Carleton University

Department of Systems and Computer Engineering

**SYSC 5801 – Advanced Network Routing Technologies**

# Segment Routing & Software-Defined Networking – Analysis, Use Cases & SR in SDN

**FINAL PROJECT REPORT**

**Presented to:**

**Prof. Chung-Horng Lung, Ph.D.**

**Department of Systems and Computer Engineering**


**Submitted by:**

**Nihal Kulkarni**

**101178458**

# Table of Contents

# Acknowledgment

I would like to express my sincere gratitude and appreciation to Professor Chung-Horng Lung for his invaluable support and guidance throughout the course. His expertise in core networking areas has been instrumental in my understanding of the subject. I thank him for his constant availability to provide suggestions and respond instantly to my emails. The dedication to answering all my queries regarding course assignments and project has been immensely valuable, and I am profoundly grateful for the willingness to guide me through every step of the way. Moreover, the remarkable insights and guidance throughout the course project have been crucial to its successful completion. Professor's mentorship has not only helped me in this course but has also equipped me with the skills and knowledge required to excel in my future endeavours. I feel privileged to have had the opportunity to work with such an exceptional professor, and I always cherish the lessons I have learned from him.

## Abstract:

Segment Routing (SR) has emerged and standardized as an efficient routing scheme to minimize the challenges faced by the current routing schemes. SR was first implemented with the MPLS architecture and later extended to IPv6 data planes as SRv6. SR has expanded from the basic architecture to real-time use cases such as service programming, and network virtualization. Software Defined Networking (SDN) on the other hand, is becoming increasingly prominent by providing the most flexible solutions for Internet traffic engineering. SDN decouples control plane and data planes and provides programmability to the network applications. Due to its programmability feature, the network goals such as bandwidth, latency, and QoS can be achieved. This project aims to elucidate the significance of segment routing and its various use cases, as well as software-defined networking and its diverse applications. Additionally, the project seeks to explore the combination of segment routing with software-defined networking and the challenges that may arise in implementing such a combination. The project report is based on a thorough literature review of various articles related to SR and SDN, which were selectively sourced from numerous reputable databases such as IEEE Xplore Digital Library, Science Direct, Elsevier, Google Scholar, and articles from Carleton Digital Library. The content of this project report is organized as follows. Section 1 introduces SR and its operation, section 2 provides insights on SR use cases, section 3 gives a brief introduction to SDN, followed by its use cases in section 4, section 5 is presented as SR in SDN and use cases, section 6 presents some key challenges, the future research directions are presented in section 7 and concludes with section 8.


## Acronyms:

| | |
|---|---|
| **SR** | Segment Routing |
| **SDN** | Software-Defined Networking |
| **MPLS** | Multi-Protocol Label Switching |
| **RSVP-TE** | Reservation Protocol-Traffic Engineering |
| **SR-TE** | Segment Routing-Traffic Engineering |
| **NF** | Network Function |
| **VNF** | Virtual Network Function |
| **SID** | Segment Identifier |
| **VPN** | Virtual Private Network |
| **LDP** | Label Distribution Protocol |
| **PE, CE** | Provider Edge, Customer Edge |
| **SLA** | Service Level Agreement |
| **CSPF** | Constrained Shortest Path First |
| **TLS** | Transport Layer Security |
| **DDoS** | Distributed Denial-of-Service |
| **PCE** | Path Computation Element |

# 1. Introduction to Segment Routing

Segment Routing (SR) is a source routing paradigm that simplifies traffic engineering and manages the networks. The state information is removed at the transit routers and nodes and places the path state information into the packet headers at an ingress node. A node can be an ordered list of instructions in packet headers that steers the packet along the path in the network. The single instruction is also called *segments*, and the sequence of the list is called a *segment list* or *SR Policy*. The intermediate nodes can insert the segment list and these nodes can be removed by any other nodes along the path developing a mechanism called *tunneling* [1] through the SR domain from the ingress node to the egress node.

## 1.1 SR Operation

The SR architecture can be implemented with the data plane and control plane components. The data plane will carry the segment list in packet headers and the control plane component will supplement the data plane by bringing additional features such as the allocation of segments by specifying the instruction to the segment identifier, and the distribution of the segment identifiers in the SR domain. The data plane can be instantiated in two ways, i.e., SR over MPLS (SR-MPLS) and SR over IPv6 (SRv6). For the IPv6 data plane, the Segment Routing Header (SRH) has an additional dedicated field called Segments Left (SL), which defines the number of active segments left in the segment list.

An example of SR operations can be shown in Fig. 1. The three segments as per the SR policy, can be termed as $P = <S1,S2,S3>$ and will traverse through node S1, S2, and reaches the destination node S3. The SR policy states that the packet should reach the destination node with SID S3, by identifying the transit nodes SIDs S1 and S2. The segment list is added to the packet header by the headend node and then steers the packet to SR policy [1].

