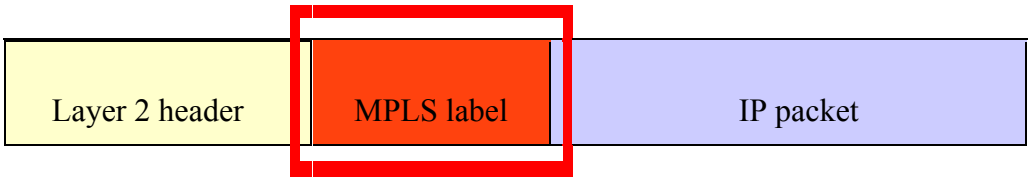
MPLS - Introduction (1/3)

http://wiki.netkit.org/netkit-labs/netkit-labs_advanced-topics/netkit-lab_mpls-forwarding/netkit-lab_mpls-forwarding.pdf

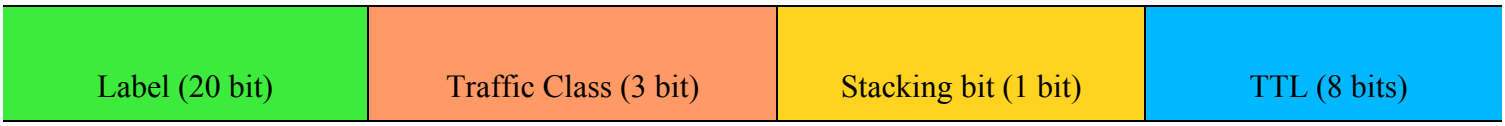
- MPLS : Multi Protocol Label Switching
- MPLS forwarde les paquets à l'intérieur du réseau en utilisant un mécanisme de "label switching"
- Ce mécanisme fonctionne comme suit :
  - Le trafic qui est supposé être forwardé à l'intérieur d'un réseau MPLS est tout d'abord *classifié* (par exemple, tous les paquets qui ont la même adresse de destination)
  - Chaque paquet classifié se voit associer un *label* (« label binding ») et il est injecté dans le réseau MPLS.
  - Le forwarding à l'intérieur du réseau MPLS a lieu en changeant les labels (*label swapping*). Il n'y a donc plus un routage en fonction de l'adresse de destination, mais seulement du label
    - "swapping" = remplacer le label avec un autre (éventuellement différent) ; le "scope" d'un label est local : un seul lien (« single link »)
  - Le label est finalement retiré à la sortie du réseau MPLS.

MPLS - Introduction (2/3)

Après insertion du label, le paquet a la structure suivante :



… et le « *MPLS label* » consiste des champs suivants :



- Label : la valeur du label
- Traffic Class : utilisé pour différentier la priorité entre différents types de trafics

- Stacking bit : un paquet peut être associé à une pile (*stack*) de labels, et non pas à un seul label ; ce bit marque la fin de la pile (du *stack*)
- TTL: Time To Live

MPLS - Introduction (3/3)

Pour effectuer le forwarding des paquets, les routeurs MPLS utilisent les tableaux suivants :

- NHLFE (Next Hop Label Forwarding Entry) : il contient les informations concernant comment forwarder un paquet MPLS, c'est à dire :
    - L'adresse IP actuelle du "next hop" pour le paquet
    - Les opérations (push/pop) qui doivent être effectuées sur le stack
- ILM (Incoming Label Map) : contient le *mapping* entre les paquets qui arrivent ("incoming") et un NHLFE : donc, ce tableau dit comment forwader un « labelled packet »
- XC (Cross Connect) : contient le mapping entre une ligne dans le ILM et le NHLFE ; il dit donc au routeur MPLS comment effectuer le « label swapping »

Laboratoire MPLS

Terminologie/acronymes utilisés en MPLS :

- LER (Label Edge Router) : les routeurs à l'entrée et à la sortie d'un réseau MPLS. Ils ajoutent (opération « *push* ») les labels aux paquets qui entrent dans les réseau MPLS et enlèvent (opération « *pop* ») les labels aux paquets qui sortent du réseau.
- LSR (Label Switching Router) : réalise le routage à l'intérieur du réseau MPLS, en se basant exclusivement sur le *label swapping*.
- FEC (Forwarding Equivalence Class) : un ensemble de paquets IP qui sont forwardés (traités) tous de la même façon.

