netkit lab

Traffic Engineering with MPLS for Linux

Version	1.1
Author(s)	F. Di Ciccio, F. Antonini (Kasko Networks S.r.l.) Reviewed by M. Rimondini (Roma Tre University)
E-mail	fra.dix87@gmail.com, fabio.antonini@kaskonetworks.it
Web	http://www.kaskonetworks.it/
Description	An example with 5 routers (2 lers, 3 lsrs) and 2 hosts to show link protection on a link and node protection on a router, with MPLS

copyright notice

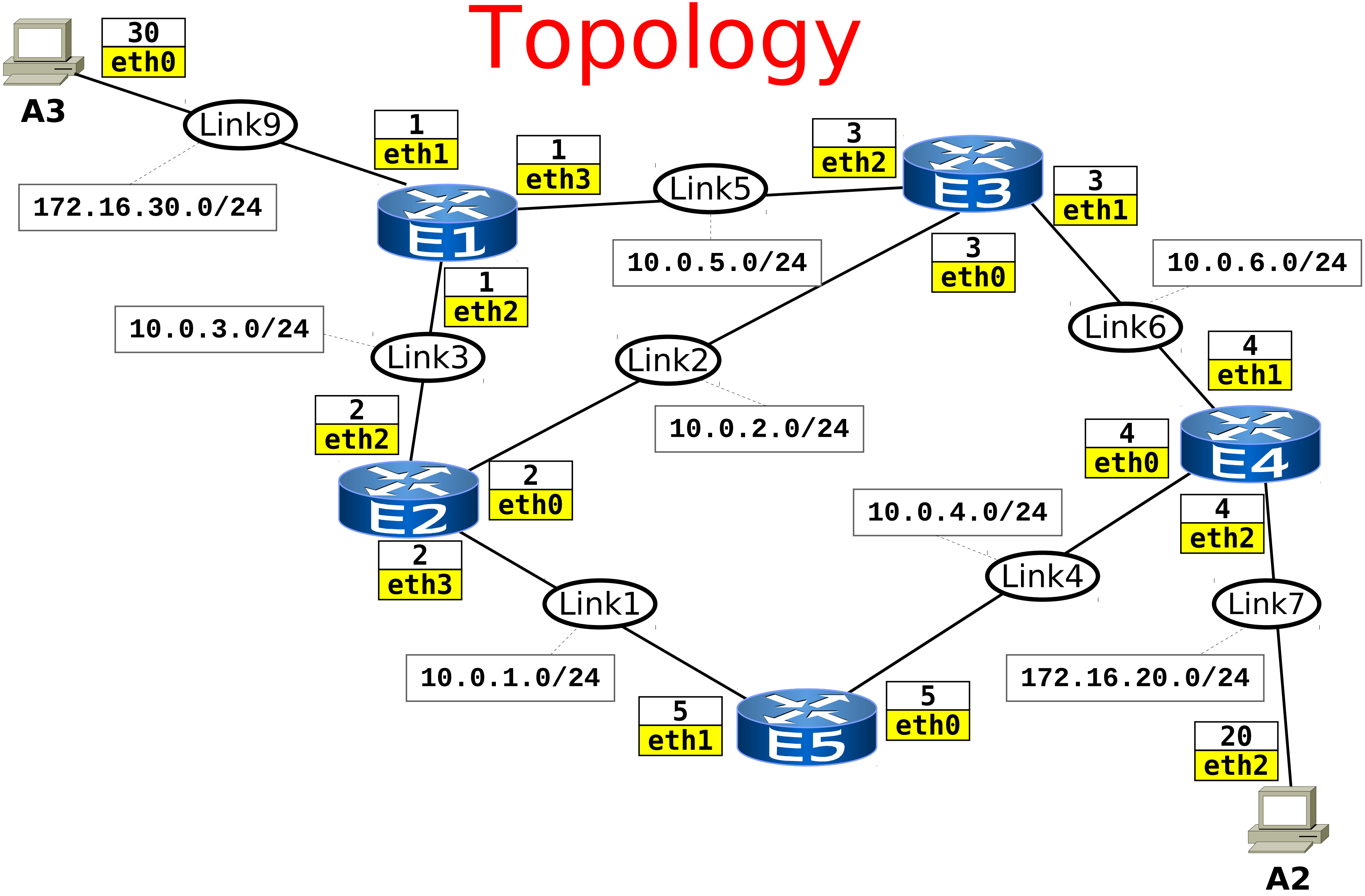
- All the pages/slides in this presentation, including but not limited to, images, photos, animations, videos, sounds, music, and text (hereby referred to as "material") are protected by copyright.
- This material, with the exception of some multimedia elements licensed by other organizations, is property of the authors and/or organizations appearing in the first slide.
- This material, or its parts, can be reproduced and used for didactical purposes within universities and schools, provided that this happens for non-profit purposes.
- Information contained in this material cannot be used within network design projects or other products of any kind.
- Any other use is prohibited, unless explicitly authorized by the authors on the basis of an explicit agreement.
- The authors assume no responsibility about this material and provide this material "as is", with no implicit or explicit warranty about the correctness and completeness of its contents, which may be subject to changes.
- This copyright notice must always be redistributed together with the material, or its portions.

The MPLS Lab

- This lab is (highly) inspired to the example “MPLS-Linux labs – 4.2 Node Protection” from [1]
- Goal: achieving fast & seamless rerouting in case of link/node failure using MPLS
- Approach: emulate fast rerouting by changing label swapping rules on the fly upon detecting link/node failures

[1] http://ontwerpen1.khlim.be/~lrutten/cursussen/comm2/mpls-linux-docs/4-2-node_protection.html

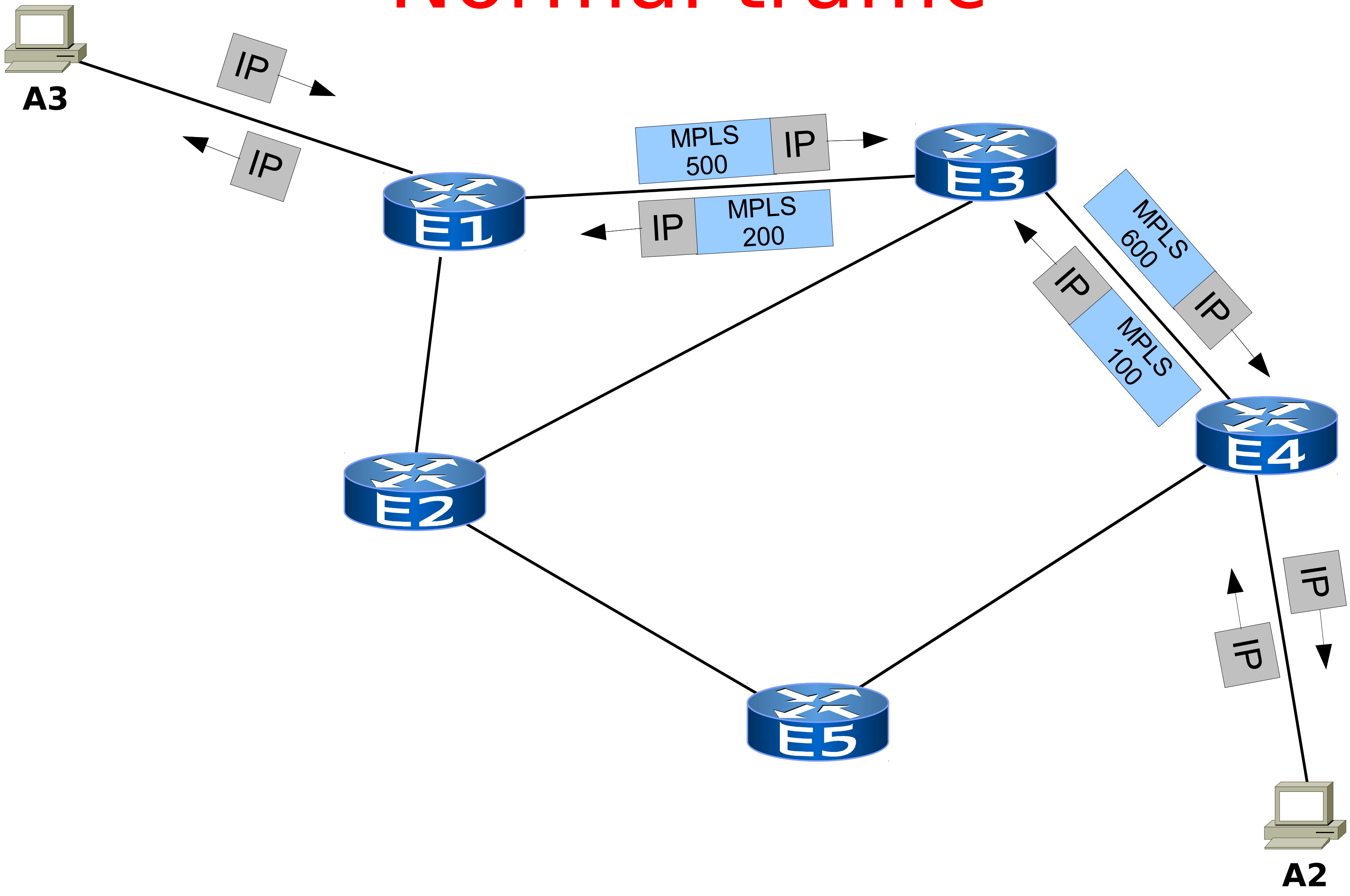
Topology


A3
30
eth0
Link9
172.16.30.0/24
1
eth1
1
eth3
E1
1
eth2
Link5
10.0.5.0/24
3
eth2
E3
3
eth1
3
eth0
10.0.6.0/24
Link6
10.0.3.0/24
Link3
Link2
4
eth1
2
eth2
10.0.2.0/24
4
eth0
E4
E2
2
eth0
10.0.4.0/24
4
eth2
2
eth3
Link1
Link4
Link7
10.0.1.0/24
172.16.20.0/24
5
eth1
E5
5
eth0
20
eth2
A2

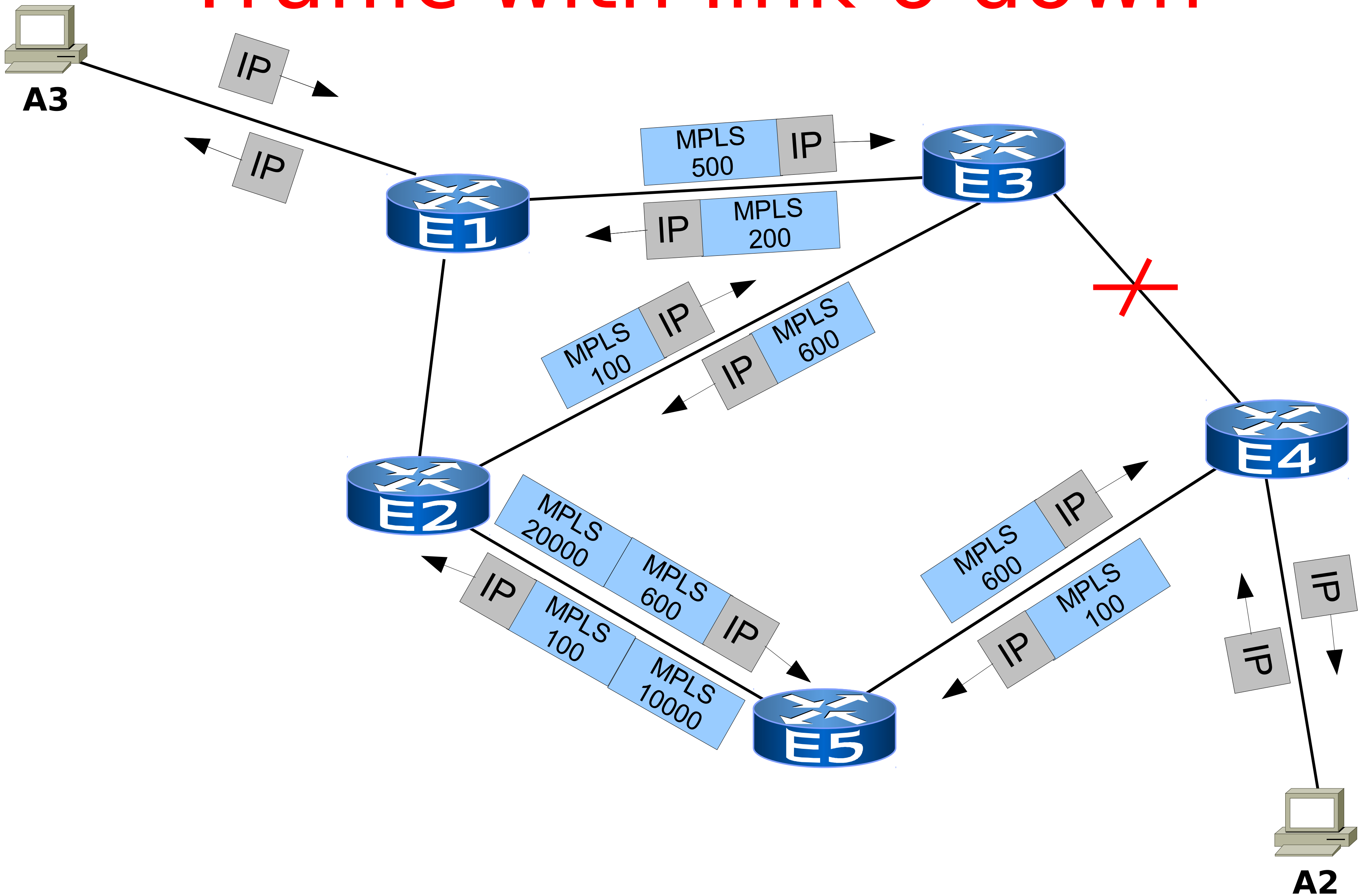
Network topology

- A3 and A2 are connected by an MPLS network (E1, E2, E3, E4, E5)
  - Note: there are no VPNs, i.e., we use a single label
- For the traffic from A2 to A3 routers are configured to choose the “best” route (E4, E3, E1)
- If link 6 fails, all traffic between A2 and A3 is switched to a backup path (E4, E5, E2, E3, E1)
- If node E3 fails, all traffic between A2 and A3 is switched to a backup route (E4, E5, E2, E1)

