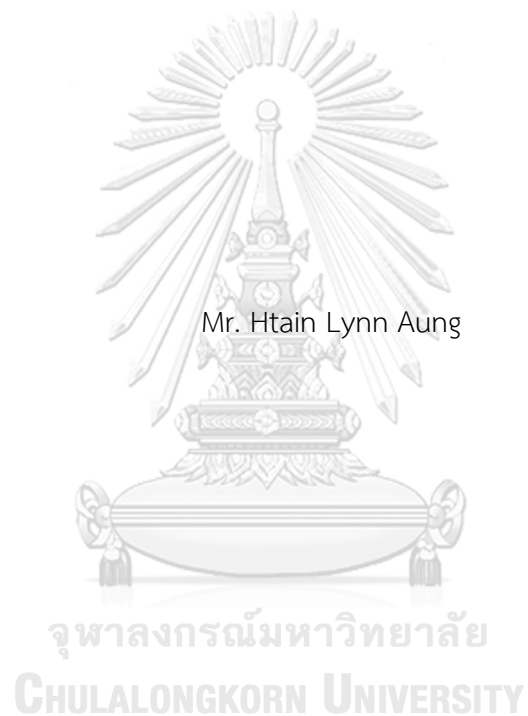


IMPLEMENTATION OF TRAFFIC ENGINEERING WITH SEGMENT ROUTING AND  
OPENDAYLIGHT CONTROLLER ON EMULATED VIRTUAL ENVIRONMENT NEXT  
GENERATION (EVE-NG)



A Thesis Submitted in Partial Fulfillment of the Requirements  
for the Degree of Master of Engineering in Electrical Engineering

Department of Electrical Engineering

FACULTY OF ENGINEERING

Chulalongkorn University

Academic Year 2020

Copyright of Chulalongkorn University

การประยุกต์ใช้วิศวกรรมโทรคมนาคมกับการจัดเส้นทางเซกเมนต์และตัวควบคุมโอเพนเดย์ไลต์บน  
สภาพแวดล้อมเสมือนจำลอง-ยุคลัดไป (อีวีอี-เอ็นจี)



วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต  
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า  
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย  
ปีการศึกษา 2563  
ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย



เอียน ลิน ออง : การประยุกต์ใช้วิศวกรรมทราฟฟิกร่วมกับการจัดเส้นทางเซกเมนต์และตัวควบคุมโอเพนเดย์ไลต์บนสภาพแวดล้อมเสมือนจำลอง-ยุคถัดไป (อีวีอี-เอ็นจี). ( IMPLEMENTATION OF TRAFFIC ENGINEERING WITH SEGMENT ROUTING AND OPENDAYLIGHT CONTROLLER ON EMULATED VIRTUAL ENVIRONMENT NEXT GENERATION (EVE-NG)) อ.ที่ปรึกษาหลัก : ลัญฉกร วุฒิสีทธิกุลกิจ

ผู้ให้บริการอินเทอร์เน็ตและเครือข่ายในระดับองค์กรเผชิญกับการเปลี่ยนแปลงและการเติบโตอย่างรวดเร็วของอินเทอร์เน็ต ซึ่งนำไปสู่ความซับซ้อนของเครือข่ายในการดำเนินงานเพื่อรองรับแอปพลิเคชันที่ต้องมีข้อตกลงระดับการบริการที่เข้มงวด การจัดเส้นทางเซกเมนต์เป็นเทคโนโลยีการกำหนดเส้นทางที่ต้นทางที่สามารถเอาชนะข้อเสียของเครือข่ายมัลติโพรโตคอล เลเบล สวิตชิงแบบเดิมในด้านความสามารถในการปรับขนาด ความยืดหยุ่น และการนำไปประยุกต์ใช้ในเครือข่ายกำหนดด้วยซอฟต์แวร์ โดยการจัดเส้นทางเซกเมนต์ช่วยให้อุปกรณ์ต้นทางสามารถส่งเส้นทางได้ด้วยการใช้เซกเมนต์หรือรายการเซกเมนต์เพื่อผ่านเครือข่าย เซกเมนต์สามารถใช้งานได้ในระบบไอพีวี 6 และมัลติโพรโตคอล เลเบล สวิตชิง เซกเมนต์สามารถกำหนดเป็นข้อมูลที่ส่งให้โหนดที่มีความสามารถจัดเส้นทางเซกเมนต์ดำเนินการบนแพ็กเก็ตขาเข้าได้ ในการจัดเส้นทางเซกเมนต์นั้นส่วนหัวของแพ็กเก็ตมีคำสั่งเพียงพอสำหรับการเปลี่ยนแพ็กเก็ตเกิดจากต้นทางไปยังปลายทาง ดังนั้นการจัดเส้นทางเซกเมนต์จึงไม่ต้องการโพรโตคอลการส่งสัญญาณที่แยกจากกัน และไม่รักษาสถานะเส้นทางในเราเตอร์ระดับกลาง

เนื่องจากค่าเริ่มต้นของการจัดเส้นทางเซกเมนต์เป็นการกำหนดแบบหลายเส้นทางที่มีต้นทุนเท่ากันซึ่งอาจทำให้เกิดการใช้งานลิงก์ในเครือข่ายสูงสุดในระดับสูง จึงหลีกเลี่ยงการใช้หลายเส้นทางที่มีต้นทุนเท่ากันและใช้การจัดเส้นทางเซกเมนต์แบบเข้มงวด วิทยานิพนธ์นี้ศึกษาแบบจำลองโปรแกรมเชิงเส้นจำนวนเต็มในบทความ [1] เพื่อประเมินประสิทธิภาพวิศวกรรมทราฟฟิกในเครือข่ายที่มีการจัดเส้นทางเซกเมนต์ และปรับปรุงแบบจำลองการโปรแกรมเชิงเส้นจำนวนเต็มซึ่งบังคับให้เลือกเส้นทางที่สั้นที่สุดเท่านั้นเพื่อลดการใช้งานสูงสุด วิทยานิพนธ์นี้เปรียบเทียบผลลัพธ์ของแบบจำลองโปรแกรมเชิงเส้นจำนวนเต็มของ [1] และแบบปรับปรุงที่เสนอ ผลลัพธ์แสดงให้เห็นว่าเราสามารถได้การใช้งานสูงสุดได้เกือบเท่ากับแบบจำลองที่เสนอของ [1] เนื่องจากความลึกของรายการเซกเมนต์ที่จะต่อท้ายเพื่อสร้างเส้นทาง SR-TE แบบเข้มงวดได้ถูกนำมาเป็นส่วนหัวของแพ็กเก็ต อีกทั้งเรายังลดจำนวนของความลึกของรายการเซกเมนต์ที่จะผนวกสุดท้ายนี้เราจะพัฒนาแอปพลิเคชันเพื่อใช้แบบจำลองโปรแกรมเชิงเส้นจำนวนเต็มที่ปรับปรุงแล้วของเราในสภาพแวดล้อมที่จำลองโดยใช้เราเตอร์เชิงพาณิชย์ ตัวควบคุมเครือข่ายที่กำหนดด้วยซอฟต์แวร์และการให้บริการโพสต์แมน

สาขาวิชา วิศวกรรมไฟฟ้า

ลายมือชื่อนิสิต .....

ปีการศึกษา 2563

ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

# # 6170512221 : MAJOR ELECTRICAL ENGINEERING

KEYWORD: Segment Routing, Integer Linear Programming, Multiprotocol Label Switching, Equal-cost Multipath

Htain Lynn Aung : IMPLEMENTATION OF TRAFFIC ENGINEERING WITH SEGMENT ROUTING AND OPENDAYLIGHT CONTROLLER ON EMULATED VIRTUAL ENVIRONMENT NEXT GENERATION (EVENG). Advisor: Assoc. Prof. LUNCHAKORN WUTTISITTIKULKIJ, Ph.D.

Internet service providers and enterprise networks face rapid changes and rapid growth of the internet, and the networks become complex in operations to support the strict Service-level Agreements (SLAs) needed applications. Segment Routing (SR) is a source routing technology that overcomes the conventional Multiprotocol Label Switching (MPLS) networks' drawbacks in scalability, flexibility, and applicability in Software-defined Networking (SDN). SR enables the source device to instruct the path using a segment or list of segments to go through the network. SR can be implemented in IPv6 and MPLS. A segment can be defined as information that instructs SR capable nodes to execute on the incoming packet. In SR, the packet header has enough instruction for packets to traverse from source to destination. So, SR does not need separate signaling protocols and does not maintain the path state in the intermediate routers.

As the default forwarding of SR, equal-cost multi-path (ECMP), can cause higher maximum utilization of links in the network, ECMP is avoided, and strict SR paths are used. This thesis studies ILP models in the paper [1] to evaluate traffic engineering performance in SR networks and enhance the integer linear programming (ILP) model, which forces to choose among only shortest paths to reduce the maximum utilization. This thesis compares the results of the ILP models of [1] and the proposed enhanced version. Results show that we can achieve the maximum utilization as nearly as the proposed model of [1]. As the Segment List Depth (SLD) to be appended to form a strict SR-TE path introduces the packet overhead, we also reduce the number of SLD to be appended. Finally, we will develop an application to implement our proposed enhanced ILP model in an emulated environment using commercial routers, SDN controller, and Postman.

Field of Study: Electrical Engineering

Student's Signature .....

Academic Year: 2020

Advisor's Signature .....

## ACKNOWLEDGEMENTS

Firstly, I am extremely grateful to my supervisor, Assoc. Prof. Lunchakorn Wuttisittikulij for patient support and valuable advice. Your expertise and suggestions were valuable and helped me to enjoy my journey at Chulalongkorn University.

My research would have been impossible without the aid and support of the scholarship program for ASEAN countries of Chulalongkorn University to study for a Master's Degree in Chulalongkorn University. I also would like to express my gratitude to Electrical Engineering Department for financial support.

I am grateful to my colleagues from Smart Wireless Communication Ecosystem Research Unit who gave valuable suggestions and advice during our meeting.

I also would like to thank the committee members of my thesis examinations for your valuable comments and suggestion.

In addition, I would like to thank my mother and sister for their support. I would like to thank Ms. Phetphailin Watthanapradit who helped, supported unconditionally, and encouraged me. My friends helped me during my journey at Chulalongkorn University. I thank Dr. Ei Ei Tun, Dr. Ei Ei Mon, and Dr. Aye Myat Myat Paing who helped me immeasurably in applying for the scholarship and completing my dissertation. I would like to thank Mr. Pruk Sasithong and Mr. Amir Pamianifard for their valuable help and suggestions. Finally, I would like to thank Mr. Soe Ye Htet, Mr. Kyaw Myo Thet, Mr. Saw Mar Nay Gaw Taw, Mr. Nyan Lynn Shwe, and Mr. Win Htet Aung who provided me discussion and help for my thesis.

Htain Lynn Aung

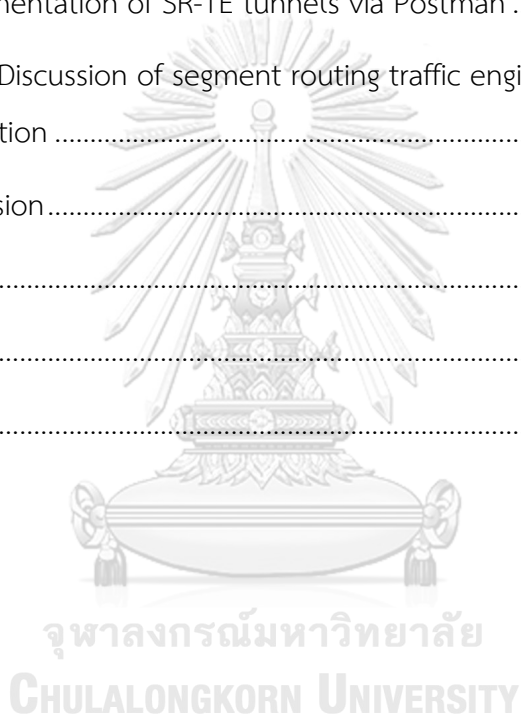
## TABLE OF CONTENTS

	Page
ABSTRACT (THAI).....	iii
ABSTRACT (ENGLISH).....	iv
ACKNOWLEDGEMENTS.....	v
TABLE OF CONTENTS.....	vi
List of Tables.....	ix
List of Figures.....	x
List of Abbreviations.....	xii
Chapter 1 Introduction.....	1
1.1: Motivation.....	2
1.2: Problem Statement.....	4
1.3: Objective.....	5
1.4: Scope of the Thesis.....	6
1.5: Contributions.....	7
1.6: Literature Review.....	7
1.7: Thesis Layout.....	9
Chapter 2 Background.....	10
2.1 Multiprotocol Label Switching (MPLS).....	10
2.1.1 MPLS Operations.....	11
2.1.2 MPLS Label Distribution Protocol (LDP).....	15
2.1.3 Resource Reservation Protocol.....	15
2.2 Segment Routing (SR).....	18

2.2.1 Segment Routing Architecture .....	19
2.2.1.1 Segment Routing Data Plane .....	20
2.2.1.2 Segment Routing Control Plane .....	23
2.2.2 Segment Routing Data Plane Operations.....	23
2.2.3 IGP Segments Segment Routing Use Cases .....	24
2.2.3.1 Simple Transport Paths .....	24
2.2.3.2 Traffic Engineering .....	25
2.3 LDP vs RSVP-TE vs SR .....	27
2.4 Path Computation Element (PCE).....	28
2.4.1 PCE Communication Protocol (PCEP) for Segment Routing.....	29
2.5 Border Gateway Protocol Link-State .....	30
2.6 Software-Defined Networking (SDN) .....	31
2.6.1 Software-Defined Networking Architecture.....	33
Chapter 3 Traffic Engineering using SR-TE in Segment Routing Networks.....	36
3.1 Equal-cost multi-path routing (ECMP) ILP Model .....	36
3.2 SHP ILP Model.....	41
3.3 SHP_En ILP Model.....	43
3.4 SEGMR ILP Model .....	46
3.5 Results and Discussion .....	50
Chapter 4 Implementation of Segment Routing network on Emulated Virtual Environment - Next Generation (EVE-NG).....	70
4.1: Emulated Virtual Environment - Next Generation (EVE-NG).....	70
4.2: Cisco IOS XRV .....	72
4.3 Creating a network topology .....	72



4.4 Setting up SDN Controller .....	73
4.5 Connecting the topology and the SDN controller .....	73
4.6 Installing SR, PCEP and BGP-LS.....	74
4.7 Configuration in OpenDaylight.....	75
4.8 Implementation of SR-TE .....	76
4.8.1 SHP_En Traffic Engineering Application .....	76
4.8.2 Implementation of SR-TE tunnels via Postman .....	82
4.9 Results and Discussion of segment routing traffic engineering (SR-TE) tunnel implementation .....	84
Chapter 5 Conclusion.....	90
5.1 Conclusion .....	90
REFERENCES .....	92
VITA.....	97



## List of Tables

	Page
Table. 1. Comparison of SR to LDP/RSVP-TE .....	27
Table. 2. Notations and Meaning of ECMP, SHP, SHP_EN .....	38
Table. 3. SID List for Connection 0-5.....	41
Table. 4. SID List .....	43
Table. 5. Notations and Meaning for SEGMR .....	46
Table. 6. Nodes and Links of Topology.....	51
Table. 7. Maximum Utilization for Same Traffic Demand .....	57
Table. 8. Maximum Utilization for Random Traffic Demand .....	58
Table. 9. Mean Utilization for Same Traffic Demand .....	59
Table. 10. Mean Utilization for Random Traffic Demand.....	60
Table. 11. Mean Hop Count for Same Traffic Demand .....	61
Table. 12. Mean Hop Count for Random Traffic Demand .....	62
Table. 13. Maximum SLD for Same Traffic Demand.....	63
Table. 14. Maximum SLD for Random Traffic Demand.....	64
Table. 15. Program Running Time for Same Traffic Demand .....	65
Table. 16. PuLP Time for Same Traffic Demand .....	66
Table. 17. Program Running Time for Random Traffic Demand .....	67
Table. 18. PuLP Time for Random Traffic Demand.....	68
Table. 19. Number of Constraints .....	69
Table. 20. IP address and Node-SID Information of Routers .....	75

## List of Figures

	Page
Fig. 1. Wireshark Capture of Multiprotocol Label Switching Header.....	11
Fig. 2. MPLS topology in EVE-NG emulator .....	12
Fig. 3. R2 MPLS label local bindings .....	13
Fig. 4. Label Forwarding Information Base (LFIB).....	13
Fig. 5. R2 MPLS local and remote labels bindings.....	14
Fig. 6. MPLS Label Operations.....	14
Fig. 7. MPLS Traffic Engineering Topology .....	16
Fig. 8. RSVP Signaling Capture at R1 e0/0 interface.....	17
Fig. 9. Intermediate nodes maintaining LSPs (a) R2 and (b) R3.....	17
Fig. 10. A segment .....	19
Fig. 11. An ordered list of Segments.....	19
Fig. 12. Types of IGP Segment.....	20
Fig. 13. IGP-Node Segment illustration.....	21
Fig. 14. IGP Anycast Segment .....	22
Fig. 15. Simple MPLS Transport Services .....	25
Fig. 16. SR-TE tunnel set up in centralized approach by SDN Controller.....	26
Fig. 17. SDN Architecture in Simple View.....	32
Fig. 18. Software-Defined Network Architecture in (a) planes and (b) layers [33].....	34
Fig. 19. ECMP Demonstrations.....	37
Fig. 20. 2 x 3 Topology for ECMP .....	39
Fig. 21. SHP Demonstrations.....	42
Fig. 22. SHP_En Demonstrations .....	44

Fig. 23. SL Computation .....	45
Fig. 24. Unique shortest paths .....	48
Fig. 25. Specifications of Server .....	52
Fig. 26. Topologies (a) Grid 2 x 2 (b) 3 x 3 (c) 4 x 4 (d) 5 x 5 (e) Eurocore.....	54
Fig. 27. Experimental Topology .....	71
Fig. 28. System Design.....	73
Fig. 29. Experimental Topology in EVE-NG emulator .....	74
Fig. 30. BGP Configuration at ODL .....	76
Fig. 31. SHP_En Traffic Engineering Application GUI .....	78
Fig. 32. Traffic Demand.....	78
Fig. 33. Infeasible Traffic.....	79
Fig. 34. Optimal Traffic Demand.....	80
Fig. 35. SR-TE Paths Information.....	81
Fig. 36. XML Files Generated by Application .....	81
Fig. 37. SR-TE Tunnel Setup from Postman .....	82
Fig. 38. XML Configuration for PE1-PE2 SR-TE tunnel .....	83
Fig. 39. Static Route Configuration for SR-TE tunnel at (a) PE1 and (b) PE2.....	84
Fig. 40. Verifying SR-TE tunnels.....	86
Fig. 41. Traceroute results (a) PE1 to P3 (b) PE1 to PE2 (c) PE1 to P2 (d) PE1 to P4 (e) PE2 to PE1 (f) PE2 to P3 and (g) PE2 to P2.....	88
Fig. 42. Verifying intermediate routers (a) P1 (b) P2 (c) P3 (d) P4 (e) P5 and (f) P6.....	89

## List of Abbreviations

API	Application Programming Interface
AS	Autonomous System
BGP	Border Gateway Protocol
BGP-LS	Border Gateway Protocol Link-State
COS	Class of Service
CPU	Central Processing Unit
CSPF	Constrained Shortest Path First
ECMP	Equal-cost Multi-path Routing
ERO	Explicit Route Object
FEC	Forwarding Equivalence Class
GMPLS	Generalized Multi-Protocol Label Switching
GUI	Graphical User Interface
iBGP	Internal BGP
IGP	Interior gateway protocol
ILP	Integer Linear Programming
IP Address	Internet Protocol address
IS-IS	Intermediate System to Intermediate System
LDP	Label Distribution Protocol
LER	Label Edge Router
LSP	Label-switched Path
LSR	Label Switch Router
MPLS	Multiprotocol Label Switching
NAT	Network Address Translation
NETCONF	Network Configuration Protocol
NOS	Network Operating System
ODL	OpenDaylight
ONOS	Open Network Operating System

OOS	Out-of-Sequence
OSPF	Open Shortest Path First
PCC	Path Computation Client
PCE	Path Computation Element
PE	Provider Edge
PHP	Penultimate Hop Popping
PPP	Point-to-Point Protocol
REST	Representational State Transfer
RR	Route reflector
RSVP	Resource Reservation Protocol
SDN	Software-defined Networking
SID	Segment ID
SLA	Service-level Agreement
SLD	Segment List Depth
SPF	Shortest Path First
SPOF	Single Point of Failure
SR	Segment Routing
SRGB	Segment Routing Global Block
SRH	Segment Routing Header
SR-TE	Segment Routing Traffic Engineering
TE	Traffic Engineering
TED	Traffic Engineering Database
TTL	Time to live
VM	Virtual Machine
VPN	Virtual Private Network
WAN	Wide Area Network
XML	Extensible Markup Language

# Chapter 1

## Introduction

Today applications such as video conferencing, video streaming need strict Service Level Agreements (SLAs) for latency, bandwidth or loss, etc. Internet service providers and enterprise networks face rapid changes and rapid growth of the internet, and the networks become complex in operations to support these strict SLAs needed applications. The operations become complex in a traditional IP/MPLS network because of complicated traffic engineering configurations and a lack of scalability and flexibility. Service providers and enterprises are demanding more scalable, flexible, and programmable solutions to keep up with the rapidly changing demand to support demanding applications.

Segment Routing (SR) [2] is a source routing technology that enables the source device to instruct the path using a segment or list of segments to go through the network. SR can be implemented in IPv6 and MPLS. A segment can be defined as information that instructs SR capable nodes to execute on the incoming packet. In SR, the packet header has enough instruction for packets to traverse from source to destination. So, SR does not need any signaling protocols and does not maintain the path state in the intermediate routers.

SR [2] keeps the network's core very light and stateless because SR [2] does not need heavy signalings such as Label Distribution Protocol (LDP) and Resource Reservation Protocol (RSVP). SR [2] helps overcome the downside of traditional IP and MPLS networks in scalability, flexibility, and applicability in Software-Defined Networking (SDN). SR [2] simplifies the transport layer by extending existing routing protocols such as Open Shortest Path First (OSPF) or Intermediate System to Intermediate System (ISIS). SR is a source-based routing that makes the IP/MPLS and IPv6 simpler and run scalier.

SDN is a technology that introduces capabilities to program and automates the traditional network infrastructures to enable end-to-end management [3]. Existing traditional network infrastructures lack automation and programmability because the control plane and data plane are tightly coupled. SDN logically centralizes the state and network intelligence in the centralized entity called the controller and provides abstractions to the underlying network infrastructure from the application running on top of the SDN controller. The application running on top of the SDN controller collects network status from the SDN controller through Web-based REST Application Programming Interface, executes algorithms, and applies its forwarding and traffic engineering rules through Southbound APIs. Southbound APIs may include Path Computation Element (PCE) Communication Protocol (PCEP), Border Gateway Protocol Link-State (BGP-LS), OpenFlow, and NETCONF.

### 1.1: Motivation

Segment Routing has been proposed by the Internet Engineering Task Force SPRING working group as an alternative TE solution to simplify the control plane. In SR, the source node or ingress node specifies the specific path by encoding the list of Segment IDs (SIDs). Intermediate nodes only look at the top SID of the segment list to perform packet forwarding.

SR can be instantiated over MPLS without any changes and makes the control plane simpler. SR instantiation over MPLS does not require signaling protocols and improves MPLS network scalability. And also, SR over MPLS doesn't need to store the path state of the information in the intermediate routers as the source node enforces a specific path to flow packets by the list of Segment IDs (SIDs). In Segment Routing with the MPLS, an MPLS label encodes a segment, and a stack of labels encodes a list of segments. SR can also be implemented in the IPv6 data plane with Segment Routing



Header (SRH). SR supports various cases such as layer 3 and 2 Virtual Private Network (VPN) tunneling services, traffic engineering by implementing SR-TE paths, network resiliency.

Several applications are running on top of the MPLS network, requiring strict SLAs such as bandwidth, latency, loss, etc., to carry the customer traffic. In Segment Routing, IGP segments are combined to steer the packets on the traffic-engineered path. SR over MPLS data plane can be deployed to recognize the routes to meet required SLAs. However, the default behavior of SR ECMP using IGP shortest paths leads to shortcomings such as congestion, higher network resource utilization [1]. TE is used in SR to overcome these drawbacks. TE can be used to place traffic in the network effectively to improve network capability [4]. SR-TE expresses the SR path with the computed segments list in order, enables the ingress device to encode segments list, and sends the traffic along the SR-TE paths. Among traffic engineering objectives, proper utilization of network resources such as bandwidth helps to serve many user's traffic demands not by expanding the network infrastructure [4].

Traffic Engineering in Segment Routing, in addition to conventional TE objectives such as minimization of maximum utilization on all network links, several objectives have to consider such as SR-TE path to be encoded with a segment or list of segments. And also, the number of segments in the list or Segment List Depth (SLD) to be encoded on the packet introduces the packet overhead and encoding segment lists have to be minimized. Also, SR-capable MPLS routers support limited SLD numbers [1].