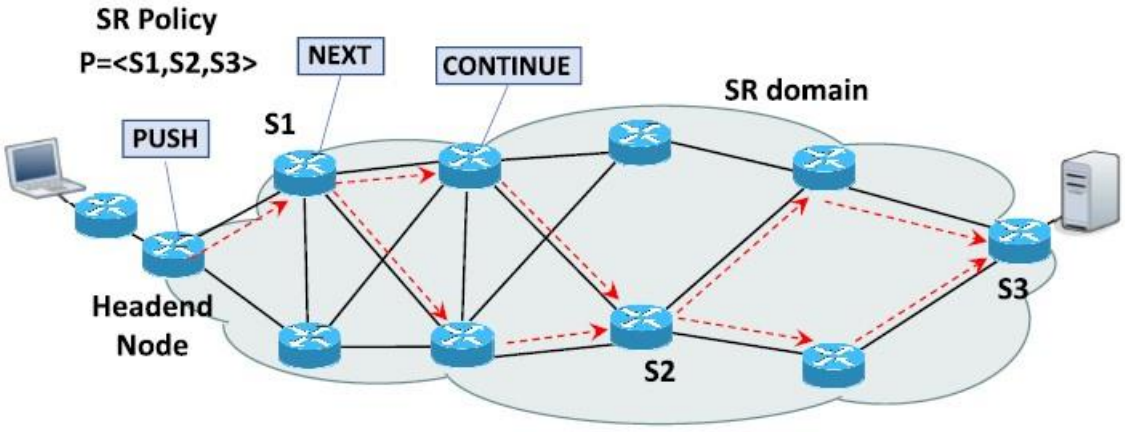


Fig. 1. SR Operation [1]

The most important motivation for selecting the SR is due to the fact of reducing the per-flow state and maintaining the network nodes that support traffic-engineered paths. Due to the SR implementation, it is not required to configure the forwarding table in the node along the path and only ingress nodes will have the information for the path. The RSVP-TE protocol need not be configured and send the configuration messages along the path. The other important motivation of SR is implementing the segments instead of running with a full sequence of hops for all the nodes, whereby all the path information can be known within a segment.

## 2. Segment Routing: Use Cases

### 2.1 Use Case 1: Service Programming with SR

SR can be instantiated over MPLS or IPv6 data plane to steer the packets through an ordered list of instructions called *segments*. The SR segments can also steer packets through the Virtual Network Functions (VNFs) or physical service appliances available in the network. Each of these services in the SR network running on physical devices is associated with SIDs, and these SIDs can be combined in a segment list to achieve Service Programming [2]. The SR thus provides a comprehensive solution for underlay, overlay, and service programming. Key terms included with this use case are *service segment* where a segment is associated with a service, *SR-aware service* which states a service is fully capable of processing SR traffic, and *SR-unaware service* which states a service is not capable of processing SR traffic.

#### 2.1.1 Implementation of SR-Aware & SR-Unaware Service Programming

The service can be referred to as a physical device or a Virtual Machine running (VM), running on a compute element and comprise of multiple sub-components running in different processors or containers. The execution of the service is based on the SR policy defined in section 1, where the SID is attached to the service in the segment list. This segment is termed as the service SID, and the service is reachable from anywhere in the SR domain.

#### 2.1.2 SR-Aware

The SR-Aware service can process the SR traffic and the information it receives from the packet. This type of service can identify the service segment as an instruction list and steers in the segment list. One practical example of this service is ***SR-aware firewall filtering***, where the destination address is retrieved from the SRH than the DA field in IPv6 [2]. If the service is configured for the physical devices, the routing scheme will implement default SRv6 endpoint behaviors, and the SIDs will steer the traffic throughout the service. In the case of a service that requires a virtual interface, the routing scheme needs to identify a particular SRv6 endpoint behavior. The SR-aware services will enable advanced network programming functionalities.

#### 2.1.3 SR-Unaware

The SR-Unaware service cannot process the SR traffic information, which results in unexpected routing information or incorrect decisions. To make use of the SR-unaware service, the SR information and other encapsulation headers are removed before the packet arrives in the segment list, and program the service in such a way that it correctly processes the packets. These modifications are termed to be *proxy behaviors* and an *SR proxy* can run on a service appliance with unique nodes or host. The two kinds of SR proxies are

***1. Static Proxy*:** The proxy receives SR traffic which can be an MPLS label stack or IPv6 address header on top of the packet.

***2. Dynamic Proxy*:** The SR information is known beforehand dynamically and all the state information is attached to the traffic.

For SR-MPLS, currently, *static proxy* behaviors are in use, and for SRv6, both the static and dynamic proxy behaviors are in use for the SR-unaware service programming.