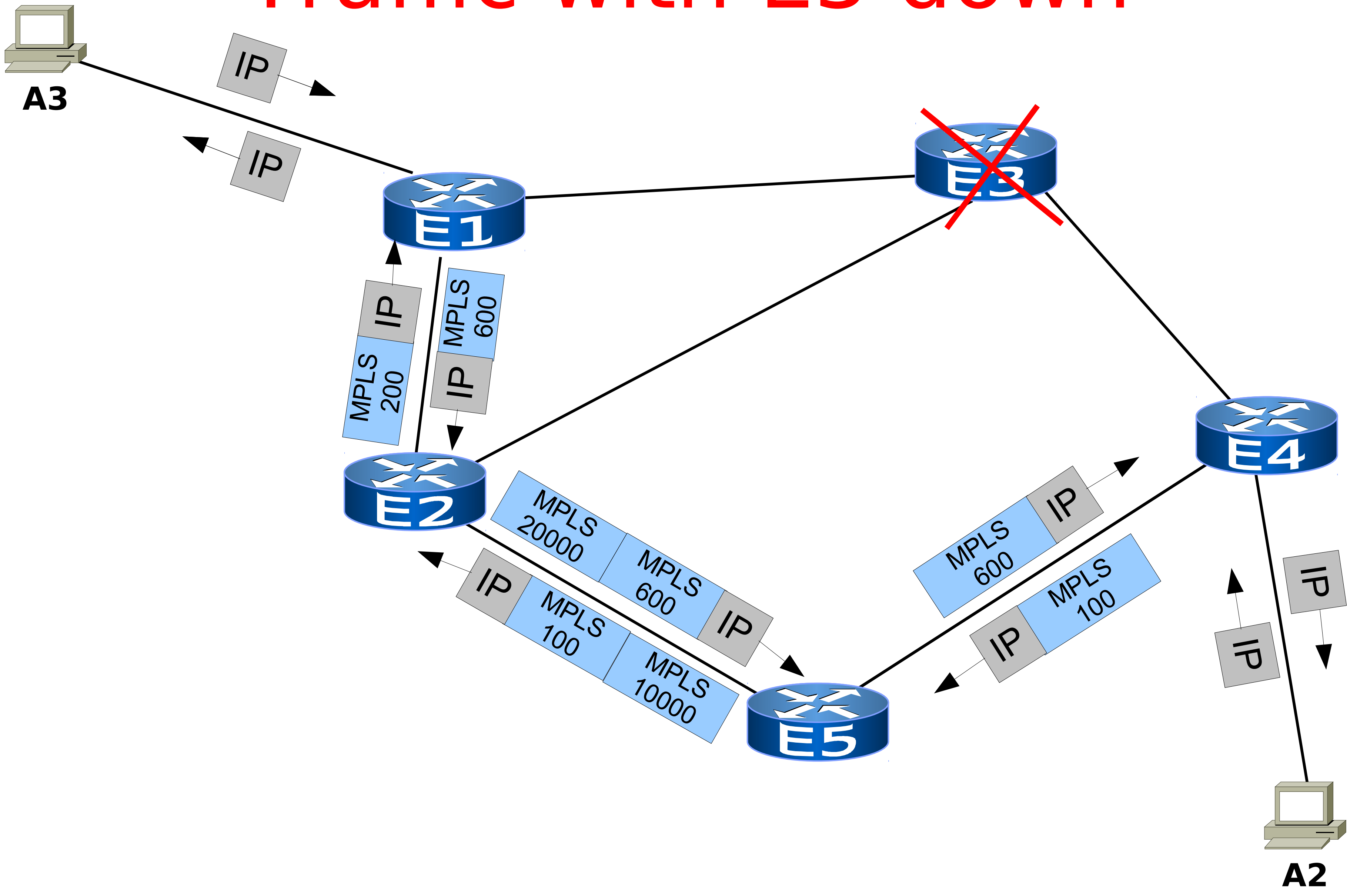
Normal traffic



Traffic with link 6 down



Traffic with E3 down



Basic configuration on end hosts

A2.startup

```
ifconfig eth2 down
ifconfig eth2 172.16.20.20 netmask 255.255.255.0 up
ip route add 172.16.10.0/24 via 172.16.20.4 dev eth2
ip route add 172.16.30.0/24 via 172.16.20.4 dev eth2
```

A3.startup

```
ifconfig eth0 down
ifconfig eth0 172.16.30.30 netmask 255.255.255.0 up
ip route add 172.16.20.0/24 dev eth0 via 172.16.30.1
ip route add 172.16.10.0/24 dev eth0 via 172.16.30.1
```

Router settings

- All routers load MPLS kernel modules at startup
- MPLS configuration on E5 is in E5.startup as usual
- MPLS configurations for E1-E4 are in external scripts found in shared/root/ (therefore /root inside virtual machines)
  - These scripts take care of redirecting traffic as soon as failures are detected
  - They periodically ping to check which routes are still available
    - best: via E4, E3, E1
    - medium: via E4, E5, E2, E3, E1
    - low: via E4, E5, E2, E1

MPLS configuration on ler E1

MPLS_on_E1.sh

```
#A2->A3 default route

mpls labelspace set dev eth3 labelspace 0

mpls ilm add label gen 200 labelspace 0
```

Enable eth3 to receive MPLS traffic

add an entry on ilm table in order to list (and pop) the incoming label (200)

```
#A2->A3 backup route

mpls labelspace set dev eth2 labelspace 0
```

MPLS configuration on ler E1

MPLS_on_E1.sh

```
#A3->A2 default route

var_best=`mpls nhlfe add key 0 instructions push gen 500 nexthop
eth3 ipv4 10.0.5.3|grep key|cut -c 17-26`

ip route add 17  6.20.0/24         5.3 mpls $var_best
```

New nhlfe entry

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

```
#A3->A2 backup route

var_low=`mpls nhlfe add key 0 instructions push gen 600 nexthop
eth2 ipv4 10.0.3.2|grep key|cut -c 17-26`
```

MPLS configuration on ler E1

MPLS_on_E1.sh

```
#A3->A2 default route

var_best=`mpls nhlfe add key 0 instructions push gen 500 nexthop
eth3 ipv4 10.0.5.3|grep key|cut -c 17-26`

ip route add 172.16.20.0/24 via 10.0 5 mpls $var_be
```

Push a label of type “gen” and value 500...

...and forward the packet to a certain router

```
#A3->A2 backup route

var_low=`mpls nhlfe add key 0 instructions push gen 600 nexthop
eth2 ipv4 10.0.3.2|grep key|cut -c 17-26`
```

MPLS configuration on ler E1

MPLS_on_E1.sh

```
#A3->A2 default route

var_best=`mpls nhlfe add key 0 instructions push gen 500 nexthop
eth3 ipv4 10.0.5.3|grep key|cut -c 17-26`

ip route add 172.16.20.0/24 via 10.0.5.3 mpls $var_best
```

label binding: instruct the router to use the previously created nhlfe to forward the packet

this is the key returned by the previous mpls nhlfe add command
note: here we are defining a fec

```
#A3->A2 backup route

var_low=`mpls nhlfe add key 0 instructi
eth2 ipv4 10.0.3.2|grep key|cut -c 17-26`
```

MPLS configuration on ler E1

MPLS_on_E1.sh

Loop

We are on the best/medium route, ping to see if E3 is still up

```
route="best"
while [ "1" = "1" ]
do
 case $route in
   best)
       status=`/usr/sbin/hping3 -c 1 --icmp 10.0.5.3 2>&1
               |grep "100% packet loss\|Host Unreachable"`
       if [ ! -z "$status" ] ;then
         # E3 is down switch to low route
         route="low";
         ip route del 172.16.20.0/24 via 10.0.5.3 mpls $var_best
         ip route add 172.16.20.0/24 via 10.0.3.2 mpls $var_low
       fi
     ;;
   low)
       status=`/usr/sbin/hping3 -c 1 --icmp 10.0.5.3 2>&1
               |grep "100% packet loss\|Host Unreachable"`
       if [  -z "$status" ] ;then
         # E3 is up again switch to best route
         route="best";
         ip route del 172.16.20.0/24 via 10.0.3.2 mpls $var_low
         ip route add 172.16.20.0/24 via 10.0.5.3 mpls $var_best
       fi
     ;;
esac
done
```

E3 is down: traffic directed to network 172.16.20.0/24 is switched from E3 to E2

MPLS configuration on ler E1

MPLS_on_E1.sh

```
route="best"
while [ "1" = "1" ]
do
 case $route in
   best)
       status=`/usr/sbin/hping3 -c 1 --icmp 10.0.5.3 2>&1
               |grep "100% packet loss\|Host Unreachable"`
       if [ ! -z "$status" ] ;then
         # E3 is down switch to low route
         route="low";
         ip route del 172.16.20.0/24 via 10.0.5.3 mpls $var_best
         ip route add 172.16.20.0/24 via 10.0.3.2 mpls $var_low
       fi
     ;;
   low)
       status=`/usr/sbin/hping3 -c 1 --icmp 10.0.5.3 2>&1
               |grep "100% packet loss\|Host Unreachable"`
       if [  -z "$status" ] ;then
         # E3 is up again switch to best route
         route="best";
         ip route del 172.16.20.0/24 via 10.0.3.2 mpls $var_low
         ip route add 172.16.20.0/24 via 10.0.5.3 mpls $var_best
       fi
     ;;
esac
done
```

we are on the low route, ping to see if E3 is up

E3 is back up: traffic directed to network 172.16.20.0/24 is switched from E2 to E3

MPLS configuration on ler E1

- Note: the script for E1 collapses the *best* and *medium* routes into one case, because both routes reach E1 via the same link

MPLS configuration on ler E4

- Similar to E1
- The loop inside the scripts distinguishes among *best*, *medium*, and *low* routes by checking the status of link 6 and of E3 using ping packets

MPLS configuration on lsr E5

```
#A2->A3
#if input label is 100, keep it and push 10000 on top of it.
mpls labelspace set dev eth0 labelspace 0
mpls ilm add label gen 100 labelspace 0

key1=`mpls nhlfe add key 0 instructions push gen 10000 nexthop
eth1 ipv4 10.0.1.2 |grep key | cut -c 17-26`

key2=`mpls nhlfe add key 0 instructions push gen 100 forward
$key1 |grep key | cut -c 17-26`



#This instruction exchange incoming label 100 with the label
#stack (100, 10000)
mpls xc add ilm_label gen 100 ilm_labelspace 0 nhlfe_key $key2
```

With this instruction label 1000 is set on top of label 100

exchange

Incoming label

MPLS configuration on lsr E5

```
#A3->A2
#if input label is 20000, pop it and keep the bottom label 600
mpls labelspace set dev eth1 labelspace 0
mpls ilm add label gen 20000 labelspace 0
mpls ilm add label gen 600 labelspace 0

key1=`mpls nhlfe add key 0 instructions push gen 600 nexthop
eth0 ipv4 10.0.4.4 |grep key | cut -c 17-26`

mpls xc add ilm_label gen 600 ilm_labelspace 0 nhlfe_key $key1
```

Exchange...