This thesis studies the traffic engineering performance of SR with ILP models as in [1]. This thesis will propose the enhanced version of the ILP model in [1] (explained details in section 3.3). This paper will present the proposed ILP model to minimize the maximum utilization of the network while using k-shortest paths. This paper will also propose the model to compute the segment lists to encode in the packet. Finally, this

paper will implement the enhanced ILP model in an emulated environment. Emulated Virtual Environment Next Generation (EVE-NG) [5] is used to emulate commercial IP/MPLS routers for the implementation. OpenDaylight [6] is used as a centralized SDN controller to collect topology information and implement the SR-TE tunnel deployed by the Python GUI application and the Postman [7]. Python GUI application is used to collect topology information from the controller, visualize the topology, take the user's traffic demand, and generate the XML files for SR-TE tunnels to be configured the network element through SDN controller using Postman [7].

## 1.2: Problem Statement

Traffic Engineering can be applied in the network to arrange traffic demands to improve network operational efficiency. However, Traffic Engineering in Segment Routing has to consider extra constraints such as the maximum number of SLD to be appended in the packet in addition to conventional traffic engineering objectives such as minimization of maximum utilization on the network links.

In SR-TE, the default behavior ECMP makes network resource utilization higher than avoiding ECMP. When the ECMP paths are wholly avoided, and only the shortest path is considered, the maximum utilization is higher compared to considering non-shortest paths. When non-shortest paths are considered, the number of segment lists or SLD to be appended in the header becomes larger. So, we also have to consider how to reduce the number of SLD to append. To implement commercial IP/MPLS routers in an SDN environment, we can implement SR-TE tunnels by sending RESTCONF commands to the controller. There is an open-source application by Cisco called Pathman-SR [8] to visualize and implement SR-TE tunnels. But Pathman-SR cannot reduce the number of segments to form strict SR-TE tunnels. So, the application to implement with the lower SLD is needed in addition to visualizing topology.

### 1.3: Objective

This thesis is designed to explore the SR-TE performance from maximum utilization among links using non-shortest paths with limited SLDs to be appended. In addition, we will implement the proposed enhanced model in the emulated environment using EVE-NG emulator, commercial IP/MPLS routers, SDN controller, and Python application.

#### *General Objectives:*

- To minimize the maximum utilization among links and reduce the number of SLD to be appended to the packet.
- To implement the proposed model in the emulated environment using commercial IP/MPLS routers, SDN controller, Python application, and Postman.

#### *Specific Objectives:*

- To study and explain the current/reference ILP models.
- To compare the results of the proposed model with the reference ILP models.
- To propose the enhanced version of the reference ILP model.
- To minimize the maximum utilization among links.
- To reduce the SLD.
- To apply the proposed model in various topologies.
- To implement the proposed model in EVE-NG emulator using commercial IP/MPLS routers, SDN controller, Python application, and Postman.
- To visualize the network infrastructure, execute the proposed enhanced ILP model, and implement SR-TE tunnels.

#### 1.4: Scope of the Thesis

As the default behavior, ECMP of the SR leads to higher network resource utilization, such as higher maximum utilization among links than cases where ECMP is avoided. This thesis will evaluate the maximum utilization among the links. In addition, the application to test the proposed model is also essential to test the applicability of the model. So, this study will develop the application running on top of the SDN controller to visualize the topology, take the user's traffic demand, run the proposed enhanced ILP model and generate the XML configuration files to implement SR-TE tunnels via Postman API.

This thesis studies the three ILP models in [1] and aims to enhance the second model (explained in detail in 3.2) to minimize the maximum utilization among links using ILP. In this thesis, Python PuLP [9] CBC solver is used to solve ILP models. This thesis will also compare the maximum utilization among links, mean utilization, mean hop count, the total number of constraints to solve the ILP model, the program running time, ILP solving time, and maximum SLD. This study analyzes 11 different topologies ranging from 4 nodes to 30 nodes with bidirectional links and unidirectional node pairs. This study will examine traffic demand with two types of traffic demand: same traffic demand and random traffic demands. Hop count is used as a metric in this thesis.

To implement the proposed model in an emulated environment, the thesis will use EVE-NG community edition [5] as an emulator, Cisco IOS XRV [10] as a router, and OpenDaylight controller [6] as an SDN controller. The application running on top of the SDN controller is written in Python. This thesis will use BGP/LS and PCEP as southbound protocol because the router used in this thesis does not support Openflow. The thesis aims to add the SR capability to the MPLS network in the intradomain network without changing to the network device that supports OpenFlow.

To implement SR-TE tunnels through the SDN controller, Postman is used to implement through RESTful interface.

### 1.5: Contributions

SR default behavior ECMP leads to higher maximum utilization among links of the network. The paper [1] proposes 3 ILP models, namely ECMP, SHP, and SEGMR. ECMP is the default behavior of SR and splits the traffic whenever it is possible. The SHP model is forced to choose only one route among all shortest paths, and the third model, SEGMR, chooses the sub-paths among unique shortest paths between every node pair (explained details in 3.4), which are equal to or less than maximum SLD. This thesis will enhance the SHP model to choose a path among k-shortest paths and minimize the maximum utilization as the third model, SEGMR. In addition, we will also have to reduce the SLD to be appended to the source of the SR-TE tunnel as labels appended on the packet header introduce packet overhead. This research will develop an application written in Python, namely SHP\_En Traffic Engineering, to apply the proposed model to commercial IP/MPLS routers in an emulated environment. This application will help the service providers to visualize the network, apply the proposed enhanced ILP model to minimize the maximum utilization so that the traffic demand can be placed efficiently without expanding network infrastructure.

### 1.6: Literature Review

The papers [11] and [12] proposes an algorithm for path encoding in a centralized environment. However, path encoding is only applied on shortest paths, and they do not consider paths that are longer than shortest paths. In paper [13],

ELEANOR architecture is proposed to reduce the size of segment lists to be appended to the packet. ELEANOR performs path computation, management, and label stack optimization based on an open-source project called Pathman-SR [8]. Paper [14] introduced 2-segment routing where traffic from source to destination flows exactly one intermediate node. This paper considers three cases: Traffic Matrix Oblivious, Online, and Traffic Matrix Aware Segment Routing exploiting ECMP. Finally, the primary reference paper [1] proposed 3 ILP models, namely, ECMP for default SR behavior ECMP, SHP for forcing to choose only one shortest path and SEGMR to choose the sub-paths among unique shortest paths and heuristic to perform traffic engineering efficiently by evaluating maximum utilization among links.

In [15], the demonstration of the implementation of segment routing is presented. The author implements segment routing in two testbeds. The first one is in OpenFlow switches with SR controller (specifically enhanced and designed OpenFlow controller). The second one is commercial IP/MPLS routers with the Segment Routing controller using PCE with an extended version. In [16], the segment routing implementation is emulated in the GNS3 network emulator. This study emulates the segment routing in commercial IP/MPLS routers with the OpenDaylight controller as an SR controller. The author implements traffic engineering tunnels from an API client, Postman using ERO through OpenDaylight Controller. In [17], traffic engineering with segment routing is implemented in Open Segment Router (OSR) routed with IP unicast data plane with standard MPLS operations using the ONOS controller. In [17], the authors implement based on the project [18]. Project [18] uses OpenFlow version 1.3 instead of the existing protocols such as OSPF or ISIS, and the ONOS controller discovers the topology and maintains information about the network. The project [18] does not aim to make for Generally Available for large-scale WAN SDN. In [19], the implementation of Segment Routing Traffic Engineering is demonstrated in an IP/MPLS network with distributed control and centralized control with a centralized controller. For the distributed control, the author implements SR-TE tunnels from the routers. For

centralized control, the author implements EROs from the OpenDaylight controller using the Postman API client. However, these studies lack the efficient TE algorithm to place the traffic demand efficiently on the network.

### 1.7: Thesis Layout

There are five chapters in this thesis. Chapter 1 describes thesis motivation, problem statement, objective, contributions, and literature review. Chapter 2 studies background technology such as MPLS, Segment Routing, BGP, etc. Chapter 3 discusses traffic engineering in segment routing networks and the results from 4 different ILP models and evaluates the performance. Chapter 4 explains the operating environment and implements the segment routing traffic engineering using “SHP\_En Traffic Engineering” in an emulated environment. Finally, chapter 5 will conclude the thesis. References are also described at the end of chapter 5.



## Chapter 2

### Background

#### 2.1 Multiprotocol Label Switching (MPLS)

Multiprotocol Label Switching (MPLS) is an efficient encapsulation mechanism. MPLS can operate on the so-called 2.5 layer, as shown in Fig. 1. In term, Multiprotocol means that MPLS can run on IP, Ethernet, PPP, frame relay, etc., and Label Switching means that forwarding is based on labels and no longer need to look up the longest best match and easily route the packet based on 32-bit MPLS header [20]. In a 32-bit MPLS header, 20 bits carry a specific label in the label field. There are 3 bits for Class of Service (COS), which can influence the discard algorithms applied to packet and queuing. There is 1 bit for the Stack field (S) to support the hierarchical label stack. There are 8 bits for the TTL (Time to Live) header to prevent infinite forwarding loops of MPLS like IP header. The labels in the MPLS network can be seen in Fig. 1. Combining Labels with a group of sites, bandwidth paths, and prefixes, new services such as MPLS VPN, Traffic Engineering (TE), and GMPLS are introduced [21]. Open Shortest Protocol (OSPF), Intermediate System to Intermediate System (ISIS), or Border Gateway Protocol (BGP) is needed for reachability. Labels are distributed using dedicated LDP or extending existing protocol - BGP. LDP is used to discover neighbors and exchange prefix/label information. RSVP is used in Traffic Engineering (TE). BGP is used in the MPLS VPN and needs to extend to multiprotocol BGP.

Forwarding Equivalence Class (FEC) is a group of packets that traverses the network in the same manner. In traditional IP forwarding, each router along the path will reexamine the packet to assign to an FEC. However, in MPLS forwarding, FEC's assignment to a particular packet is done only once a packet enters the network. A



label that is an index for the router to specify the next label is sent with the packet when the packet is forwarded [22].

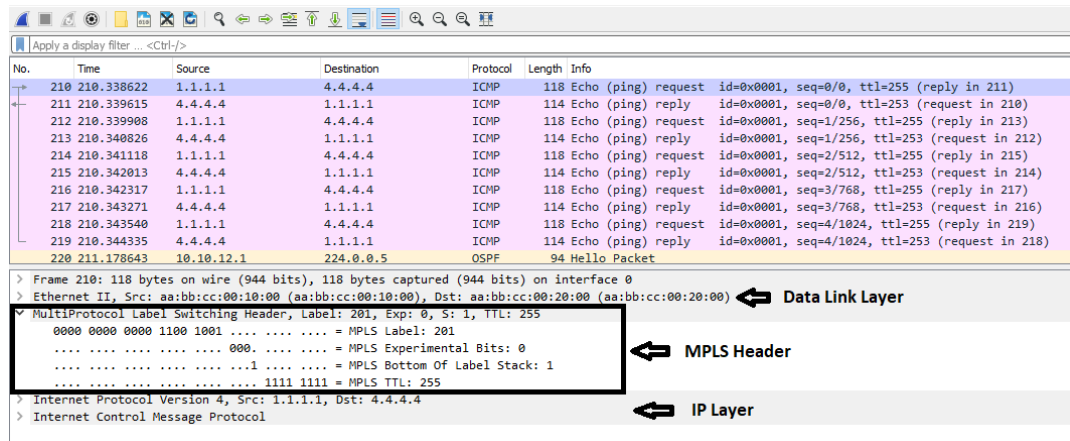


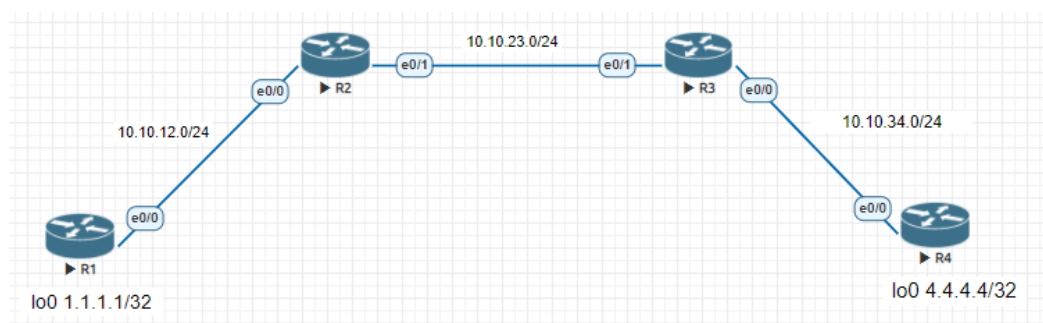
Fig. 1. Wireshark Capture of Multiprotocol Label Switching Header

MPLS Traffic Engineering (TE) is used to route the data traffic to balance the network's capacity among available resources in the network. MPLS TE is mainly used in the network where multiple back-ups or parallel paths exist in the network. TE can improve network congestion because of the traffic changed patterns such as famous football matches, emergency news, etc. TE can also help route the traffic on better network utilization by steering traffic from the non-shortest paths. TE can also improve the aggregate availability of the network.

### 2.1.1 MPLS Operations

There are three different operations treated to the packet by the MPLS router when the packet comes to the MPLS network. Three various operations are **PUSH**, **SWAP**, and **POP**. When the packet with no MPLS label arrives at the Label Switch Router (LSR), the **PUSH** appends an associate label to the incoming packet. In **SWAP**

operation, the new label is swapped with the existing label of the packet and sends the packet through the outgoing interface. Before the label reaches the egress router, the MPLS router does the **POP** operation by removing the label and sends the packet to the destination router or egress router.



*Fig. 2. MPLS topology in EVE-NG emulator*

To explain the operation of MPLS operations, the author simulates the above network topology shown in Fig. 2 in EVE-NG [5]. In this network, MPLS is enabled in the routers. When MPLS is enabled, LDP is automatically enabled. After MPLS and LDP are enabled, the labels will be generated locally. The local label bindings of R2 can be seen in Fig. 3. As shown in Fig. 3, the labels are generated by R2 locally and unique to R2. The label forwarding table of R2 can be seen in Fig. 4. From the label forwarding table, we can see that which labels will be popped or swapped. The label bindings for the MPLS network at R2 can be seen in Fig. 5. We can see the labels generated by the adjacent routers R1 and R3 and the local labels generated by R2 in Fig. 5.

```

R2#show mpls ldp bindings local
lib entry: 1.1.1.1/32, rev 15
    local binding: label: 203
lib entry: 2.2.2.0/24, rev 2
    local binding: label: imp-null
lib entry: 3.3.3.3/32, rev 8
    local binding: label: 200
lib entry: 4.4.4.4/32, rev 13
    local binding: label: 202
lib entry: 10.10.12.0/24, rev 4
    local binding: label: imp-null
lib entry: 10.10.23.0/24, rev 6
    local binding: label: imp-null
lib entry: 10.10.34.0/24, rev 10
    local binding: label: 201

```

Fig. 3. R2 MPLS label local bindings

```

R2#show mpls forwarding-table
Local      Outgoing  Prefix          Bytes Label  Outgoing  Next Hop
Label      Label     or Tunnel Id   Switched     interface
200        Pop Label  3.3.3.3/32     0            Et0/1     10.10.23.3
201        Pop Label  10.10.34.0/24  0            Et0/1     10.10.23.3
202        300       4.4.4.4/32     360          Et0/1     10.10.23.3
203        Pop Label  1.1.1.1/32     0            Et0/0     10.10.12.1

```

Fig. 4. Label Forwarding Information Base (LFIB)

MPLS label operation is demonstrated in Fig. 6. When the packet, which is a source from R1 to the loopback 0 of R4 – 4.4.4.4/32, is forwarded, the ingress Label Edge Router (LER) (which is R1) performs **PUSH** operation by adding the MPLS label (202) to the packet. Label Switch Router (LSR), which is R2 in our case, performs the **SWAP** operation by changing the existing label (202) with the new label (300). When the packet arrives at LSR R3, R3 performs Penultimate Hop Popping (PHP) by removing the label (300) before sending it to the R4, egress LER. PHP is an operation that reduces the load of the egress LER by removing the label before passing it to the LER. The one cycle of label lookup is reduced by performing PHP and reducing the LER's processing power. The path that the packet is transmitted in the network is called the Label Switch Path (LSP). LSR is a router that does **SWAP**, **PUSH**, and **POP** operations depending on the router's position.

```

R2#show mpls ldp bindings
lib entry: 1.1.1.1/32, rev 15
  local binding: label: 203
  remote binding: lsr: 1.1.1.1:0, label: imp-null
  remote binding: lsr: 3.3.3.3:0, label: 303
lib entry: 2.2.2.0/24, rev 2
  local binding: label: imp-null
lib entry: 2.2.2.2/32, rev 12
  remote binding: lsr: 3.3.3.3:0, label: 301
  remote binding: lsr: 1.1.1.1:0, label: 102
lib entry: 3.3.3.3/32, rev 8
  local binding: label: 200
  remote binding: lsr: 3.3.3.3:0, label: imp-null
  remote binding: lsr: 1.1.1.1:0, label: 101
lib entry: 4.4.4.4/32, rev 13
  local binding: label: 202
  remote binding: lsr: 3.3.3.3:0, label: 300
  remote binding: lsr: 1.1.1.1:0, label: 100
lib entry: 10.10.12.0/24, rev 4
  local binding: label: imp-null
  remote binding: lsr: 3.3.3.3:0, label: 302
  remote binding: lsr: 1.1.1.1:0, label: imp-null
lib entry: 10.10.23.0/24, rev 6
  local binding: label: imp-null
  remote binding: lsr: 3.3.3.3:0, label: imp-null
  remote binding: lsr: 1.1.1.1:0, label: 104
lib entry: 10.10.34.0/24, rev 10
  local binding: label: 201
  remote binding: lsr: 3.3.3.3:0, label: imp-null
  remote binding: lsr: 1.1.1.1:0, label: 103

```

Fig. 5. R2 MPLS local and remote labels bindings

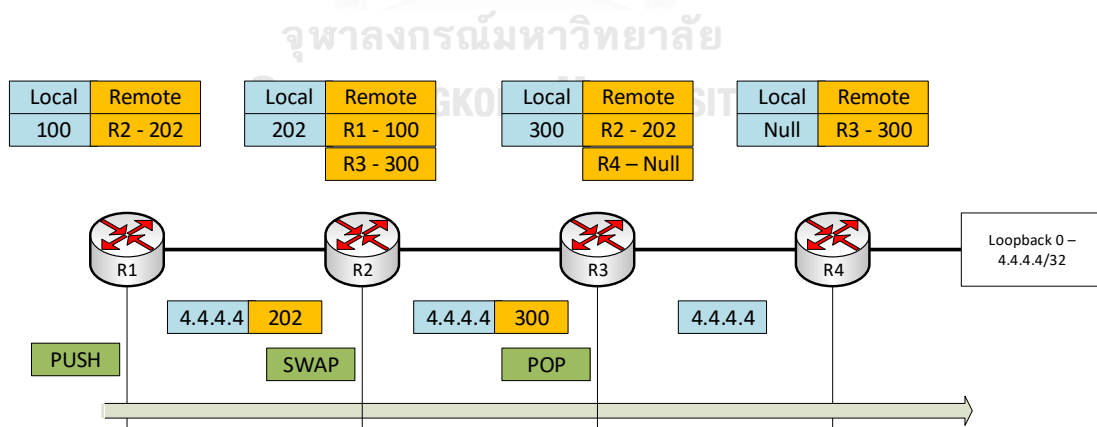


Fig. 6. MPLS Label Operations

### 2.1.2 MPLS Label Distribution Protocol (LDP)

MPLS LDP enables the peer LSRs to exchange label binding information such as the label bindings information shown in Fig. 5 to support hop-by-hop forwarding in the MPLS network. LDP allows LSRs to request, distribute, and release label prefix information to neighbor LSR and establish LDP sessions to exchange label binding information [23]. As shown in Fig. 5, the labels are locally significant, and locally significant labels make the network troubleshooting hard.

When LSRs form neighbors by exchanging hello messages, LSRs start negotiating LDP parameters. A router becomes the active router if the router has a higher transport IP address, and the other router is the passive router. After LSPs finish negotiating the LDP parameters, they form the LSPs and assign labels to each route based on the information learned from the IGPs.

### 2.1.3 Resource Reservation Protocol

RSVP enables the routers to request resource reservations from the network as a signaling protocol. RSVP is used in MPLS traffic engineering. MPLS TE is a process to route the traffic to balance among network devices in the network. Traffic engineering is helpful for network congestion due to changing network data traffic patterns such as breaking news and sports events. Traffic engineering enables better utilization of available bandwidth by routing the traffic on the non-shortest path. RSVP-TE uses the Constrained Shortest Path (CSPF) and explicit route object (ERO) to compute the paths, unlike LDP, which only depends on IGP.

When MPLS uses RSVP to signal the LSPs, the headend router signals a PATH message along the path until the tail-end router. The PATH messages reserve resources

for all the nodes on the path. The tail-end node sends a reservation (RESV) message with a label back to the headend node. To demonstrate RSVP, the topology in Fig. 7 is implemented, and we set up the path from R1 to R4 through R2 and R3. The example of Wireshark's capture of the RSVP signaling between R2 and R1 can be seen in Fig. 8. We can see that the PATH message is sent from R1(1.1.1.1) to R4 (4.4.4.4), and the RESV message is sent from the R2 interface to the R1 interface. As we can see in the output of Fig. 9, every node along the path must maintain the LSPs, and signal the session state every 30 seconds. In our topology, we can see from the output of Fig. 9 (a) and (b) that intermediate nodes R2 and R3 maintain the LSP information created for R1 to R4.

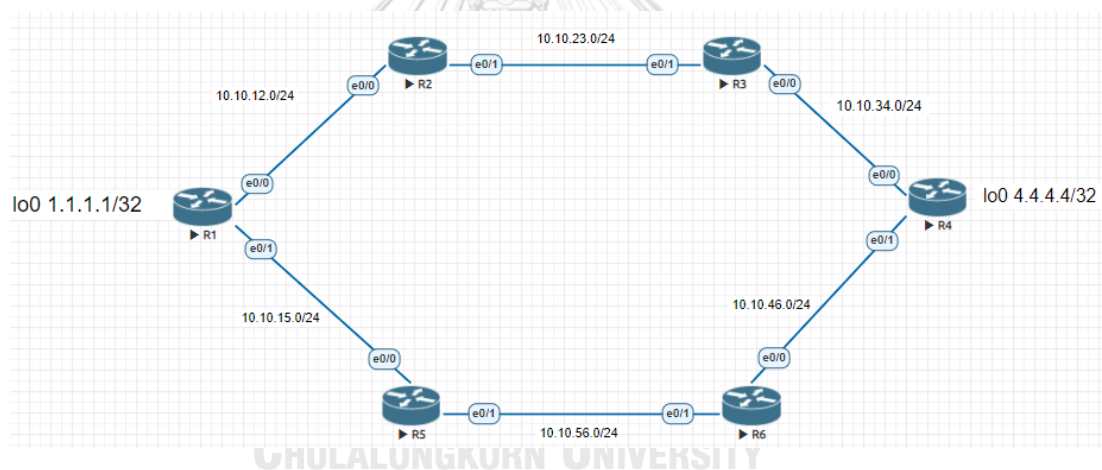


Fig. 7. MPLS Traffic Engineering Topology

No.	Time	Source	Destination	Protocol	Length	Info
18	16.663874	10.10.12.2	10.10.12.1	RSVP	142	RESV Message. SESSION: IPv4-LSP, Destination 4.4.4.4, Short Call ID 0, Tunnel ID 1, Ext ID 1010101. FILTERSPEC: IPv4-LSP, Tunnel Source: 1.1.1.1, Tunnel ID 1, Ext ID 1010101.
31	27.241683	1.1.1.1	4.4.4.4	RSVP	254	PATH Message. SESSION: IPv4-LSP, Destination 4.4.4.4, Short Call ID 0, Tunnel ID 1, Ext ID 1010101. SEND...

