DEVELOPMENT OF MPLS TEST-BED FOR NETWORK TRAFFIC ENGINEERING

MOHD TAUFIK BIN JUSOH @ TAJUDIN

A thesis submitted in fulfilment of the requirements for the award of the Degree of Master of Engineering (Electrical-Electronic & Telecommunication)

Faculty of Electrical Engineering
Universiti Teknologi Malaysia

30 NOVEMBER 2005

To

My beloved Parents, Sisters and Brothers

for Their

Love, Sacrifices and Best Wished

ACKNOWLEDGEMENT

In the name of ALLAH S.W.T, the Most Beneficent and the Merciful, praise be to ALLAH S.W.T for the incredible gift endowed upon me and for the health and strength given to me in order to finish the project and to prepare this thesis.

First of all, I would like to thank my supervisor, Prof. Dr. Norsheila Binti Fisal for her keen effort, interest, advice, consistent guidance and insightful comments throughout the period of this study. I also would like to take this opportunity to thank Mr. Adel and all masters student at Telekom Lab, Faculty of Electrical Engineering and also not to forget a lot of thanks to all technicians at Switching Lab and Acoustic Lab. Without their helps, this project could not be done.

Special words of gratitude are dedicated to my classmate batch 04/05 especially to Miss Anis Shahida Niza Binti Mohktar and Mr. Kamaru for their help and advice. A lot of thank to my best buddy, Mr. Ismail Ibrahim and Mr. Jack for his moral support on me. To my entire clique, I deeply appreciate all your helps in finishing this project.

Finally, extraordinary thanks and love to my beloved parents, siblings and relatives for their full support, encouragement and love throughout my studies in UTM. With them, this life is very meaningful.

ABSTRACT

Providing Quality of Service (QoS) and Traffic Engineering (TE) capabilities in the Internet is essential, especially in supporting the requirement of real-time traffic, as well as mission critical applications. For that reason, the current Internet must be enhanced with new technology that enables it to offer capabilities for controlling its behaviour as needed. Multiprotocol Label Switching (MPLS) is an emerging technology that provides QoS and traffic engineering features in IP network. This study is mainly concerned on how to develop MPLS test-bed in order to assess MPLS functionalities. The goal is to develop MPLS test-bed using Linux operating system with kernel version 2.6.5 as the platform. In this work the MPLS test-bed consists of four MPLS routers and two host terminals. MPLS software package version 1.946 has been used to build up routers similar to existing routers in MPLS domain called as Label Switched Router (LSR). Once the routers are well configured, the connection between two routers is established to create Label Switched Path (LSP). This LSP connection is also used to create new LSP from ingress router to egress router. IP packets are sent from ingress router to host terminals to validate the test-bed. These packets were encapsulated with MPLS header and they are examined by using 'tcpdump' command in Linux terminal which shows that the test-bed is successfully developed. In order to enhance test-bed functionalities, packet generator can be added to the test-bed so that UDP and TCP throughput measurement can be done. Besides, RSVP and Diffserv can be integrated into the test-bed for future study.