## 2.2 Use Case 2: Transport Paths for L2VPN & L3VPN Services

The applications and services running over MPLS networks require strict SLA requirements like latency, bandwidth, jitter, and end-to-end paths to carry traffic. The data plane of MPLS can be programmed by SR mechanisms to meet the SLA requirements. The distributed path computation approach in SR at the ingress node calculates the shortest path first for the destination. Service providers employ MPLS data plane to provide Layer 2 VPN (L2VPN) and Layer 3 VPN (L3VPN) services. The full mesh MPLS tunnels provide connectivity among the Provider Edge (PE) routers. The PE device at the ingress network encapsulates the traffic with a VPN header and transports to more tunnels to an egress PE device. SR in this case (Fig. 2) will automate tunnel setup using the IGP protocol without the need for LDP or RSVP [3]. SR also automates the ECMP-aware tunnels and improves the scalability of the network.

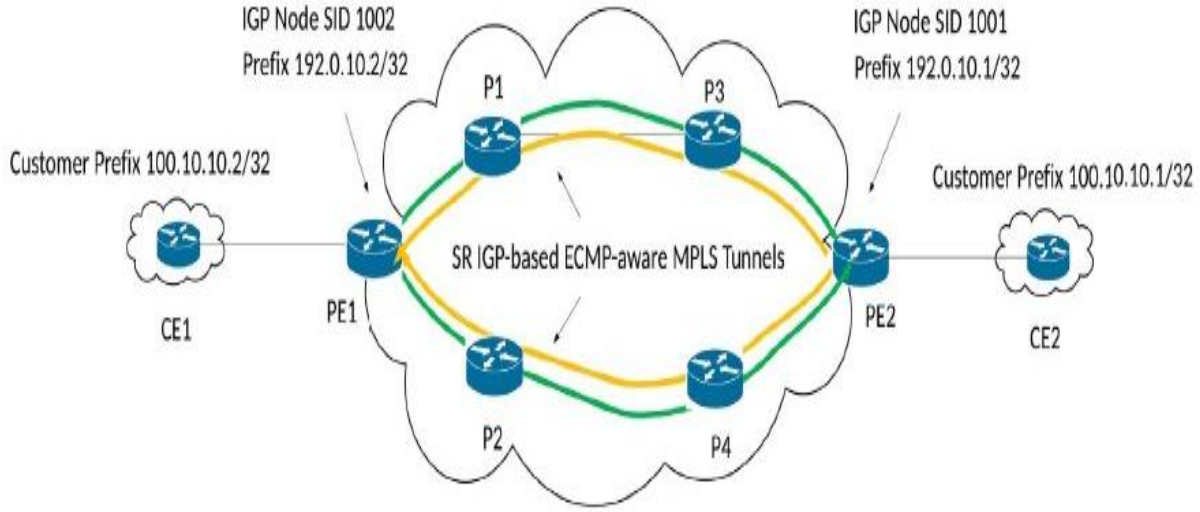


Fig. 2. SR-based transport paths for MPLS VPN service [3]

## 2.3 Use Case 3: Segment Routing IPv6 for Mobile User Plane

Mobile networks are getting increasingly challenging to operate, as the traffic demand is growing, and the latency parameters are tighter. The number of devices connected is growing rapidly and causing scalability issues. The statelessness of SRv6 and the ability to control the service layer and transport layer is invaluable in mobile user-plane by providing flexibility, network slicing, and SLA control for different applications. In the user plane, a tunnel between the mobile node and anchor node is established over IP backhaul networks. SRv6 when applied to mobile networks enables a source routing-based mobile architecture, where the mobile operators can provide an explicit route for packets to traverse from different mobile nodes. The SRv6 endpoint nodes act as user planes.

The IPv6 data plane instantiation in SRv6 integrates with the data path and transport layer into a single protocol, which allows operators to optimize networks and removes the forwarding state from the network [4]. In terms of mobility, SRv6 can provide user-plane behaviors for mobility management, takes advantage of transport awareness, provides flexibility, and includes services for end-to-end mobile data plane optimization.

## 3. Introduction to Software-Defined Networking:

Software-Defined Networking (SDN) is an emerging networking paradigm where the network control is decoupled from the forwarding plane and is directly programmable [5]. SDN decouples the control plane and data plane and is programmable on the control plane. SDN challenges the limitations of current network infrastructures. SDN breaks down the integration by separating control logic from routers and switches that forward traffic from the data plane. The SDN separation feature allows the network switches to behave as simple forwarding devices and the logic is implemented in a logically centralized controller [6]. SDN ensures the performance, scalability, flexibility, and reliability of the network. The control and data plane separation are implemented by the programming interface between switches and the SDN controller. The basic SDN architecture is shown in Fig. 3 below.