> Frame 18: 142 bytes on wire (1136 bits), 142 bytes captured (1136 bits) on interface 0  
 > Ethernet II, Src: aa:bb:cc:00:20:00 (aa:bb:cc:00:20:00), Dst: aa:bb:cc:00:10:00 (aa:bb:cc:00:10:00)  
 > Internet Protocol Version 4, Src: 10.10.12.2, Dst: 10.10.12.1  
 > Resource Reservation Protocol (RSVP): RESV Message. SESSION: IPv4-LSP, Destination 4.4.4.4, Short Call ID 0, Tunnel ID 1, Ext ID 1010101. FILTERSPEC: IPv4-LSP, Tunnel Source: 1.1.1.1,

Fig. 8. RSVP Signaling Capture at R1 e0/0 interface

```
R2(config)#do show mpls traffic-eng tunnels brief
Signalling Summary:
  LSP Tunnels Process:      running
  Passive LSP Listener:    running
  RSVP Process:            running
  Forwarding:              enabled
  Periodic reoptimization: every 3600 seconds, next in 2880 seconds
  Periodic FRR Promotion:  Not Running
  Periodic auto-bw collection: every 300 seconds, next in 181 seconds
TUNNEL NAME                DESTINATION    UP IF          DOWN IF        STATE/PROT
R1_t1                      4.4.4.4       Et0/0         Et0/1         up/up
Displayed 0 (of 0) heads, 1 (of 1) midpoints, 0 (of 0) tails
```

(a)

```
Router#show mpls traffic-eng tunnels brief
Signalling Summary:
  LSP Tunnels Process:      running
  Passive LSP Listener:    running
  RSVP Process:            running
  Forwarding:              enabled
  Periodic reoptimization: every 3600 seconds, next in 2789 seconds
  Periodic FRR Promotion:  Not Running
  Periodic auto-bw collection: every 300 seconds, next in 89 seconds
TUNNEL NAME                DESTINATION    UP IF          DOWN IF        STATE/PR
OT
R1_t1                      4.4.4.4       Et0/1         Et0/0         up/up
Displayed 0 (of 0) heads, 1 (of 1) midpoints, 0 (of 0) tails
```

(b)

Fig. 9. Intermediate nodes maintaining LSPs (a) R2 and (b) R3

## 2.2 Segment Routing (SR)

Segment Routing (SR) [24] is a source routing technique that instructs the ingress node to give complete information about the path to go to the egress node. A segment is an instruction that a Segment Routing enabled node can execute based on the packet that comes to the node [3]. The packet can be steered by the header information from the traffic engineering approach because the packet's header has enough instructions to guide the packets to the destination from the source. So, the intermediate nodes no longer need to maintain the states of every path in the network. The ability not to maintain states at the intermediate nodes improves the traditional MPLS network's traffic engineering's scalability, which requires fatty signaling and path states, as explained in section 2.1.3. Segment Routing can be used in the MPLS and IPv6. SR can be deployed in the SR instantiation MPLS network with no changes over the MPLS data plane. As shown in Fig. 10, an MPLS label encodes a segment, and as shown in Fig. 11, a stack of MPLS labels contains an ordered list of segments. SR Header (SRH) [25] is used to realize SR over the IPv6 data plane. However, this thesis only focuses on IPv4 segment routing, which is implemented using MPLS.

Constrained SPF and IGP Shortest Path First (SPF) that considers bandwidth, latency, and the explicit path that network operator configures can derive the SR path. Segment Routing path from the Shortest Path First algorithm is the SR-SPF path that enables ECMP shortest path and path derived from the operator's explicit configuration, or CSPF is the SR-TE path that steers the path that is not IGP SPF.



```

> Frame 4: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 0
> Ethernet II, Src: 50:00:00:02:00:02 (50:00:00:02:00:02), Dst: 50:00:00:06:00:01 (50:00:00:06:00:01)
v MultiProtocol Label Switching Header, Label: 17004, Exp: 0, S: 1, TTL: 255 ← A segment
    0000 0100 0010 0110 1100 .... .. = MPLS Label: 17004
    .... .. = MPLS Experimental Bits: 0
    .... .. = MPLS Bottom Of Label Stack: 1
    .... .. 1111 1111 = MPLS TTL: 255
> Internet Protocol Version 4, Src: 10.10.2.100, Dst: 10.10.4.100
> Internet Control Message Protocol

```

*Fig. 10. A segment*

```

> Frame 23: 122 bytes on wire (976 bits), 122 bytes captured (976 bits) on interface 0
> Ethernet II, Src: 50:00:00:01:00:01 (50:00:00:01:00:01), Dst: 50:00:00:05:00:01 (50:00:00:05:00:01)
> MultiProtocol Label Switching Header, Label: 17006, Exp: 0, S: 0, TTL: 255 } Ordered list of
> MultiProtocol Label Switching Header, Label: 17002, Exp: 0, S: 1, TTL: 255 } segments
> Internet Protocol Version 4, Src: 10.10.1.100, Dst: 10.10.2.100
> Internet Control Message Protocol

```

*Fig. 11. An ordered list of Segments*

### 2.2.1 Segment Routing Architecture

A segment is a topological or service-related instruction that SR enabled node processes on the packet that came to the node. Segment Routing uses the list of segments that steer the packet through an explicit path. A segment can guide the packet to send a packet using an explicit path to the destination, send packets through the shortest path, or a specific service instance such as firewall, DPI (Deep Packet Instruction) to get to the destination. A local segment is a locally significant segment to an SR-enabled node, and a globally significant segment inside an SR domain is a global segment. In SR architecture, there are two primary components, which are the SR data and control planes.

### 2.2.1.1 Segment Routing Data Plane

The SR data plane includes defining the procedure for encoding instructions or segments and processing the list of segments or segments to send the packet. MPLS or IPv6 header contains the list of the segments and the pointer to indicate the active segment to be executed and points to the next segment after the previous segment has been executed. A segment of the list is defined with SID – Segment ID, which can be globally or locally significant. The globally significant segment or global segment is unique within an SR domain and is advertised through the domain. The local segment is locally generated by an originated node and assigned to a label outside of Segment Routing Global Block (SRGB). For SR implementation in the MPLS network, SR enabled node maintains global segment labels in the SRGB. SRGB range, which is 16000 to 23999, can be different in each SR node, but Cisco strongly recommends the same SRGB range to ease troubleshooting and administration. Different segments can be defined in Segment Routing with MPLS, such as RSVP-TE LSP segment, IGP segments, BGP peer segment, and LDP segment. In the scope of this thesis, the thesis will only focus on IGP segments. IGP segment or IGP SID is a segment that is information that the SR node advertises, such as connected or adjacency prefixes. Types of IGP segments can be seen in Fig. 12.

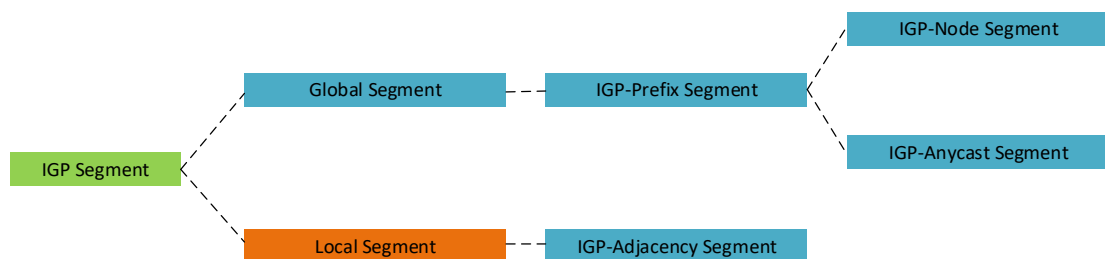


Fig. 12. Types of IGP Segment

## (i) IGP-Node Segment

IG-Prefix Segment can be the IGP-Node segment or IGP-Anycast segment [26]. IGP-prefix segment is global inside the SR domain. IGP-Node segment indicates an exact node, such as a loopback interface. The IGP-Node segment's SID is Node-SID. A Node-SID represents the instruction to send the packet to the computed path by specific algorithms such as ECMP aware shortest paths to the node. To demonstrate Node SID, the illustration is shown in Fig. 13. Let's assume that node SID for R6 is 16001. SR control plane advertises the node Segment ID 16001 in the IGP domain. When a packet is sent from node R1 to node R6, R1 pushes the SID - 16001 on the packet header. Since the node segment is globally significant inside the IGP domain, the node received the active segment - 16001 sends the packet to node R6 through ECMP-aware shortest-path.

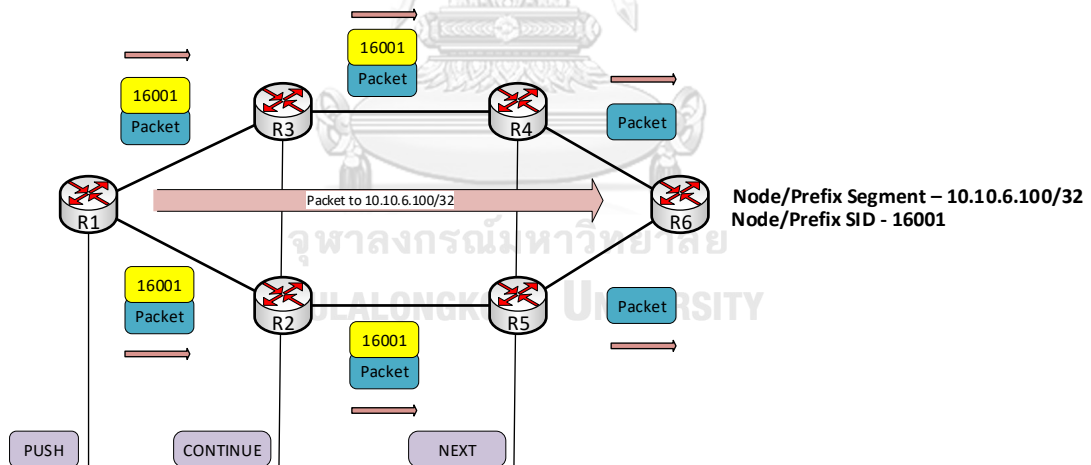


Fig. 13. IGP-Node Segment illustration

## (ii) IGP-Anycast Segment

Routers advertise the IGP Anycast segment in the anycast set. The set of routers that advertised the same IGP-Anycast segment are in the anycast set. Using the Anycast segment, ECMP-aware shortest path forwarding, and resiliency can be achieved in the

SR network. Routers included in the anycast set are set to the same loopback address and the same SID value (same Node-SID). As an example, an illustration of the Anycast segment is shown in Fig. 14. As P2 and P3 are the members of the anycast set, P2 and P3 are configured with the same anycast prefix, which is 10.10.23.23/32, and the same Anycast-SID, which is 17023. If P1 receives the packet with the Anycast-SID 17023, P1 will load-balance the packet to P2 and P3 because P2 and P3 are set of anycast sets that are ECMP-paths reachable from P1. If any router of anycast sets fails, such as P2 or P3, P1 can still send the packet with the Anycast-SID of 17023. If P2 fails, P1 can still send with the Anycast-SID of 17023 through P3, and it is also the same for the failure of P3. We can see that Anycast-SID also provides node resiliency. Anycast-SID is also useful when the TE path is not required to pass through a specific SR-TE path, and the only set of nodes or anycast set is needed to pass through.

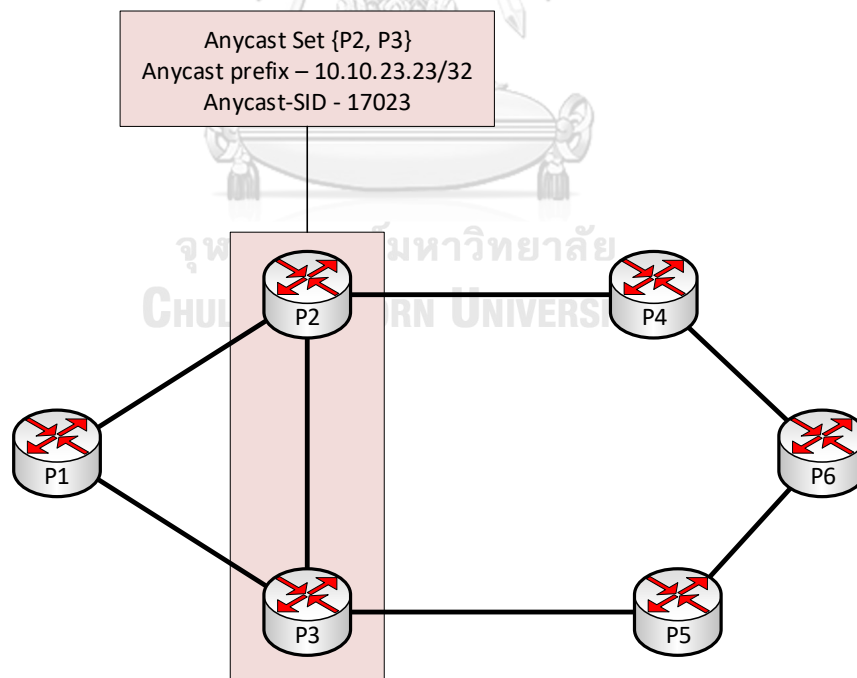


Fig. 14. IGP Anycast Segment

### (iii) IGP-Adjacency Segment

An adjacency segment is a segment advertised for an adjacency advertised by local or remote adjacent nodes in the IGP domain. Remote and local nodes create IGP adjacencies. IGP-Adjacency segment or Adj-SID is locally significant to the node that advertises the link. The use of the adjacency segment enables the packet to steer through a specific interface of a node. By combining Adj-SID and Node-SID, packets can be steered through the ECMP-aware shortest path.

#### 2.2.1.2 Segment Routing Control Plane

Link state protocols for the IGP control plane include ISIS and OSPF to exchange routing information among nodes. The Segment Routing node advertises the SIDs of each related IGP prefixes and adjacencies through the control plane of IGP. To be able to advertise segment identifiers, extensions to OSPF and ISIS routing protocols are needed. OSPF and ISIS routing protocols' extensions allow the SR node to maintain the information about all IGP segments and update the information according to the topology changes. The SR path that allows the packet depends on the IGP control plane. When one or more prefixes are configured on the SR node, the SR node must advertise the IGP SIDs for each associated prefix through the IGP control plane. Whenever the prefix is deleted from the Segment Routing node, the Segment Routing node removes the IGP SID associated with the prefix removed.

#### 2.2.2 Segment Routing Data Plane Operations

SR can be implemented on the data plane of MPLS without changes. An ingress network device appends the lists of segments on the packet to steer the desired SR

path. The receiving SR node executes the label on top of the segment list. Upon completing the SID, the top label is popped, and the next label becomes active. There will be three operations in SR data plane operations: **PUSH**, **CONTINUE**, and **NEXT**, as shown in Fig. 13 [3]. **PUSH** is an instruction that instructs to add a SID to the segment list's top. **CONTINUE** is an instruction that instructs to keep the active segment that is still processing. **NEXT** instructs to execute the following segment. Fig. 14 demonstrates how the SR data-plane operates to forward a packet from source to destination.

## 2.2.3 IGP Segments Segment Routing Use Cases

### 2.2.3.1 Simple Transport Paths

As the networking devices are fast enough to process, service providers apply the MPLS network to get the services such as Layer 3 VPN and Layer 2 VPN services. It is crucial to create fully mesh MPLS tunnels to connect any PE device to enable these services. The ingress router encapsulates the traffic Virtual Private Network header and sends the traffic over one or more tunnels, as shown in Fig. 15. When SR is implemented with an MPLS data plane, three main benefits can be achieved, such as

- (i) ECMP aware tunnels
- (ii) Simple operations because of the ability to create tunnel automatically by using only IGP and without additional protocols such as LDP and RSVP,
- (iii) Improved scalability because the intermediate node maintains less state (one IGP-node SID per PE device).

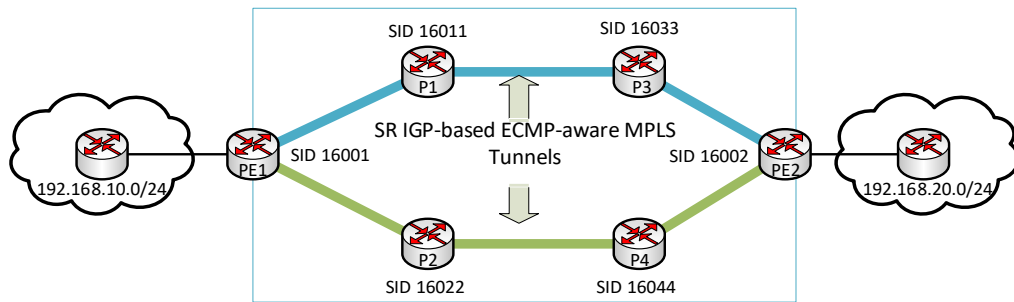


Fig. 15. Simple MPLS Transport Services

### 2.2.3.2 Traffic Engineering

Many applications run on top of the MPLS network, requiring strict Service Level Agreements (SLAs) with bandwidth, end-to-end paths loss, and latency. In a segment routing network, IGP segments that can be combined can steer the TE paths' traffic. Segment Routing provides a technique that the MPLS network's data plane will realize whether paths met the SLA requirements in a centralized or distributed manner. The SR node (headend router) computes the path subject to desired constraints based on the TE topology in a distributed approach. In a centralized system, an SR controller or SDN controller calculates the route. SR-TE route can include one or more nodes or prefix segments. SDN controller learns about the latest information on a network topology from the BGP-LS protocol. The SR-TE path calculated from the controller is deployed in an Explicit Route Object (ERO) to the headend router by communicating with Path Computation Entity Communication Protocol (PCEP) messages. A Segment Routing-Traffic Engineering tunnel set up by the SDN Controller can be seen in Fig. 16.

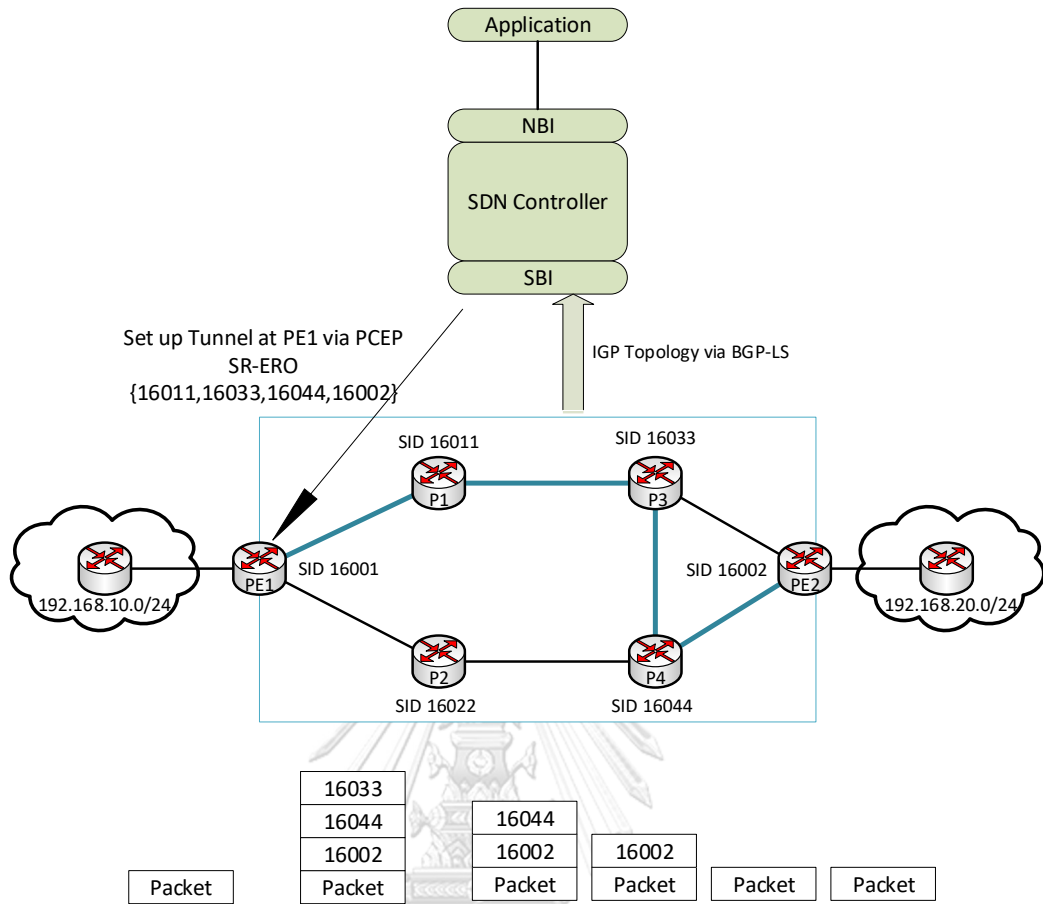


Fig. 16. SR-TE tunnel set up in centralized approach by SDN Controller



### 2.3 LDP vs RSVP-TE vs SR

Table. 1. Comparison of SR to LDP/RSVP-TE

	SR	LDP/RSVP-TE
Operational Simplicity	Simple	RSVP-TE is complex LDP is simple
Traffic Engineering	Yes	No - LDP Yes - RSVP-TE - complex tunnel configuration
TE Scalability	High – Core is stateless, and the state is in the headend node	Low – RSVP creates full LSPs, and the transit router keeps a lot of transit information
Number of Protocols	No RSVP, LDP Fewer protocols in the network	LDP, RSVP, or both
Label Allocation	Node segment – Global Adjacency segment - Local	Locally significant
ECMP	Yes	Yes – LDP No – RSVP-TE
IPv6	Native	Need extensions

Comparison among LDP, RSVP-TE, and SR can be seen in Table. 1 [27]. In a conventional MPLS network, resource reservation and signaling are made by implementing signaling protocols such as LDP and RSVP. As LDP largely depends on IGP protocol, LDP and IGP have to be synchronized, and SR removes the requirement

to be in sync by extending the IGP protocol to support SID. In Segment Routing, additional signaling protocols are not needed, and IGP link-state protocols such as OSPF and ISIS extensions are used for label distribution. SR is scalable because it does not need to keep thousands of labels in the LDP database and does not depend on additional LDP and RSVP protocols. RSVP-TE is not very popular because of the complicated tunnel configuration and is not scalable. SR-TE keeps the core network very scalable and light because of no additional protocols and signaling.

#### 2.4 Path Computation Element (PCE)

PCE architecture includes two components, which are PCE and PCC (Path Computation Client). A PCE can be an application, component, or network device and can compute the path based on topology information and apply constraints. A PCC can be any client application requested to the PCE to compute the path. PCE Communication Protocol (PCEP) [28] communicates between PCC and PCE or between two PCEs [29]. PCE can configure the paths for nodes that are configured to be PCC. PCEP defines messages and objects to transmit PCEP sessions and paths. PCEP relies on TCP to guarantee a reliable session and uses TCP port 4196. PCE can be used in RSVP-TE and SR-TE. As mentioned in previous sessions, RSVP-TE lacks scalability because every intermediate router maintains state information of every TE path. SR solves the problem of RSVP because SR does not keep every TE state information in the intermediate nodes.

Client-server communication is used between PCE and PCC. PCEP sessions include Open message, Keepalive, Path Computation Request (PCReq), Path Computation Reply (PCRep), Notification, Error message, and close. In PCEP sessions, a TCP 3-way handshake is performed first, and the nodes start to initiate the PCEP session. Parameters such as keepalive and dead timer are negotiated at initiating PCEP

session. After initiating the PCEP session, PCC sends PCReq to the PCE to compute the paths based on constraints and attributes. When PCE receives PCReq from the PCC, PCE computes the path and replies with PCRep. When the PCEP session is terminated, the underlying TCP connection is terminated.

Client-server communication is used between PCE and PCC. PCEP sessions include Open message, Keepalive, Path Computation Request (PCReq), Path Computation Reply (PCRep), Notification, Error message, and close. In PCEP sessions, a TCP 3-way handshake is performed first, and the nodes start to initiate the PCEP session. Parameters such as keepalive and dead timer are negotiated at initiating PCEP session. After initiating the PCEP session, PCC sends PCReq to the PCE to compute the paths based on constraints and attributes. When PCE receives PCReq from the PCC, PCE computes the path and replies with PCRep. When the PCEP session is terminated, the underlying TCP connection is terminated.

#### **2.4.1 PCE Communication Protocol (PCEP) for Segment Routing**

Segment Routing-Traffic Engineering can be implemented by a centralized entity called PCE and distributed approach. A centralized approach can achieve several advantages, such as effective network utilization in multidomain networks and optimized traffic engineering because of the controller's global view. In the PCEP session, information about the SR-TE path is put inside the Explicit Route Object (ERO), which contains a sub-object or ordered sequence of sub-objects. Each sub-object represents one SID. When PCC gets ERO from PCE, PCC puts the SIDs to the incoming packets. PCE computed SR-TE paths can be implemented to PCC in the following ways

1. Ordered lists of sub-objects that contain IP addresses, and in this case, PCC translates IP address to SID by looking up at the TED

2. Ordered lists of sub-objects that contain IP addresses and SIDs, and in this case, PCC also has to translate
3. Ordered lists of sub-objects that include SIDs.

## 2.5 Border Gateway Protocol Link-State

Border Gateway Protocol (BGP) is a path vector protocol and one of the essential internet protocols. BGP is used to exchange information among 16-bit ASs (Autonomous Systems). AS refers to a large network or a group of networks within a common administration. To know network topology information across multiple domains and calculate end-to-end path across domains, the controller must see the link states information across domains. To get to know the topology information across multiple domains, the controller can be configured to support ISIS and OSPF as passive IGP peers. But link-state protocols such as IS-IS and OSPF are very chatty and will give the controller frequent updates about the topology. It is challenging for the SDN controller to place as passive IGP peers as controller peers with every different IGP domain. So BGP-LS is used to carry link-state information obtained by IGP domains and shared with the controller. BGP speakers in IGP domains get the IGP link-state information and share it with the SDN controller. BGP speakers can communicate directly to the controller, or BGP speakers form internal BGP (iBGP) mesh by connecting to a dedicated BGP node called BGP Route-Reflector (RR), and BGP RR nodes share the information with the controller [30]. As the benefits offered by the BGP-LS protocol, BGP-LS helps the centralized path computation, implement SR-TE from centralized control together with PCE. BGP-LS enables sharing of segment routing network information required to complement SR-TE paths.