Topologie du réseau

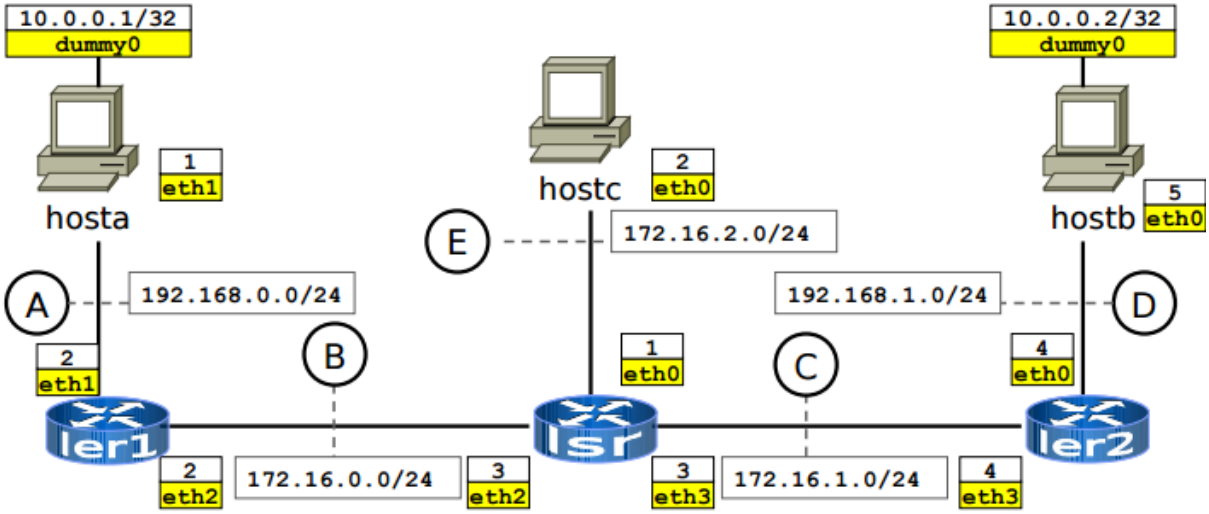
Nous allons utiliser le laboratoire disponible ici :

http://wiki.netkit.org/netkit-labs/netkit-labs_advanced-topics/netkit-lab_mpls-forwarding/netkit-lab_mpls-forwarding.tar.gz

Sur la machine réelle utiliser les commandes :