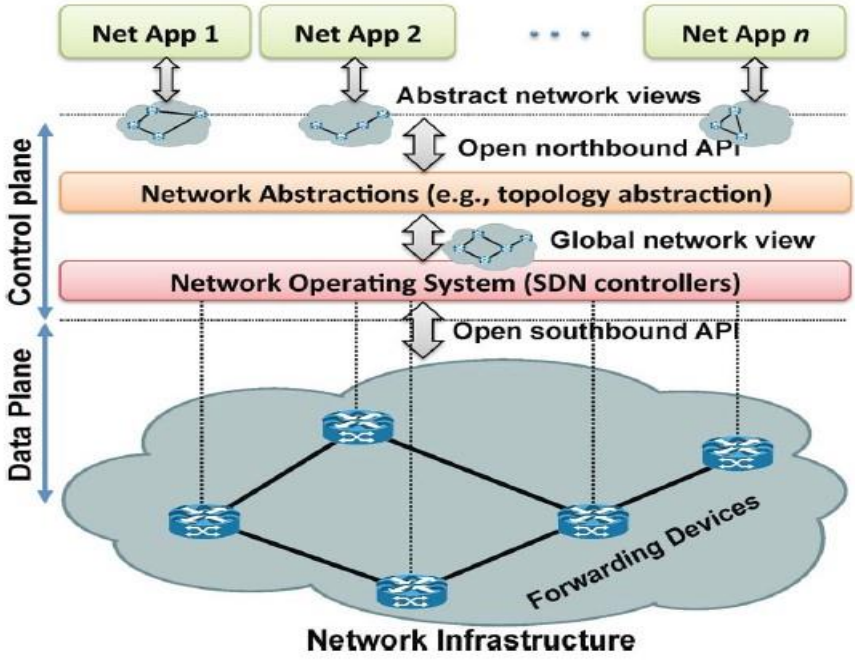


Fig. 3. Basic SDN Architecture [6]

### 3.1 Advantages of SDN

SDN decoupling mechanism offers great benefits and control over the network through programming.

- SDN breaks the barrier of layering by controlling the packet forwarding at the switching level and not at the data link level [7].
- SDN permits real-time ability for centralized control of the network based on the network status.
- With the unification of the control plane over other network devices such as switches, routers, and firewalls, the SDN will automatically configure from a single point through controlling software.
- SDN offers an opportunity to improve network performance by allowing centralized control and feedback control with information exchange between different layers.
- SDN provides solutions to classic problems such as traffic scheduling, Quality-of-Service (QoS), congestion control, and energy-efficient algorithms [8].

## 4. Software-Defined Networking Use Cases:

### 4.1 Use Case 1: Resource Allocation in Cloud Computing with SDN

Resource allocation is a major issue in cloud computing especially in Infrastructure-as-a-Service (IaaS). Cloud data centers require more elastic and agile network control functions to control cloud computing resources ensuring the functioning of VMs, traffic performance, and energy conservation [9]. SDN provides opportunities for network resource management to handle the cloud services that run on VMs. SDN implementation in the cloud makes it easier for allocating resources in cloud computing. The cloud computing based on SDN architecture is shown in Fig. 4.

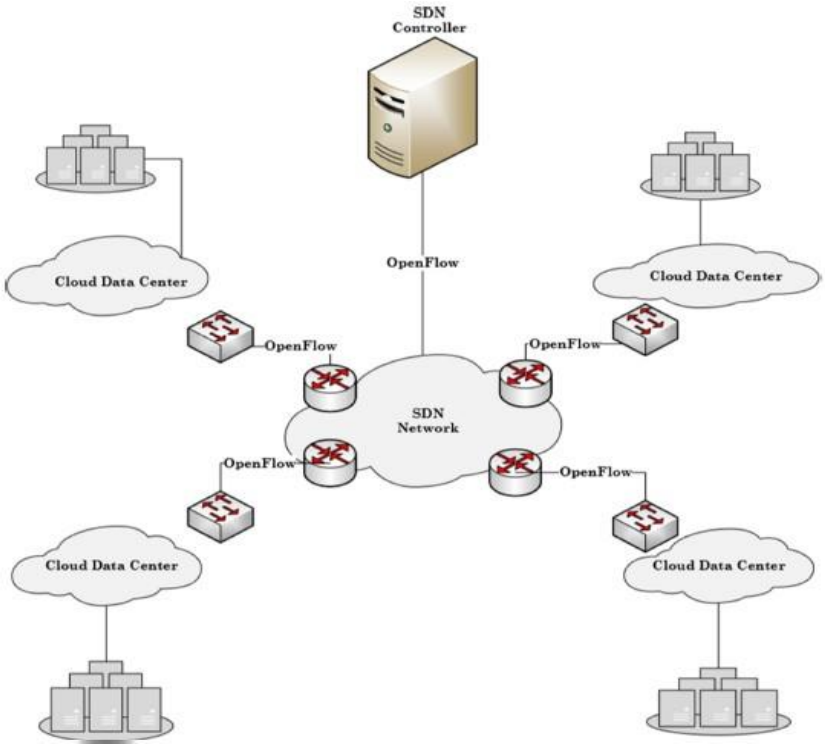


Fig. 4. Cloud computing based SDN architecture [9]

Cloud computing has thousands of servers that are attached to the virtualization technology to facilitate scalability and flexibility. Cloud operators have adopted the SDN in traditional data centers to mitigate the challenges. The controllers communicate through OpenFlow to their forwarding tables which allows it to change the network parameters and provides efficient, flexible, agile, scalable solutions to the cloud providers.

