancing traffic engineering performance and flow manageability in hybrid SDN." 2016 IEEE Global Communications Conference (GLOBECOM). IEEE, 2016.

[120] Chen, Ming-Hung, et al. "Incremental hybrid SDN deployment for enterprise networks." 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech). IEEE, 2017.

[121] Yang, Hui, et al. "Blockchain-based secure distributed control for software-defined optical networking." China Communications 16.6 (2019): 42-54.

[122] Xu, Hongli, et al. "Joint deployment and routing in hybrid SDNs." 2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS). IEEE, 2017.

[123] Jia, Xuya, et al. "Intelligent path control for energy-saving in hybrid SDN networks." Computer networks 131 (2018): 65-76.

[124] Yu, Changhe, et al. "DROM: Optimizing the routing in software-defined networks with deep reinforcement learning." IEEE Access 6 (2018): 64533-64539.

[125] Lin, Shih-Chun, et al. "QoS-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach." 2016 IEEE International Conference on Services Computing (SCC). IEEE, 2016.

[126] Xu, Zhiyuan, et al. "Experience-driven networking: A deep reinforcement learning based approach." IEEE INFOCOM 2018-IEEE conference on computer communications. IEEE, 2018.

[127] Huang, Xiaohong, et al. "Deep reinforcement learning for multimedia traffic control in software defined networking." IEEE Network 32.6 (2018): 35-41.

[128] Zhang, Junjie, et al. "Smartentry: Mitigating routing update overhead with reinforcement learning for traffic engineering." Proceedings of the Workshop on Network Meets AI & ML. 2020.

[129] Zhang, Junjie, et al. "CFR-RL: Traffic engineering with reinforcement learning in SDN." IEEE Journal on Selected Areas in Communications 38.10 (2020): 2249-2259.

[130] Sun, Penghao, et al. "Improving the scalability of deep reinforcement learning-based routing with control on partial nodes." ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). IEEE, 2020.

[131] Egilmez, Hilmi E., et al. "OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks." Proceedings of the 2012 Asia Pacific signal and information processing association annual summit and conference. IEEE, 2012.

[132] Yan, Jinyao, et al. "HiQoS: An SDN-based multipath QoS solution." China Communications 12.5 (2015): 123-133.

[133] Sieber, Christian, et al. "Network configuration with quality of service abstractions for SDN and legacy networks." 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM). IEEE, 2015.

[134] Alouache, Lylia, et al. "HSDN-GRA: A hybrid software-defined networking-based geographic routing protocol with the multi-agent approach." International Journal of Communication Systems 33.15 (2020): e4521.

[135] Yang, Hui, et al. "SUDOI: Software defined networking for ubiquitous data center optical interconnection." IEEE Communications Magazine 54.2 (2016): 86-95.

[136] Al-Fares, Mohammad, et al. "Hedera: dynamic flow scheduling for data center networks." Nsdi. Vol. 10. No. 8. 2010.

[137] Long, Hui, et al. "LABERIO: Dynamic load-balanced routing in OpenFlow-enabled networks." 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA). IEEE, 2013.

[138] Juttner, Alpar, et al. "Lagrange relaxation-based method for the QoS routing problem." Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213). Vol. 2. IEEE, 2001.

[139] Qin Z. et al. "A software defined networking architecture for the internet-of-things," in Proc. IEEE Network Operations and Management Symp., May 2014, pp. 1-9.

[140] Bari, Md Faizul, et al. "PolicyCop: An autonomic QoS policy enforcement framework for software defined networks." 2013 IEEE SDN for Future Networks and Services (SDN4FNS). IEEE, 2013.

[141] Karakus, Murat, and Arjan Durresi. "Quality of service (QoS) in software defined networking (SDN): A survey." Journal of Network and Computer Applications 80 (2017): 200-218.

[142] Moshref, Masoud, Minlan Yu, and Ramesh Govindan. "Resource/accuracy tradeoffs in software-defined measurement." Proceedings of the second ACM SIGCOMM workshop on Hot topics in Software defined networking. 2013.

[143] S Chowdhury, Shihabur Rahman, et al. "Payless: A low-cost network monitoring framework for software defined networks." 2014 IEEE Network Operations and Management Symposium (NOMS). IEEE, 2014.

[144] Sanvito, Davide, Ilario Filippini, Antonio Capone, Stefano Paris, and Jeremie Leguay. "Adaptive robust traffic engineering in software defined networks." In 2018 IFIP Networking Conference (IFIP Networking) and Workshops, pp. 145-153. IEEE, 2018.

[145] J.M. Kleinberg, Single-source unsplittable flow, in: IEEE Symposium on Foundations of Computer Science (FOCS), 1996, pp. 68–77

[146] T. Hartman, A. Hassidim, H. Kaplan, D. Raz, M. Segalov, How to split a flow?, in: IEEE International Conference on Computer Communications (INFOCOM), 2012, pp. 828–836.