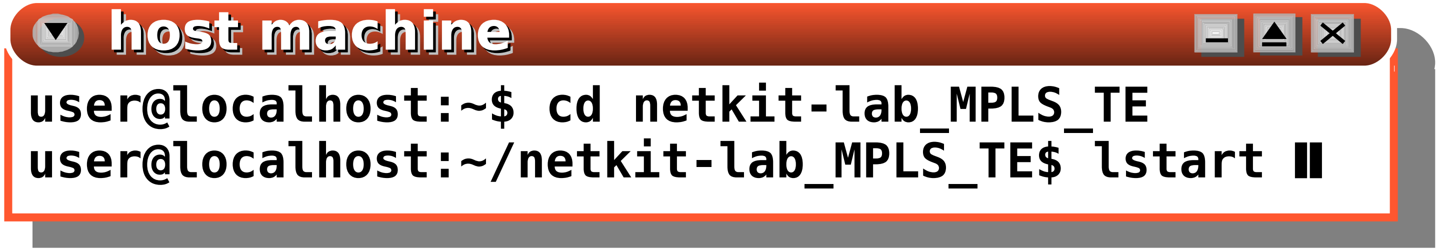
...incoming label 600 at the bottom of the stack (so it is removed from the stack)...

...with another label 600

MPLS configuration on other lsrs

- E2 and E3 are similar to E5. In addition to E5 they only have the loop instructions (very similar to the E1's):
  - E2 to check the status of E3
  - E3 to check the status of Link6
    - Side note: we simulate the fault of Link6 by bringing E3’s eth1 down. Therefore, upon restoring this interface we need to also redefine the nhlfe

Starting the lab

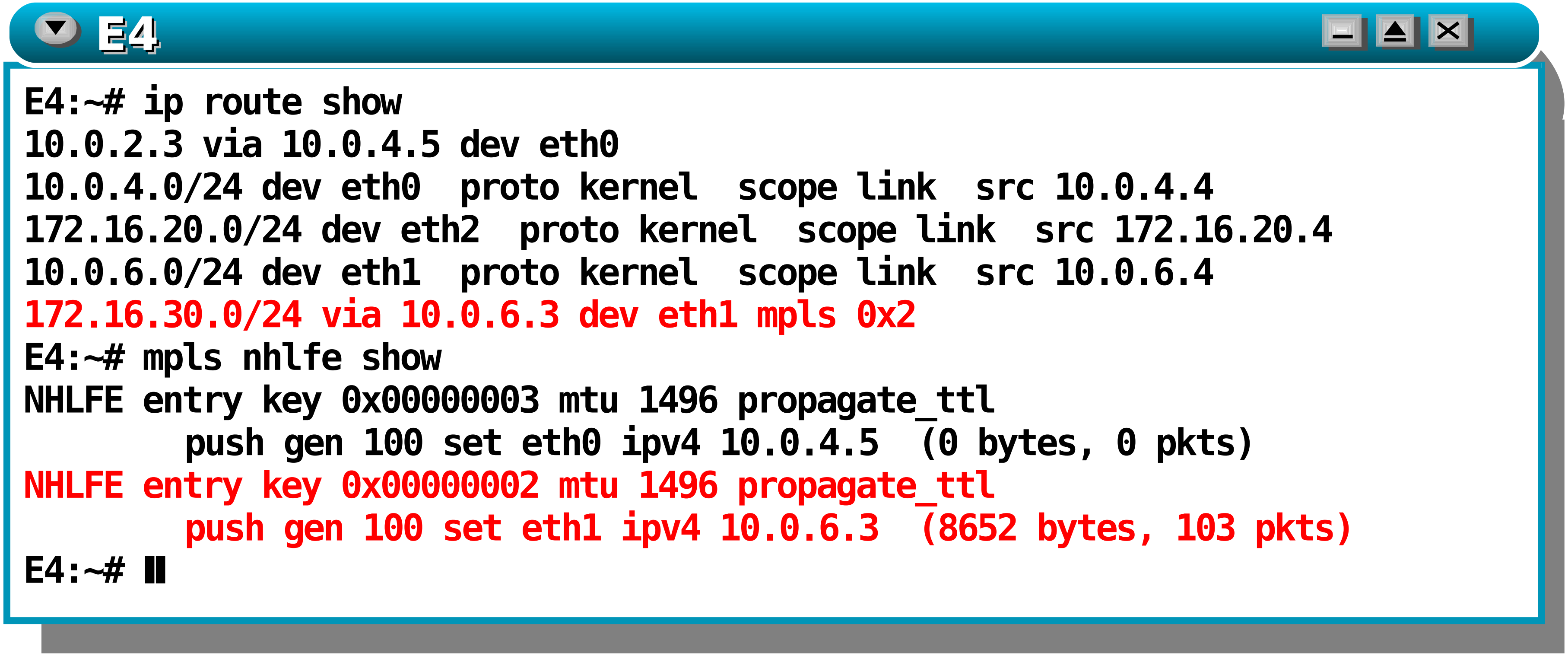


```
user@localhost:~$ cd netkit-lab_MPLS_TE
user@localhost:~/netkit-lab_MPLS_TE$ lstart
```

- Currently selected routes are shown at the top of terminal windows of relevant routers

MPLS traffic analisys

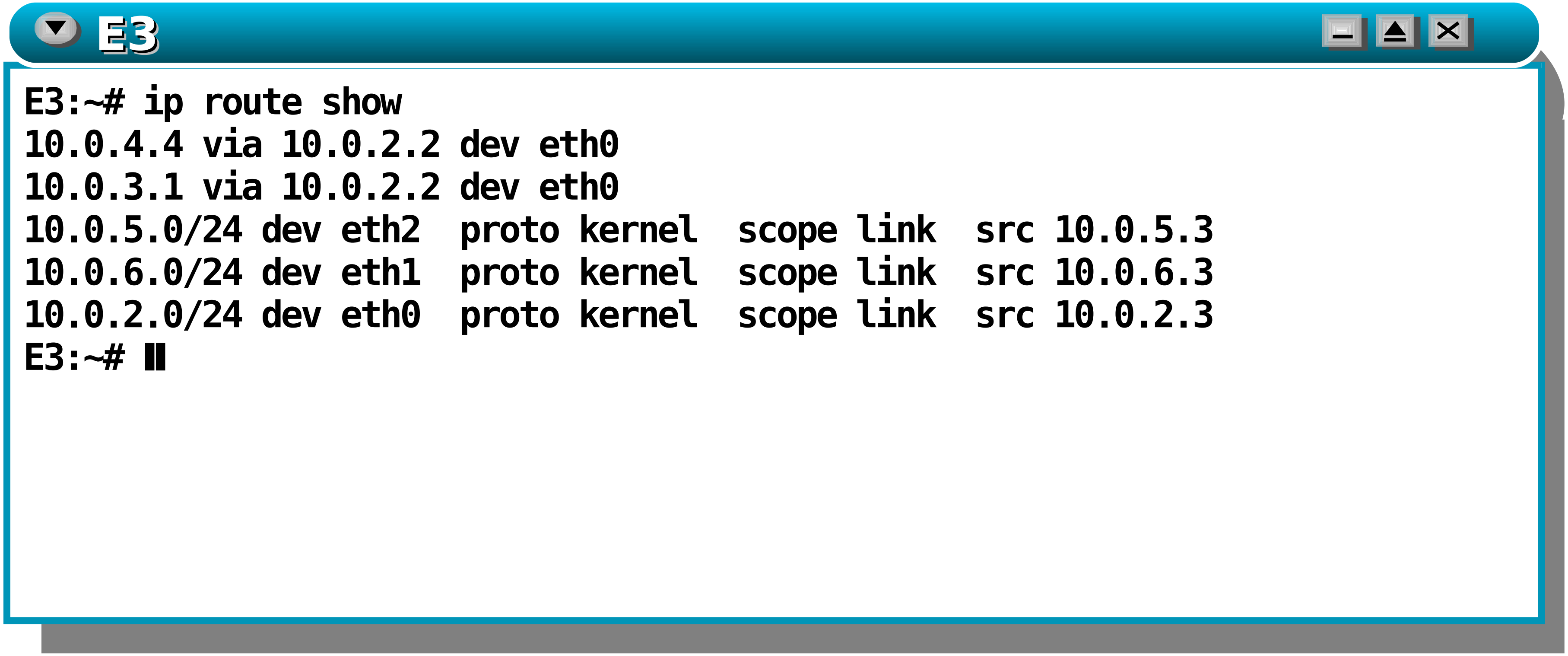
- ler E4 applies NHLFE 0x2 (=”push label 100”) to traffic directed to 172.16.30.0/24 and forwards it to E3



```
E4:~# ip route show
10.0.2.3 via 10.0.4.5 dev eth0
10.0.4.0/24 dev eth0  proto kernel  scope link  src 10.0.4.4
172.16.20.0/24 dev eth2  proto kernel  scope link  src 172.16.20.4
10.0.6.0/24 dev eth1  proto kernel  scope link  src 10.0.6.4
172.16.30.0/24 via 10.0.6.3 dev eth1 mpls 0x2
E4:~# mpls nhlfe show
NHLFE entry key 0x00000003 mtu 1496 propagate_ttl
        push gen 100 set eth0 ipv4 10.0.4.5  (0 bytes, 0 pkts)
NHLFE entry key 0x00000002 mtu 1496 propagate_ttl
        push gen 100 set eth1 ipv4 10.0.6.3  (8652 bytes, 103 pkts)
E4:~# ▮
```

MPLS traffic analisys

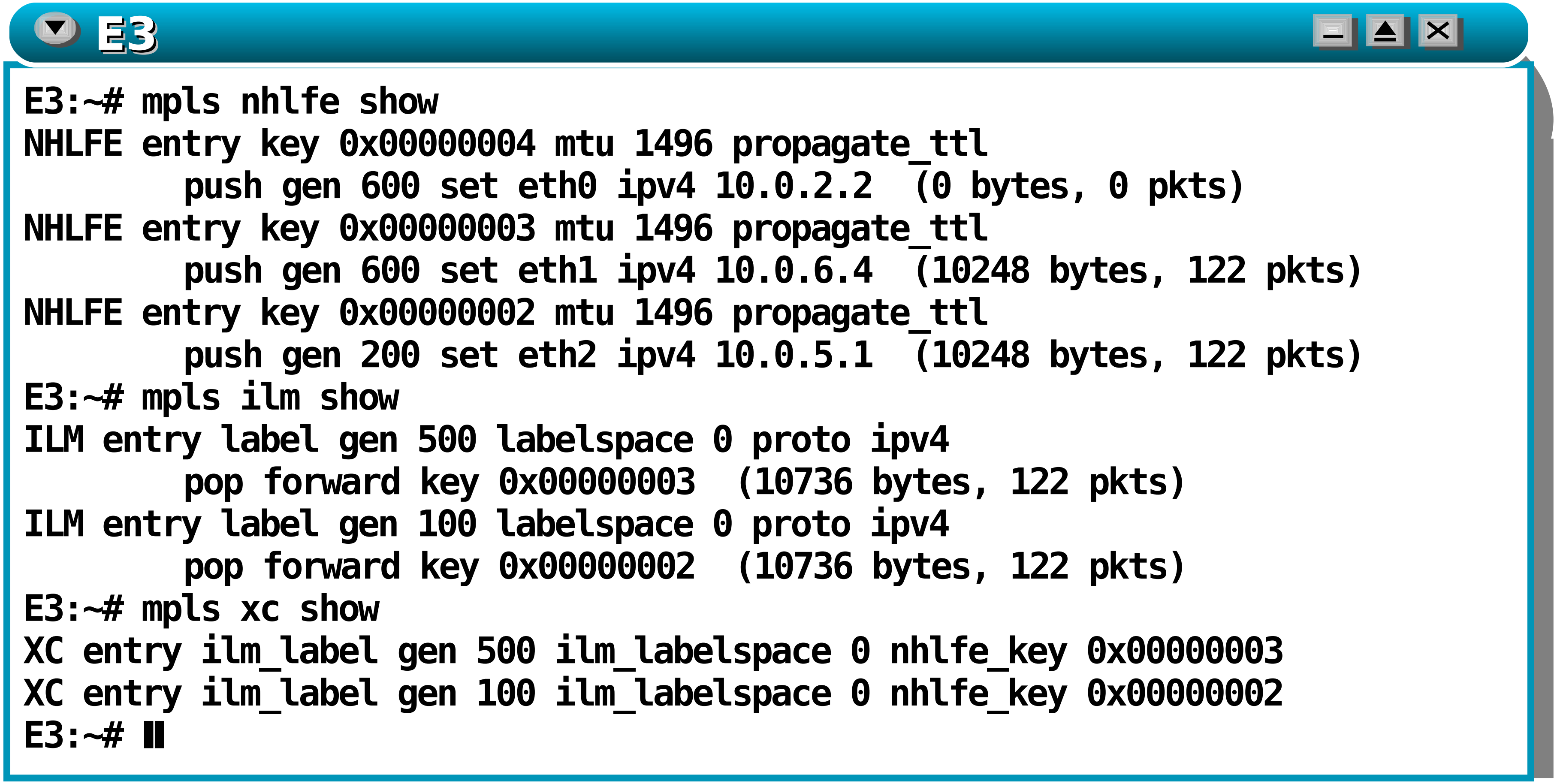
- lsr E3's IP (correctly) knows nothing about 172.16.30.0/24 and 172.16.20.0/24



```
E3:~# ip route show
10.0.4.4 via 10.0.2.2 dev eth0
10.0.3.1 via 10.0.2.2 dev eth0
10.0.5.0/24 dev eth2  proto kernel  scope link  src 10.0.5.3
10.0.6.0/24 dev eth1  proto kernel  scope link  src 10.0.6.3
10.0.2.0/24 dev eth0  proto kernel  scope link  src 10.0.2.3
E3:~# ▮
```