ABSTRAK

Menyediakan keupayaan Kualiti Perkhidmatan (QoS) dan Kejuruteraan Trafik (TE) dalam Internet amatlah penting, lebih-lebih lagi bagi menyokong keperluan aplikasi masa nyata seperti aplikasi-aplikasi yang lebih kritikal. Oleh itu, teknologi Internet masa kini perlulah diperluaskan dengan teknologi terbaru untuk membolehkan ia menyediakan keupayaan yang diperlukan bagi mengawal tingkahlakunya sendiri seperti yang diinginkan. Pensuisan Label Pelbagai Protokol (MPLS) adalah satu teknologi yang sedang berkembang dimana ia menyediakan ciri-ciri QoS dan TE dalam rangkaian Protokol Internet (IP). Kajian tesis ini tertumpu bagaimana membangunkan rangkaian-uji MPLS (MPLS test-bed) bagi menilai fungsi-fungsinya. Dengan objektif membangunkan rangkaian-uji MPLS, sistem operasi Linux dengan kernel versi 2.6.5 diperlukan sebagai asas pembangunan. Dalam kajian ini pakej perisian MPLS versi 1.946 telah digunakan untuk membangunkan *router* yang berfungsi sama seperti *router* yang terdapat dalam domain MPLS dan ia dipanggil sebagai *Router* Pensuisan Label (LSR). Setelah *router* siap dibina, penyambungan antara dua *router* dilakukan untuk menyediakan Laluan Label Tersuis (LSP). Penyambung LSP ini juga turut digunakan untuk membina satu LSP baru yang menghubungkan *ingress router* dan *egress router*. Paket-paket IP dihantar dari *ingress router* ke terminal *host* bagi mengesahkan rangkaian-uji MPLS. Paket-paket IP yang telah digabungkan dengan kepala MPLS ini akan diuji menggunakan arahan ‘tcpdump’ yang terdapat dalam terminal Linux untuk menunjukkan bahawa rangkaian-uji ini telah siap dibangunkan. Sebagai langkah untuk memperluaskan fungsi rangkaian-uji ini, penjana paket boleh diganding bersama. Dengan itu, kualiti penghantaran TCP dan UDP boleh ditentukan. Selain itu, RSVP dan Diffserv boleh juga digabungkan bersama untuk kajian dimasa akan datang.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ATM	-	Asynchronous Transfer Mode
CoS	-	Class of Service
CR-LDP	-	Constraint-based Routing Label Distribution Protocol
DLCI	-	Data Link Connection Identifier
DWDM	-	Dense Wavelength Division Multiplexing
FEC	-	Forwarding Equivalent Class
FSF	-	Free Software Foundation
GMPLS	-	Generalized Multiprotocol Label Switching
GPL	-	General Public License
IETF	-	Internet Engineering Task Force
IGP	-	Interior Gateway Protocol
IntServ	-	Integrated Services
IP	-	Internet Protocol
IS-IS	-	Intermediate System-Intermediate System
LDP	-	Label Distribution Protocol
LER	-	Label Edge Router
L-LSP	-	Label inferred Label Switched Path
LMP	-	Link Management Protocol
LSR	-	Label Switching Router
LSP	-	Label Switched Path
MEM	-	Micro-Electric Mechanical Systems

MPLS	-	Multiprotocol Label Switching
MPλS	-	Multiprotocol Lambda Switching
OADM	-	Optical Add-Drop Multiplexer
OS	-	Operating System
OSI	-	Open System Interconnection
OSPF	-	Open Shortest Path First
OXC	-	Optical Cross-Connects
PXC	-	Photonic Cross-Connect
PVC	-	Permanent Virtual Circuits
PC	-	Personal Computer
QoS	-	Quality of Service
RSVP	-	Resource Reservation Protocol
RSVP-TE	-	Reservation Protocol with Traffic Engineering
RTT	-	Round Trip Time
SONET	-	Synchronous Optical Network
TDM	-	Time Division Multiplexing
TE	-	Traffic Engineering
TLV	-	Type Length Value
TTL	-	Time to Live
VCI	-	Virtual Circuit Identifier
VoIP	-	Voice over IP
VPI	-	Virtual Path Identifier
VPN	-	Virtual Private Network
WAN	-	Wide Area Network
WWW		World Wide Web

LIST OF APPENDICES

CHAPTER I

1.1 Overview of Internet Challenges

The overwhelming growth of the Internet and the growing popularity of real-time applications set new challenges to the Internet community. Big Internet service providers are growing ever larger and supporting increasingly a variety of services, with different requirements both from different applications and their customers. Service providers need for a commercially viable and scalable tools to make the most of their networks in order to increase their revenues by supporting the needs of time or and mission critical applications [24]. This is because different applications have varying needs for delay, delay variation (jitters), bandwidth, and packet loss.

For example, real-time applications such as Voice over IP (VoIP) and video conferencing are extremely latency-dependent. Here, the timeliness of data delivery is an issue of utmost importance. But in the Internet, where there is no predictable traffic control, these applications do not run effectively. Service differentiation for traffic flows and performance optimization of the operational networks are very critical for the Internet to remain as successful as before. However, the Internet, particularly its core protocol IP (Internet Protocol) was never designed with Quality of Service (QoS) in mind. Instead it was originally designed as a research and educational resource, and thus the underlying technology that forms the backbone of today's Internet is largely based on that philosophy [24]. But times have changed a lot since, and service differentiation using QoS mechanisms while optimizing the operational network has become quite an important issue in the Internet for these applications to run effectively and efficiently.