## 2.6 Software-Defined Networking (SDN)

A new emerging networking paradigm, Software-Defined Networking, is proposed to defeat the traditional network's limitations. In SDN, the control plane is not bundled with the data plane. Control logic is centralized to the control entity called the Network Operating System (NOS) or SDN controller, and the forwarding device remains only a pure forwarding device. The simple SDN overview can be seen in Fig. 17.

The four main pillars of SDN are

1. Decoupling of control and data plane – The control plane is not tightly coupled with forwarding devices, and forwarding devices remain pure forwarding devices.
2. Programmable network – Programmable networks can be considered as the principal value of SDN. Software applications installed on the NOS or SDN controller can program the network through NOS, which communicates with the forwarding devices through a well-defined API.
3. Centralized control – The control plane is allocated to the external entity called NOS or SDN controller.
4. Flow-based forwarding decision – Decisions for forwarding are dependent on the flow instead of the destination. A flow is a packet stream between destination and source.

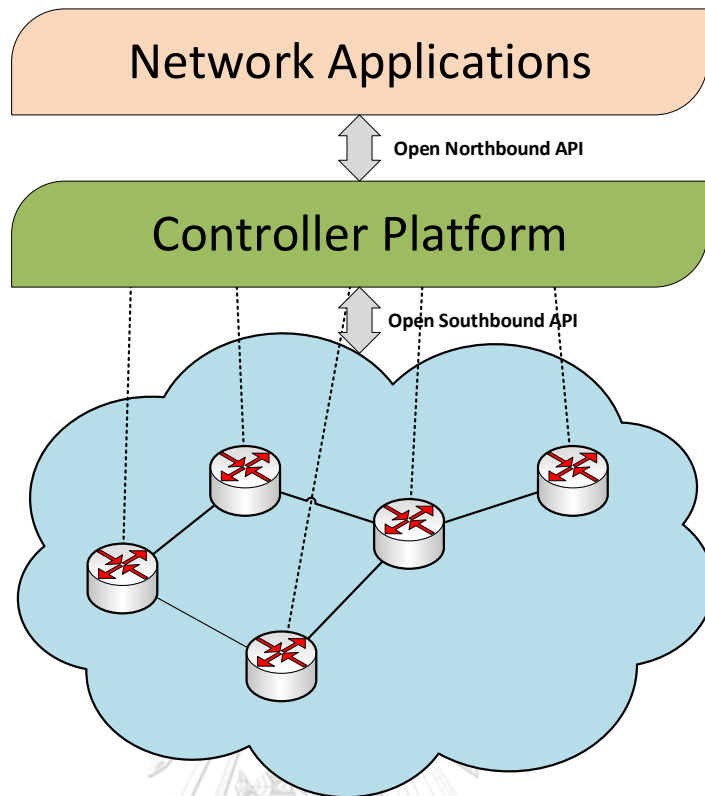


Fig. 17. SDN Architecture in Simple View

As seen in Fig. 17, the well-defined programming interface API enables the communication between the controller platform and data forwarding elements such as OpenFlow [31], BGP-LS [32], and PCEP [28]. Open northbound API allows communication between network application(s) and controller platform [33]. According to [34], three main fundamental abstractions include

1. Forwarding abstraction – Control plane such as OpenFlow [31], BGP-LS [32] and PCEP [28] enables any forwarding behavior while hiding the hardware architecture details.
2. Distribution abstraction – allows the network applications installed on the NOS from various states and centralizes the control problem.
3. Specification abstraction – allows the network applications to implement the required network action without implementing that network.

### 2.6.1 Software-Defined Networking Architecture

SDN architecture can be composed of different layers. From the Bottom-up approach, the different layers consist of network infrastructure, southbound interface, network hypervisor, network operating system, northbound interface, language-based virtualization, and network applications. Among these layers, layers such as southbound APIs, northbound APIs, NOSs, and network applications can present always in the SDN architecture [33].

In SDN architecture, as seen in Fig. 18 [33], the network infrastructure is composed of forwarding elements such as switches, routers, firewalls, and load balancers, etc. Southbound interfaces such as OpenFlow, BGP-LS, and PCEP lie between SDN controller and forwarding devices. Controllers program forwarding devices via the southbound interface. Forwarding devices are hardware or software configured to forward the packet, and the controller is a stack of software running on hardware.

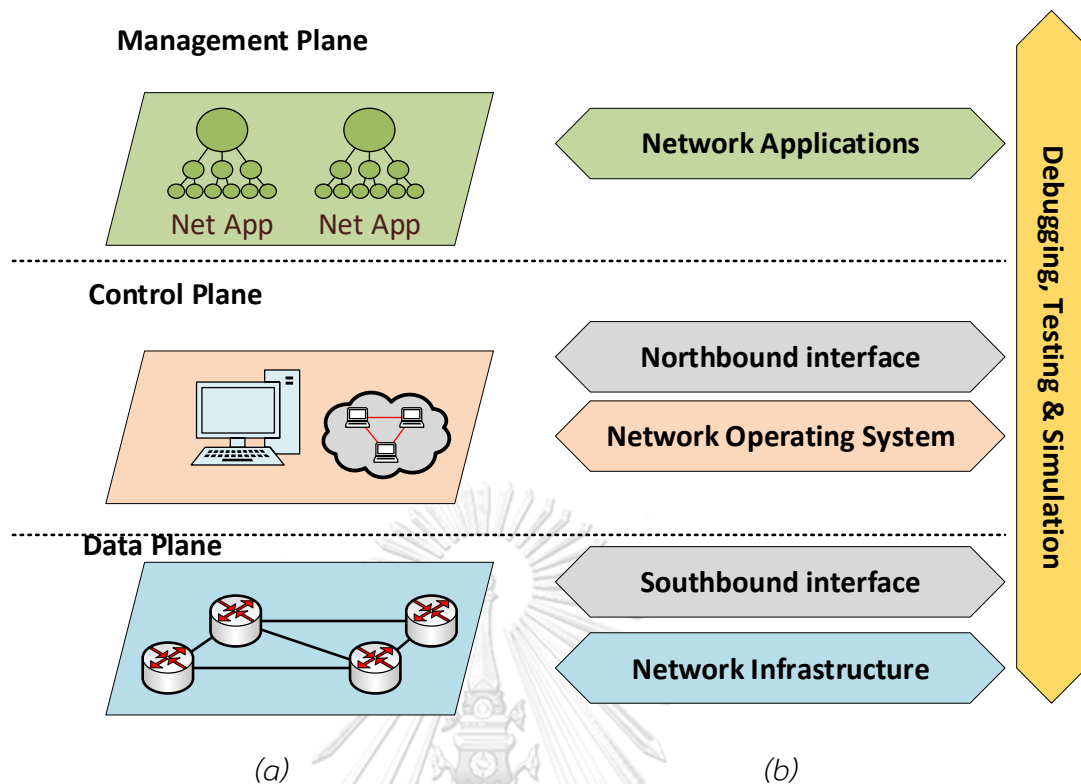


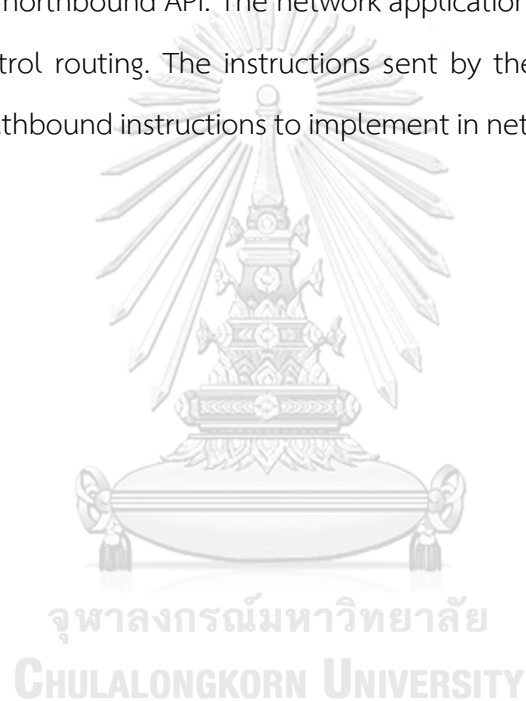
Fig. 18. Software-Defined Network Architecture in (a) planes and (b) layers [33]

The Southbound interface or Southbound API enables the connection between the controller and the forwarding devices such as routers, switches, etc. There is a Network Operating System layer or Controller layer in the control plane, "brain of the network". The controller makes it easy to solve the problems of networking and facilitates the management of the network. The controller is the critical supporting element to configure the network according to the network operator's policies. One of the critical designs of SDN control platforms is centralized and distributed platforms. For centralized controllers, the controllers manage all network elements by one controller, such as Ryu NOS [35]. As a single controller handles the whole network, there are drawbacks such as a Single Point of Failure (SPOF) and many scaling problems. Handling a large number of nodes with a single controller may not be enough. Distributed controllers such as OpenDaylight [6] can adjust to meet the requirement of potentially all kinds of network environments, from large to small



networks. In distributed design, controllers can be centralized in a cluster or physically distributed.

Northbound API is as essential as Southbound API. Northbound API helps to communicate between SDN controllers and network applications and presents a network abstraction interface to the application running on the SDN architecture. Northbound API allows network programmability from the application level. In the management plane, network applications implement operational logic and network behaviors through northbound API. The network applications can monitor the network topology and control routing. The instructions sent by the network applications are translated into southbound instructions to implement in networking devices in network infrastructure.



## Chapter 3

### Traffic Engineering using SR-TE in Segment Routing Networks

This chapter will study 3 different ILP models from [1], namely ECMP, SHP, and SEGMR. The reason to learn three other ILP models is to understand the proposed enhanced version of the SHP model as a thesis contribution. We test all 4 ILP models with 11 topologies, including grid and real topologies. We will evaluate the performance of ILP models with the maximum utilization among links. In addition to maximum utilization, we will measure and compare mean utilization, mean hop count, number of constraints, maximum SLD, program running time, and ILP solving time.

#### 3.1 Equal-cost multi-path routing (ECMP) ILP Model

ECMP is the default behavior of SR. In ECMP, traffic is split among all equal paths. The ECMP concept is explained in [1]. Let's consider the topology in Fig. 19 (a) with six routers. For connection **PE1 – PE2**, there are 3 ECMP paths which are **PE1 – P1 – P2 – PE2**, **PE1 – P3 – P2 – PE2**, and **PE1 – P3 – P4 – PE2**. If the SID label for egress router **PE2** is pushed at the ingress router **PE1**, traffic will be split among all three ECMP paths. If four traffic units from **PE1** to **PE2** are sent, load balancing is performed efficiently only on links **PE1 – P1** and **PE1 – P3**, as shown in Fig. 19 (b). When two traffic units arrive at **P1**, **P1** will send two units of traffic to **PE2** as there are no paths to split the traffic as in Fig. 19 (c). Another two traffic units from the link **PE1 – P3** split one unit each again on **P3**, one unit traffic on link **P3 – P2**, and another one unit on **P3 – P4** as in Fig. 19 (d). Finally, Fig. 19 (e) shows that three traffic units on link **P2 – PE2** and one traffic unit on link **P4 – PE2** arrive at **PE2**. We can see that by exploiting ECMP, depending on specific topology and traffic demand, traffic on the network can be distributed in an unbalanced way, leading to excessive latency. In addition, some

network forwarding devices cannot guarantee per-flow forwarding and can result in out-of-sequence (OOS) packet delivery. In such a case, we should avoid SR ECMP paths and use explicit strict SR routes.

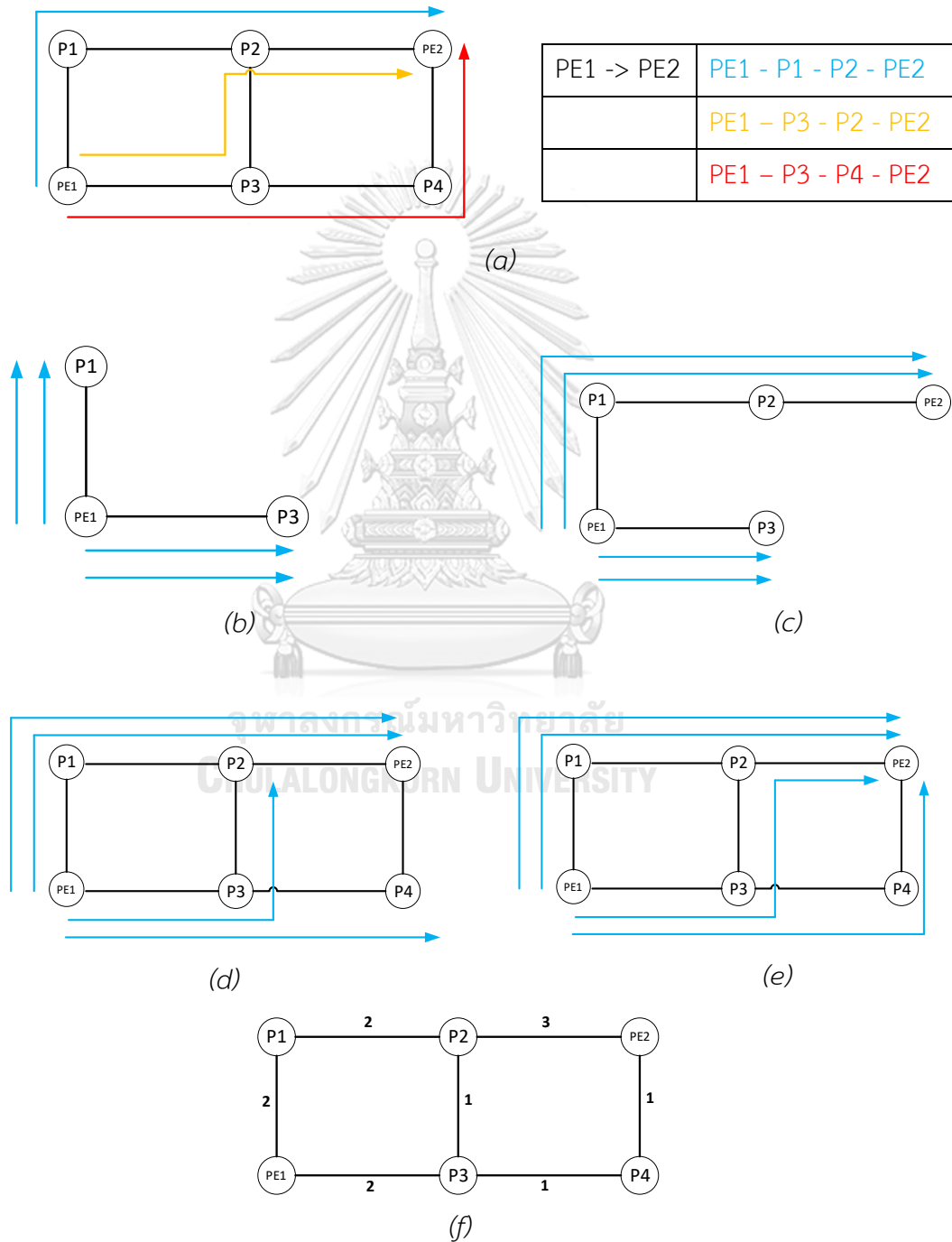


Fig. 19. ECMP Demonstrations

The notations and constraints are referenced from [1]. The first model ECMP is used as a benchmark in [1] and exploits all possible shortest paths. Notations for the ECMP model can be seen in Table. 2. The following ILP formulations are referenced from [1], and we will learn details in this sub-section.

Table. 2. Notations and Meaning of ECMP, SHP, SHP\_EN

Notation	Meaning	Notation	Meaning
$G$	Graph representing network	$d(c)$	The demand of a connection $c$
$N$	Nodes	$P_c$	Shortest paths of connection $c$
$L$	Links	$p$	A path $p \in P_c$
$K_l$	Maximum capacity of $l \in L$	$s(p)$	Source or ingress of $p$
$C$	Set of connections	$t(p)$	Destination or egress of $p$
$s(c)$	Source or ingress of a connection $c$	$x_c^p$	A variable describing the fraction of amount of traffic for $c$ through $p$ , $p \in P_c$ , $c \in C$
$t(c)$	Destination or egress of a connection $c$	$\alpha$	Maximum utilization of the network

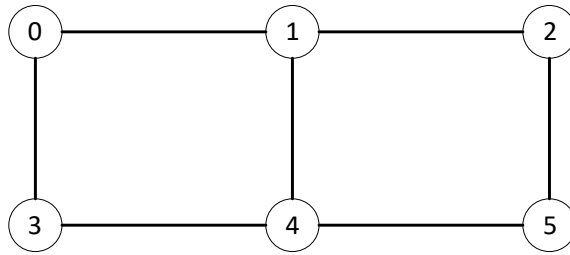


Fig. 20. 2 x 3 Topology for ECMP

To explain the ECMP model, we will consider the topology in Fig. 20 and a connection from node 0 to 5, denoted as  $c = 0-5$ . Let's assume that the maximum capacity of every link on the topology  $K_l$  is 1. We will explain the ECMP model with constraints generated from Python PuLP using the default CBC solver. In  $P$  of connection  $c = 0-5$ , there will be three ECMP paths,

0. 0-1-2-5 and the fraction of traffic flow on this path denoted as  $x_{0-5}^0$
1. 0-1-4-5 and the fraction of traffic flow on this path denoted as  $x_{0-5}^1$
2. 0-3-4-5 and the fraction of traffic flow on this path denoted as  $x_{0-5}^2$ .

This ECMP ILP model discovers the paths utilized by each  $c \in C$  such that maximum utilization is minimized.

$$\min \alpha \tag{1}$$

$$\sum_{p \in P_c} x_c^p = 1 \quad \forall c \in C \tag{2}$$

$$C1: x_{0-5}^0 + x_{0-5}^1 + x_{0-5}^2 = 1$$

Constraint (2) implies that the demand of each connection  $c$  has to send traffic totally on the shortest paths of connection  $c$ . As we can see the constraint in C1, the sum of fractions  $x_{0-5}^0$  for path 0-1-2-5,  $x_{0-5}^1$  for path 0-1-4-5, and  $x_{0-5}^2$  for 0-3-4-5 have to be equal to 1.

$$\sum_{c \in C} \sum_{p \in P_c: l \in p} d_c x_c^p \leq \alpha \cdot K_l \quad \forall l \in L \tag{3}$$

$$C2: x_{0-5}^0 + x_{0-5}^1 \leq \alpha \cdot 1$$

$$C3: x_{0-5}^2 \leq \alpha \cdot 1$$

$$C4: x_{0-5}^0 \leq \alpha \cdot 1$$

$$C5: x_{0-5}^1 \leq \alpha \cdot 1$$

$$C6: x_{0-5}^0 \leq \alpha \cdot 1$$

$$C7: x_{0-5}^2 \leq \alpha \cdot 1$$

$$C8: x_{0-5}^1 + x_{0-5}^2 \leq \alpha \cdot 1$$

(3) specifies the amount of traffic over each link in the network to be equal or less to  $\alpha$  times the maximum capacity of each link. The example topology has seven bi-directional links, 0-1, 0-3, 1-2, 1-4, 2-5, 3-4 and 4-5. C2 is a constraint for link 0-1, and two paths are flowing on that link: paths 0-1-2-5 and 0-1-4-5. C2 ensures the flow on the link 0-1 to be less or equal to  $\alpha$  times the capacity on link 0-1, and the objective function (1) makes the maximum utilization  $\alpha$  to be minimized. And this procedure proceeds for all remaining links, as we can see in C3 to C8.

$$0 \leq x_c^p \leq 1 \quad \forall c \in C, \quad \forall p \in P_c \quad (4)$$

$$\alpha \geq 0 \quad (5)$$

$$\sum_{p \in P_c: l_1 \in p} x_c^p = \sum_{p' \in P_c: l_2 \in p'} x_c^{p'} \quad \forall c \in C, \quad p, \quad p' \in P_c, \quad \text{all outgoing links } l_1, l_2 \text{ sharing} \quad (6)$$

same node

$$C9: x_{0-5}^0 + x_{0-5}^1 = x_{0-5}^2$$

$$C10: x_{0-5}^0 = x_{0-5}^1$$

Constraint (6) is constructed to enable ECMP. To enable ECMP, we have to distribute traffic at node 0 and node 1. In the example topology, we can see that there will be three ECMP paths 0-1-2-5, 0-1-4-5, and 0-3-4-5. At node 0, there are two outgoing links, link 0-1 and link 0-3. On Link 0-1, shortest paths 0-1-2-5 and 0-1-4-5 are

used, and on link 0-3, the shortest path 0-3-4-5 is used. As we can see in C9, the traffic combined between 0-1-2-5 and 0-1-4-5 and the traffic on 0-3-4-5 are forced to equal. At node 1, 0-1-2-5 on link 1-2 and 0-1-4-5 on link 1-4 are forced to equal, as we can see in C10. In that way, constraint (6) forces split traffic among outgoing links on each node if the shortest paths belong to the link. In SR, we have to encode only the SID of the terminal node at the source node to perform ECMP if there is more than one shortest path. In our example topology, the SID of node 5 is enough to perform ECMP on 0-1-2-5, 0-1-4-5, and 0-3-4-5 for connection 0-5, as shown in Table. 3. In this case, the SLD value is 1.

Table. 3. SID List for Connection 0-5

SID List	Paths
5	0-1-2-5, 0-1-4-5, 0-3-4-5

### 3.2 SHP ILP Model

After the ECMP model is studied, we will review the second model of [1], the SHP model. SHP model forces to select only one route if there is more than one shortest path. For example, considering 2 x 3 topology, there are 3 ECMP paths for connection 0 – 5. SHP model will force to choose only one of them, such as 0-1-2-5, 0-1-4-5, or 0-3-4-5, as shown in Fig. 21. Annotations for the SHP model are the same as the ECMP model, as shown in Table. 2. To build the SHP model, we will change to variable  $x_c^p$  to be 0 or 1 only. In addition, we will discard the ECMP model's constraint (6). and add constraint (7) [1] for consistency among connections that have the same terminal node.

$$x_c^p \geq x_{c'}^{p'} \text{ for all } c \text{ and } c' \in C, p \in P_c, p' \in P_{c'} \text{ such that } t(c) = t(c') \text{ and } p \subset p' \quad (7)$$

$$C11: x_{1-5}^0 \geq x_{0-5}^0$$

$$C12: x_{1-5}^1 \geq x_{0-5}^1$$

$$C13: x_{3-5}^0 \geq x_{0-5}^2$$

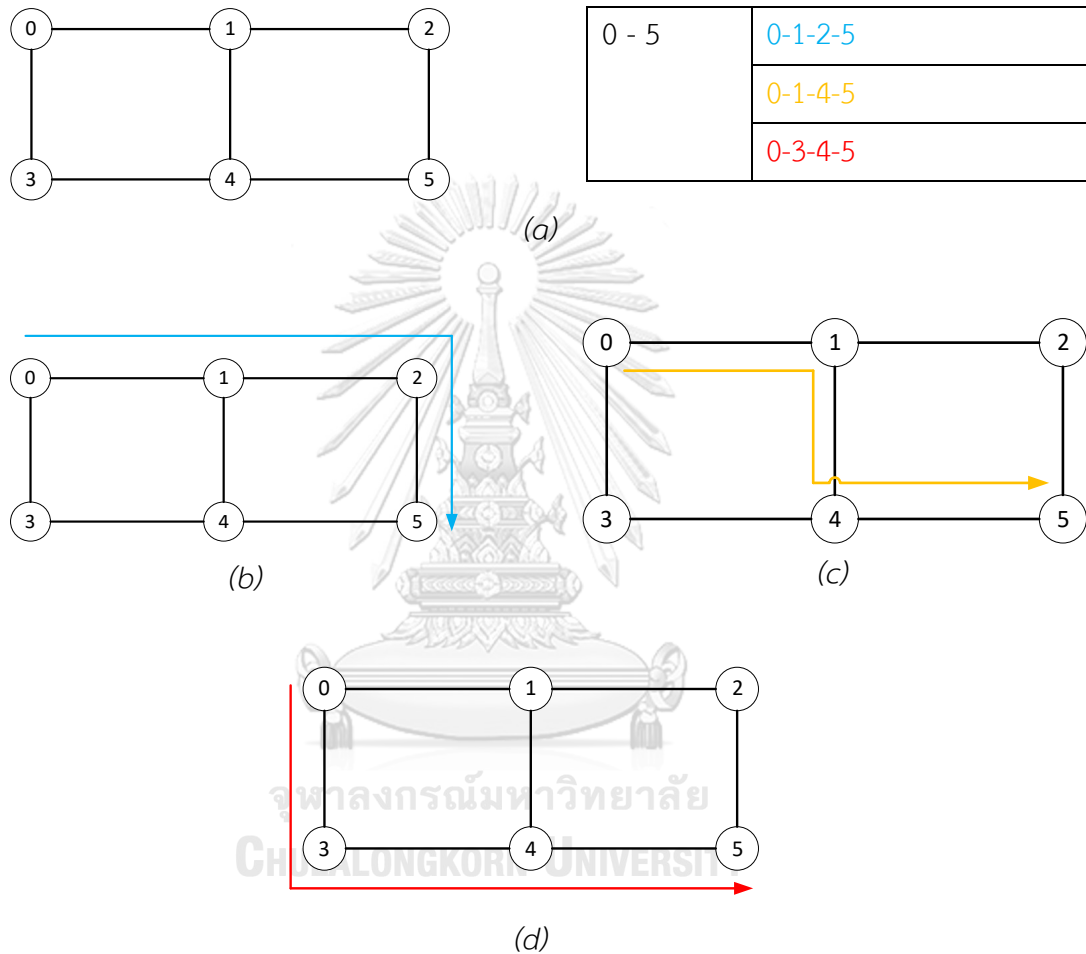


Fig. 21. SHP Demonstrations

To explain constraint (7), let's consider we have three connections, 0-5, 1-5, and 3-5. Connection 0-5 has three shortest paths which are 0-1-2-5, 0-1-4-5, and 0-3-4-5. Connection 1-5 has two shortest paths, 1-2-5 and 1-4-5. Connection 3-5 has only one path, which is 3-4-5. As we see C11 and C12, if connection 0-5 chooses 0-1-2-5, connection 1-5 has to choose 1-2-5 and if connection 0-5 chooses 0-1-4-5, connection 1-5 has to choose 1-4-5. However, as we can see C13, if connection 0-5 chooses 0-3-



4-5, connection 1-5 can choose any of the shortest paths between 1-2-5 and 1-4-5, and connection 3-5 has to choose 3-4-5. In this way, the paths among connections that have the same terminal node maintain consistency.