MPLS traffic analisys

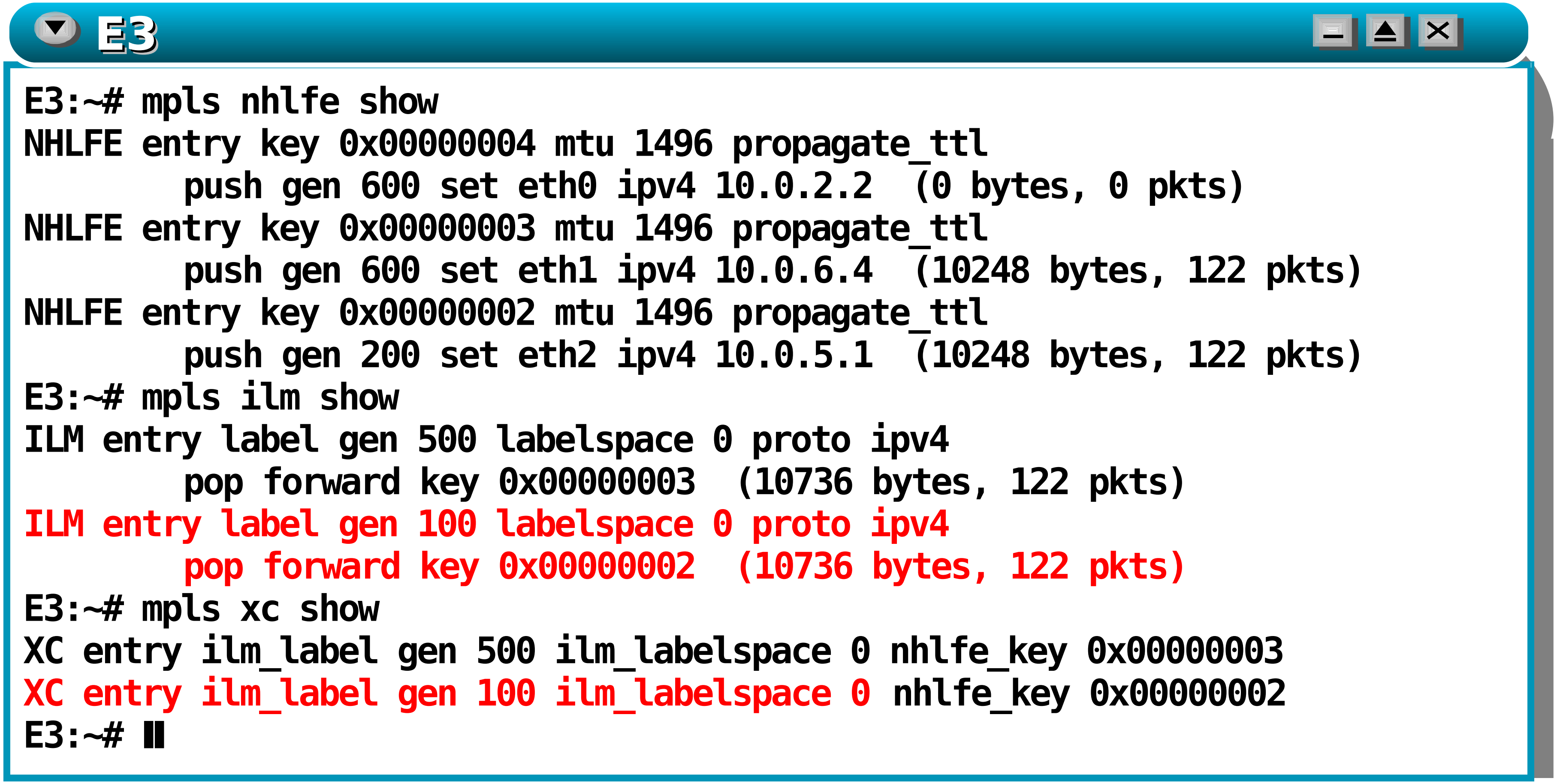
- lsr E3's MPLS knows how to forward labeled packets



```
E3:~# mpls nhlfe show
NHLFE entry key 0x00000004 mtu 1496 propagate_ttl
        push gen 600 set eth0 ipv4 10.0.2.2  (0 bytes, 0 pkts)
NHLFE entry key 0x00000003 mtu 1496 propagate_ttl
        push gen 600 set eth1 ipv4 10.0.6.4  (10248 bytes, 122 pkts)
NHLFE entry key 0x00000002 mtu 1496 propagate_ttl
        push gen 200 set eth2 ipv4 10.0.5.1  (10248 bytes, 122 pkts)
E3:~# mpls ilm show
ILM entry label gen 500 labelspace 0 proto ipv4
        pop forward key 0x00000003  (10736 bytes, 122 pkts)
ILM entry label gen 100 labelspace 0 proto ipv4
        pop forward key 0x00000002  (10736 bytes, 122 pkts)
E3:~# mpls xc show
XC entry ilm_label gen 500 ilm_labelspace 0 nhlfe_key 0x00000003
XC entry ilm_label gen 100 ilm_labelspace 0 nhlfe_key 0x00000002
E3:~#
```

MPLS traffic analisys

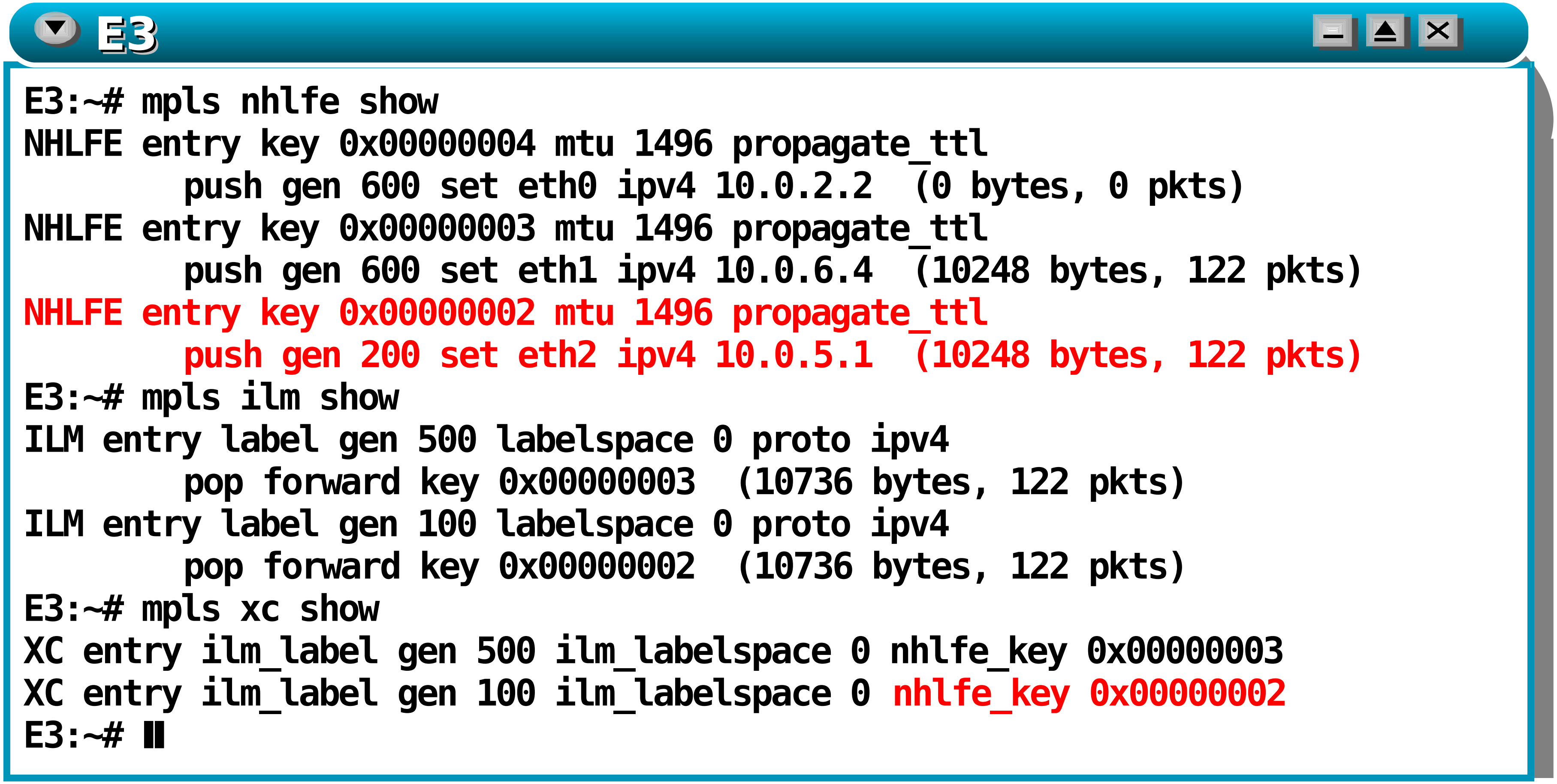
- incoming packets with label 100 have their label recognized and popped...



```
E3:~# mpls nhlfe show
NHLFE entry key 0x00000004 mtu 1496 propagate_ttl
        push gen 600 set eth0 ipv4 10.0.2.2  (0 bytes, 0 pkts)
NHLFE entry key 0x00000003 mtu 1496 propagate_ttl
        push gen 600 set eth1 ipv4 10.0.6.4  (10248 bytes, 122 pkts)
NHLFE entry key 0x00000002 mtu 1496 propagate_ttl
        push gen 200 set eth2 ipv4 10.0.5.1  (10248 bytes, 122 pkts)
E3:~# mpls ilm show
ILM entry label gen 500 labelspace 0 proto ipv4
        pop forward key 0x00000003  (10736 bytes, 122 pkts)
ILM entry label gen 100 labelspace 0 proto ipv4
        pop forward key 0x00000002  (10736 bytes, 122 pkts)
E3:~# mpls xc show
XC entry ilm_label gen 500 ilm_labelspace 0 nhlfe_key 0x00000003
XC entry ilm_label gen 100 ilm_labelspace 0 nhlfe_key 0x00000002
E3:~# ▮
```