*"Google Cloud has officially implemented SDN model in its data center to improve scalability and flexibility"* [9].

*"IBM developed the state-of-the-art SDN cloud platform called Meridian for cloud services."*

#### 4.1.1 Resource allocation techniques based on SDN:

The cloud network architecture causes degradation issues in QoS and power consumption metrics. SDN when combined with the cloud, facilitates improved QoS and reduces power consumption. Energy-aware resource allocation and QoS-aware resource allocation are two resource allocation mechanisms currently in place for cloud computing based on SDN.

**a. Energy-aware resource allocation**:

The energy-efficient algorithm in combination with particle swarm optimization will be applied to Open Daylight Controller is an efficient routing strategy that consumes less power and saves energy. *Elastic-Tree* [9] is an SDN-based network power manager that reduces the power consumption in the network by shutting down unused links and switches. The performance of the network is improved and increases robustness.

**b. QoS-aware resource allocation:**

QoS parameters involve assured bandwidth, network latency, loss rate, and congestion control. The *Q-Ctrl* mechanism proposed in [10] breaks down the network into slices and monitors the QoS metrics based on the SDN infrastructure, OVS switch, and Mininet emulator. This method pre-calculates the path between two VMs using a cloud data center network which maintains delay performance and the QoS objectives are satisfied.

## 4.2 Use Case 2: Cloud-Native Network Slicing using SDN

Cloud-native is essential for 5G Network Functions (NFs). The cloud-native applications run on Kubernetes-enabled infrastructure and these network applications can be deployed in the public cloud, distributed core based on the service requirements [11]. The VM-based virtualization techniques add complexity to the networks as the system requires complete packaging of operating systems. The containers reduce the overhead packaging of applications and the dependencies on OS are greatly reduced. The implementation of this use case is categorized as follows.

### 4.2.1 SDN & Service Chaining

SDN controller provides effective network service chaining and management by enforcing independent rules for each network slice. Service chaining is a network ability that allows the applications to network through the set of NFs [12]. Service chaining provides flexible solutions for both control and data plane functions. The network operators will be able to scale and eliminate the NFs for specific network slices with the service chaining. Network slicing also allows mobile operators to add VNs on the same physical networks. SDN frameworks are implemented for NF connectivity.

### 4.2.2 VNF Traffic Steering through SDN

VNFs altogether are combined to form a service due to their functions. VNF traffic steering through SDN allows multiple domains to steer traffic through desired NFs with manual configuration. In the case of network slicing, the real-time deployment of VNFs creates services on-prem. Network slicing allows dynamic resource allocations to VNFs and has traffic steering capabilities. SDN provides traffic steering towards VNFs due to its centralized logic control. Extending SDN capabilities to Layer 2 and Layer 3 [13] allows dynamic traffic steering through VNFs.

### 4.2.3 Traffic Splitting through SDN

The communication between the controllers and the data forwarding tables is performed through the OpenFlow protocol which also allows one to make network decisions and carry out network monitoring. The control plane specifies the data plane of the forwarding instructions based on the primary flow entries.

SDN network will be utilized for efficient traffic splitting for *Differential Services (DiffServ)* by using OpenFlow 1.3 [14]. OpenFlow enables meter entries that define per-flow state to perform QoS operations such as DiffServ. The table also defines the packet forwarding rules and *Differentiated Service Code Point (DSCP)* numbers respectively. The DSCP is a 6-bit field included in the IP header that identifies the service levels of the packet. In summary, the SDN OpenFlow 1.3 will be used for traffic splitting to provide DiffServ to the users and provides diverse service requirements.

## 5. Segment Routing in Software-Defined Networking – Use Cases

Traffic engineering manages the network traffic and optimizes the utilization of network resources in an efficient manner. The existing traffic engineering solutions are facing issues related to scalability, flexibility, and complexity. SR has emerged as a promising solution for the routing paradigms. One of the most important applications of SR [15] is Traffic Engineering (SR-TE) which enables a headend node to steer the traffic along the defined paths. The analysis of SR traffic engineering with SDN is explained briefly in sections 5.1 and 5.2 respectively.

### 5.1 Segment Routing Traffic Engineering (SR-TE) with Efficient Routing Design

SDN decides the forwarding behavior for different combinations of packet header fields and provides fine-grained traffic allocation and distribution control throughout the network. In general, SDN requires larger flow tables than the traditional switch for storing the flow table information. SR introduces a method for packet forwarding where the need for a large number of information states is minimized. For scalability and agility, SR avoids huge label coding stored in network devices along the path [16]. SR maintains a large number of forwarding rules between the network devices and the SDN controller.