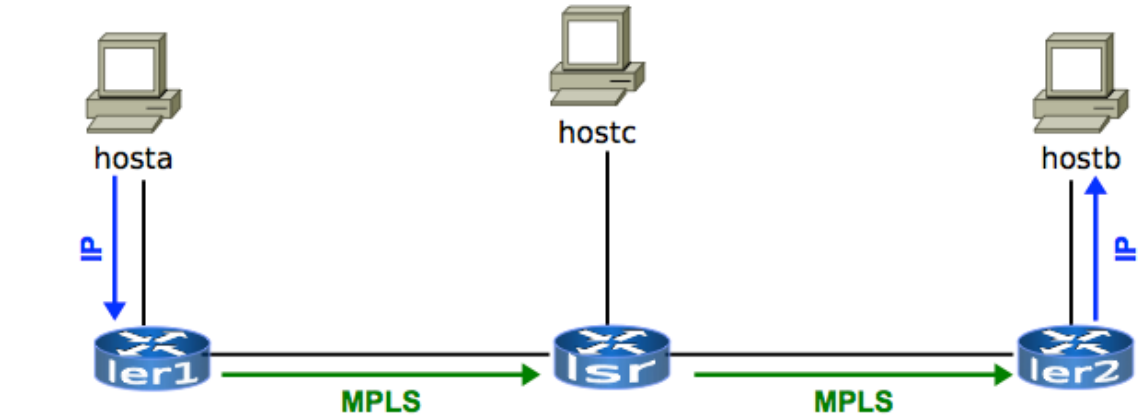
cd netkit-lab_mpls

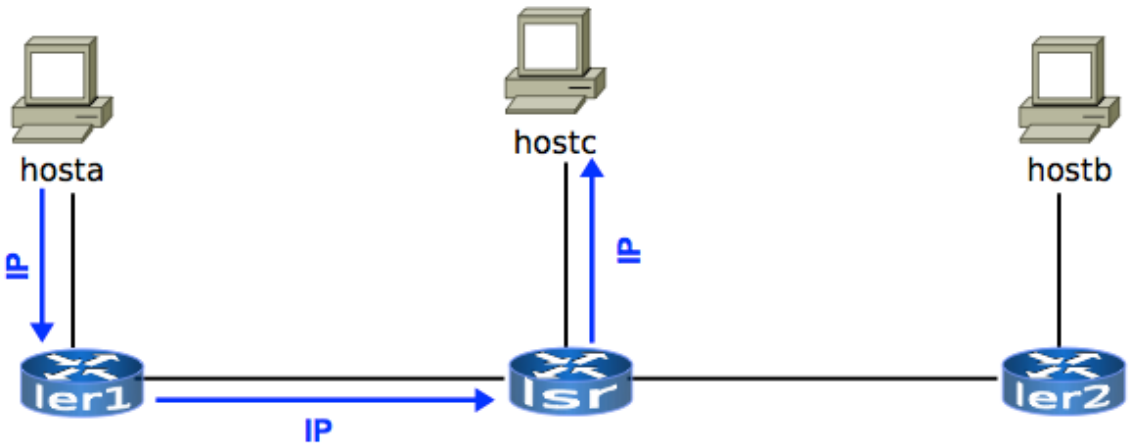
lstart

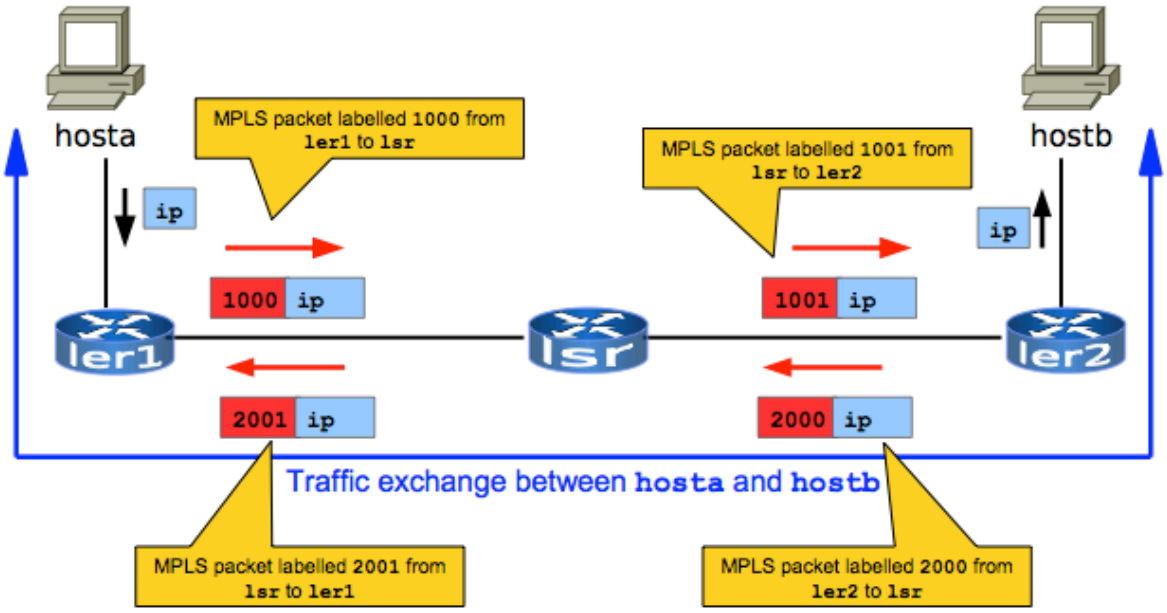


- **hosta** est connecté par un réseau MPLS (**ler1** – **lsr** – **ler2**) à **hostb**.
- **ler1**, **lsr**, et **ler2** doivent être configurés pour permettre l'échange de paquets entre **hosta** et **hostb** en utilisant le protocole MPLS.
- En même temps, **hosta** est connecté à **hostc** via le protocole IP, exclusivement. Pour cela, il faut ajouter des routes IP à **hosta**, **hostc**, **ler1**, et **lsr** pour permettre de supporter l'échange de trafic IP conventionnel.

Types de trafics







Configuration de base (au "*startup*")

hosta.startup

ifconfig eth1 192.168.0.1 netmask 255.255.255.0 broadcast 192.168.0.255 up ifconfig dummy0 10.0.0.1/32
ip route add 10.0.0.2/32 via 192.168.0.2 src 10.0.0.1

#traffic to hostc

route add -net 172.16.2.0 netmask 255.255.255.0 gw 192.168.0.2 dev eth1

hostb.startup

ifconfig eth0 192.168.1.5 netmask 255.255.255.0 broadcast 192.168.1.255 up ifconfig dummy0 10.0.0.2/32
ip route add 10.0.0.1/32 via 192.168.1.4 src 10.0.0.2

hostc.startup

ifconfig eth0 172.16.2.2 netmask 255.255.255.0 broadcast 172.16.2.255 up

#traffic to hosta

route add default gw 172.16.2.1

Configuration MPLS : **ler1**

```
modprobe mpls4
modprobe mplsbr
modprobe mpls_tunnel

#************ MPLS TRAFFIC FROM HOSTA TO HOSTB  ****************

mpls nhlfe add key 0 instructions \
        push gen 1000 nexthop eth2 ipv4 172.16.0.3

ip route add  .0.0.2/32 via 172.16.0.3 mpls 0x2
```