If ECMP is avoided and only one shortest path is forced to choose, we cannot encode only the SID of the terminal node as in the ECMP model. We have to encode more than one label in case there are ECMP paths. As shown in Table. 3, only one label is needed to encode in the ECMP model. Considering the same topology in Fig. 21, we must encode more than one to use a strict path for connection 0-5. To use the explicit path 0-1-2-5, we can encode SID 1,2,5, or 2,5, but more SIDs in the packet header mean more packet overhead in the packet. So, we can encode SL 2,5 for 0-1-2-5, 1,4,5 for 0-1-4-5 and 3,5 for 0-3-4-5 as shown in Table. 4.

Table. 4. SID List

SID List	Paths
2,5	0-1-2-5
1,4,5	0-1-4-5
3,5	0-3-4-5

### 3.3 SHP\_En ILP Model

This sub-section proposes an enhanced version of the SHP model (explained in section 3.2). In the SHP model, we use only the shortest paths for  $P_c$  and force to select only one if there are one or more shortest paths. In our SHP\_En ILP model, we will use k-shortest paths instead of shortest paths for  $P_c$ . For example, considering the topology in Fig. 22, when we limit the possible shortest paths to have at most nine paths, connection 0 – 5 will have four possible paths, which are 0-1-2-5, 0-1-4-5, 0-3-4-5, and 0-3-4-1-2-5. We limit to at most nine paths because it will take too much time

for computing every possible path in the network if the network becomes larger and SLD to form an explicit will be very big. All the constraints and objective function are the same as SHP except using the k-shortest paths for each connection and forcing to choose among them. In this way, we can minimize the maximum utilization among links to be nearly identical to the SEGMR model (the proposed model of the paper [1]), as we will explain in section 3.4.

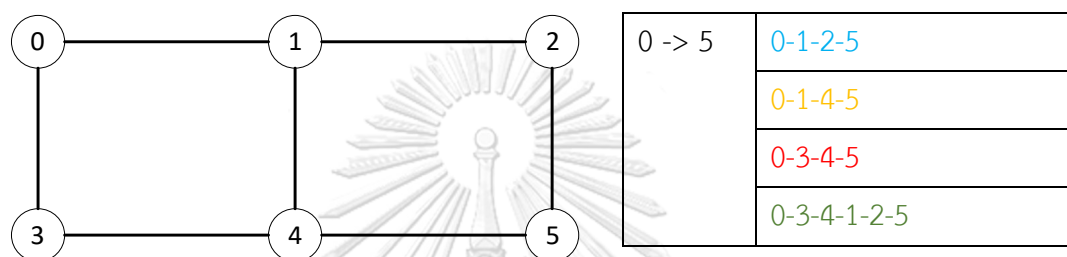
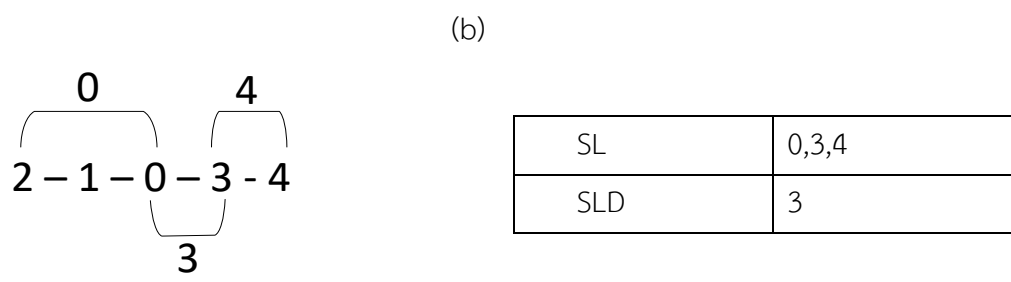
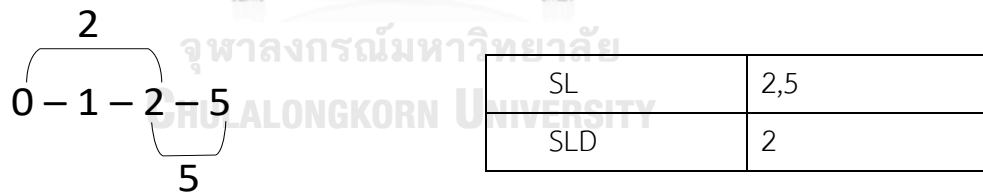
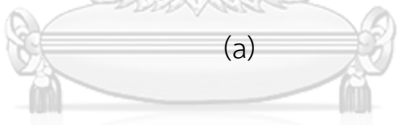
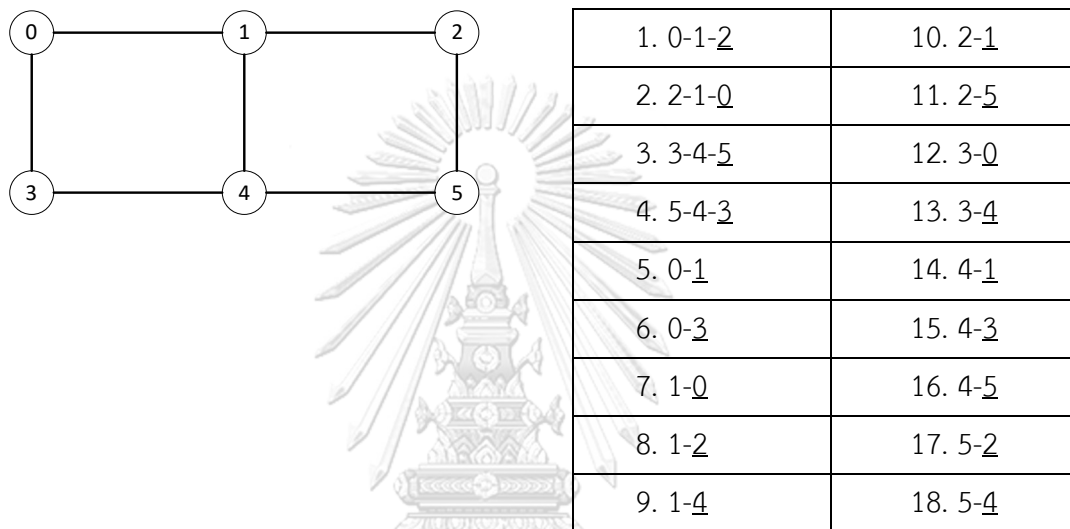


Fig. 22. SHP\_En Demonstrations

As the SHP\_En model can choose the longer paths among k-shortest paths, we cannot use the encoding algorithm in [1] to generate the SLD of the paths. The reason we cannot use is that the algorithm can only generate for the shortest paths. So, we make a program to encode every possible path. To encode every possible path, we first list every unique shortest path among every connection from larger to smaller hop counts. As shown in Fig. 23, there are 18 unique shortest paths in the 2 x 3 topology. In segment routing, we can use the SID of the terminal node as the SID for the path if the path is unique. For example, we can refer to path 0-1-2 with SID 2 only. If the SHP\_En model chooses path 2-1-0-3-4 for connection 2-4, we can use 1,0,3,4 as SID, but more SIDs mean more packet overhead. So, we will try to reduce SLD by the following procedure: we try to match the unique path from 1 to 18 until we get all the labels. For 2-1-0-3-4, 2-1-0 is first matched, so we put SID 0 for that segment and 0-3 is matched, so we put SID 3, and lastly, 3-4 is matched, so we put SID 4. The final segment list is 0,3,4, and SLD (segment list depth) is 3. The illustrations for reducing SLD to path 2-1-0-3-4 can be seen in Fig. 23 (c). The same procedure is also applied to

encode the explicit path for the SHP model. If we want to encode for the path 0-1-2-5 from the SHP model, the first 0-1-2 is matched. So, we put SID 2. 2-5 is matched, so we put 5, the final SL is 2,5, and SLD is 2. The SL procedure for SHP can be seen in Fig. 23 (b).



(c)

Fig. 23. SL Computation

Table. 5. Notations and Meaning for SEGMR

Notation	Meaning	Notation	Meaning
$G$	Graph representing network	$d(c)$	Demand of a connection $c$
$N$	Nodes	$\rho$	All unique shortest paths between node pairs
$L$	Links	$p$	A path $p \in \rho$
$K_l$	Maximum capacity of $l \in L$	$s(p)$	Source of $p$
$C$	Set of connections	$t(p)$	Destination of $p$
$s(c)$	Source node of a connection $c$	$y_c^p$	A binary variable represents the sub-path $p$ is used or not, $\forall c \in C, p \in \rho$
$t(c)$	Destination node of a connection $c$	$\alpha$	Maximum utilization of the network

### 3.4 SEGMR ILP Model

In this section, we will study the SEGMR model of the paper [1]. This paper proposed the SEGMR model to exploit the Segment Routing traffic engineering. In this model, there are a few changes in notations from previous ILP models as shown in notations as shown in Table. 5. Instead of  $P_c$ , this paper uses  $\rho$ . In  $\rho$ , there will be all unique shortest paths between node pairs. As shown in Fig. 24, there are 18 unique shortest paths in  $\rho$ . Choosing a unique shortest path is that 0-5 has three paths which are 0-1-2-5, 0-1-4-5, and 0-3-4-5, and they are not selected. 0-2 has only one path (0-1-2), and 0-1-2 is selected. If we choose unique shortest paths in  $\rho$ , we can use the SID of the terminal node to define a segment. For example, 0-1-2 path can be just defined by the SID of node 2.

The notations and constraints are referenced from [1]. To explain the SEGMR model, we will consider the topology in Fig. 24 and three connections 0-5, 1-2, and 1-5. Constraint (9) can be defined as flow conservation constraint. Constraint (9) makes sure every connection goes accurately through source, intermediate, and destination. For example, C1 and C6 ensure that connection 0-5 goes correctly through the source, intermediate, and destination. If segments 0-3 and 3-4-5 are chosen, C1 makes sure that segment 0-3 is chosen at node 0. C2, C3, and C5 ensure that no paths are starting at nodes 1,2, and 4. C4 ensures that segments 0-3 and 3-4-5 are chosen. C6 ensures that 3-4-5 ends at node 5. The same goes on for connections 1-2 from C7 to C12 and 1-5 from C13 to C18. Constraint (10) is the same as a constraint (3) from the ECMP model (explained in detail in section 3.1). Constraint (11) ensures that segments are not chosen more than  $\kappa$ . Limiting the number of  $\kappa$  can be considered as the maximum number of SLD. C20 limits the connection 0-5 to choose sub-paths less than or equal to  $\kappa$ . C21 and C22 limit the connection 1-2 and 1-5 respectively. Constraint (14) is for consistency. As we can see in C29 if connection 0-5 uses segment 1-2 as a subpath, connection 1-2 have to use segment 1-2, but if connection 0-5 doesn't use the segment 1-2, connection 1-2 can or cannot use segment 1-2. Also, with the connection 1-5, if connection 1-5 uses segment 1-2, connection 1-2 have to use segment 1-2, and if connection 1-5 doesn't use segment 1-2, connection 1-2 can or cannot use segment 1-2.

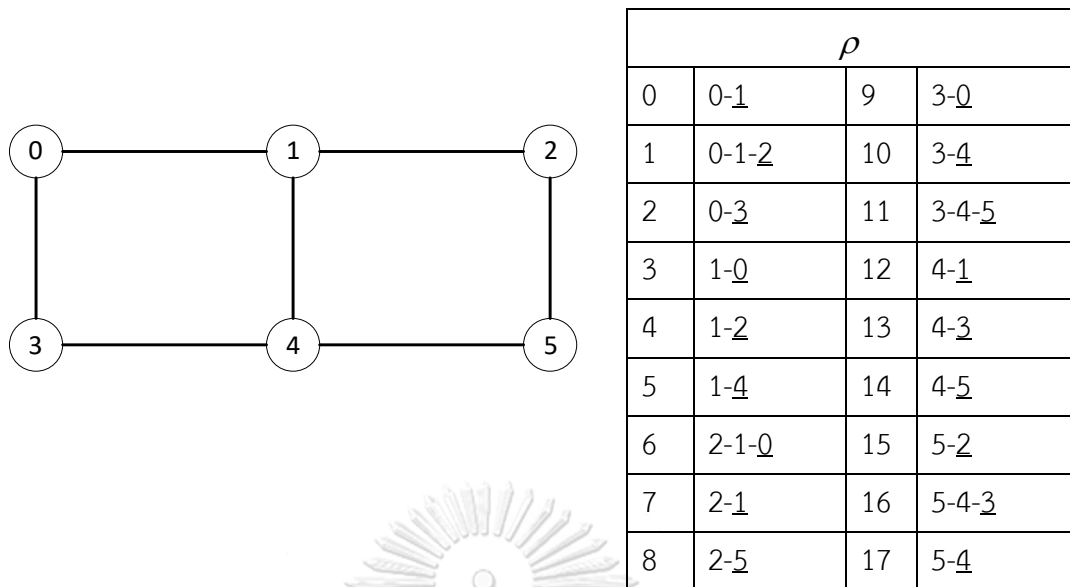


Fig. 24. Unique shortest paths

$$\min \alpha \quad (8)$$

$$\sum_{p \in \rho: s(p)=n} y_c^p - \sum_{p \in \rho: t(p)=n} y_c^p = b_{c,n} \quad \forall c \in C, n \in N \quad (9)$$

where

$$b_{c,n} = \begin{cases} 1 & \text{if } n = s(c) \\ -1 & \text{if } n = t(c) \\ 0 & \text{otherwise} \end{cases}$$

$$C1: y_{0-5}^0 + y_{0-5}^1 + y_{0-5}^2 - (y_{0-5}^3 + y_{0-5}^6 + y_{0-5}^9) = 1$$

$$C2: y_{0-5}^3 + y_{0-5}^4 + y_{0-5}^5 - (y_{0-5}^0 + y_{0-5}^7 + y_{0-5}^{12}) = 0$$

$$C3: y_{0-5}^6 + y_{0-5}^7 + y_{0-5}^8 - (y_{0-5}^1 + y_{0-5}^4 + y_{0-5}^{15}) = 0$$

$$C4: y_{0-5}^9 + y_{0-5}^{10} + y_{0-5}^{11} - (y_{0-5}^2 + y_{0-5}^{13} + y_{0-5}^{16}) = 0$$

$$C5: y_{0-5}^{12} + y_{0-5}^{13} + y_{0-5}^{14} - (y_{0-5}^5 + y_{0-5}^{10} + y_{0-5}^{17}) = 0$$

$$C6: y_{0-5}^{15} + y_{0-5}^{16} + y_{0-5}^{17} - (y_{0-5}^8 + y_{0-5}^{11} + y_{0-5}^{14}) = -1$$

$$C7: y_{1-2}^0 + y_{1-2}^1 + y_{1-2}^2 - (y_{1-2}^3 + y_{1-2}^6 + y_{1-2}^9) = 0$$

$$C8: y_{1-2}^3 + y_{1-2}^4 + y_{1-2}^5 - (y_{1-2}^0 + y_{1-2}^7 + y_{1-2}^{12}) = 1$$

$$C9: y_{1-2}^6 + y_{1-2}^7 + y_{1-2}^8 - (y_{1-2}^1 + y_{1-2}^4 + y_{1-2}^{15}) = -1$$

$$C10: y_{1-2}^9 + y_{1-2}^{10} + y_{1-2}^{11} - (y_{1-2}^2 + y_{1-2}^{13} + y_{1-2}^{16}) = 0$$

$$C11: y_{1-2}^{12} + y_{1-2}^{13} + y_{1-2}^{14} - (y_{1-2}^5 + y_{1-2}^{10} + y_{1-2}^{17}) = 0$$

$$C12: y_{1-2}^{15} + y_{1-2}^{16} + y_{1-2}^{17} - (y_{1-2}^8 + y_{1-2}^{11} + y_{1-2}^{14}) = 0$$

$$C13: y_{1-5}^0 + y_{1-5}^1 + y_{1-5}^2 - (y_{1-5}^3 + y_{1-5}^6 + y_{1-5}^9) = 0$$

$$C14: y_{1-5}^3 + y_{1-5}^4 + y_{1-5}^5 - (y_{1-5}^0 + y_{1-5}^7 + y_{1-5}^{12}) = 1$$

$$C15: y_{1-5}^6 + y_{1-5}^7 + y_{1-5}^8 - (y_{1-5}^1 + y_{1-5}^4 + y_{1-5}^{15}) = 0$$

$$C16: y_{1-5}^9 + y_{1-5}^{10} + y_{1-5}^{11} - (y_{1-5}^2 + y_{1-5}^{13} + y_{1-5}^{16}) = 0$$

$$C17: y_{1-5}^{12} + y_{1-5}^{13} + y_{1-5}^{14} - (y_{1-5}^5 + y_{1-5}^{10} + y_{1-5}^{17}) = 0$$

$$C18: y_{1-5}^{15} + y_{1-5}^{16} + y_{1-5}^{17} - (y_{1-5}^8 + y_{1-5}^{11} + y_{1-5}^{14}) = -1$$

$$\sum_{c \in C} \sum_{p \in \rho: l \in p} d_c y_c^p \leq \alpha \cdot K_l \quad \forall l \in L \quad (10)$$

$$\sum_{p \in \rho} y_c^p \leq \kappa \quad \forall c \in C \quad (11)$$

$$C20: y_{0-5}^0 + \dots + y_{0-5}^{17} \leq \kappa$$

$$C21: y_{1-2}^0 + \dots + y_{1-2}^{17} \leq \kappa$$

$$C22: y_{1-5}^0 + \dots + y_{1-5}^{17} \leq \kappa$$

$$y_c^p \in \{0, 1\} \quad \forall c \in C, p \in \rho \quad (12)$$

$$\alpha \geq 0 \quad (13)$$

$$y_c^p \geq y_{c'}^p \text{ for all } c, c' \in C, p \in \rho \text{ such that } s(p) = s(c), t(p) = t(c) \quad (14)$$

$$C29: y_{1-2}^4 \geq y_{0-5}^4$$

$$C30: y_{1-2}^4 \geq y_{1-5}^4$$

In the SEGMR model, the resulting segments from each connection can get the SLD. For example, if connection 0-5 chooses segment 0-3 and 3-4-5, segment list or SL will be 3 and 5, and SLD will be 2.

As the SEGMR ILP model can have multiple optimal solutions, the SEGMR model may choose some extra links for some connections as long as the maximum utilization is minimized. For example, the connection 1-2 may choose the sub-paths [1-2], [3-4], and [4-3] as long as the maximum utilization is minimized. We can see that [3-4] and [4-3] do not violate the constraints because connection 1-2 can reach with sub-path [1-2], and the total sub-paths are equal to  $\kappa$ . But, when we measure the performance of the SEGMR model after completing the ILP model, we remove [3-4] and [4-3] to evaluate the performance of the SEGMR model.

### 3.5 Results and Discussion

This section will present the topologies we used to evaluate the four ILP models we discussed and the results by four ILP models. We use four grid topologies and seven real topologies to test and assess the performance of the four ILP models. Topologies can be seen in Fig. 26, and the number of nodes and links of topologies can be seen in Table. 6. This thesis uses Python PuLP [9] and free default solver CBC to solve ILP models. We assume that every node pair or connection have traffic demand to send. We test with two kinds of traffic demands: same traffic demand and different traffic demand ranging from 1 to 10, including 10. The links in the topologies are bidirectional. We will set the parameters such as link capacity in a way that our main reference paper [1] conducts because we would like to compare results among them. The maximum capacity of each link is fixed to 1. Setting link capacity to 1 is not practical because the traffic demand cannot go through the link more than link capacity. But by limiting capacity 1, we will know the maximum utilization  $\alpha$  on a link



or the maximum number of traffic on a link. For the SHP\_En ILP model, we limit possible paths to 9, and for the SEGMR model, we use  $\kappa = 3$  and  $\kappa = 8$ . According to paper [1], networking devices support maximum SLD between 5 and 8, but according to Cisco, Cisco ASR 1000 routers support up to maximum SLD 16 [36]. We measure maximum utilization (Table. 7 and Table. 8), mean utilization (Table. 9 and Table. 10), mean hop count (Table. 11 and Table. 12), maximum segment list depth (SLD) (Table. 13 and Table. 14), time (Table. 15, Table. 16, Table. 17, and Table. 18), and the number of constraints (Table. 19). All 4 ILP models are run on Ubuntu 18.04.4 LTS server version, and specifications can be seen in Fig. 25.

Table. 6. Nodes and Links of Topology.