On the other hand, as the Internet is required to support different types of services, effective and efficient bandwidth management tools in IP networks becomes increasingly important, especially when dealing with how to allocate the available network resources in order to optimize the overall performance of the networks. And yet, when the network has to sustain heavy traffic load, and has limited resources, the situation of having some congested links, while others remain underutilized is almost an inevitable phenomenon [24]. One of the main reasons to cause such congestion events in IP networks is that of the destination based forwarding paradigm.

In IP networks, the Interior Gateway Protocols (IGPs), such as Open Shortest Path First (OSPF), and Intermediate System-Intermediate System (IS-IS) routing protocols use destination-based forwarding algorithm, without considering other network parameters, such as the available bandwidth. In effect, all traffic between any two nodes traverses across the IGP shortest path. Hence, it is obvious that such situation can create hot spots on the shortest distance between two points, while other alternative routes may still be underutilized. As a result, degradation of throughput, and long delay, and packet losses can be noticed. In such situation, minimizing the effects of congestion by optimizing the performance of the operational networks becomes more critical.

Traffic Engineering (TE) is very important in this regard [11], and plays a key role in that it offers service providers a means for performance optimization and bandwidth provisioning. In fact, without TE, it is also difficult to support QoS on a large scale and at reasonable cost [12]. Therefore, the key to address the problem of traffic engineering is to have the ability to place the traffic onto the network as flexibly as needed, so that the congestion in the network can be minimized before it leads to poor network performance. Because minimizing congestion by optimizing the distribution of traffic on a given network is the central goal of TE [13].

In order to address the QoS issue, however, the ability to introduce connection-oriented forwarding techniques to connectionless IP networks becomes necessary. In effect, this allows IP networks to reserve resources, such as bandwidth over predetermined paths for service differentiation in order to provide QoS guarantees.

1.2 Project Background

Multiprotocol Label Switching (MPLS) is an emerging technology, which plays a key role in IP networks by delivering QoS, as well as traffic engineering features. MPLS has been developed and standardized by the Internet Engineering Task Force (IETF) to address these issues, in a more scalable and cost effective way.

A lot of research has been done on MPLS performance study, and MLPS does provide QoS and TE. In University Teknologi Malaysia, MPLS test-bed not yet been developed and used. Hence, this project is prominent since it would be a pioneer in MPLS study in our university.

1.3 Objectives of Research

Internet nowadays is facing a lot of challenges due to the gigantic numbers of users and fast growing of Internet applications. One way to overcome this problem is to provide QoS. However QoS cannot be achieved without having TE along. Due to these challenges, this project aim is to develop MPLS test-bed for network traffic engineering and to preserve QoS through explicit routing in Internet using MPLS.

1.4 Research Scopes

In order to develop MPLS test-bed, the components in MPLS domain itself need to be set up previously. For this project, four routers are needed to provide alternative routes between routers in MPLS network. Routers in MPLS test-bed which are able to forward IP packets in MPLS domain are called Label Switching Router (LSR). The routers basically based on Linux operating system since this operating system is open-source system, and the software to turn ordinary Linux PC into LSR are freely available in the Internet. Other scopes of this project are to enable the connection between routers and to create Label Switched Path (LSP). Finally IP packets passing through routers using LSP are validated using Linux command.

1.5 Thesis Outline

The contents of the project are further subdivided into chapters that described in details. Chapter I discussed about the overviews of current challenges in Internet technology. The factors why core network must provide new technology and must be improved are also specified in details. This chapter also introduced project background, objective of the research and scopes that are involved in order to finish the project. The overall project's process flow chart also shown.