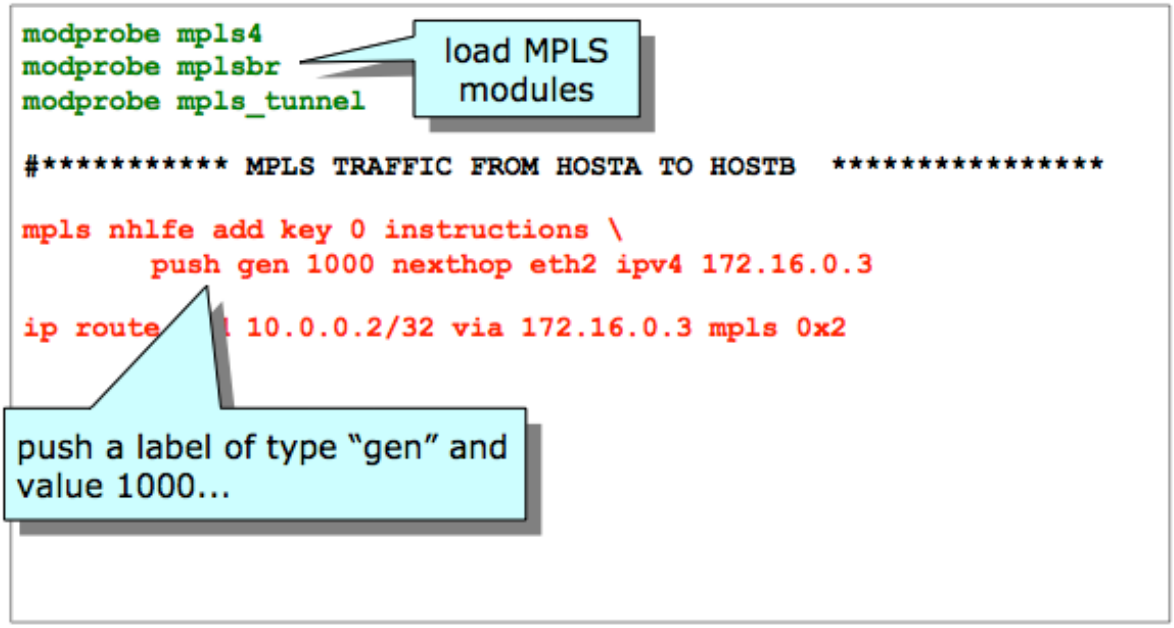
load MPLS modules

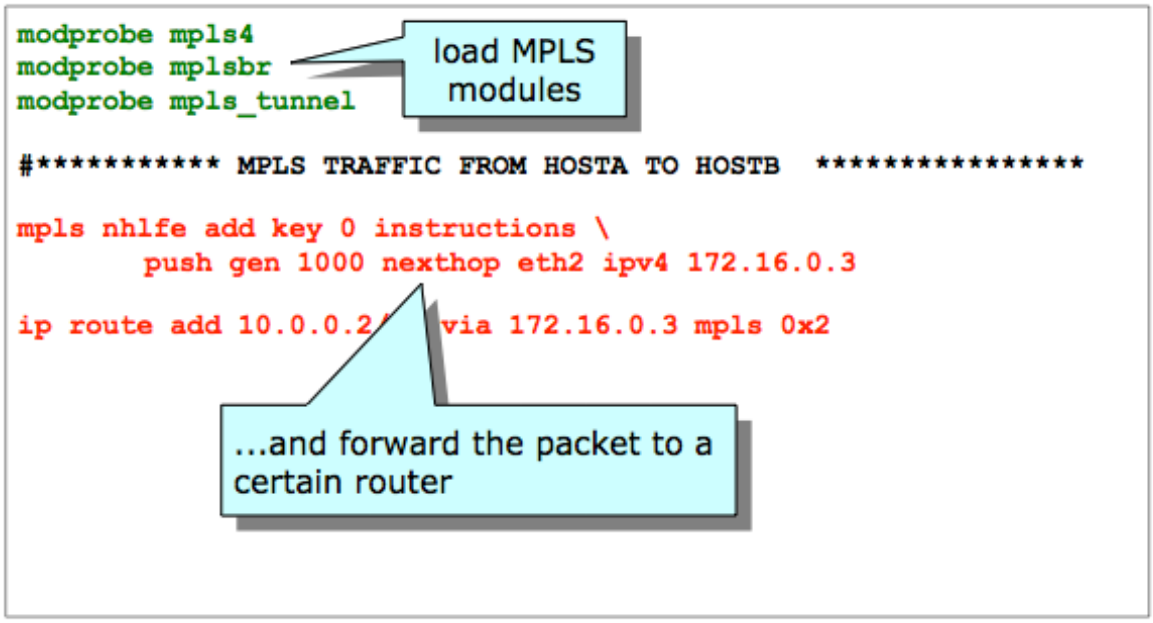
new nhlfe entry

```
modprobe mpls4
modprobe mplsbr
modprobe mpls_tunnel

#************ MPLS TRAFFIC FROM HOSTA TO HOSTB  ****************

mpls nhlfe add key 0 instructions \
        push ge 1000 nexthop eth2 ipv4 172.16.0.3

ip route add    0.0.2/32 via 172.16.0.3 mpls 0x2
```

load MPLS modules

sequential number identifying the entry (0="new entry": a number will be automatically assigned)