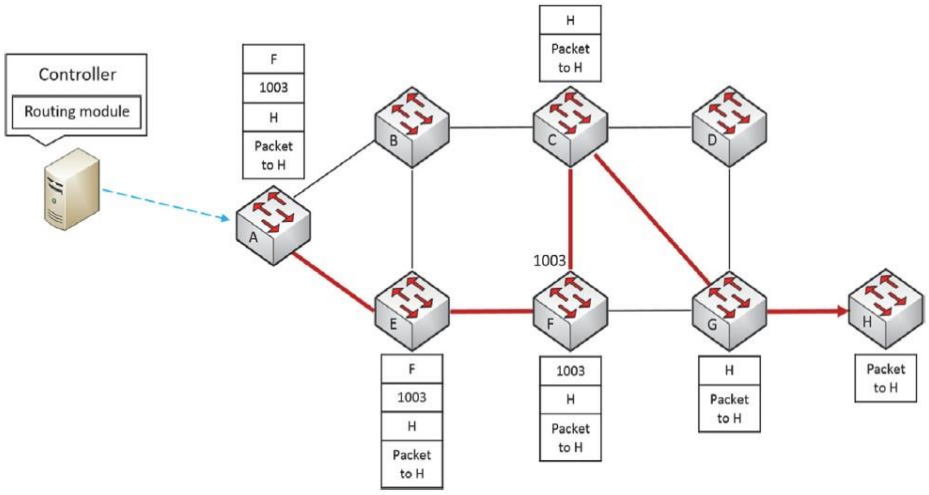


Fig. 5. SR-TE with SDN controller [16]

The SDN controller shown in Fig. 5, will encode only the end-to-end routing information into the packet header as an order list of labels at the ingress network device through SR. The SDN controller need not add the forwarding information at each router placed on the network on its path. The intermediate nodes need not maintain any per-flow state and can simply act on routed behaviors as per the segment list which is a more simple and scalable TE solution. Fig. 5 gives an overview of the SR function through the SDN controller. The network has node segments

and adjacency segments. The routing module in the SDN controller determines the explicit route and configures the forwarding table at the ingress device with the ordered list of segments [16]. The implementation shows the path from router A to H. SDN controller encodes the routing information in the packet header by choosing the path *(A-E-F-C-G-H)* at the ingress node and removes the path information rules from the intermediate nodes. The router A processes the top label and forwards the packet along the shortest path with the node segment F. The node segment F again validates the adjacency segment 1003 and forwards the packet to the next shortest path and so on. This way, SR can eliminate the number of forwarding rules and creates the shortest path in the network with the SDN-based environment.

## 5.2 Segment Routing Traffic Engineering (SR-TE) with SDN Open-Source Application

One of the main applications that is benefited from the SDN paradigm is TE. In traditional per-flow routing, the SDN controller interacts with each node in the traffic paths. SR simplifies this by configuring the per-flow state at the border network.

### 5.2.1 Open-Source Implementation of Control & Data Planes for MPLS-SR

For Open-Source implementation, in MPLS-SR, the ISP network is managed by SDN centralized controller. The network nodes are classified as Provider Edge (PE) routers and Core Routers (CR), as MPLS nodes. The PE nodes are connected to Customer Edge (CE) routers as the traffic source and destinations respectively. MPLS data plane is utilized for SR forwarding as seen in Fig. 6. SIDs are derived from the MPLS router in an ISP network [17]. There is no IGP or OSPF protocol implemented in this case for the distribution of SIDs.

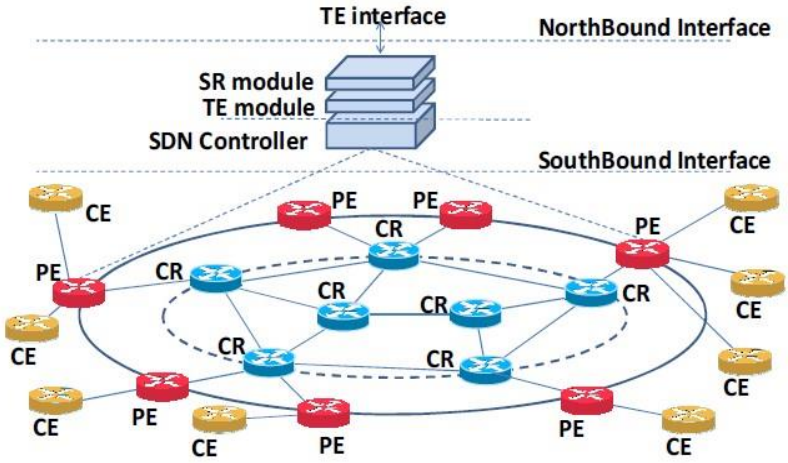


Fig. 6. TE in SDN-enabled network [17]

Packet forwarding is based on the MPLS label switching and no modifications to the MPLS plane are required. The PE & CE routers will act as hybrid SDN nodes. In summary, the SR forwarding is implemented through SDN by OpenFlow protocol interacting with the IP control plane of nodes.