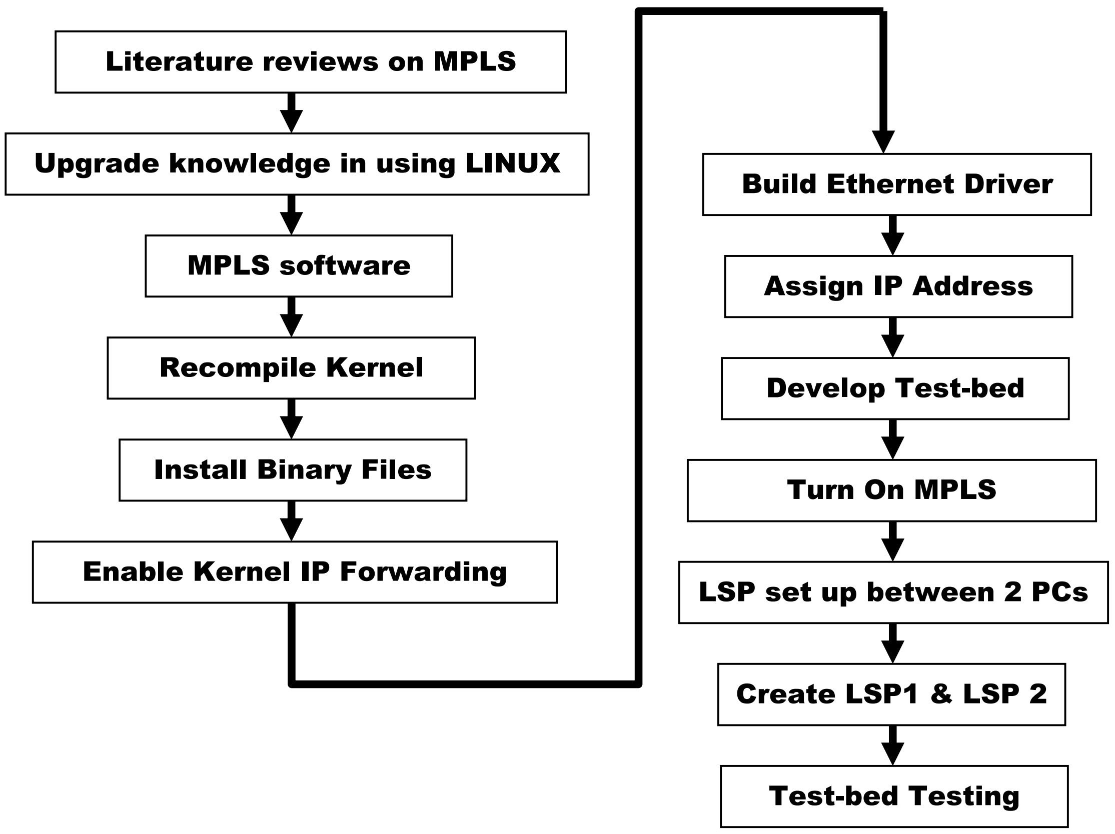
The Chapter II discussed literature studies and related works. The basic fundamentals and architecture of MPLS are explained in details. Other MPLS applications such as Virtual Private Network (VPN) and Asynchronous Transfer Mode (ATM) with IP integration also discussed. How MPLS provides Traffic Engineering (TE), Quality of service (QoS) and Class of Service (CoS) are elaborated in this chapter. Moreover, operating system used in this project is explained briefly. Finally the implementations of MPLS in Linux are explained in details such as its data structures and the processes happened when Linux machines run MPLS.

Chapter III explained about design and procedures done for this project. The steps to configure Linux machine into MPLS router are also explained. Label Switched Path (LSP) establishment between routers are described step by step for easy understanding.

In Chapter IV, results are shown and discussed. This chapter covers the results of LSP set up between two routers and the results of LSP between ingress and ingress router. Finally in this chapter, the results of Round Trip Time (RTT) for the test-bed are shown.

Lastly in Chapter V, the conclusions of this project are described. Some future works that can be done in order to enhance are explained.

1.6 Project Flow Chart



Progress of this study can be summarized into flow chart shown above. First of all, literature reviews on MPLS need to be done to acquire basic knowledge about MPLS. Journals and papers were referred to enhance the understanding and to get the overview of the project. Since this project requires Linux operating system as the platform, the knowledge using Linux must be enhanced. The understanding of Linux critical scripts that exists in Linux kernel is also very important. MPLS software is taken from Sourceforge homepage. This software package is still under development thus MPLS package version 1.946 is chosen since it is the latest up date package. Before Linux PC can be change into MPLS router, the kernel need to be compiled. The appropriate binary files are installed in order to make MPLS router to function.