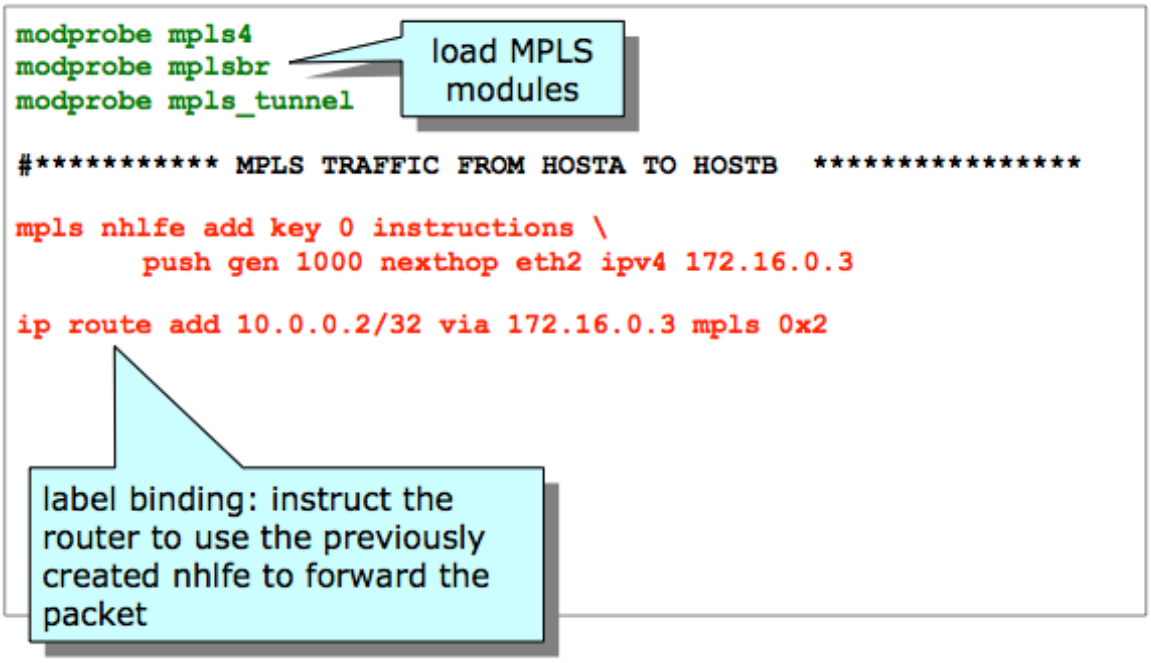
En fait, il semble proche du routage IP traditionnel.

Explications dans la suite.

Configuration MPLS : **ler1**

- Les machines à l'extérieur du réseau MPLS...
  - forwardent les paquets en utilisant l'adresse IP
  - connaissent seulement les chemins vers les "ingress point" du réseau MPLS
- Les routeurs à l'intérieur du réseau MPLS...
  - forwardent les paquets en utilisant la technique de "label swapping" vu auparavant
  - connaissent seulement comment rejoindre les "edge routers" du réseau MPLS...
    - ... et ils obtiennent cette connaissance en lisant l'information contenue dans les tableau de routage IP construites par les protocoles IGP sous-jacents (tells que RIP, OSPF, etc.)
  - dans notre cas, nous utilisons exclusivement des routes statiques
- Cela permet une séparation complète du routage à l'intérieur et à l'extérieur du réseau

Configuration MPLS: **ler1**



```
modprobe mpls4
modprobe mplsbr
modprobe mpls_tunnel

#************ MPLS TRAFFIC FROM HOSTA TO HOSTB  ****************

mpls nhlfe add key 0 instructions \
        push gen 1000 nexthop eth2 ipv4 172.16.0.3

ip route add 10.0.0.2/32 via 172.16.0.3 mpls 0x2
```

load MPLS modules

this is the key returned by the previous **mpls nhlfe add** command

note: here we are defining a fec

```
#************* MPLS TRAFFIC FROM HOSTB TO HOSTA ******************

mpls labelspace set dev eth2 labelspace 0

mpls ilm add label gen 2001 labelspace 0

mpls nhlfe add key 0 instructions nexthop eth1 ipv4 192.168.0.1

mpls xc add ilm_label gen 2001 ilm_labelspace 0 nhlfe_key 0x3
```