[147] A. Morton, L. Ciavattone, G. Ramachandran, S. Shalunov, J. Perser, Packet reordering metrics, 2006, RFC 4737

[148] S. Bohacek, J.P. Hespanha, J. Lee, C. Lim, K. Obraczka, A new TCP for persistent packet reordering, IEEE/ACM Trans. Netw. 14 (2) (2006) 369–382.

[149] K.-C. Leung, V.O. Li, D. Yang, An overview of packet reordering in transmission control protocol (TCP): problems, solutions, and challenges, IEEE Trans. Parallel Distrib. Syst. 18 (4) (2007) 522–535

[150] Garg, Naveen, and Jochen Könemann. "Faster and simpler algorithms for multicommodity flow and other fractional packing problems." SIAM Journal on Computing 37, no. 2 (2007): 630-652.

[151] Wang, Yufei, Zheng Wang, and Leah Zhang. "Internet traffic engineering without full mesh overlaying." In Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213), vol. 1, pp. 565-571. IEEE, 2001.

[152] G. Optimization, Inc.," Gurobi optimizer reference manual." 2024.

[153] Cplex, IBM ILOG. "V12. 1: User's Manual for CPLEX." International Business Machines Corporation 46.53 (2009): 157.

[154] Orlowski, Sebastian, Roland Wessäly, Michal Pióro, and Artur Tomaszewski. "SNDlib 1.0—Survivable network design library." Networks: An International Journal 55, no. 3 (2010): 276-286.

[155] Garey, D. S. "Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness." (1979).

[156] Panchen, Sonia, Neil McKee, and Peter Phaal. "sFlow: A method for monitoring traffic in switched and routed networks." Internet Eng. Task Force (2001).

[157] Guo, Yingya, et al. "Routing optimization with path cardinality constraints in a hybrid SDN." Computer Communications 165 (2021): 112-121.

[158] Balon, S., and G. Monfort. "The traffic matrices and topology of the Abilene network", (http://totem.run.montefiore.ulg.ac.be/datatools.html).

[159] Zhang, Baobao, et al. "Cte: cost-effective intra-domain traffic engineering." ACM SIGCOMM Computer Communication Review 44.4 (2014): 115-116.

[160] Uhlig, Steve, et al. "Providing public intradomain traffic matrices to the research community." ACM SIGCOMM Computer Communication Review 36.1 (2006): 83-86.

[161] C. Jin, C. Lumezanu, Q. Xu, H. Mekky, Z.-L. Zhang, and G. Jiang, "Magneto: Unified fine-grained path control in legacy and openflow hybrid networks," in Proc. SOSR 17 Symp. SDN Res., Apr. 2017, pp. 75–87.

[162] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," in Proc. IEEE INFOCOM, Apr. 2013, pp. 2211–2219.

[163] H. Wang, Y. Li, D. Jin, P. Hui, and J. Wu, "Saving energy in partially deployed software-defined networks," IEEE Trans. Comput., vol. 65, no. 5, pp. 1578–1592, May 2016.

[164] Al-Jumaily, Abdulmajeed, et al. "Architecture Design of B5G and 6G Millimeter-Wave Radio Access Network: Using Wireless Communications to Increase Coverage, Capacity and Performance." 2023 16th International Conference on Developments in eSystems Engineering (DeSE). IEEE, 2023.

LIST OF PUBLICATIONS

Paper published in International Journals:

1. Mehraban, S., & Yadav, R. K. (**2024**). Traffic engineering and quality of service in hybrid software-defined networks. *China Communications*, *21*(2), 96-121. **(SCIE-3.1)**

2. Mehraban, S., & Yadav, R. K. (**2025**). Traffic Engineering Optimization in Hybrid Software-Defined Networks: A Mixed Integer Non-Linear Programming Model and Heuristic Algorithm. *International Journal of Network Management*, *35*(3), e70017. **(SCIE – 1.5)**

3. Mehraban, S., & Yadav, R. K. (**2025**). Routing Optimization in Hybrid Software-Defined Networks: A Heuristic Approach. *IETE Journal of Research*, 1-19. **(SCIE – 1.5)**.

Papers Published in International Conferences:

1. Mehraban, S., & Yadav, R. K. (**2022**, June). Quality of services in hybrid sdn (hsdn): A review. In *2022 7th International Conference on Communication and Electronics Systems (ICCES)* (pp. 652-658). IEEE.

2. Mehraban, S., & Yadav, R. K. (**2024**, September). Interconnection of SDN node with conventional network using an ONOS controller. In *2024 IEEE International Conference on Communication, Computing and Signal Processing (IICCCS)* (pp. 1-6). IEEE.

Book Chapter

1. Mehraban, S., & Yadav, R. K. (**2025**). Quality of Service Challenges in Hybrid IP/Software-Defined Networks. In *Quality of Service (QoS)-Challenges and Solutions*. IntechOpen.