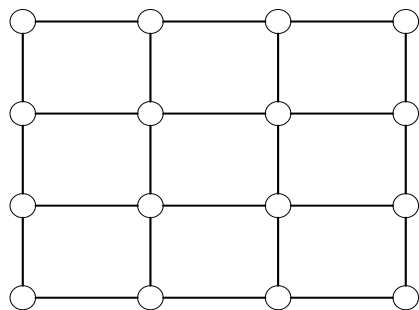
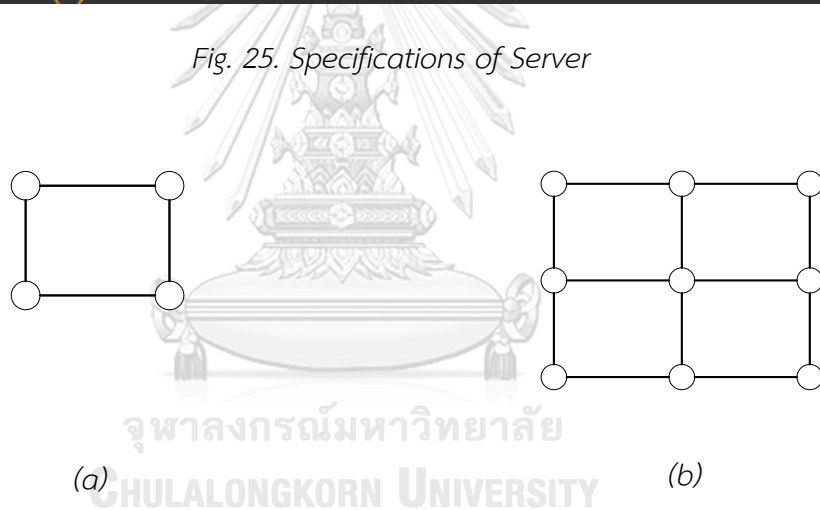
Topology	Node	Link
2 x 2	4	4
3 x 3	9	12
4 x 4	16	24
5 x 5	25	40
Eurocosre	11	25
NFSNET	14	21
EON	20	39
UKNet	21	39
ITALNet	21	36
Arpanet	20	31
30-Node European	30	48

```

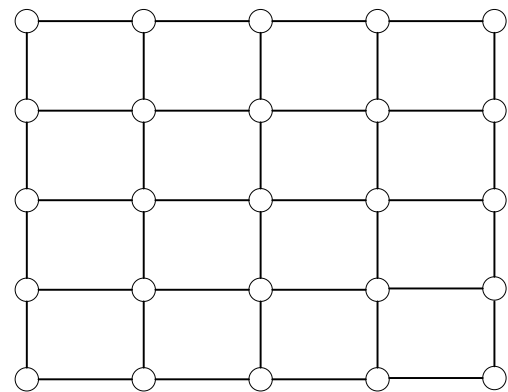
Architecture:      x86_64
CPU op-mode(s):    32-bit, 64-bit
Byte Order:        Little Endian
CPU(s):            20
On-line CPU(s) list: 0-19
Thread(s) per core: 1
Core(s) per socket: 1
Socket(s):         20
NUMA node(s):     2
Vendor ID:         GenuineIntel
CPU family:        6
Model:            85
Model name:        Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz
Stepping:          7
CPU MHz:           2194.843
BogoMIPS:          4389.68
Hypervisor vendor: VMware
Virtualization type: full
L1d cache:         32K
L1i cache:         32K
L2 cache:          1024K
L3 cache:          14080K
NUMA node0 CPU(s): 0-9
NUMA node1 CPU(s): 10-19

```

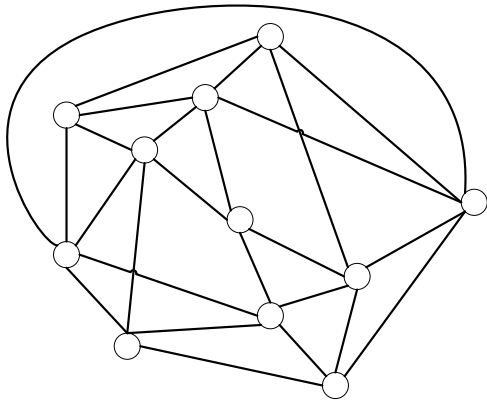
Fig. 25. Specifications of Server



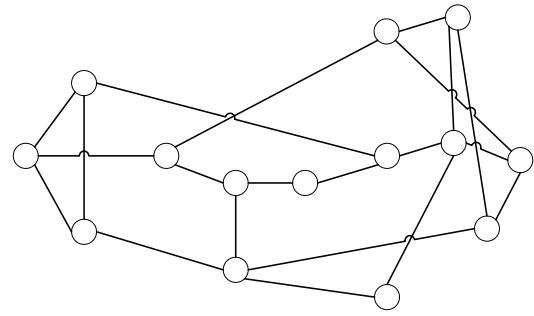
(c)



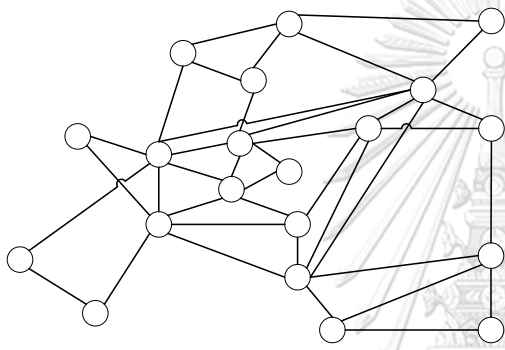
(d)



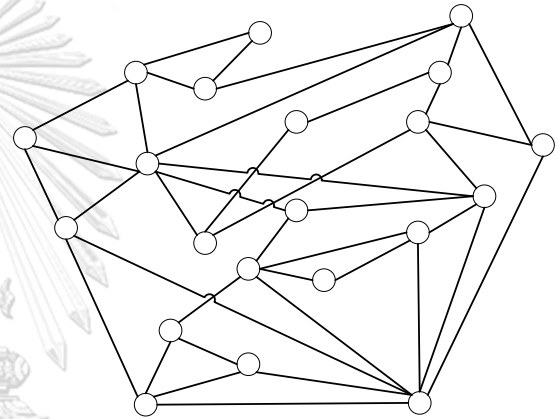
(e)



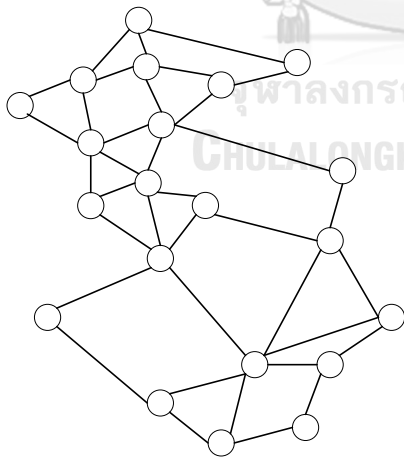
(f)



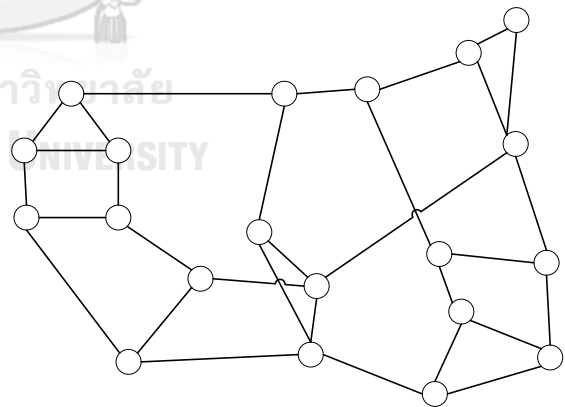
(g)



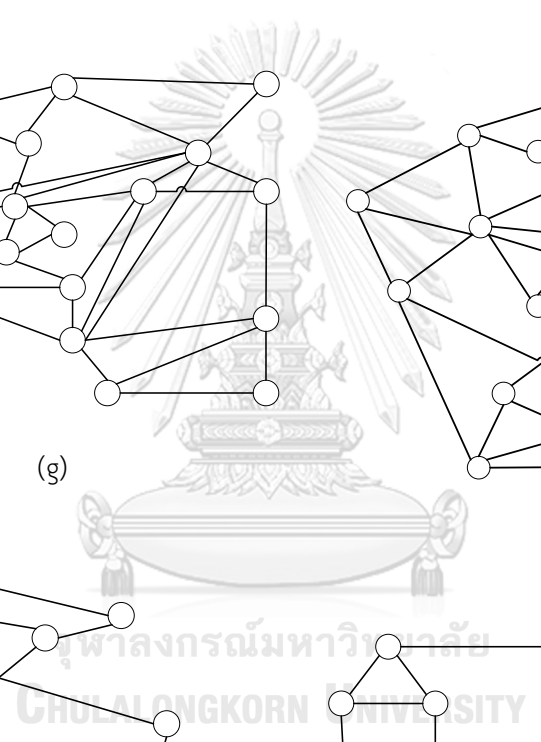
(h)



(i)



(j)



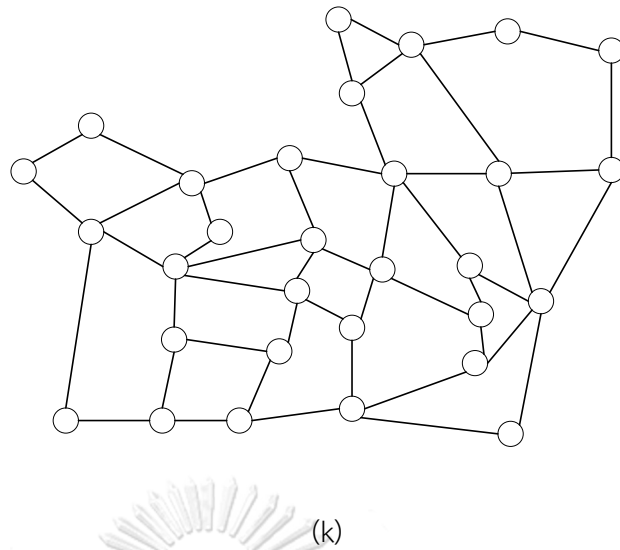


Fig. 26. Topologies (a) Grid 2 x 2 (b) 3 x 3 (c) 4 x 4 (d) 5 x 5 (e) Eurocore (f) NFSNET (g) EON (h) UKNET (i) ITALNET (j) Arpanet (k) 30-Node European

The maximum utilization is measured as the maximum number of traffic demands flowing among network topology links. For the same traffic demand of each connection = 1 or the same traffic demand, we can see the maximum utilization in **Table. 7**. We limit the ILP models to find the solution within 6 hours or 21600 seconds. Some topology such as UKNet in SHP takes a lot of time to find the optimal solution and denote with \* for results at the maximum amount of time (21600 seconds). For maximum utilization results with the same traffic demand, ECMP always generates higher maximum utilization except 2 x 2 topology compared to SHP, SHP\_En, and SEGMR. In 30 node European topology, maximum utilization is 42% higher than SHP\_En and SEGMR. Comparing SHP to SHP\_En and SEGMR for the same traffic demand, SHP\_En and SEGMR generate 14% in average smaller maximum utilization in NFSNet, UKNET, ITALNET, and 30-node European topologies. Comparing SHP\_En to SEGMR, by giving the 9-shortest paths to SHP to choose, we can achieve the maximum utilization as the same as SEGMR. In the SEGMR model,  $\kappa = 3$  is enough to achieve the results in **Table. 7**. To test the ILP models with random traffic demand, we create traffic matrices ranging from 1 to 10 (including 10) for each topology and use them for each ILP model. In random traffic demand, ECMP generates less maximum utilization in only 2 x 2

topology. Compared to ECMP, SHP\_En increases 2% maximum utilization while SHP and SEGMR increase by 6%. Except for 2 x 2 topology, ECMP generates higher maximum utilization in other topologies. Comparing ECMP to SHP\_En and SEGMR, ECMP generates 26% on average higher maximum utilization. SHP generates 16% on average higher maximum utilization than SHP\_En and SEGMR. SHP\_En generates 4% less maximum utilization than SEGMR in 2 x 2 and other than that, generates the same as SEGMR. Although we set the maximum capacity of each link is set to 1, we can see that the maximum utilization results are more than 1. The reason the maximum utilization results are more than 1 is that as we explained in ECMP, SHP, SHP\_En and SEGMR model, constraint (3) and (10) limits the capacity not to be more than  $\alpha$  times the capacity but we set  $\alpha$  to be greater than or equal to 0. As we explained, the maximum utilization more than the capacity of the link is not practical, but we can know  $\alpha$  as the maximum number of demands flowing on a link.

The utilization of a link is the amount of traffic flowing on a link. Mean utilization for the same traffic demand can be seen in Table. 9. As SHP\_En and SEGMR can choose longer paths than shortest paths, the mean utilization in SHP\_En and SEGMR is higher than ECMP and SHP except 2 x 2 and 3 x 3 topologies. SHP\_En produces 13% in average higher mean utilization than SHP and ECMP model while SEGMR produces 15% in average higher with  $\kappa = 3$ . Table. 10 shows the mean utilization for random traffic demand. ECMP and SHP generate the same mean utilization. In 2 x 2 topology, SHP\_En generates 2% higher than other models, but SHP\_En's maximum utilization is lower than SHP and SEGMR, as shown in Table. 8. In other topologies, SHP\_En and SEGMR cause 12% in average higher mean utilization than ECMP and SHP.

The mean hop count for each ILP model can be seen in Tables. 11 and 12. Table. 11 shows that SHP\_En and SEGMR have longer hop count than ECMP and SHP as SHP\_En and SEGMR can travel longer paths for the same traffic demand. All ILP models generate the same mean hop count in 2 x 2 and 3 x 3 topologies. In other topologies except 2 x 2 and 3 x 3, we can see that SHP\_En produces 13% in average

higher mean hop count and SEGMR produces 15% in average higher than SHP and ECMP. In topology 2 x 2 with random traffic demand, SHP\_En generates 11% higher than other ILP models but SHP\_En maximum utilization is lower than SHP and SEGMR. In other topologies except 2 x 2, SHP\_En produces 11% on average higher for random traffic demand, and SEGMR produces 12% in average higher mean hop count than ECMP and SHP.

Tables. 13 and 14 show the maximum SLD results to be appended to form SR-TE paths for each ILP model. Table. 13 indicates that ECMP only needs one label to encode to create a path for the same traffic demand. SHP uses up to five labels as SHP is forced to choose only one strict path. SHP\_En uses up to a maximum of six labels to extend the SHP ILP model and provides lower maximum utilization in the network. We already tested six labels to form the SR-TE paths in Cisco XRV 6.1.3. The wonderful proposed SEGMR of [1] needs only three labels to provide efficient maximum utilization in the network. ECMP only needs one label, and SHP and SHP\_En need up to a maximum of six labels for random traffic demand. SEGMR only needs three labels.

Table. 7. Maximum Utilization for Same Traffic Demand

Topology	Maximum utilization – Same Traffic Demand				
	ECMP	SHP	SHP_En (9)	SEGMR $\kappa = 8$	SEGMR $\kappa = 3$
2 x 2	4.0	4.0	4	4	4
3 x 3	14.0	12.0	12	12	12
4 x 4	37.25	32.0	32.0	32.0	32.0
5 x 5	73.5	60.0	60.0	60.0	60.0
Eurocore	9.67	8.0	8.0	8.0	8.0
NFSNet	30.67	26.0	25	25.0	25.0
EON	49.78	36.0	36.0	36.0	36.0
UKNET	54.88	42*	37.0	37.0	37.0
ITALNET	93.75	66.0	55	55.0	55.0
Arpanet	71.29	66.0	66	66.0	66.0
30-node European	194.38	146	112	112	112.0

\* Result at 6 Hour



Table. 8. Maximum Utilization for Random Traffic Demand

Topology	Maximum utilization – Random Traffic Demand (1-10)				
	ECMP	SHP	SHP_En (9)	SEGMR $\kappa = 8$	SEGMR $\kappa = 3$
2 x 2	23.5	25.0	24	25.0	25.0
3 x 3	84.25	65.0	65.0	65.0	65.0
4 x 4	230.06	179*	179.0	179.0	179.0
5 x 5	439	347*	347.0	347.0	347
Eurocore	56.42	45.0	41.0	43*	41.0
NFSNet	170.0	151	128.0	128.0	128
EON	265.69	199	199.0	199	199.0
UKNET	295.17	235.0*	198.0	198.0	198.0
ITALNET	525.67	387.0	325.0	325.0	325.0
Arpanet	401.71	363.0	363.0	363.0	363.0
30-node European	969.27	754.0	584.0	584.0	584.0

\* Result at 6 Hour





Table. 9. Mean Utilization for Same Traffic Demand

Topology	Mean utilization – Same Traffic Demand				
	ECMP	SHP	SHP_En	SEGMR $\kappa = 8$	SEGMR $\kappa = 3$
2 x 2	4.0	4.0	4.0	4	4
3 x 3	12.0	12.0	12	12	12
4 x 4	26.67	26.67	29.08	31.17	29
5 x 5	50	50.0	52.65	58.7	53.85
Eurocore	6.96	6.96	7.28	7.56	7.52
NFSNet	18.57	18.57	22.95	22.62	22.29
EON	23.03	23.03	27.03	32.26	27.64
UKNET	26.97	26.97*	30.67	34.49	31.92
ITALNET	34.06	34.06	40.28	48.36	41.25
Arpanet	34.38	34.39	46.23	49.26	44.13
30-node European	61.38	61.38	73.90	98.42	80.02

\* Result at 6 Hour



Table. 10. Mean Utilization for Random Traffic Demand

Topology	Mean utilization – Random Traffic Demand (1-10)				
	ECMP	SHP	SHP_En	SEGMR $\kappa = 8$	SEGMR $\kappa = 3$
2 x 2	22.5	22.5	23	22.5	22.5
3 x 3	60.42	60.42	62.25	61.92	62.25
4 x 4	145.21	145.21*	151.54	161.96	156.63
5 x 5	274.83	274.83*	292.78	310.18	298.18
Eurocore	38.24	38.24	40.32	40.76	40.56
NFSNet	98.10	98.10	122.38	110.81	110.57
EON	117.21	117.21	142.28	155.85	143.38
UKNET	137.97	137.97*	155.44	171.28	162.05
ITALNET	199.58	199.58	233.69	261.86	230.03
Arpanet	187.81	187.81	240.90	233.35	221.58
30-node European	322.13	322.13	380.19	481.06	410.27

\* Result at 6 Hour



Table. 11. Mean Hop Count for Same Traffic Demand

Topology	Mean Hop Count – Same Traffic Demand				
	ECMP	SHP	SHP_En	SEGMR $\kappa = 8$	SEGMR $\kappa = 3$
2 x 2	1.33	1.33	1.33	1.33	1.33
3 x 3	2.0	2.0	2.0	2.0	2.0
4 x 4	2.67	2.67	2.91	3.12	2.9
5 x 5	3.33	3.33	3.51	3.91	3.59
Eurocore	1.58	1.58	1.65	1.72	1.71
NFSNet	2.14	2.14	2.65	2.61	2.57
EON	2.36	2.36	2.77	3.31	2.84
UKNET	2.50	2.50*	2.85	3.20	2.96
ITALNET	2.92	2.92	3.45	4.15	3.54
Arpanet	2.81	2.81	3.77	4.02	3.6
30-node European	3.39	3.39	4.08	5.43	4.41

\* Result at 6 Hour



Table. 12. Mean Hop Count for Random Traffic Demand

Topology	Mean Hop Count – Random Traffic Demand (1-10)				
	ECMP	SHP	SHP_En	SEGMR $\kappa = 8$	SEGMR $\kappa = 3$
2 x 2	1.33	1.33	1.5	1.33	1.33
3 x 3	2.0	2.0	2.06	2.08	2.06
4 x 4	2.67	2.67*	2.78	2.97	2.89
5 x 5	3.33	3.33*	3.54	3.83	3.66
Eurocore	1.58	1.58	1.65	1.74*	1.69
NFSNet	2.14	2.14	2.65	2.51	2.41
EON	2.36	2.36	2.79	3.24	2.86
UKNET	2.50	2.50*	2.78	3.26	2.96
ITALNET	2.92	2.92	3.41	3.91	3.39
Arpanet	2.81	2.81	3.56	3.54	3.36
30-node European	3.39	3.39	3.97	5.16	4.29

\* Result at 6 Hour



Table. 13. Maximum SLD for Same Traffic Demand

Topology	Maximum SLD – Same Traffic Demand				
	ECMP	SHP	SHP_En	SEGMR $\kappa = 8$	SEGMR $\kappa = 3$
2 x 2	1	2	3	2	2
3 x 3	1	2	3	4	3
4 x 4	1	5	5	7	3
5 x 5	1	5	6	8	3
Eurocore	1	3	3	5	3
NFSNet	1	2	4	7	3
EON	1	4	4	8	3
UKNET	1	4*	5	8	3
ITALNET	1	4	5	8	3
Arpanet	1	3	5	8	3
30-node European	1	4	5	8	3

\* Result at 6 Hour

Table. 14. Maximum SLD for Random Traffic Demand

Topology	Maximum SLD – Random Traffic Demand (1-10)				
	ECMP	SHP	SHP_En	SEGMR $\kappa = 8$	SEGMR $\kappa = 3$
2 x 2	1	2	2	2	2
3 x 3	1	2	4	5	3
4 x 4	1	4*	5	5	3
5 x 5	1	6*	6	7	3
Eurocore	1	3	3	6*	3
NFSNet	1	2	4	7	3
EON	1	4	4	8	3
UKNET	1	4*	5	7	3
ITALNET	1	4	5	8	3
Arpanet	1	3	5	8	3
30-node European	1	4	5	8	3

\* Result at 6 Hour

Tables. 15, 16, 17 and 18 discuss the program running time and ILP problem-solving time. Program running time is the time to finish the whole program, including ILP problem-solving time, and ILP problem-solving time is the time to solve the ILP problem only. ECMP takes the shortest time to find an optimal solution among the 4 ILP models for both traffic demands as ECMP requires fewer constraints than other ILP models. Tables. 15 and 16 show the results for the same traffic demand. We limit the time for UKNET in the SHP model to get the result at 21600 seconds as it takes too long to find the optimal solution. SHP\_En requires more time compared to SHP except for UKNet topology. In UKNET topology, SHP can find an optimal solution in 8.60 seconds. Comparing SHP\_En and SEGMR, SHP\_En takes less time to find the optimal solution except 3 x 3 topology requires 0.08 seconds. For random traffic demand as

shown in Tables. 17 and 18, three topologies, namely, 4 x 4, 5 x 5, and UKNET in SHP, take a long time to find the solution and are limited to 6 hours. Comparing SHP\_En and SEGMR, SHP\_En takes less time to find the optimal solution than SEGMR. Table. 19 shows the number of constraints for 4 ILP models. We can see that SEGMR requires the most number of constraints compared to the three other models. The enhanced version of the SHP model, SHP\_En, requires more than the SHP model but less than the SEGMR model.

Table. 15. Program Running Time for Same Traffic Demand

Topology	Program Running Time(s) – Same Traffic Demand				
	ECMP	SHP	SHP_En	SEGMR $\kappa = 8$	SEGMR $\kappa = 3$
2 x 2	0.04	0.05	0.05	0.07	0.06
3 x 3	0.08	0.10	0.70	0.96	0.95
4 x 4	0.42	0.65	4.41	14.82	16.53
5 x 5	2.85	16.06	55.06	231.29	262.94
Eurocore	0.12	0.34	0.76	3.95	3.85
NFSNet	0.14	0.17	2.63	14.53	15.11
EON	0.57	0.70	6.87	89.66	89.33
UKNET	0.63	21675.49*	11.31	144.91	381.25
ITALNET	0.55	0.74	12.64	117.83	239
Arpanet	0.39	0.51	10.05	96.08	115.14
30-node European	1.64	2.06	40.33	1828.01	7045.61

\* Result at 6 Hour

Table. 16. PuLP Time for Same Traffic Demand

Topology	PuLP Time (s) – Same Traffic Demand				
	ECMP	SHP	SHP_En	SEGMR $\kappa = 8$	SEGMR $\kappa = 3$
2 x 2	0.01	0.01	0.01	0.02	0.02
3 x 3	0.01	0.03	0.45	0.39	0.37
4 x 4	0.04	0.26	3.16	5.79	7.48
5 x 5	0.30	13.46	50.46	151.71	180.65
Eurocore	0.01	0.22	0.31	1.23	1.16
NFSNet	0.02	0.03	1.79	5.62	5.99
EON	0.04	0.13	4.52	46.67	45.77
UKNET	0.04	21674.86*	8.60	84.30	320.81
ITALNET	0.04	0.14	9.92	60.88	182.23
Arpanet	0.03	0.07	7.67	39.88	59.51
30-node European	0.07	0.26	32.65	1505.92	6722.77

\* Result at 6 Hour





Table. 17. Program Running Time for Random Traffic Demand

Topology	Program Running Time (s) – Random Traffic Demand (1-10)				
	ECMP	SHP	SHP_En	SEGMR $\kappa = 8$	SEGMR $\kappa = 3$
2 x 2	0.04	0.05	0.06	0.06	0.06
3 x 3	0.08	0.12	1.45	3.85	4.20
4 x 4	0.41	22317.51*	13.83	68.57	59.42
5 x 5	2.89	21637.17*	31.14	571.53	543.01
Eurocore	0.13	1.47	1919.10	21702.89*	2554.95
NFSNet	0.14	0.17	11.45	43.23	36.56
EON	0.58	0.71	6.78	61.65	80.18
UKNET	0.62	21674.15*	28.60	335.08	534.95
ITALNET	0.59	0.72	50.54	139.86	328.79
Arpanet	0.38	0.54	7.30	170.67	85.32
30-node European	1.56	2.10	23.60	786.31	966.50

\* Result at 6 Hour



Table. 18. PuLP Time for Random Traffic Demand

Topology	PuLP Time (s) – Random Traffic Demand (1-10)				
	ECMP	SHP	SHP_En	SEGMR $\kappa = 8$	SEGMR $\kappa = 3$
2 x 2	0.01	0.02	0.02	0.04	0.04
3 x 3	0.01	0.05	1.19	3.28	3.66
4 x 4	0.04	22317.13*	12.62	59.31	50.25
5 x 5	0.27	21634.52*	26.76	491.15	461.53
Eurocore	0.01	1.36	1918.63	21700.14*	2552.30
NFSNet	0.01	0.03	10.61	33.96	27.44
EON	0.03	0.14	4.42	17.44	35.88
UKNET	0.04	21673.52*	25.88	273.53	472.90
ITALNET	0.04	0.12	47.83	82.24	271.80
Arpanet	0.03	0.09	4.92	114.69	28.51
30-node European	0.07	0.27	15.97	464.83	635.97

\* Result at 6 Hour



Table. 19. Number of Constraints

Topology	Number of constraints				
	ECMP	SHP	SHP_En	SEGMR $\kappa = 8$	SEGMR $\kappa = 3$
2 x 2	20	24	40	152	152
3 x 3	148	296	2340	3288	3288
4 x 4	664	2304	8314	27048	27048
5 x 5	2240	14240	21954	135440	135440
Eurocore	211	299	2039	8539	8539
NFSNet	255	505	6107	28091	28091
EON	767	1887	9520	79271	79271
UKNET	823	2121	11589	107325	107325
ITALNET	764	2438	13376	105646	105646
Arpanet	593	1659	14341	112615	112615
30-node European	1648	5878	33220	412854	412854

## Chapter 4

### Implementation of Segment Routing network on Emulated Virtual Environment - Next Generation (EVE-NG)

#### 4.1: Emulated Virtual Environment - Next Generation (EVE-NG)

Implementation and testing the traffic engineering algorithms and changes in a real production network are complex and challenging in terms of costs. In this thesis, the EVE-NG community edition [5] is used as a tool to emulate the network. EVE-NG is an emulated virtual environment - the next generation. EVE-NG supports the real-life scenario by using virtual devices by connecting to real devices or physical devices. In this thesis, Cisco IOS XRV 6.1.3 routers and a Cisco switch are implemented in EVE-NG as routers and switch.

We created an experimental testbed in Emulated Virtual Environment Next Generation (EVE-NG) community edition [5]. Firstly, the EVE-NG is installed on the VM with specifications

Memory: 16 GB

Processors: 4

Hard Disk: 50 GB

Network Adapter: one for NAT and another one to connect to ODL with VMware workstation 16 Pro [37]. EVE-NG is a web-based application and very light to use. Three parts in this testbed are network devices, the SDN controller, and the SHP\_En Traffic Engineering application running on top of the SDN controller. The author uses two VMs. The first one is for EVE-NG to implement network topology, and the latter one is to implement OpenDaylight controller [6] (SDN controller) and SHP\_En Traffic Engineering application with Postman [7] on top of the controller as shown in Fig. 27. A network adapter in a VMware workstation connects two VMs.