MPLS traffic analisys

- ...and are forwarded to E1 after swapping the label with 200

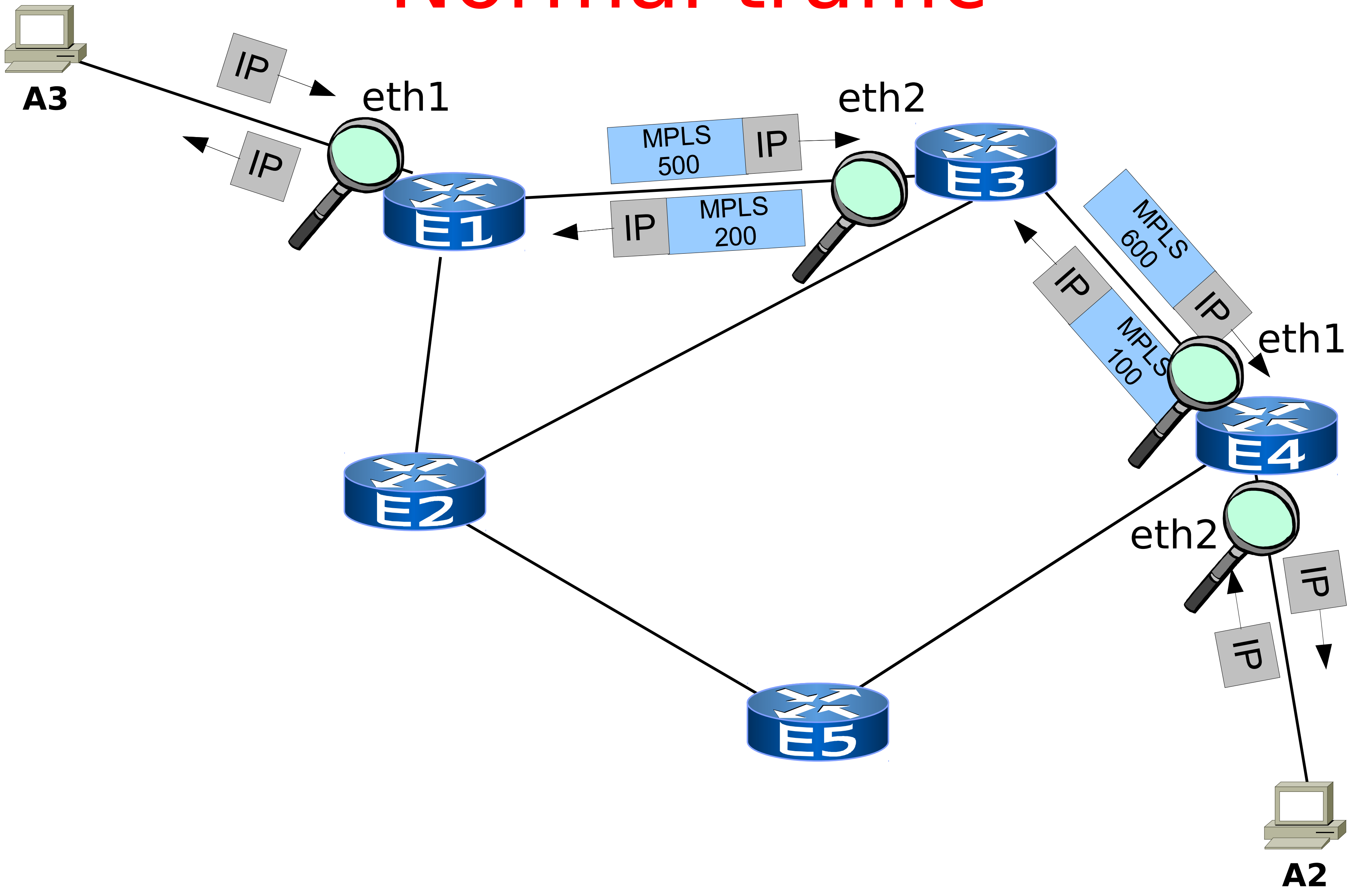


```
E3:~# mpls nhlfe show
NHLFE entry key 0x00000004 mtu 1496 propagate_ttl
        push gen 600 set eth0 ipv4 10.0.2.2  (0 bytes, 0 pkts)
NHLFE entry key 0x00000003 mtu 1496 propagate_ttl
        push gen 600 set eth1 ipv4 10.0.6.4  (10248 bytes, 122 pkts)
NHLFE entry key 0x00000002 mtu 1496 propagate_ttl
        push gen 200 set eth2 ipv4 10.0.5.1  (10248 bytes, 122 pkts)
E3:~# mpls ilm show
ILM entry label gen 500 labelspace 0 proto ipv4
        pop forward key 0x00000003  (10736 bytes, 122 pkts)
ILM entry label gen 100 labelspace 0 proto ipv4
        pop forward key 0x00000002  (10736 bytes, 122 pkts)
E3:~# mpls xc show
XC entry ilm_label gen 500 ilm_labelspace 0 nhlfe_key 0x00000003
XC entry ilm_label gen 100 ilm_labelspace 0 nhlfe_key 0x00000002
E3:~# ▮
```

MPLS traffic analisys

- We see how MPLS fills its tables and the integration between IP and MPLS
- You can try, for exercise, to see other ler's and lsr's routing tables to check their correctness

Normal traffic

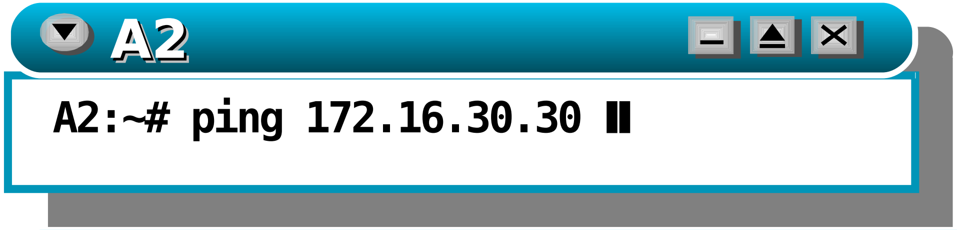


MPLS Traffic analisys

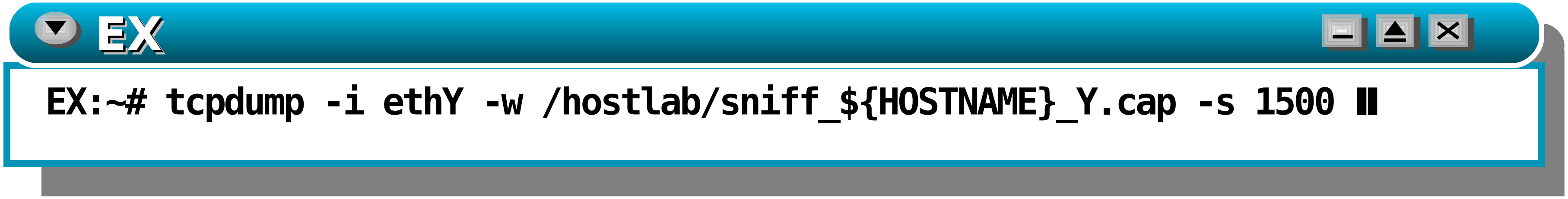
- Some checkpoints have been identified along the path from A2 to A3
  - E4: both interfaces, to observe traffic before and after the insertion of the MPLS header
  - E3: interface eth2 to observe label switching
  - E1: interface eth1, where the MPLS header is removed
- Similar checkpoints can be considered for the traffic from A3 to A2

MPLS Traffic analisys

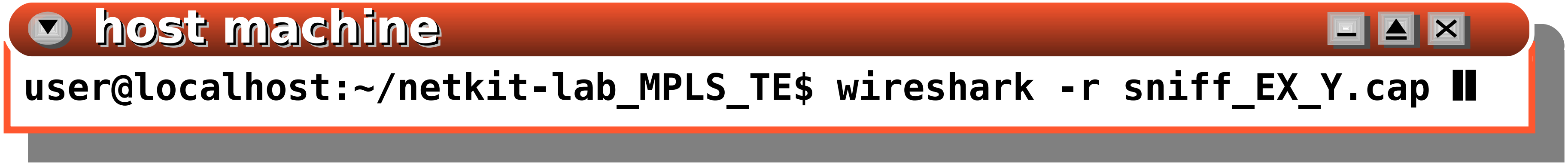
- We sniff packets using tcpdump and examine the dumps on the host using wireshark



```
A2:~# ping 172.16.30.30
```



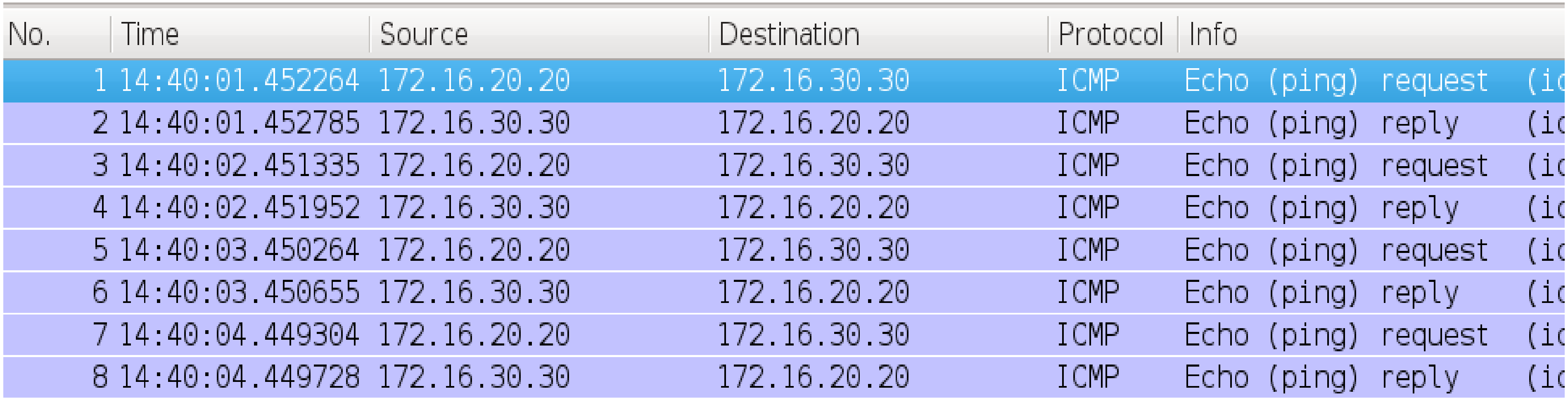
```
EX:~# tcpdump -i ethY -w /hostlab/sniff_${HOSTNAME}_Y.cap -s 1500
```



```
user@localhost:~/netkit-lab_MPLS_TE$ wireshark -r sniff_EX_Y.cap
```

MPLS Traffic analisys

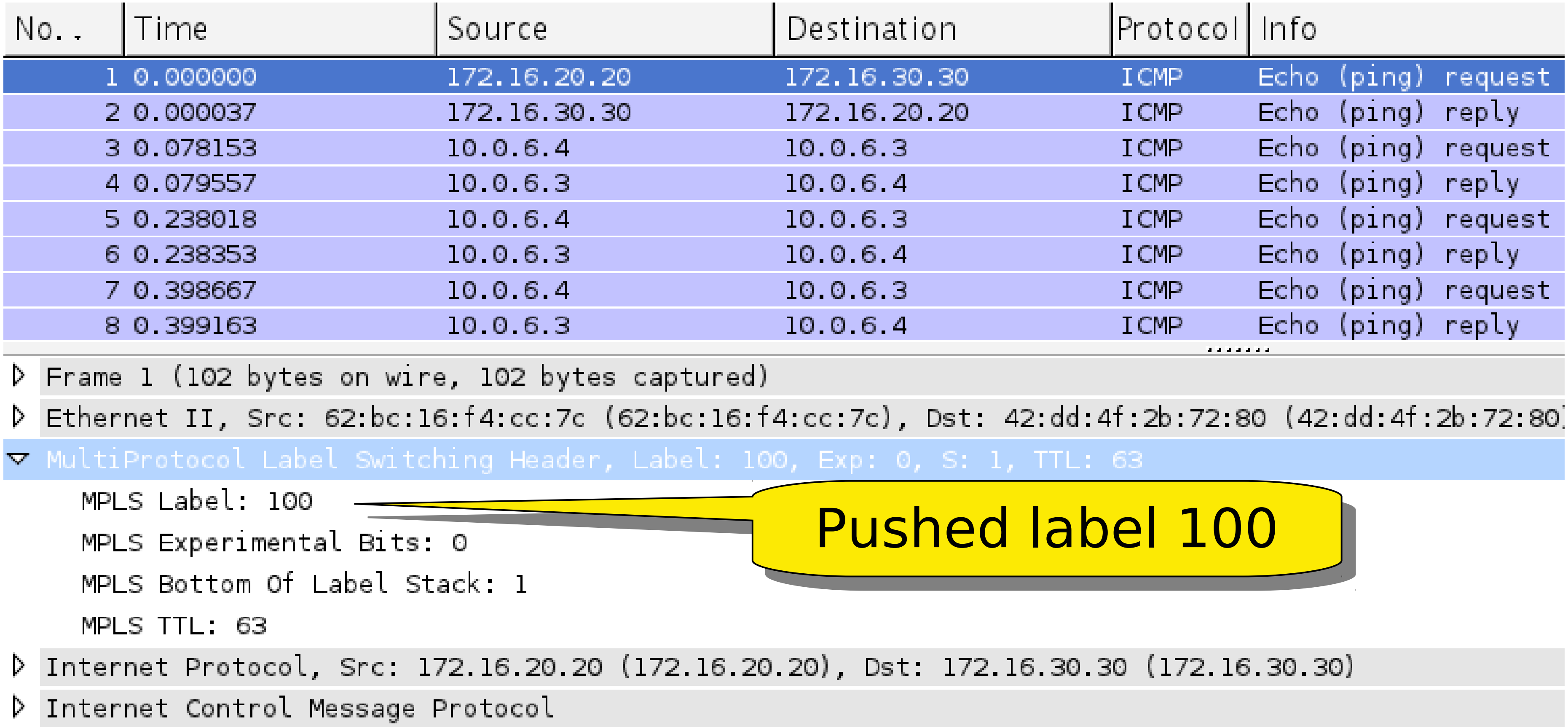
- Sniffing on E4’s eth2: plain ICMP packets