### 5.2.2 SR-TE Implementation with SDN

The SDN controller will allocate the traffic flows through a specified bit rate and the known link capacity. For implementation, the SR/TE modules installed as shown in Fig. 6 will solve the classical path flow problem by assigning the *hop-by-hop* TE path [17]. For each TE path, the SR will instruct the flow through assigned TE paths. For the allocation of the first flow, the

*Constrained Shortest Path First (CSPF*) is utilized. The SR will continuously perform the selection of a minimal path corresponding to the TE path. This implementation will result in an enhancement of the flow rate by 20% on average and path selection accuracy increases by 10% between the source and destination PE routers.

## 6. Key Challenges

For any system, the two main parameters we need to consider in terms of efficiency are the performance and programmability or flexibility of the system. In terms of networks, performance refers to the processing speed of the node taking throughput and latency into consideration. Programmability can be termed as an ability to accept new changes and alter functional behaviors when a new set of instructions are created. Flexibility is when the system adapts to new changes or behaviors such as application protocols and security enhancements [18]. The challenges in this paper from sections 6.1 to 6.3 are described in terms of scalability, security, and interoperability respectively considering the performance and flexibility parameters as stated above.

### 6.1 Scalability

The scalability challenge for SDN controllers can be categorized into two ways such as latency, which is introduced due to the exchange of network information, second is how SDN controllers communicate with other controllers. The latency issue can be resolved or minimized by implementing the distributed or peer-to-peer [19] controller architecture, where each controller will have an equal share of communication information. To minimize the second challenge, the overall network view should be taken into consideration. Routing protocols have been designed in such a way that each network node is autonomous and only requires less knowledge regarding the network. To maintain the controller interactions, alternative paths, and secondary controller equipment are required. In this case, even if there is a failure in a single controller, there wouldn't be any disruption in the service with other controllers.

A specific solution to these scalability issues is the design of *HyperFlow* [20], a controller, which allows an event propagation system. This controller application will publish the network and forwarding information, and all other controllers replay the same set of events to maintain the state. In this way, network scalability can be achieved by sharing the resources and distribution of the state through multiple controllers.

### 6.2 Security

Potential security issues exist across the SDN platform. The SDN controllers are an attractive target for attacks like unauthorized access and malicious threats. In the absence of the robustness of the controller, there is a possibility for a potential attack like Distributed Denial-of-Service (DDoS) [21]. A security technology such as Transport Layer Security (TLS) can provide mutual authentication between the controllers and can mitigate the risk. SDN can support network forensics, security policy alteration, and security service insertion as security threats mitigation methodologies. Network forensics allows for identifying threats and managing the network intelligence by updating security policy and then reprogramming the network. Security policy alteration is where a policy is defined throughout the network elements and minimizes misconfigurations. Security service insertion can facilitate applications like firewalls and Intrusion Detection Systems (IDS) and can be applied to the traffic in the network.

*"Fortinet focusses on providing the security solutions such as firewalls and IDS integration with SDN"*

*"Cisco security solutions aims to integrate its security features with SDN as Cisco ACI, which provides compatible firewall solutions."* [22]

### 6.3 Interoperability

Interoperability can be precisely defined as the straightforward deployment of completely new infrastructure based on SDN. Each device and element in the network should be SDN-enabled. Path Computational Element (PCE) helps in the migration of traditional networks to the SDN. The PCE will move the networking node to the centralized node, and the traditional network nodes not using PCE will remain with the existing path computation solutions. With the current solutions, it is possible to integrate the SDN-enabled architectures. Various vendors have come forward and joined Open Networking Foundation (ONF) to provide SDN solutions [22] for interoperability challenges. Cisco, Dell, and Juniper have partnered to provide innovative solutions on the limited resources of traditional networks.

## 7. Future Research Directions

The SR and its use cases we reviewed in this article are focused on service programming with SR-aware and SR-Unaware technologies, transport paths for L2VPN and L3VPN services, and IPv6 mobile user plane functionalities when SR is employed. These use cases show SR can provide efficient solutions and enhancements concerning MPLS and SRv6. The future research direction can be worth exploring in terms of use cases like Service Function Chaining (SFC), SRv6 end-to-end implementations, Cloud orchestration, SR in 5G, and IoT applications. The SDN use cases were reviewed in terms of resource allocation in cloud computing, and cloud-native network slicing. Further research on SDN can be extended by focusing on areas such as updating and modernizing cloud networks and enhancing QoS constraints for web-based services.

## 8. Conclusion

Internet traffic and applications demand high performance concerning network management, scalability, and traffic engineering. To better meet these demands and replace the current routing schemes, the SR paradigm plays a pivotal role by being highly flexible and efficient. SR implementation can resolve many traffic engineering problems and facilitates path failure protection and restoration, service functioning, and network monitoring. SR also proves to perform better when compared to traditional routing schemes and improves performance metrics such as link utilization and network resiliency. Besides, SDN also plays an important role and has led to many opportunities in networking that enabled unique configuration techniques with its separation of control and data planes as a key feature. SDN can be deployed in wireless carrier networks, and wireless mesh networks, which provide high reliability. Furthermore, the emergence of SDN has brought many advantages to cloud data centers in terms of utilization and efficiency. In conclusion, the implementations of SDN and SR has been shown to significantly enhance the performance of networks. With greater flexibility and centralized control, SDN and SR provide a more efficient and scalable approach to network management, resulting in improved network performance and reliability.