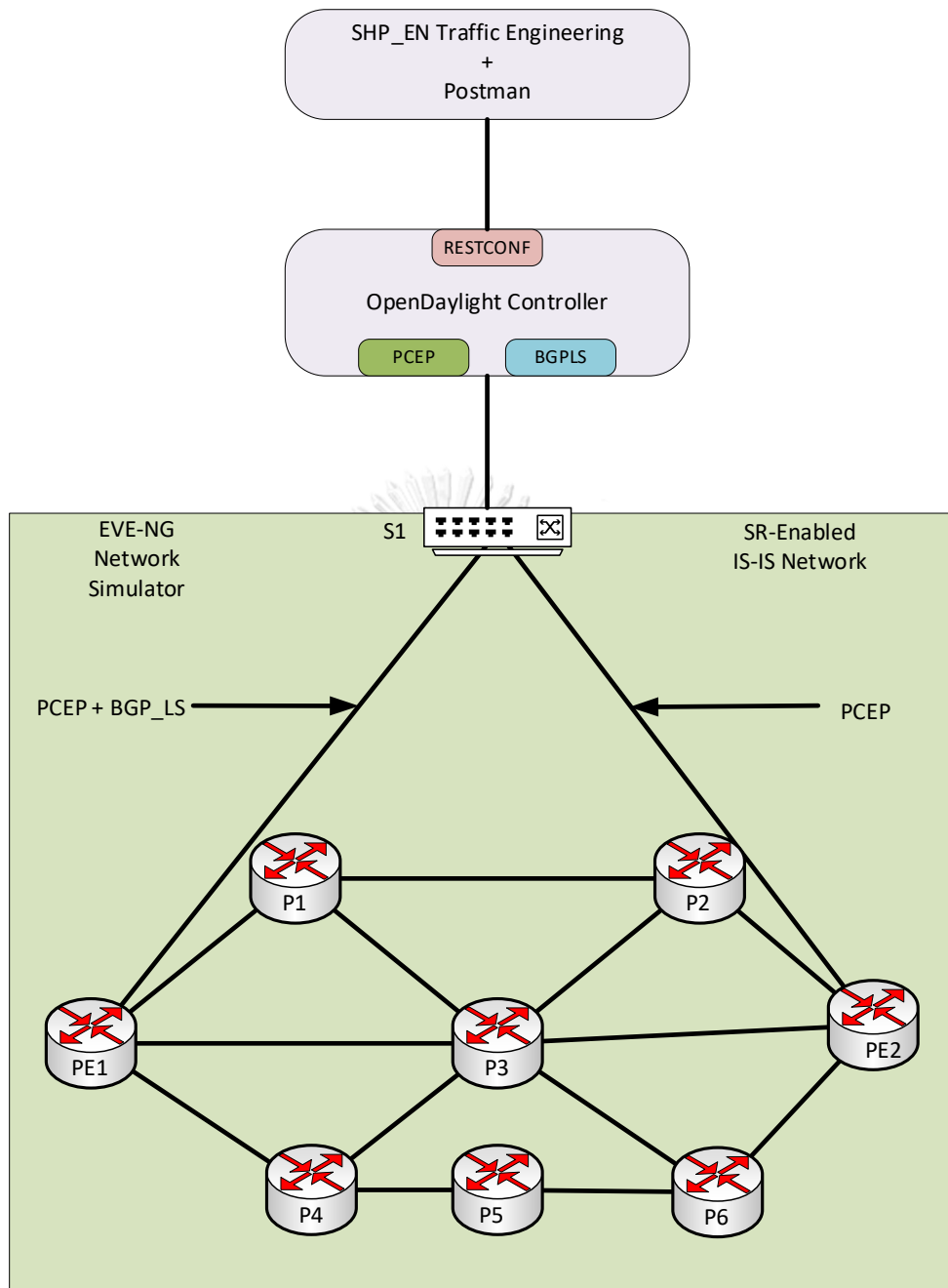


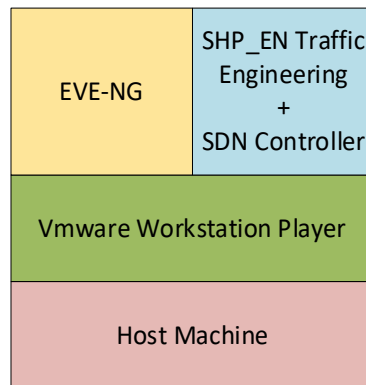
Fig. 27. Experimental Topology

## 4.2: Cisco IOS XRV

The Cisco IOS XRV is a router based on Virtual Machine based platform with 32-bit IOS XR software [10]. This VM includes the forwarding and routing functionality, manageability, and control-plane features. Cisco IOS XRV does not represent an emulation of any hardware component or any physical router. In a demonstration environment, this VM supports up to eight CPUs on a single VM. This VM is also hardware-independent, and it runs on x86 hardware supported by the virtualization platform.

## 4.3 Creating a network topology

To create an experimental network topology, routers and a switch are added to the EVE-NG. Cisco IOS XRV 6.1.3 image is added to the EVE-NG as a router, and l2-adventure is added as layer two switch. The system design of the experimental topology is shown in Fig. 28. PE1 (Provider Edge) and PE2 are edge routers, and P1 (Provider), P2, P3, P4, P5, and P6 are core routers. A switch connects between the SDN controller and PE routers. We can connect to the actual internet or another VM in EVE-NG, as shown in Fig. 27. As seen in Fig. 27, PE1 and PE2 are connected to the SDN controller (Net) via a switch (s1).



*Fig. 28. System Design*

#### 4.4 Setting up SDN Controller

OpenDaylight [6] is used as an SDN controller. OpenDaylight controller is installed on Ubuntu 18.04.5 LTS Desktop Version [38] with memory 4 GB, 1 Processor, 20 GB Hard Disk, and two network adapters. OpenDaylight can be downloaded on [6]. After installing and starting OpenDaylight in our VM machine, we must enable the essential features that we will use in our topology using the odl command line: `install odl-bgpcep-bgp-all`, `install odl-restconf-noauth`, `install odl-bgpcep-pcep-pcep-all`.

จุฬาลงกรณ์มหาวิทยาลัย  
CHULALONGKORN UNIVERSITY

#### 4.5 Connecting the topology and the SDN controller

Connecting the SDN controller to the routers can be done in EVE-NG. We connect two VM with the same virtual network interface at the VMware workstation. Moreover, we choose the network cloud in the EVE-NG and connect via a switch, as shown in Fig. 29. We have to set the required IP address (172.16.1.128) to the controller.

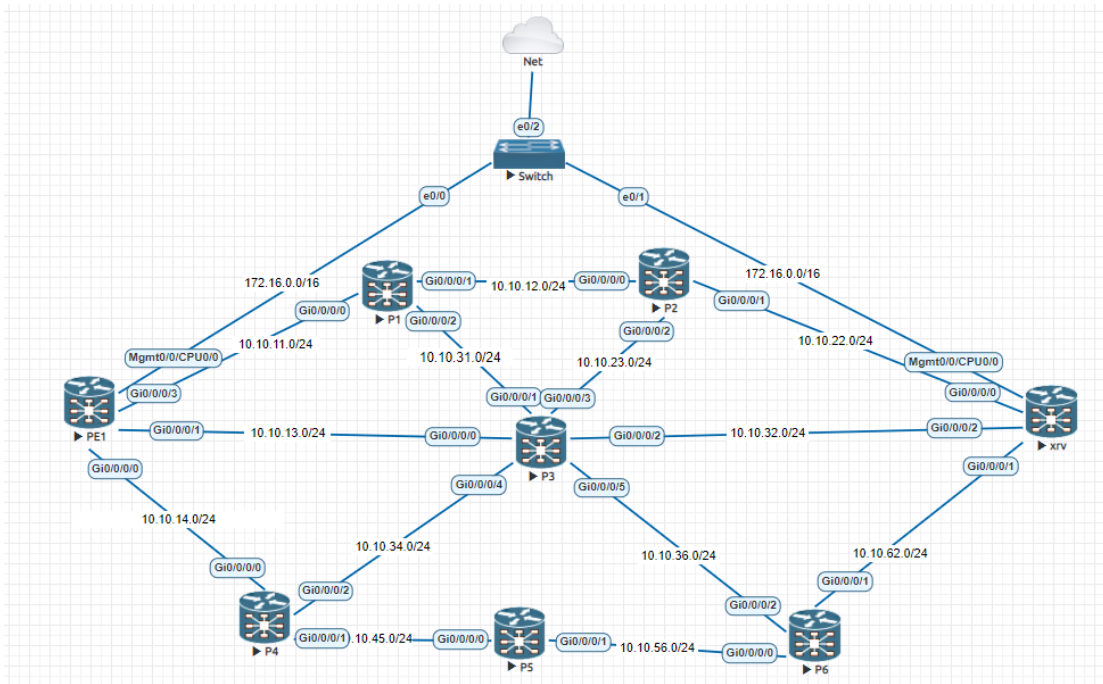


Fig. 29. Experimental Topology in EVE-NG emulator

#### 4.6 Installing SR, PCEP and BGP-LS

We will demonstrate provisioning the segment routing in our network. As seen in our topology in Fig. 29, there are eight routers, which are PE1, PE2, P1, P2, P3, P4, P5, and P6. All the routers are using the Cisco XRV 6.1.3 version and support the Segment Routing feature. The EVE emulator can connect the router to the internet or another VM by choosing Management Cloud. Edge routers PE1 and PE2 are connected to the SDN controller by the management interface. PE nodes and P nodes are connected by a gigabit Ethernet interface which supports a maximum bandwidth of 1 Gbps. First, we give IP addresses of loopback, management, and interface to each router. IP address information can be found in Table. 20. After IP addresses are correctly configured, we install IGP protocol (in this thesis, we use ISIS) on each router. As we explained in background theories, SR uses IGP extensions instead of LDP or RSVP-TE. So, we enabled segment routing under ISIS configuration, and we configure node-SID



on each loopback interface. Node-SID of each router can be seen in Table. 20. After we configured IGP, we will enable MPLS on all routers. For PCEP, the SDN controller is PCE, and PE1 and PE2 are PCC. As PE1 and PE2 are PCC, the SDN controller can only install the SR-TE path to PE1 and PE2. In PE1 and PE2, we configure them to refer to the SDN controller as PCE. The next step is to configure BGP-LS. To configure BGP-LS, we choose PE1 to form BGP peer with the SDN controller, OpenDaylight.

*Table. 20. IP address and Node-SID Information of Routers*

Router	IP Address	Node-SID
PE1	Mgmt - 172.16.1.82 Loopback - 10.10.1.100	17001
PE2	Mgmt - 172.16.2.82 Loopback - 10.10.2.100	17002
P1	Loopback - 10.10.10.100	17010
P2	Loopback - 10.10.20.100	17020
P3	Loopback - 10.10.30.100	17030
P4	Loopback - 10.10.40.100	17040
P5	Loopback - 10.10.50.100	17050
P6	Loopback - 10.10.60.100	17060

#### 4.7 Configuration in OpenDaylight

After installing the required features in ODL as described in section 4.4, the ODL will generate configuration files. In our thesis, we will configure ODL to form a BGP neighbor with PE1. We will change the default configurations of **41-bgp-example.xml** located in `"/usr/local/karaf/distribution-karaf-0.5.2-Boron-SR2/etc/opendaylight/karaf"`. Configurations can be seen in Fig. 30. 172.16.1.82 is the management interface of PE1 that is connected to the ODL controller.

```

<name>example-bgp-peer</name>
<host>172.16.1.82</host>
<holdtimer>180</holdtimer>
<retrytimer>10</retrytimer>
<peer-role>ibgp</peer-role>

```

Fig. 30. BGP Configuration at ODL

## 4.8 Implementation of SR-TE

After we finished the configuration of segment routing control and data plane, Segment Routing Traffic Engineering (SR-TE) can be implemented in two different ways: centralized and distributed. As this thesis focuses on a centralized approach using an SDN controller, we implement SR-TE with a centralized system. The controller obtains the network topology information for the centralized approach, such as the network device's maximum link bandwidth, IP addresses, metrics, and SIDs using the southbound protocol - BGP-LS and PCEP.

We will use the SHP\_En ILP model to implement SR-TE for the topology in Fig. 29. We develop a Python GUI application called "SHP\_En Traffic Engineering" for network administrators to visualize the topology, input the required traffic demand, compute the SR-TE paths for the given traffic demand and implement SR-TE tunnels with Postman.

### 4.8.1 SHP\_En Traffic Engineering Application

SHP\_En Traffic Engineering Application is written in Python programming language. The application is on the same VM with the ODL controller. The application use RESTCONF to connect to the ODL and the httplib2 library [39] to request a JSON file from ODL and use JSON library [40] to load the JSON file. The URL we use to get name, IP address, router-id, connectivity among routers, bandwidth from the ODL is

<http://172.16.1.128:8181/restconf/operational/network-topology:network-topology/topology/example-linkstate-topology>. 172.16.1.128 is the IP address of the ODL controller, and the URL to get the SID from the SDN controller is <http://172.16.1.128:8181/restconf/operational/bgp-rib:bgp-rib/rib/example-bgprib/loc-rib/>. We use URL <http://172.16.1.128:8181/restconf/operational/network-topology:network-topology/topology/pcep-topology> to know which nodes are installed for pcep.

SHP\_En Traffic Engineering Application GUI can be seen in Fig. 31. As shown in Fig. 31, the application allows the network administrator to visualize the topology instead of reading the complicated XML files and visualizing them. We use networkx [41] python library to draw the topology in the application. The network topology provided by using the information from the ODL controller can be seen as in Fig. 31. There are two buttons on the right side of the application. The button “Traffic Demand” allows the network administrator to enter the traffic demand. As shown in Fig. 32, the application only allows the network administrator to put traffic demand from PE1 and PE2 because we only configure PE1 and PE2 as PCC in our experimental topology. PCE can only configure the network devices configured as PCC. After entering traffic demand, we can press the button “Finish” and send the traffic demand to the SHP\_En ILP model. After traffic demand is set, we can press “Implement” to generate the SR-TE paths for the assigned traffic.

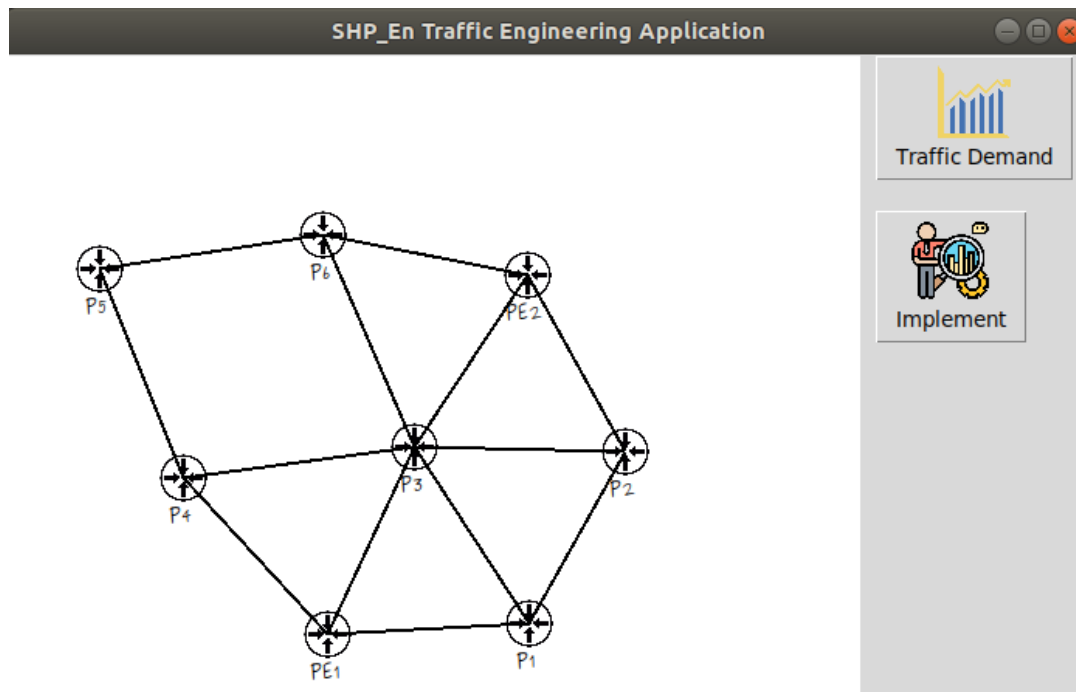


Fig. 31. SHP\_En Traffic Engineering Application GUI

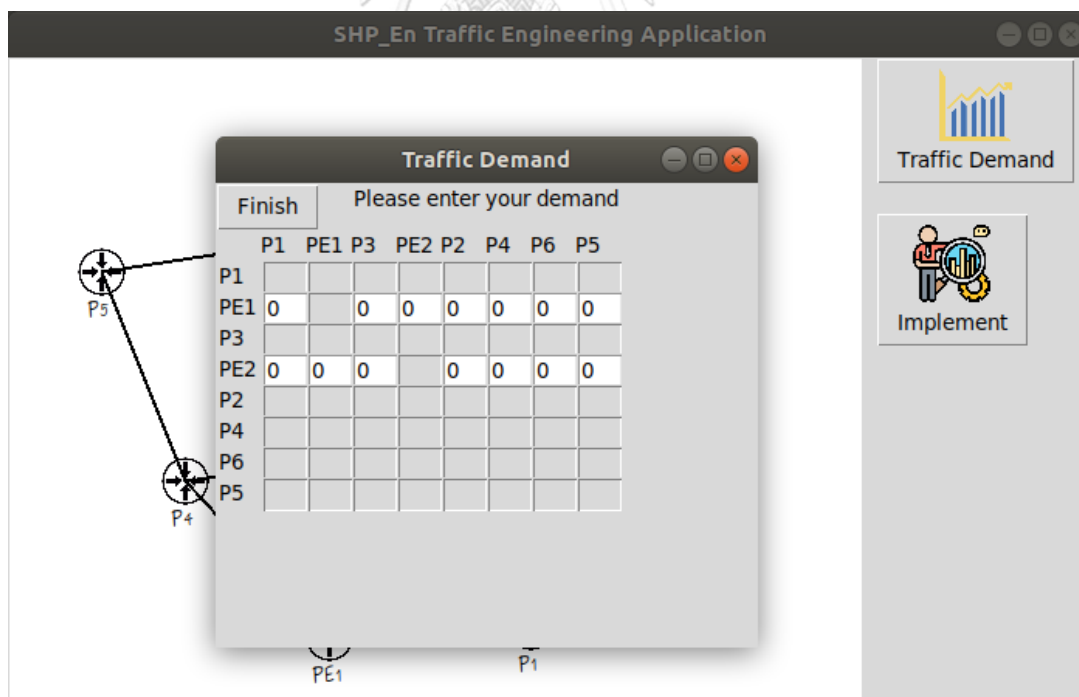


Fig. 32. Traffic Demand

To use the SHP\_En ILP model (explained detail in section 3.3), we must change some parameters. We must change the maximum utilization value  $\alpha$  to be within 0 and 1. Please note that in this emulation, the maximum bandwidth on each router's port is 1 Gbps, and we assume that each link can carry up to 1 Gbps traffic. So, we keep the maximum utilization value  $\alpha$  to be within maximum bandwidth on the link 0 to 1 Gbps. When the traffic demand more than the topology can route is put, the status will show as "Infeasible" and show the message "Select Traffic Again", as shown in Fig. 33. For example, in Fig. 33, we assume that PE1 to PE2, P4, P6, and P5 have 1 Gb of traffic demand each and PE2 to P3, P4 and P5 have 1 Gb traffic demand each. The example network infrastructure cannot handle the traffic we input, and the application asks the user to select the traffic again.

The screenshot shows the SHP\_En Traffic Engineering Application interface. A 'Traffic Demand' dialog box is open, displaying a table for entering traffic demand between various ports. The table has columns for P1, PE1, P3, PE2, P2, P4, P6, and P5, and rows for P1, PE1, P3, PE2, P2, P4, P6, and P5. The values in the table are as follows:

	P1	PE1	P3	PE2	P2	P4	P6	P5
P1								
PE1	0		0	1	0	1	1	1
P3								
PE2	0	0	1		0	1	0	1
P2								
P4								
P6								
P5								

The dialog box also has a 'Finish' button and a 'Please enter your demand' label. On the right side of the application, there is a 'Traffic Demand' button with a bar chart icon and an 'Implement' button with a person and gear icon. Below these buttons, the status is displayed as 'Status: Infeasible Select Traffic Again'.

Fig. 33. Infeasible Traffic

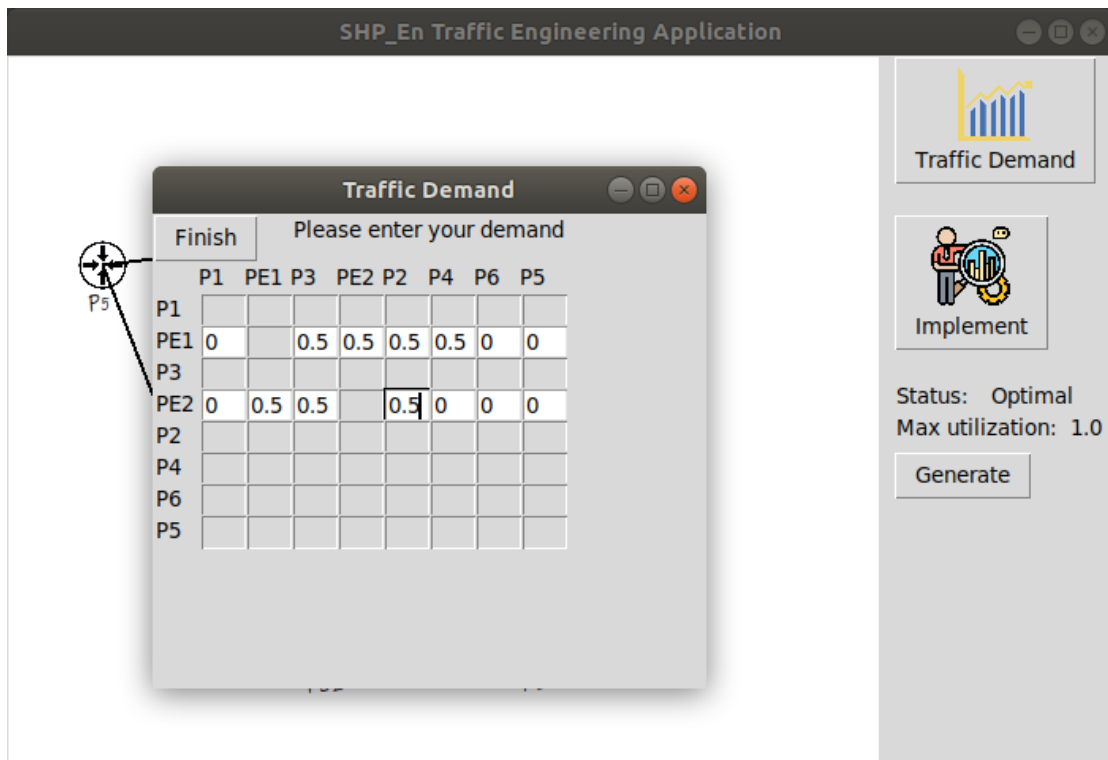


Fig. 34. Optimal Traffic Demand

When the traffic demand can be routed on the topology, the status will show as “Optimal”, and the application shows the maximum utilization on the network as shown in Fig. 34. Let’s assume we want to send 0.5 Gb (500 Mb) traffic demand from PE1 to P3, PE2, P2, and P4 and 0.5 Gb (500 Mb) traffic demand from PE2 to PE1, P3, and P2. When the user implements the traffic demand, the application will show the status: “Optimal” and the maximum utilization of 1 Gb in the network. After the traffic demand is set and the status is optimal, the application will show the “Generate” button. When the “Generate” button is pressed, the application will show SR-TE paths information and generate the XML files to set up the SR-TE tunnel via Postman. As shown in Fig. 35, the application shows the number of SR-TE tunnels, source to destination, and the path. For example, path No.2 is from PE1 to PE2, and the path will be PE1-P3-P6-PE2. XML files generated to set up SR-TE tunnels can be seen in Fig. 36.