No.	Time	Source	Destination	Protocol	Info
1	14:40:01.452264	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request (ic
2	14:40:01.452785	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply (ic
3	14:40:02.451335	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request (ic
4	14:40:02.451952	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply (ic
5	14:40:03.450264	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request (ic
6	14:40:03.450655	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply (ic
7	14:40:04.449304	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request (ic
8	14:40:04.449728	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply (ic

MPLS Traffic analisys

- Sniffing on E4’s eth1: ICMP packets encapsulated in MPLS



No. .	Time	Source	Destination	Protocol	Info
1	0.000000	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
2	0.000037	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply
3	0.078153	10.0.6.4	10.0.6.3	ICMP	Echo (ping) request
4	0.079557	10.0.6.3	10.0.6.4	ICMP	Echo (ping) reply
5	0.238018	10.0.6.4	10.0.6.3	ICMP	Echo (ping) request
6	0.238353	10.0.6.3	10.0.6.4	ICMP	Echo (ping) reply
7	0.398667	10.0.6.4	10.0.6.3	ICMP	Echo (ping) request
8	0.399163	10.0.6.3	10.0.6.4	ICMP	Echo (ping) reply

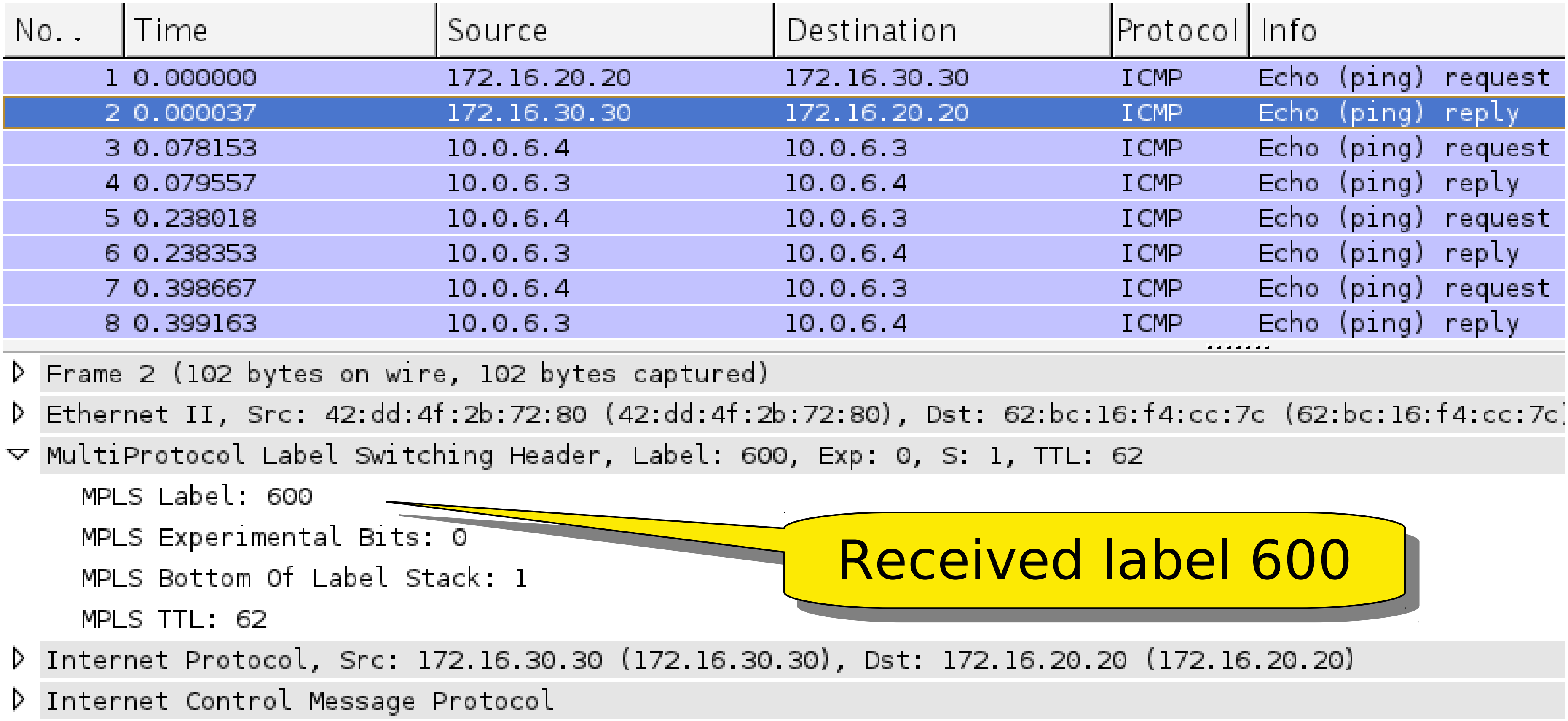
....

```
▷ Frame 1 (102 bytes on wire, 102 bytes captured)
▷ Ethernet II, Src: 62:bc:16:f4:cc:7c (62:bc:16:f4:cc:7c), Dst: 42:dd:4f:2b:72:80 (42:dd:4f:2b:72:80
▽ MultiProtocol Label Switching Header, Label: 100, Exp: 0, S: 1, TTL: 63
    MPLS Label: 100
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 1
    MPLS TTL: 63
▷ Internet Protocol, Src: 172.16.20.20 (172.16.20.20), Dst: 172.16.30.30 (172.16.30.30)
▷ Internet Control Message Protocol
```

Pushed label 100

MPLS Traffic analisys

- Sniffing on E4’s eth1: ICMP packets encapsulated in MPLS



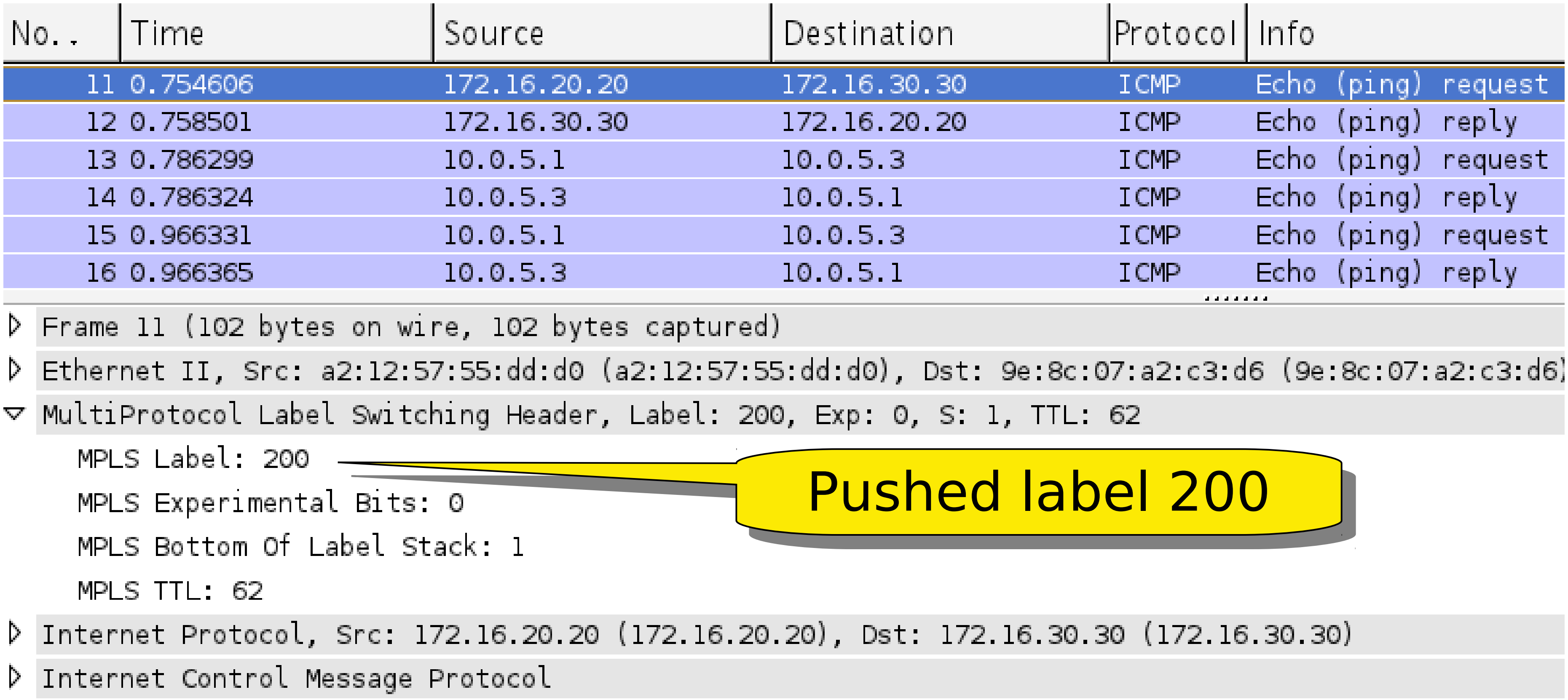
No..	Time	Source	Destination	Protocol	Info
1	0.000000	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
2	0.000037	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply
3	0.078153	10.0.6.4	10.0.6.3	ICMP	Echo (ping) request
4	0.079557	10.0.6.3	10.0.6.4	ICMP	Echo (ping) reply
5	0.238018	10.0.6.4	10.0.6.3	ICMP	Echo (ping) request
6	0.238353	10.0.6.3	10.0.6.4	ICMP	Echo (ping) reply
7	0.398667	10.0.6.4	10.0.6.3	ICMP	Echo (ping) request
8	0.399163	10.0.6.3	10.0.6.4	ICMP	Echo (ping) reply

```
▷ Frame 2 (102 bytes on wire, 102 bytes captured)
▷ Ethernet II, Src: 42:dd:4f:2b:72:80 (42:dd:4f:2b:72:80), Dst: 62:bc:16:f4:cc:7c (62:bc:16:f4:cc:7c)
▽ MultiProtocol Label Switching Header, Label: 600, Exp: 0, S: 1, TTL: 62
    MPLS Label: 600
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 1
    MPLS TTL: 62
▷ Internet Protocol, Src: 172.16.30.30 (172.16.30.30), Dst: 172.16.20.20 (172.16.20.20)
▷ Internet Control Message Protocol
```