enable eth2 to receive mpls traffic

```
#************* MPLS TRAFFIC FROM HOSTB TO HOSTA ******************
mpls labelspace set dev eth2 labelspace 0
mpls ilm add label gen 2001 labelspace 0
mpls nhlfe add key 0 instructions nexthop eth1
mpls xc add ilm_label gen 2001 ilm_labelspace 0
```

put label 2001 in the ilm

required in order to recognize the incoming label and be able to put an entry in the xc list later on

```
#************* MPLS TRAFFIC FROM HOSTB TO HOSTA ******************
mpls labelspace set dev eth2 labelspace 0
mpls ilm add label gen 2001 labelspace 0
mpls nhlfe add key 0 instructions nexthop eth1 ipv4 192.168.0.1
mpls xc add ilm_label gen 2001 ilm_labelspace 0 nhlfe_key 0x3
```

a nhlfe entry that tells where the packet is to be forwarded

```
#************* MPLS TRAFFIC FROM HOSTB TO HOSTA ******************
mpls labelspace set dev eth2 labelspace 0
mpls ilm add label gen 2001 labelspace 0
mpls nhlfe add key 0 instructions nexthop eth1 ipv4 192.168.0.1
mpls xc add ilm_label gen 2001 ilm_labelspace 0 nhlfe_key 0x3
```

perform label "swapping":
upon receiving a packet with label 2001 we execute the nhlfe indexed by key 0x3 (returned by the last **`mpls nhlfe add`** command)

note: this instruction "consumes" (=pops) label 2001, which is all we need to do because packets directed to **`hosta`** are exiting the MPLS network

Configuration MPLS : **ler2**

Très similaire à celle du **ler1**

Configuration MPLS : **lsr**

```
modprobe mpls4
modprobe mplsbr
modprobe mpls_tunnel

#********** MPLS TRAFFIC FROM HOSTA TO HOSTB

mpls labelspace set dev eth2 labelspace 0
mpls ilm add label gen 1000 labelspace 0
mpls nhlfe add key 0 instructions \
       push gen 1001 nexthop eth3 ipv4 172.16.1.4
mpls xc add ilm_label gen 1000 ilm_labelspace 0 nhlfe_key 0x2


#********** MPLS TRAFFIC FROM HOSTB TO HOSTA

mpls labelspace set dev eth3 labelspace 0
mpls ilm add label gen 2000 labelspace 0
mpls nhlfe add key 0 instructions \
       push gen 2001 nexthop eth2 ipv4 172.16.0.2
mpls xc add ilm_label gen 2000 ilm_labelspace 0 nhlfe_key 0x3
```

swap incoming label 1000 with 1001 and forward on to **ler2**

swap incoming label 2000 with 2001 and forward on to **ler1**

```
route add -net 192.168.0.0 netmask 255.255.255.0 gw
172.16.0.2 dev eth2
```

static route used to allow IP test communications between **hosta** and **hostc**

Démarrer le Lab

Nous allons utiliser le laboratoire disponible ici :

http://wiki.netkit.org/netkit-labs/netkit-labs_advanced-topics/netkit-lab_mpls-forwarding/netkit-lab_mpls-forwarding.tar.gz

Sur la machine réelle utiliser les commandes :

cd netkit-lab_mpls

lstart

Commandes pour analyser le trafic MPLS

Machine virtuelle « hosta »

hosta:~# ping 10.0.0.2

Machine virtuelle « lsr »

lsr:~# tcpdump -i eth3

Machine virtuelle « ler2 »