Furthermore, kernel IP forwarding in Linux machine has to be change into enable state since it is disabled by default. Linux machines are connected together with appropriate IP addresses. But this can only be done when the ethernet card's driver is installed in every Linux machine. Before LSP between two routers can be established, each router must be MPLS enabled by turning on MPLS. The connection between two routers is used again to create new LSP but now it connects ingress router to egress router.

In this study, two LSP that are LSP 1 and LSP 2 which connects ingress to egress are created. LSP 2 connects four routers while LSP 1 connects only three routers. The test-bed is tested by sending IP packet into it. When the packets captured were encapsulated with MPLS header it shown that the MPLS test-bed was well developed.

time. Since MPLS test-bed is already developed it can be used to study about traffic performance such as TCP and UDP throughput measurement. Furthermore packet generator can be added with this test-bed to provide real traffic environment. Finally other QoS such as RSVP and DiffServ also can be integrated into this test-bed for further study.

REFERENCES

1. Steve Maxwell (2000). *RedHat Linux Network Management Tools*. N.Y.: Mc Graw Hill. 105-171.
2. Bryan Pfaffenberger (2001). *Linux Networking Clearly Explained*. University of Virginia.: Morgan Kaufmann. 93-114.
3. Christopher Negus (2003). *Red Hat® Linux® 9 Bible*. Canada.: Wiley. 1-16, 123-155.
4. Naba Barbakati (2004). *Red Hat® Fedora™ Linux® 2 All-In-One Desk Referencefor Dummies®*. N.J.: Wiley. 319-325.
5. Eric Foster-Johnson (2003). *Red Hat® RPM Guide*. IN.: Wiley. 33-58.
6. Terry Collings and Kurt Wall (2004). *Red Hat® Linux® Networking and System Administration 2nd Edition*. IN.: Wiley. 206-211.
7. Richard Petersen (2004). *Red Hat®: The Complete Reference Enterprise Linux & Fedora™ Eedition*. N.Y.: Mc Graw Hill. 774-776.
8. Stephen A.Thomas (2002). *IP Switching and Routing Essentials*. N.Y.: Wiley. 46-57.
9. David E. McDysan and Dave Paw (2002). *ATM & MPLS Theory & Application*: *Foundations of Multi-Service Networking*. N.Y.: Mc Graw Hill. 387-413, 505-506,520-522, 697.
10. Ilyas Mohammad and T. Hussein Mouftah (2003). *Handbook of Optical Communication Networks*. N.Y.: CRC PRESS. Chapter 5: Multiprotocol Label Switching (Matthew N.O. Sadiku).
11. G. Swallow. *MPLS Advantages for Traffic Engineering*. IEEE Communication. December 1999.

12. Zheng Wang. *Internet QoS: Architectures and Mechanism for Quality of Service*. Morgan Kufmann 2001.
13. D. Awduche. *MPLS and Traffic Engineering in IP Networks*. IEEE Communication. December 1999.
14. D. Awduche et al. *Overview and Principles of Internet Traffic Engineering*. IETF RFC 3722. May 2002.
15. B. Jamoussi et al. *Constraint-Based LSP Setup using LDP*. Draft-ietf-mpls-cr-ldp-0.5.txt. Febuary 2001.
16. D. Awduche et al. *RSVP-TE: Extensions to RSVP for LSP Tunnels*. Draft-ietf-mpls-rsvp-lsp-tunnel-0.8.txt. Febuary 2001.
17. R. Braden, D. Clark and S. Shenkar. *Intergrated Services in the Internet Architecture*. RFC 1633. July 1994.
18. Braden et al. *Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification*. RFC 2205 September 1997.
19. S. Blake et al. *An architecture for Differentiated Services*. IETF RFC 2475. December 1998.
20. Le Faucheur et al. *MPLS Support of Differentiated Services*. Draft-ietf-mpls-diff-ext-08.txt. February 2001.
21. E. Rosen et al. *Multiprotocol Label Switching Architecture*. IETF RFC 3031. January 2001.
22. D. Awduche et al. *Requirements for Traffic Engineering Over MPLS*. IETF RFC 2702. September 1999.
23. R. Callon, et al. *A Framework for Multiprotocol Label Switching*. IETF *Internet draft*. Febuary 1999.
24. T. Byle, R. Aibara and K. Mishimura. *Performance Measurements of MPLS Traffic Engineering and QoS*. Proceeding of the 11th annual Internet Society Conference, INET 2001, Stockholm, Sweden, June 2001.
25. MPLS for Linux Home Page (SourceForge) http://sourceforge.net/projects/mpls-linux