SHP\_En Traffic Engineering Application

**SR-TE Paths Information**

ID	Path Name	Path
1	PE1->P3	PE1 P3
2	PE1->PE2	PE1 P3 P6 PE2
3	PE1->P2	PE1 P1 P2
4	PE1->P4	PE1 P4
5	PE2->PE1	PE2 P2 P1 PE1
6	PE2->P3	PE2 P3
7	PE2->P2	PE2 P2

Traffic Demand

Implement

Status: Optimal  
Max utilization: 1.0

Generate

Fig. 35. SR-TE Paths Information

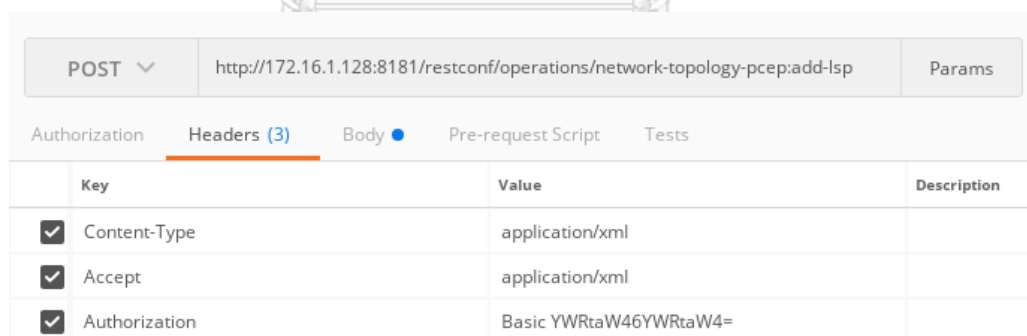
PE1->P2 PE1->P3 PE1->P4 PE1->PE2 PE2->P2

PE2->P3 PE2->PE1

Fig. 36. XML Files Generated by Application

#### 4.8.2 Implementation of SR-TE tunnels via Postman

To implement SR-TE tunnels, we will use the REST API from Postman [7] application. We will send the REST command to the ODL PCEP module and add the SR-TE tunnels from the Postman application. As we demonstrated in section 4.8.1, the application will generate the XML configuration files to implement SR-TE tunnels via Postman. As we will send the ERO to ODL, we will choose the POST method and configure the URL with the controller IP address, as shown in **Fig. 37**. The three headers type is used, such as Content-type, Accept, and Authorization. XML configuration for PE1-PE2 SR-TE tunnel generated by the application can be seen in Fig. 38. In the XML configuration generated by the application, we will see the PCC node, which is 10.10.1.100, IP address of PE1 and PE2, which are 10.10.1.100 and 10.10.2.100. For ERO, as the traffic has to go to P6 and then PE2, there will be two SIDs which are 17060 and 17002.



	Key	Value	Description
<input checked="" type="checkbox"/>	Content-Type	application/xml	
<input checked="" type="checkbox"/>	Accept	application/xml	
<input checked="" type="checkbox"/>	Authorization	Basic YWRtaW46YWRtaW4=	

*Fig. 37. SR-TE Tunnel Setup from Postman*



```

<?xml version="1.0" ?><input xmlns="urn:opendaylight:params:xml:ns:yang:topology:pcep">
<node:pcc://10.10.1.100/<node> ← PCC
<name>PE1PE2</name>
<arguments>
<lap xmlns="urn:opendaylight:params:xml:ns:yang:pcep:ietf:stateful">
<delegate>true</delegate>
<administrative>true</administrative>
</lap>
<endpoints-obj>
<ipv4>
<source-ipv4-address>10.10.1.100</source-ipv4-address> ← PE1
<destination-ipv4-address>10.10.2.100</destination-ipv4-address> ← PE2
</ipv4>
</endpoints-obj>
<path-setup-type xmlns="urn:opendaylight:params:xml:ns:yang:pcep:ietf:stateful">
<pet:l</pet>
</path-setup-type>
<ero>
<subject>
<loose>false</loose>
<sid-type xmlns="urn:opendaylight:params:xml:ns:yang:pcep:segment:routing">ipv4-node-id</sid-type>
<m-flag xmlns="urn:opendaylight:params:xml:ns:yang:pcep:segment:routing">true</m-flag>
<sid xmlns="urn:opendaylight:params:xml:ns:yang:pcep:segment:routing">17060</sid> ← P6 SID
<ip-address xmlns="urn:opendaylight:params:xml:ns:yang:pcep:segment:routing">10.10.60.100</ip-address> ← P6 IP Address
</subject>
<subject>
<loose>false</loose>
<sid-type xmlns="urn:opendaylight:params:xml:ns:yang:pcep:segment:routing">ipv4-node-id</sid-type>
<m-flag xmlns="urn:opendaylight:params:xml:ns:yang:pcep:segment:routing">true</m-flag>
<sid xmlns="urn:opendaylight:params:xml:ns:yang:pcep:segment:routing">17002</sid> ← PE2 SID
<ip-address xmlns="urn:opendaylight:params:xml:ns:yang:pcep:segment:routing">10.10.2.100</ip-address> ← PE2 IP Address
</subject>
</ero>
</arguments>
<network-topology-ref xmlns:topo="urn:YTD:params:xml:ns:yang:network-topology"/>topo:network-topology/topo:topology[topo:topology-id="pcep-topology"</network-topology-ref>
</input>

```

Fig. 38. XML Configuration for PE1-PE2 SR-TE tunnel

After we configured SR-TE tunnels from the Postman application, we have to configure the static route for each tunnel at PE1 and PE2 as ODL does not support PCEP protocol to configure a static route. We also have to use a static route or “autoreroute announce” in the tunnel’s source in the distributed approach. The configurations of the static route on PE1 and PE2 can be seen in Fig. 39.

จุฬาลงกรณ์มหาวิทยาลัย  
CHULALONGKORN UNIVERSITY

```

router static
address-family ipv4 unicast
10.10.2.100/32 tunnel-te23
10.10.20.100/32 tunnel-te24
10.10.30.100/32 tunnel-te20
10.10.40.100/32 tunnel-te25
!
!

```

(a)

```

router static
 address-family ipv4 unicast
  10.10.1.100/32 tunnel-te20
  10.10.20.100/32 tunnel-te25
  10.10.30.100/32 tunnel-te23
!
!

```

(b)

Fig. 39. Static Route Configuration for SR-TE tunnel at (a) PE1 and (b) PE2

According to the author's best knowledge, Pathman-SR [8] is the only open-source application available to implement SR-TE tunnels. This thesis is not using Pathman-SR because we cannot control the segment lists to describe an SR-TE path. For example, the path from PE1 to PE2 should go through the path PE1–P3–P6–PE2. We can use segment list [17030,17060,17002] or [17060,17002] to form a PE1 to PE2 path. If we use Pathman-SR, Pathman-SR will use SL [17030,17060,17002]. Our application, SHP\_En Traffic Engineering, will use [17060, 17002] as seen in Fig. 38. In segment routing, the labels to be appended to the packet introduces packet overhead. So, we should reduce the labels to reduce the packet overhead. SHP\_En Traffic Engineering use less labels to configure compared to Pathman-SR.

จุฬาลงกรณ์มหาวิทยาลัย  
CHULALONGKORN UNIVERSITY

#### 4.9 Results and Discussion of segment routing traffic engineering (SR-TE) tunnel implementation

We can verify whether the SR-TE tunnel is up using commands “show mpls traffic-eng tunnels” and “show mpls traffic-eng tunnels brief”, as shown in Fig. 40. To verify briefly, we can use “show mpls traffic-eng tunnels brief” to verify whether the tunnel is up or down, as shown in Fig. 40 (b) and (c) for PE1 and PE2, respectively. We can see four tunnels up from PE1 and three tunnels up from PE2 as the application generated.

To verify details of the SR-TE tunnel, we can use the command “show mpls traffic-eng tunnels”, as shown in **Fig. 40 (a)**. As we can see in **Fig. 40 (a)**, at the auto PCC section, the tunnel for PE1 to PE2 is delegated to and created by 172.16.1.128, ODL controller, which means SDN controller creates the tunnel. The SR-TE tunnel we set up is for PE1-PE2 through PE1 (SID - 17001) – P3 (SID - 17030) – P6 (SID - 17060) – PE2 (SID - 17002). We can reduce one SID, P3 (SID - 17030), to form a path. As shown in the results in **Fig. 40 (a)**, we can see segment routing path info, which includes segment list, which is 17060 and 17002.

To verify whether the packet routes through SR-TE path and segment routing data plane operations, the traceroute result is shown in **Fig. 41**. The segment lists and segment routing data plane operations went through the SR-TE tunnel for each connection. To verify whether the intermediate nodes maintain the tunnel created, we will see every intermediate router, P1, P2, P3, P4, P5, and P6. As shown in **Fig. 42**, intermediate routers do not maintain any path or tunnel states that we created for PE1 and PE2.

```

Name: tunnel-te23 Destination: 10.10.2.100 Ifhandle:0x110 (auto-tunnel pcc)
Signalled-Name: PE1PE2
Status:
  Admin: up Oper: up Path: valid Signalling: connected

  path option 10, (Segment-Routing) type explicit (autopcc_te23) (Basis for Setup)
  G-PID: 0x0800 (derived from egress interface properties)
  Bandwidth Requested: 0 kbps CT0
  Creation Time: Sun Jun 27 15:45:31 2021 (01:03:06 ago)
Config Parameters:
  Bandwidth: 0 kbps (CT0) Priority: 7 7 Affinity: 0x0/0xffff
  Metric Type: TE (global)
  Path Selection:
    Tiebreaker: Min-fill (default)
    Protection: any (default)
    Hop-limit: disabled
    Cost-limit: disabled
    Path-invalidation timeout: 10000 msec (default), Action: Tear (default)
    AutoRoute: disabled LockDown: disabled Policy class: not set
    Forward class: 0 (default)
    Forwarding-Adjacency: disabled
    Autoroute Destinations: 0
    Loadshare: 0 equal loadshares
    Auto-bw: disabled
    Path Protection: Not Enabled
    BFD Fast Detection: Disabled
    Reoptimization after affinity failure: Enabled
    SRLG discovery: Disabled
Auto PCC:
  Symbolic name: PE1PE2
  PCEP ID: 24
  Delegated to: 172.16.1.128 ← ODL SDN Controller IP Address
  Created by: 172.16.1.128
History:
  Tunnel has been up for: 01:03:06 (since Sun Jun 27 15:45:31 UTC 2021)
  Current LSP:
    Uptime: 01:03:06 (since Sun Jun 27 15:45:31 UTC 2021)

Segment-Routing Path Info (PCE controlled)
  Segment0(Node): 10.10.60.100, Label: 17060 ← Segment List
  Segment1(Node): 10.10.2.100, Label: 17002

```

(a)

```

RP/0/0/CPU0:PE1#show mpls traffic-eng tunnels brief | include up
Sun Jun 27 15:53:34.019 UTC
>tunnel-te20      10.10.30.100      up up
>tunnel-te23      10.10.2.100       up up
>tunnel-te24      10.10.20.100      up up
>tunnel-te25      10.10.40.100      up up

```

(b)

```

RP/0/0/CPU0:PE2#show mpls traffic-eng tunnels brief | include up
Sun Jun 27 15:59:30.855 UTC
>tunnel-te20      10.10.1.100       up up
>tunnel-te23      10.10.30.100      up up
>tunnel-te25      10.10.20.100      up up

```

(c)

Fig. 40. Verifying SR-TE tunnels

```
RP/0/0/CPU0:PE1#traceroute 10.10.30.100
Sun Jun 27 15:55:59.059 UTC

Type escape sequence to abort.
Tracing the route to 10.10.30.100

 1 10.10.13.3 9 msec * 0 msec
```

(a)

```
RP/0/0/CPU0:PE1#traceroute 10.10.2.100
Sun Jun 27 15:57:00.465 UTC

Type escape sequence to abort.
Tracing the route to 10.10.2.100

 1 10.10.13.3 [MPLS: Labels 17060/17002 Exp 0] 39 msec 9 msec 0 msec
 2 10.10.36.6 [MPLS: Label 17002 Exp 0] 9 msec 0 msec 0 msec
 3 10.10.62.2 0 msec * 19 msec
```

(b)

```
RP/0/0/CPU0:PE1#traceroute 10.10.20.100
Sun Jun 27 15:57:41.622 UTC

Type escape sequence to abort.
Tracing the route to 10.10.20.100

 1 10.10.11.10 [MPLS: Label 17020 Exp 0] 0 msec 0 msec 0 msec
 2 10.10.12.20 0 msec * 0 msec
```

(c)

```
RP/0/0/CPU0:PE1#traceroute 10.10.40.100
Sun Jun 27 15:58:24.689 UTC

Type escape sequence to abort.
Tracing the route to 10.10.40.100

 1 10.10.14.4 9 msec * 0 msec
```

(d)

```

RP/0/0/CPU0:PE2#traceroute 10.10.1.100
Sun Jun 27 16:01:44.726 UTC

Type escape sequence to abort.
Tracing the route to 10.10.1.100

 1 10.10.22.20 [MPLS: Labels 17010/17001 Exp 0] 9 msec 0 msec 0 msec
 2 10.10.12.1 [MPLS: Label 17001 Exp 0] 0 msec 0 msec 0 msec
 3 10.10.11.1 0 msec * 0 msec

```

(e)

```

RP/0/0/CPU0:PE2#traceroute 10.10.30.100
Sun Jun 27 16:02:52.501 UTC

Type escape sequence to abort.
Tracing the route to 10.10.30.100

 1 10.10.32.3 0 msec * 0 msec

```

(f)

```

RP/0/0/CPU0:PE2#traceroute 10.10.20.100
Sun Jun 27 16:03:26.359 UTC

Type escape sequence to abort.
Tracing the route to 10.10.20.100

 1 10.10.22.20 0 msec * 0 msec

```

(g)

Fig. 41. Traceroute results (a) PE1 to P3 (b) PE1 to PE2 (c) PE1 to P2 (d) PE1 to P4 (e) PE2 to PE1 (f) PE2 to P3 and (g) PE2 to P2

```

RP/0/0/CPU0:P1#show mpls traffic-eng tunnels brief
Sun Jun 27 16:04:48.653 UTC

```

(a)

```

RP/0/0/CPU0:P2#show mpls traffic-eng tunnels brief
Sun Jun 27 16:05:17.511 UTC

```

(b)

```
RP/0/0/CPU0:P3#show mpls traffic-eng tunnels brief  
Sun Jun 27 16:05:49.499 UTC
```

(c)

```
RP/0/0/CPU0:P4#show mpls traffic-eng tunnels brief  
Sun Jun 27 16:06:42.725 UTC
```

(d)

```
RP/0/0/CPU0:P5#show mpls traffic-eng tunnels brief  
Sun Jun 27 16:08:16.139 UTC
```

(e)

```
RP/0/0/CPU0:P6#show mpls traffic-eng tunnels brief  
Sun Jun 27 16:08:46.197 UTC
```

(f)

Fig. 42. Verifying intermediate routers (a) P1 (b) P2 (c) P3 (d) P4 (e) P5 and (f) P6

## Chapter 5

### Conclusion

#### 5.1 Conclusion

This thesis studied the ILP models of the primary reference paper [1], namely ECMP, SHP, SEGMR, and proposed the enhanced version of SHP, SHP\_En, to evaluate TE performance in SR networks. We can conclude from the obtained results that depending on the traffic demands and topologies, the default SR behavior ECMP generally leads to higher maximum utilization of links in the network than SHP, SHP\_En, and SEGMR. In addition to higher maximum utilization, we should also avoid ECMP because ECMP may cause packet mis-reordering at the destination, and some hardware capabilities cannot support per-flow granularity. SHP\_En can achieve the maximum utilization as the same as SEGMR by extending shortest paths to k-shortest paths. As SHP and SHP\_En are forced to choose only one shortest path, the SLD to be appended to form an SR-TE path can be larger. We also proposed how to reduce SLD, and we can see that we have to use only up to labels that can be deployed to the commercial routers for the considered topologies. SEGMR requires only three labels to achieve effective utilization, but SEGMR takes longer to provide the results than the SHP\_En model. SHP\_En model can be applied to low to medium size networks because ILP can take a long time to provide optimal results, and we may need to provide more k-shortest paths and higher SLD in large networks.

This thesis also implements the SHP\_En ILP models to the emulated environment. This thesis developed an application called “SHP\_En Traffic Engineering Application” to visualize the network infrastructure, take the traffic demand, implement the SHP\_En, and generates XML files to configure the SR-TE tunnels to Cisco IOS XRV through SDN controller. We can see that “SHP\_En Traffic Engineering



Application” can provide the network administrator with the forwarding abstraction and give the effective TE solution to apply traffic demand. In addition, the application can reduce SLDs compared to Cisco Pathman-SR. From the traceroute results of the routers, we can see that the intermediate nodes do not maintain any tunnel state that we created, which improves the scalability of the traditional MPLS network.



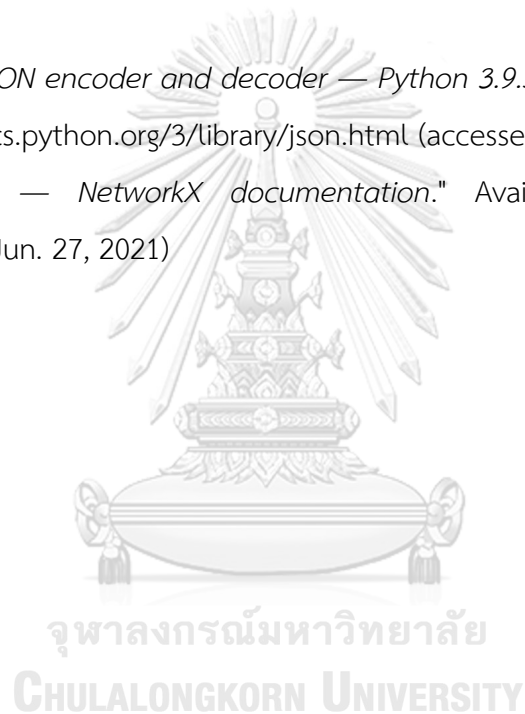
## REFERENCES

- [1] E. Moreno, A. Beghelli, and F. J. C. N. Cugini, "Traffic engineering in segment routing networks," *Comput. Netw.*, vol. 114, pp. 23-31, Feb. 2017.
- [2] C. Filsfils, S. Previdi, B. Decraene, S. Litkowski, and R. J. N. W. G. Shakir, Internet-Draft, "Segment Routing Architecture. draft-ietf-springsegment-routing-01," pp. 1-19, 2015.
- [3] Z. N. Abdullah, I. Ahmad, I. J. I. C. S. Hussain, and Tutorials, "Segment routing in software defined networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 464-486, 1st Quart., 2018.
- [4] A. Mendiola, J. Astorga, E. Jacob, M. J. I. C. S. Higuero, and Tutorials, "A survey on the contributions of software-defined networking to traffic engineering," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 918-953, 2nd Quart., 2017.
- [5] "Eve-ng." Available: <https://www.eve-ng.net/> (accessed Nov. 14, 2020).
- [6] "Home - OpenDaylight." Available: <https://www.opendaylight.org/> (accessed Nov. 18, 2020).
- [7] "Postman | The Collaboration Platform for API Development." Available: <https://www.postman.com/> (accessed Nov. 21, 2020).
- [8] "pathman-sr/dCloud at master CiscoDevNet/pathman-sr." Available: <https://github.com/CiscoDevNet/pathman-sr/tree/master/dCloud> (accessed Nov. 21, 2020).
- [9] "PuLP · PyPI." Available: <https://pypi.org/project/PuLP/> (accessed Jun. 26, 2021).
- [10] A. Headquarters, "Cisco IOS XRv Router Installation and Configuration Guide," no. 6387, 2014.
- [11] F. Lazzeri, G. Bruno, J. Nijhof, A. Giorgetti, and P. Castoldi, "Efficient label encoding in segment-routing enabled optical networks," in *Proc. IEEE Int. Conf. Opt. Netw. Design Model. (ONDM)*, 2015, pp. 34-38.
- [12] A. Giorgetti, P. Castoldi, F. Cugini, J. Nijhof, F. Lazzeri, and G. Bruno, "Path encoding in segment routing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2015, pp. 1-6.

- [13] O. Dugeon, R. Guedrez, S. Lahoud, and G. Texier, "Demonstration of segment routing with sdn based label stack optimization," in *Proc. IEEE 20th Conf. Innov. Clouds Internet Netw. (ICIN)*, 2017, pp. 143-145.
- [14] R. Bhatia, F. Hao, M. Kodialam, and T. Lakshman, "Optimized network traffic engineering using segment routing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2015, pp. 657-665.
- [15] A. Sgambelluri, F. Paolucci, A. Giorgetti, F. Cugini, and P. J. J. o. L. T. Castoldi, "Experimental demonstration of segment routing," *J. Lightw. Technol.*, vol. 34, no. 1, pp. 205-212, Jan. 1, 2016.
- [16] S. M. J. Yasini, "Implementation of segment routing and MPLS traffic engineering in software-defined network based on GNS3 network emulator and OpenDayLight SDN controller," 2018.
- [17] M. T. Z. Win, K. T. Mya, and Y. Ishibashi, "Traffic Engineering with Segment Routing in ONOS Controller," *Sixteenth International Conferences on Computer Applications (ICCA 2018)*, 2018.
- [18] "[Archived] ONF's SPRING-OPEN project - ONOS - Wiki." Available: <https://wiki.onosproject.org/display/ONOS/%5BArchived%5D+ONF%27s+SPRING-OPEN+project> (accessed Nov. 16, 2020).
- [19] I. Šeremet and S. Čaušević, "Advancing Multiprotocol Label Switching Traffic Engineering with Segment Routing in Software Defined Network environment," in *2020 19th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pp. 1-6, 2020.
- [20] "Introduction to MPLS." Available: <https://networklessons.com/mpls/introduction-to-mpls> (accessed Jul. 01, 2021).
- [21] S. Smith, "Introduction to MPLS," *2003 Tech. Symp*, 2003.
- [22] "MPLS Basics." Available: <https://mplstutorial.com/mpls-basics> (accessed Oct. 27, 2020).
- [23] S. Jose, "MPLS Label Distribution Protocol Configuration Guide, Cisco IOS XE Release 3S," no. 6387, 2013.
- [24] "draft-ietf-spring-segment-routing-09 - Segment Routing Architecture." Available: <https://tools.ietf.org/html/draft-ietf-spring-segment-routing-09> (accessed Nov. 24, 2020).

- [25] C. S. e. al., "RFC 8754 IPv6 Segment Routing Header (SRH) Abstract," pp. 1–27, 2020.
- [26] I. P. I. I. S. S. Is-is, and V. Ipv, "Configure Segment Routing for IS-IS Protocol Enabling Segment Routing for IS-IS Protocol," vol. 4, pp. 1–14, 1995.
- [27] SDNLAB, "简介：什么是Segment Routing?," pp. 1–7, 2020.
- [28] "RFC 4657 - Path Computation Element (PCE) Communication Protocol Generic Requirements." Available: <https://tools.ietf.org/html/rfc4657> (accessed Nov. 24, 2020).
- [29] "PCEP Configuration - TechLibrary - Juniper Networks." Available: [https://www.juniper.net/documentation/en\\_US/junos/topics/topic-map/pcepconfiguration.html](https://www.juniper.net/documentation/en_US/junos/topics/topic-map/pcepconfiguration.html) (accessed Nov. 23, 2020).
- [30] P. L. B.-l. B. Gateway, F. Identifier, and S. Family, "Border Gateway Protocol Link-State Information About Border Gateway Protocol Link-State Overview of Link-State Information in Border Gateway Protocol," pp. 1–12.
- [31] N. M. e. al., "OpenFlow," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74.
- [32] "RFC 7752 - North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP." Available: <https://tools.ietf.org/html/rfc7752> (accessed Nov. 24, 2020).
- [33] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. J. P. o. t. I. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14-76, 2014.
- [34] S. Shenker, M. Casado, T. Koponen, and N. J. O. N. S. McKeown, "The future of networking, and the past of protocols," vol. 20, pp. 1-30, 2011.
- [35] "Welcome to RYU the Network Operating System(NOS) — Ryu 4.34 documentation." Available: <https://ryu.readthedocs.io/en/latest/> (accessed Sep. 10, 2020).
- [36] "Segment Routing Configuration Guide, Cisco IOS XE Gibraltar 16.11.x - Segment Routing Traffic Engineering With IS-IS [Cisco IOS XE 16] - Cisco." Available:

- [https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/seg\\_routing/configuration/xr-16-11/segrrt-xe-16-11-book/seg-rout-traffic-engg.html](https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/seg_routing/configuration/xr-16-11/segrrt-xe-16-11-book/seg-rout-traffic-engg.html) (accessed Jul. 03, 2021).
- [37] "Download VMware Workstation Pro | LATAM." Available: <https://www.vmware.com/latam/products/workstation-pro/workstation-pro-evaluation.html> (accessed Jun. 27, 2021).
- [38] "Ubuntu 18.04.5 LTS (Bionic Beaver)." Available: <https://releases.ubuntu.com/18.04/> (accessed Nov. 23, 2020).
- [39] "httplib2 · PyPI." Available: <https://pypi.org/project/httplib2/> (accessed Jun. 27, 2021).
- [40] "json — JSON encoder and decoder — Python 3.9.5 documentation." Available: <https://docs.python.org/3/library/json.html> (accessed Jun. 27, 2021)
- [41] "NetworkX — NetworkX documentation." Available: <https://networkx.org/> (accessed Jun. 27, 2021)





จุฬาลงกรณ์มหาวิทยาลัย  
**CHULALONGKORN UNIVERSITY**

## VITA

**NAME** Htain Lynn Aung

**DATE OF BIRTH** 24 Jan 1996

**PLACE OF BIRTH** Taunggyi, Myanmar

**INSTITUTIONS ATTENDED** Chulalongkorn University  
University of Information Technology

**HOME ADDRESS** Ratchathewi, Bangkok

**PUBLICATION** H. L. Aung, S. Chaudhary, M. Saadi, and L. Wuttisittikulki,  
"Implementing TrafficEngineering with Segment Routing  
using OpenDaylight Controller and Pathman-SR," in 2021  
36th International Technical Conference on  
Circuits/Systems, Computers and Communications (ITC-  
CSCC), 2021, pp. 1-4