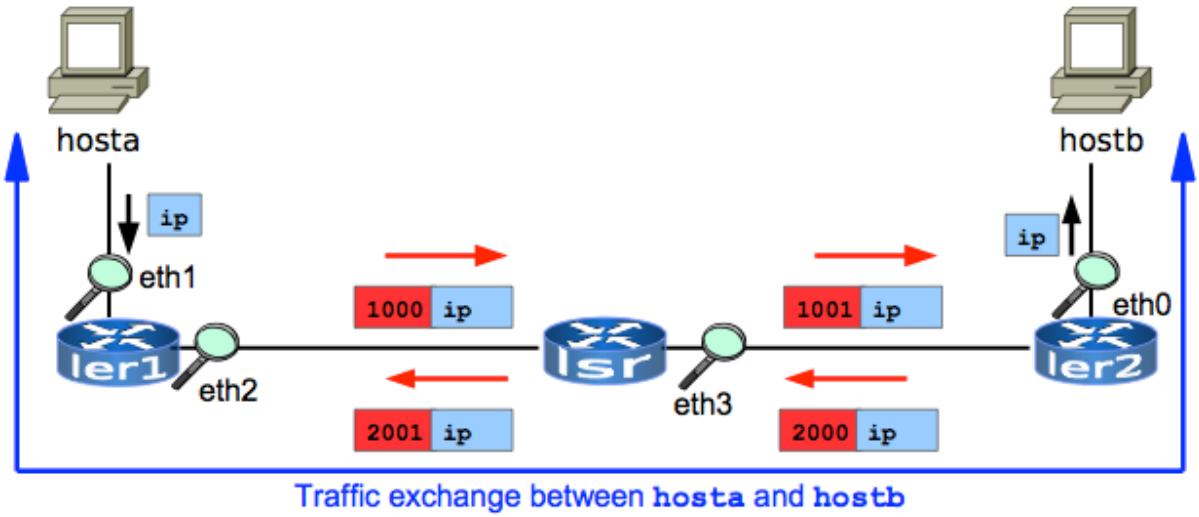
ler2:~# tcpdump -i eth0

Machine virtuelle « ler1 »

ler1:~# tcpdump -i eth1

ler1:~# tcpdump -i eth2

Analyser le trafic MPLS



Analyser le trafic MPLS

- Des « checkpoints » (points où l'on effectue les mesures) ont été identifiés au long du chemin depuis **hosta** vers **hostb**
    - **ler1** : ses deux interfaces, pour observer le trafic *avant* et *après* l'insertion de l'en-tete (header) MPLS
    - **lsr** : l'interface **eth3**, pour observer l'operation de « label switching »
    - **ler2** : l'interface **eth0**, où l'en-tête MPLS est enlevé du paquet
- Des checkpoints similaires peuvent etre bien evidemment consideres pour le trafic entre **hostb** et **hosta**

Analyser le trafic MPLS : *ler1*

La machine **ler1** applique le NHLFE (Next Hop Label Forwarding Entry) 0x2 (="push label **1000**") au

trafic destiné vers **10.0.0.2** et le forwarde vers **lsr**

Q1. Exécuter dans la machine virtuelle « *ler1* » les commandes suivantes et observer/analyser les résultats :

- ip route show

et ensuite

- mpls nhlfe show

Q2 : dans la machine virtuelle « *lsr* » observer le tableau de routage IP. Est-ce que *ler* connaît une route (IP) vers les destinations 10.0.0.0 ?

Q3 : toujours dans la machine virtuelle « *lsr* » observer le tableau de routage MPLS (utiliser dans la machine *lsr* les commandes « mpls nhlfe show », « mpls ilm show », « mpls xc show »). Est-ce que *lsr* sait comment gérer (forwarder) les paquets MPLS ?

Observer/analyser dans la machine *lsr*, en particulier,

- à l'aide des commandes « mpls ilm show » et « mpls xc show » comment les paquets qui entrent en *lsr* (les « incoming packets ») avec label 1000 son reconnus et leur label et géré par *lsr*.
- à l'aide de la commande « mpls nhlfe show » comment ces paquets sont forwardés vers *ler2* et avec quel nouveau label.

Q4 : nous allons exécuter les mêmes opérations décrites dans les questions précédentes (Q2-Q3), cette fois dans la machine virtuelle *ler2*.

En particulier, dans la machine virtuelle « ler2 » observer le tableau de routage IP. Est-ce que *ler2* connaît une route (IP) vers la destination 10.0.0.2 ?

Ensuite, toujours dans la machine virtuelle « *ler2* » observer le tableau de routage MPLS (utiliser dans la machine *ler2* les commandes « mpls nhlfe show », « mpls ilm show », « mpls xc show »). Est-ce que *ler2* sait comment gérer (forwarder) les paquets MPLS ?

Observer/analyser dans la machine *ler2*, en particulier,

- à l'aide des commandes « mpls ilm show » et « mpls xc show » comment les paquets qui entrent en *ler2* (les « incoming packets ») avec label 1001 son reconnus et leur label et géré par *ler2*.
- à l'aide de la commande « mpls nhlfe show » comment ces paquets sont forwardés vers *hostb*.

Analyser le trafic MPLS à l'aide de Wireshark (tcpdump)

Nous capturons les paquets en utilisant *tcpdump* et nous analysons les « dumps » en utilisant *wireshark* (dans la machine réelle).