APPENDIX A

Kernel Script for R-4

```
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local
echo "1" > /proc/sys/net/ipv4/ip_forward
echo "1" > /proc/sys/net/ipv4/ip_dynaddr
#iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

#Kernel IP table 6.9.2005

      route add 10.0.1.1 dev eth0

# Assign ethernet device

      mpls labelspace add dev eth0 labelspace 0

# Create "incoming label map"

      mpls ilm add label gen 17 labelspace 0
      mpls ilm add label gen 37 labelspace 0
      mpls ilm add label gen 57 labelspace 0
```

```
# LSP between 0.1(R-4) to 0.2(R-1)

      mpls nhlfe add key 0
      mpls nhlfe change key 0x2 instructions push gen 16 nexthop eth0
ipv4 10.0.1.2
      ip route add 10.0.1.2/32 via 10.0.1.2 spec_nh 0x8847 0x2

# Bind label 36 for network 10.0.5.0/24 (FEC)

      mpls nhlfe add key 0
      mpls nhlfe change key 0x3 instructions push gen 36 nexthop eth0
ipv4 10.0.1.2
      ip route add 10.0.5.0/24 via 10.0.1.2 spec_nh 0x8847 0x3

# Bind label 56 for network 10.0.6.0/24 (FEC)

      mpls nhlfe add key 0
      mpls nhlfe change key 0x4 instructions push gen 56 nexthop eth0
ipv4 10.0.1.2
      ip route add 10.0.6.0/24 via 10.0.1.2 spec_nh 0x8847 0x4
```

APPENDIX B

Kernel Script for R-1

```
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local
echo "1" > /proc/sys/net/ipv4/ip_forward
echo "1" > /proc/sys/net/ipv4/ip_dynaddr
#iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE


# Kernel IP routing table 6.9.2005

      route add 10.0.1.2 dev eth0
      route add 10.0.3.1 dev eth1
      route add 10.0.2.1 dev eth2

# Assign ethernet device

      mpls labelspace add dev eth0 labelspace 0
      mpls labelspace add dev eth1 labelspace 1
      mpls labelspace add dev eth2 labelspace 2
```

```
# Create "incoming label map"

      mpls ilm add label gen 16 labelspace 0
      mpls ilm add label gen 36 labelspace 0
      mpls ilm add label gen 56 labelspace 0
      mpls ilm add label gen 27 labelspace 1
      mpls ilm add label gen 19 labelspace 2

# LSP set up R-1 to R-4

      mpls nhlfe add key 0
      mpls nhlfe change key 0x2 instructions push gen 17 nexthop eth0
ipv4 10.0.1.1
      ip route add 10.0.1.1/32 via 10.0.1.1 spec_nh 0x8847 0x2

# Enable returned packet

      mpls xc add ilm_label gen 19 ilm_labelspace 2 nhlfe_key 0x2
      mpls xc add ilm_label gen 27 ilm_labelspace 1 nhlfe_key 0x2

# LSP set up R-1 to R-2

      mpls nhlfe add key 0
      mpls nhlfe change key 0x3 instructions push gen 18 nexthop eth2
ipv4 10.0.2.2
      ip route add 10.0.2.2/32 via 10.0.2.2 spec_nh 0x8847 0x3

# Swap incoming label 56 to outgoing label 18

      mpls xc add ilm_label gen 56 ilm_labelspace 0 nhlfe_key 0x3

# LSP set up R-1 to R-3

      mpls nhlfe add key 0
      mpls nhlfe change key 0x4 instructions push gen 26 nexthop eth1
ipv4 10.0.3.2
      ip route add 10.0.3.2/32 via 10.0.3.2 spec_nh 0x8847 0x4
```

```
# Swap incoming label 36 to outgoing label 26

      mpls xc add ilm_label gen 36 ilm_labelspace 0 nhlfe_key 0x4
```