Received label 600

MPLS Traffic analisys

- Sniffing on E3’s eth2: ICMP packets encapsulated in MPLS with a label swapped



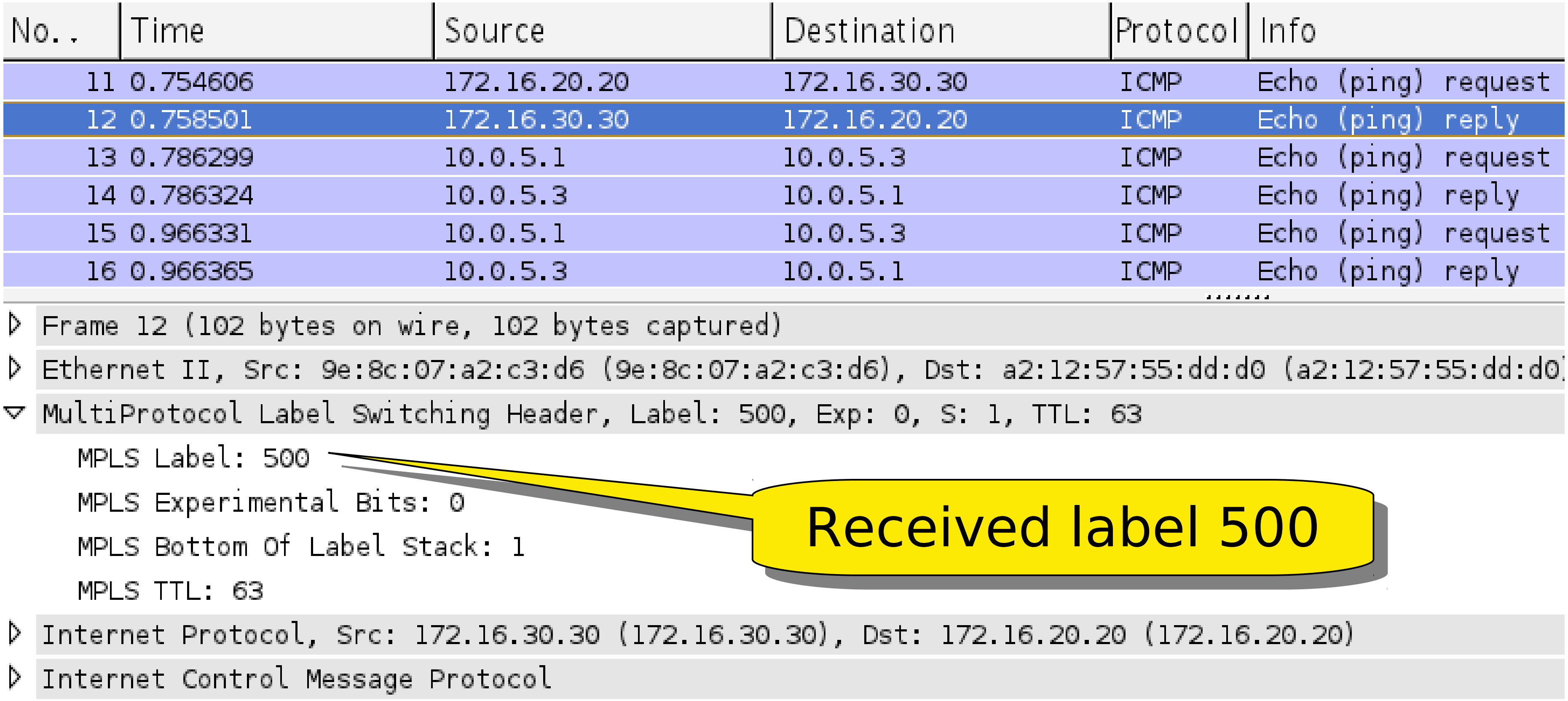
No. .	Time	Source	Destination	Protocol	Info
11	0.754606	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
12	0.758501	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply
13	0.786299	10.0.5.1	10.0.5.3	ICMP	Echo (ping) request
14	0.786324	10.0.5.3	10.0.5.1	ICMP	Echo (ping) reply
15	0.966331	10.0.5.1	10.0.5.3	ICMP	Echo (ping) request
16	0.966365	10.0.5.3	10.0.5.1	ICMP	Echo (ping) reply

```
▷ Frame 11 (102 bytes on wire, 102 bytes captured)
▷ Ethernet II, Src: a2:12:57:55:dd:d0 (a2:12:57:55:dd:d0), Dst: 9e:8c:07:a2:c3:d6 (9e:8c:07:a2:c3:d6)
▽ MultiProtocol Label Switching Header, Label: 200, Exp: 0, S: 1, TTL: 62
    MPLS Label: 200
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 1
    MPLS TTL: 62
▷ Internet Protocol, Src: 172.16.20.20 (172.16.20.20), Dst: 172.16.30.30 (172.16.30.30)
▷ Internet Control Message Protocol
```

Pushed label 200

MPLS Traffic analisys

- Sniffing on E3’s eth2



No..	Time	Source	Destination	Protocol	Info
11	0.754606	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
12	0.758501	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply
13	0.786299	10.0.5.1	10.0.5.3	ICMP	Echo (ping) request
14	0.786324	10.0.5.3	10.0.5.1	ICMP	Echo (ping) reply
15	0.966331	10.0.5.1	10.0.5.3	ICMP	Echo (ping) request
16	0.966365	10.0.5.3	10.0.5.1	ICMP	Echo (ping) reply

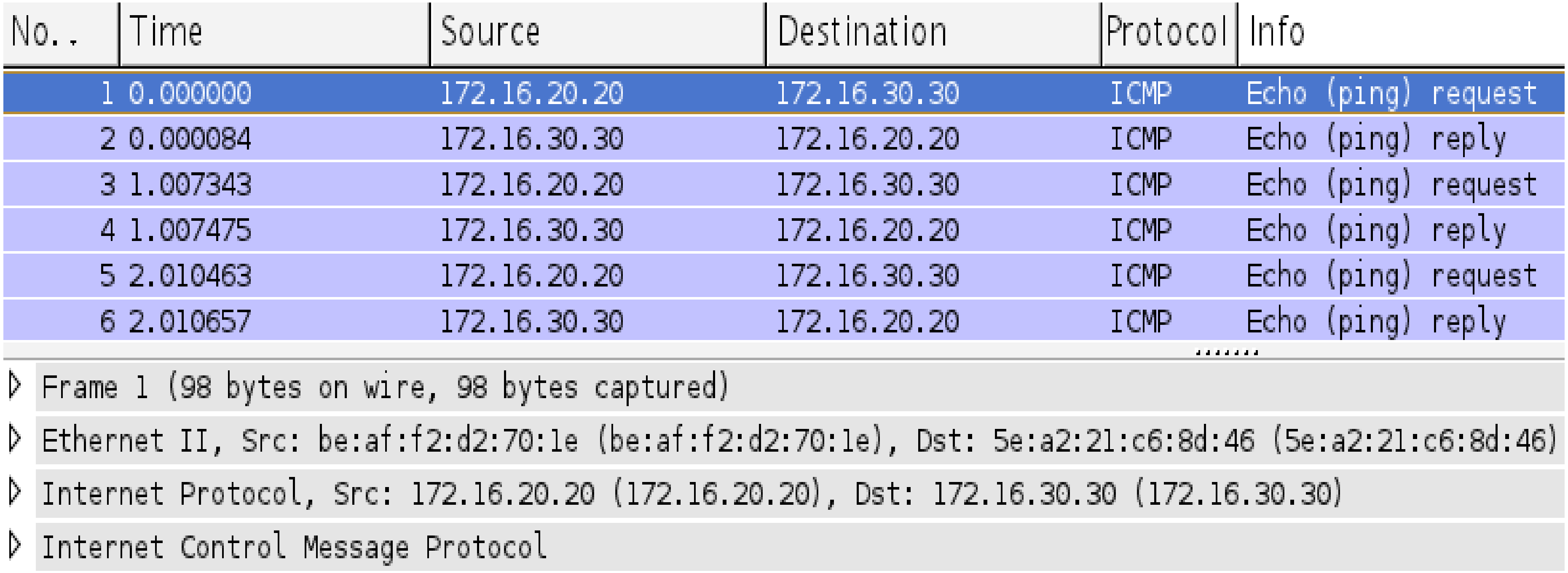
....

```
▷ Frame 12 (102 bytes on wire, 102 bytes captured)
▷ Ethernet II, Src: 9e:8c:07:a2:c3:d6 (9e:8c:07:a2:c3:d6), Dst: a2:12:57:55:dd:d0 (a2:12:57:55:dd:d0)
▽ MultiProtocol Label Switching Header, Label: 500, Exp: 0, S: 1, TTL: 63
    MPLS Label: 500
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 1
    MPLS TTL: 63
▷ Internet Protocol, Src: 172.16.30.30 (172.16.30.30), Dst: 172.16.20.20 (172.16.20.20)
▷ Internet Control Message Protocol
```

Received label 500

MPLS Traffic analisys

- Sniffing on E1’s eth1: (back to) plain ICMP packets

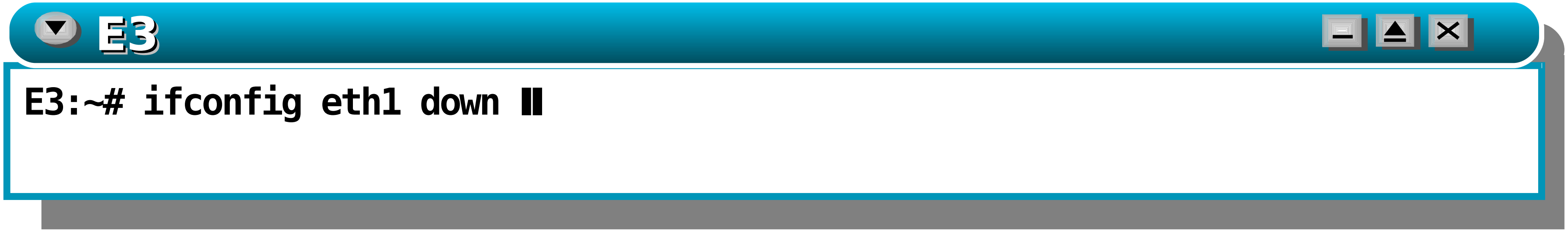


No..	Time	Source	Destination	Protocol	Info
1	0.000000	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
2	0.000084	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply
3	1.007343	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
4	1.007475	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply
5	2.010463	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
6	2.010657	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply

```
▷ Frame 1 (98 bytes on wire, 98 bytes captured)
▷ Ethernet II, Src: be:af:f2:d2:70:1e (be:af:f2:d2:70:1e), Dst: 5e:a2:21:c6:8d:46 (5e:a2:21:c6:8d:46)
▷ Internet Protocol, Src: 172.16.20.20 (172.16.20.20), Dst: 172.16.30.30 (172.16.30.30)
▷ Internet Control Message Protocol
```

MPLS Traffic analisys

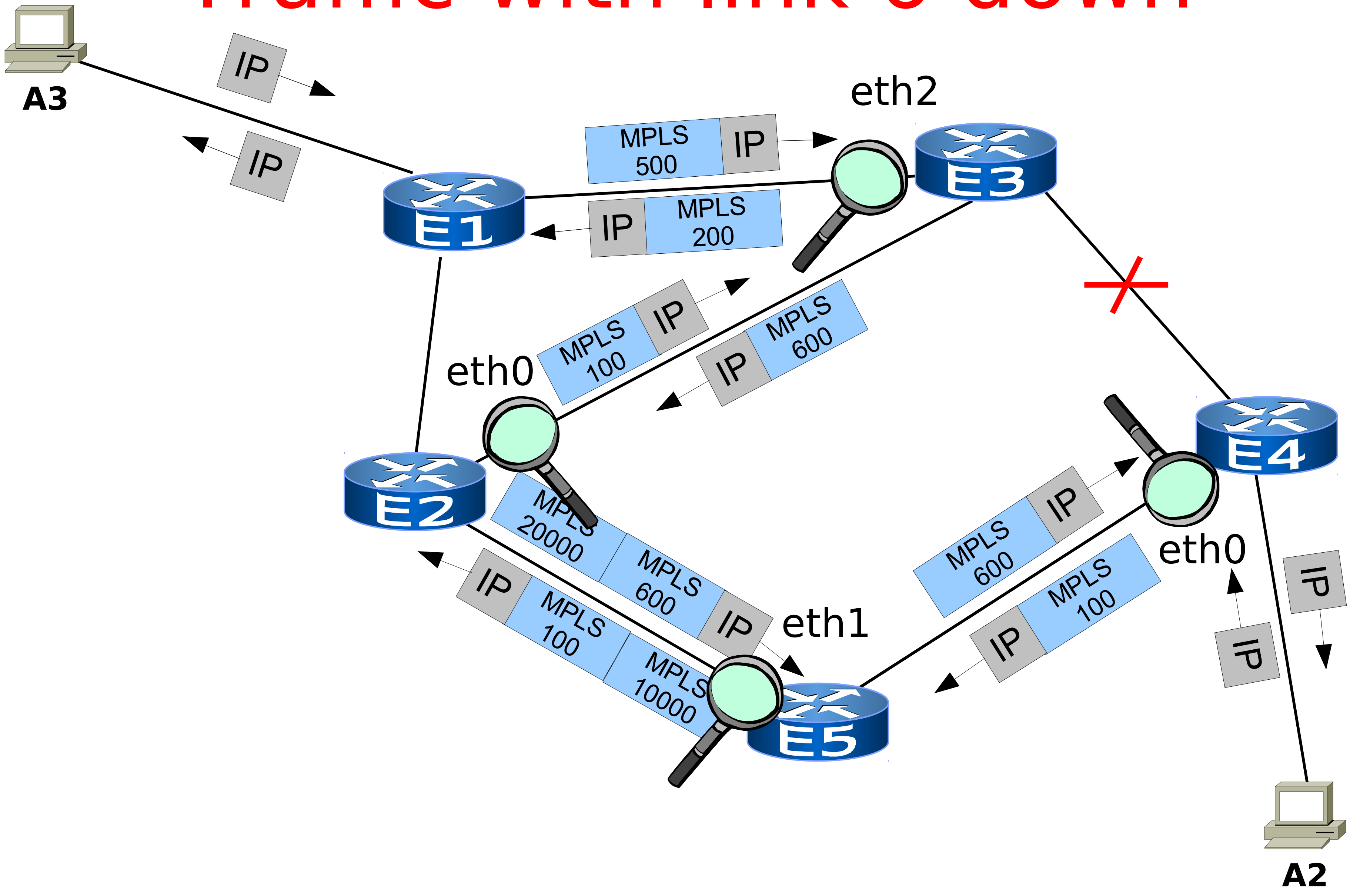
- Let's try to put link 6 down



```
E3:~# ifconfig eth1 down
```