## 9. References


[1] Pier Luigi Ventre, S Salsano, “Segment Routing: A Comprehensive Survey of Research Activities, Standardization Efforts and Implementation Results”, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 23, NO. 1, FIRST QUARTER 2021

[2] F. Clad, X. Xu, C. Filsfils, D. Bernier, “Service Programming with Segment Routing” IETF, Feb. 2023

[3] Z. N. Abdulla, I. Ahmad, “Segment Routing in Software-Defined Networks: A Survey”, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 21, NO. 1, FIRST QUARTER 2019.

[4] S. Matsushima, C. Filsfils, M. Kohno, P. Camarillo, “Segment Routing IPv6 for Mobile User Plane”, IETF, Jan. 2023

[5] Wengfang Xia, Y Wen, Dusit Niyato, “A Survey on Software-Defined Networking”, IEEE COMMUNICATION SURVEYS & TUTORIALS, VOL. 17, NO. 1, FIRST QUARTER 2015

[6] Diego Kreutz, Paulo Esteves, Siamak Azodolmolky, “Software-Defined Networking: A Comprehensive Survey”, Proceedings of the IEEE | Vol. 103, No. 1, Jan. 2015

[7] Kuntal Gaur, P Choudary, Ayush Jain, Pradeep Kumar, “Software Defined Networking: A review on Architecture, Security and Applications”, IOP Conference Series: Materials Science and Engineering, Mar. 2021

[8] Claudio Urrea, David Benitez, “Software-Defined Networking Solutions, Architecture and Controllers for the Industrial Internet of Things: A Review”, Library of Medicine, National Center for Biotechnology Information, Oct. 2021

[9] Arwa Mohamed, Mosab Hamdan, Suleman Khan, “SDN for resource allocation in cloud computing: A Survey”, Elsevier, 2021.

[10] K. Govindarajan, K.C. Meng, H. Ong, W.M. Tat, S. Sivanand, L.S. Leong, “Realizing the Quality of Service (QoS) in Software-Defined Networking (SDN) based Cloud infrastructure”, 2nd Int. Conf. Inf. Commun. Technol. ICoICT 2014, 2014, pp. 505–510, 2014

[11] Syed Danial Ali Shah, Mark A. Gregory, Shuo Li, “Cloud-native Network Slicing using SDN based Multi-Access Edge Computing: A Survey”, IEEE Access, 10.1109/ACCESS.2021.3050155, Jan. 2021

[12] S. Zhang, “An overview of network slicing for 5G”, IEEE Wireless Commun., vol. 26, no. 3, pp. 111117, Jun. 2020

[13] Feng, Q. Pei, F. R. Yu, X. Chu, J. Du, and L. Zhu, “Dynamic network slicing and resource allocation in mobile edge computing systems”, IEEE Trans. Veh. Technol., vol. 69, no. 7, pp. 78637878, Jul. 2020

[14] N. Kitsuwan and E. Oki, “Implementation of traffic splitting using meter table in software-defined networking”, J. Eng., vol. 2017, no. 12, pp. 662665, Dec. 2017

[15] Duo Wu, Lin Cui, “A comprehensive survey on Segment Routing Traffic Engineering”, ScienceDirect, Feb. 2022

[16] Ming-Chieh Lee, Jang-Ping Sheu. “An efficient routing algorithm based on SR in SDN”, Elsevier, Apr. 2016

[17] Luca Davoli, Luca Veltri, Pier L Ventre, Stefano Salsano, “Traffic Engineering with Segment Routing: SDN-based Architecture Design and Open-Source Implementation”, Conf. Publishing Service, 2021

[18] Yousef Fazea, Fathey Mohammed, “Software Defined Networking based Information Centric Networking: An Overview”, IEEE, International Congress of Advanced Technology and Engineering (ICOTEN), 2021

[19] Sakir Sezer, Sandra Scott, Barbara Fraser, David Lake, “Implementation Challenges for SDN”, IEEE Communications Magazine, Jul. 2013

[20] A. Tootoonchian and Y. Ganjali, “HyperFlow: A Distributed Control Plane for Open-Flow” ,Proc. 2010 Internet Network Management Conf. Research on Enterprise Networking, 2010.

[21] Namita Ashodia, Kishan Makadia, “Detection and Mitigation of DDoS attack in Software Defined Networking: A Survey”, Proceedings of the International Conference on Sustainable Computing and Data Communication Systems (ICSCDS-2022), IEEE, Apr. 2022

[22] B Sokappadu, A Hardin, A Mungur, “Software Defined Networks: Issues and Challenges”, Conference on Next Generation Computing Applications (NextComp), September 2019

## 10. Declaration

I, Nihal Kulkarni, hereby declare that the course final project is my original work, and the sources have been cited and acknowledged appropriately.