On utilise pour le trafic un simple « ping » entre hosta (10.0.0.1) et hostb (10.0.0.2)

Exécuter les commandes suivantes :

Dans les machines virtuelles « ler1 », « lsr » et « ler2» . Par exemple, en lsr :

lsr:~# tcpdump -i ethX -w /hostlab/sniffXX.cap -s 1500

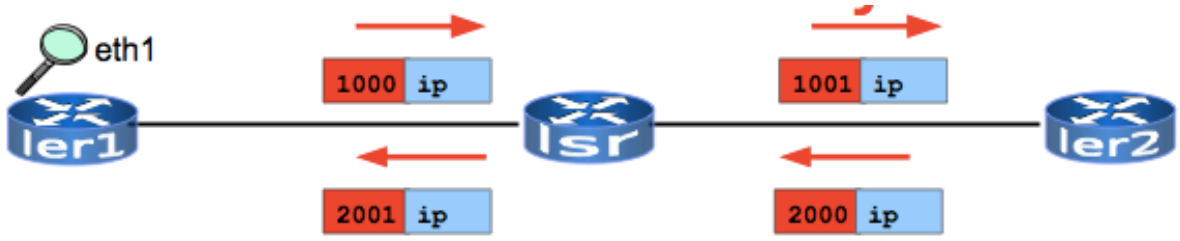
(substituer à « ethX » l'interface eth que l'on souhaite analyser)

Dans la machine réelle (host machine)

user@localhost:~$ cd netkit-lab_mpls

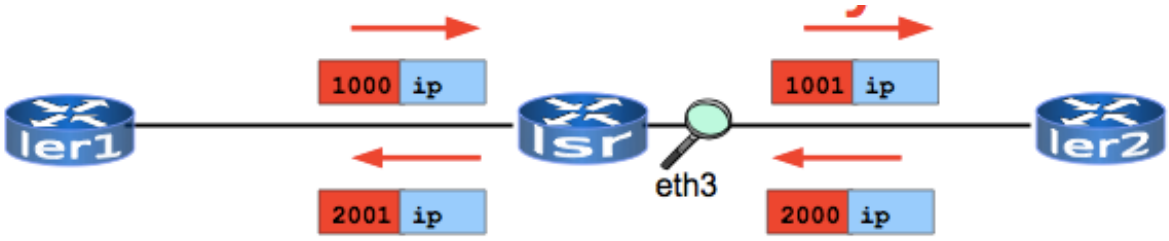
user@localhost:~/netkit-lab_mpls$ wireshark -r sniffXX.cap

Q5 : analyser le trafic sur l'interface eth1 de la machine *ler1* (comme indiqué dans la figure suivante). Indiquer en particulier quelle typologie de paquets (quel protocole, adresse de source/de destination l'on observe, etc …)



Q6 : analyser le trafic sur l'interface eth2 de la machine *ler1* (comme indiqué dans la figure suivante). Indiquer en particulier quelle typologie de paquets (quel protocole, adresse de source/de destination l'on observe, etc …). Observer aussi comment les paquets ICMP sont encapsulés avec une en-tete MPLS (quel est le numéro de label ?).

Q7 : analyser le trafic sur l'interface eth3 de la machine *lsr* (comme indiqué dans la figure suivante). Indiquer en particulier quelle typologie de paquets (quel protocole, adresse de source/de destination l'on observe, etc …). Observer aussi comment les paquets ICMP sont encapsulés avec une en-tete MPLS (quel est le nouveau numéro de label (dans les « echo request » et dans les « echo reply ») ?).



Q8 : analyser le trafic sur l'interface eth0 de la machine *ler2* (comme indiqué dans la figure suivante). Indiquer en particulier quelle typologie de paquets (quel protocole, adresse de source/de destination l'on observe, etc …)

Analyser du trafic IP (entre hosta et hostc)

Nous capturons les paquets en utilisant *tcpdump* et nous analysons les « dumps » en utilisant *wireshark* (dans la machine réelle).

On utilise pour le trafic un simple « ping » entre hosta (10.0.0.1) et hostc (172.16.2.2). Donc dans la machine virtuelle *hosta* on execute « ping 172.16.2.2 », tandis que l'on capture le trafic sur l'interface *eth0* de *hostc*.
Q9 : analyser le trafic sur l'interface eth0 de la machine *hostc*. Indiquer en particulier quelle typologie de paquets (quel protocole, adresse de source/de destination l'on observe, etc …)

Exercices proposés

Q10 : capturer et analyser le trafic sur toutes les interfaces des machines dans le réseau à l'aide de tcpdump/wireshark

Q11 : sur un routeur virtuel, essayez de donner les commandes que vous avez trouvez dans les configurations .startup "by hand", c'est à dire une après l'autre directement dans la machine virtuelle pour observer les messages (keys) que la machine va vous restituer

Q12 : essayez de changer les tableaux NHLFE dans le laboratoire tandis qu'il est en place, en ajoutant et en enlevant des entries pour voir le comportement du réseau (utiliser les commandes "**mpls -help**" à l'intérieur des machines virtuelles pour voir le help)