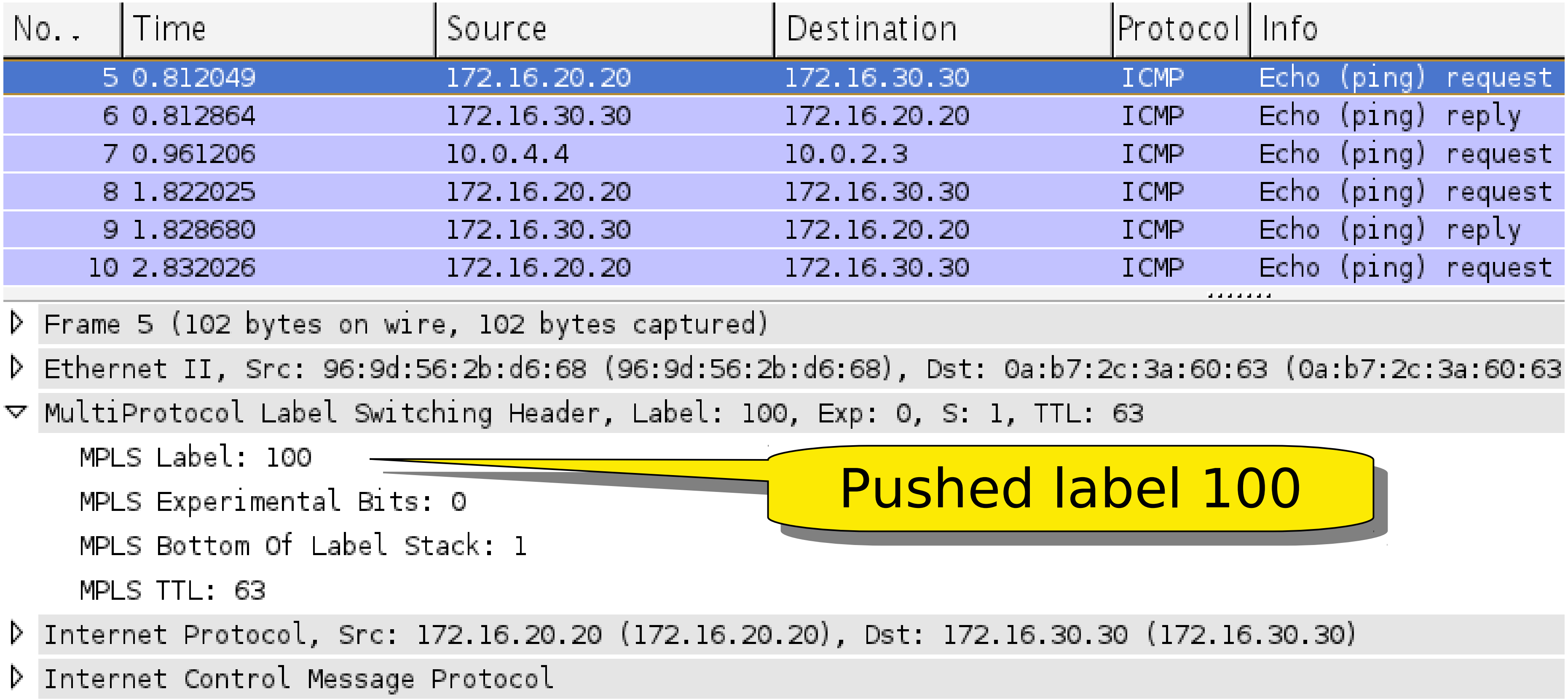
  - Routes taken by E1-E4 are updated (look at the currently selected routes in their windows)
  - Note: due to the time required for detection, a few ICMP packets may be lost (or enqueued at some interface until it is brought back up)

Traffic with link 6 down



MPLS traffic analisys

- Sniffing on E4’s eth0



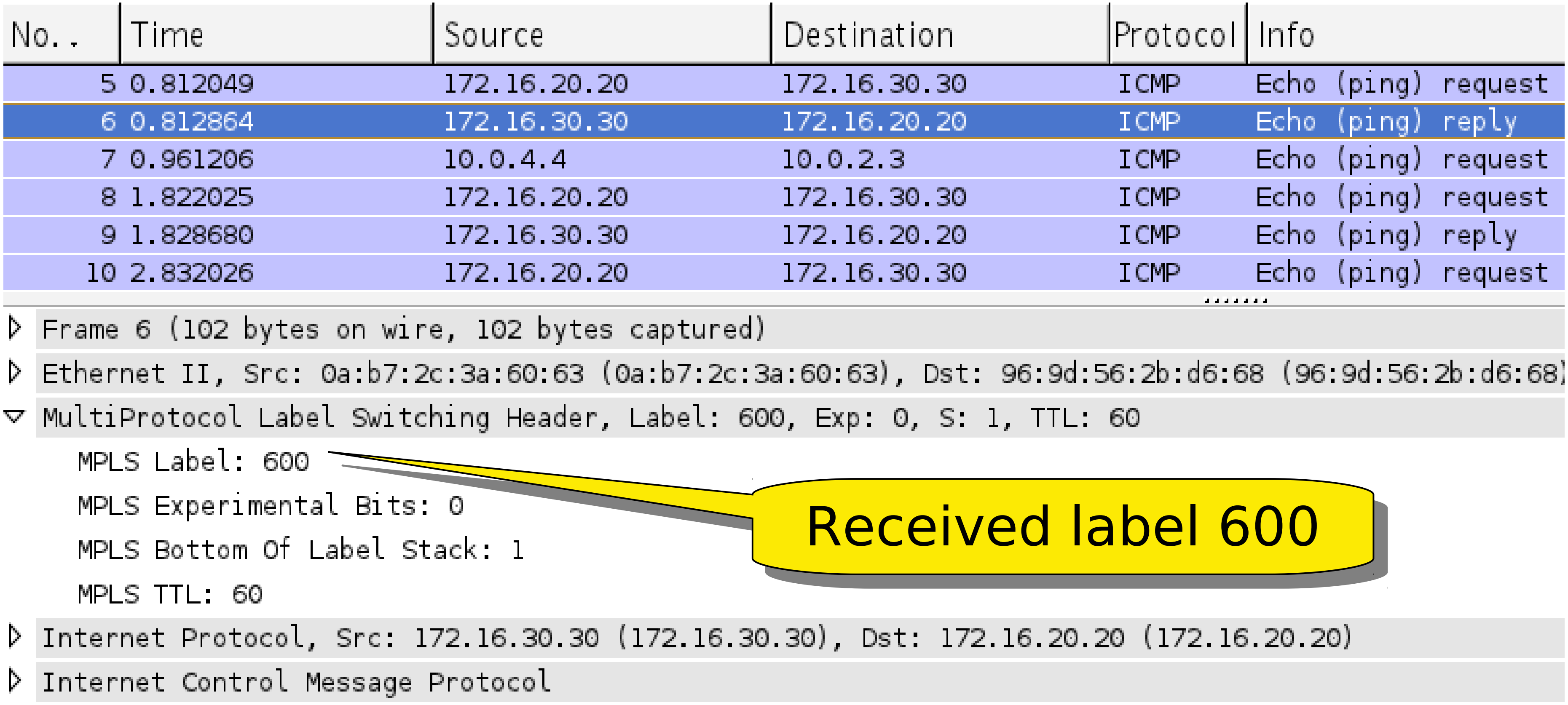
No..	Time	Source	Destination	Protocol	Info
5	0.812049	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
6	0.812864	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply
7	0.961206	10.0.4.4	10.0.2.3	ICMP	Echo (ping) request
8	1.822025	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
9	1.828680	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply
10	2.832026	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request

```
▷ Frame 5 (102 bytes on wire, 102 bytes captured)
▷ Ethernet II, Src: 96:9d:56:2b:d6:68 (96:9d:56:2b:d6:68), Dst: 0a:b7:2c:3a:60:63 (0a:b7:2c:3a:60:63
▽ MultiProtocol Label Switching Header, Label: 100, Exp: 0, S: 1, TTL: 63
    MPLS Label: 100
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 1
    MPLS TTL: 63
▷ Internet Protocol, Src: 172.16.20.20 (172.16.20.20), Dst: 172.16.30.30 (172.16.30.30)
▷ Internet Control Message Protocol
```

Pushed label 100

MPLS traffic analisys

- Sniffing on E4’s eth0



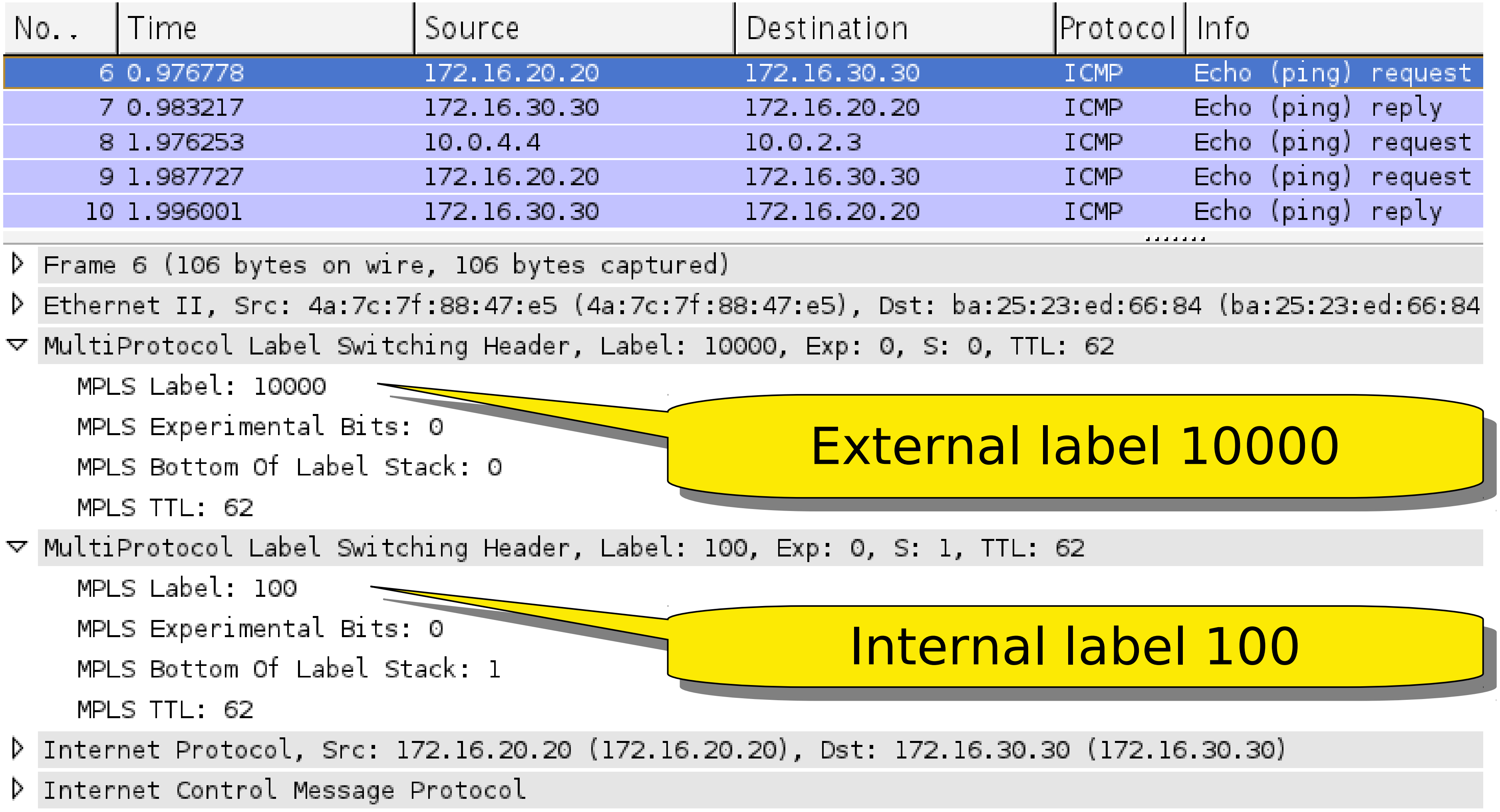
No..	Time	Source	Destination	Protocol	Info
5	0.812049	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
6	0.812864	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply
7	0.961206	10.0.4.4	10.0.2.3	ICMP	Echo (ping) request
8	1.822025	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
9	1.828680	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply
10	2.832026	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request

```
▷ Frame 6 (102 bytes on wire, 102 bytes captured)
▷ Ethernet II, Src: 0a:b7:2c:3a:60:63 (0a:b7:2c:3a:60:63), Dst: 96:9d:56:2b:d6:68 (96:9d:56:2b:d6:68
▽ MultiProtocol Label Switching Header, Label: 600, Exp: 0, S: 1, TTL: 60
     MPLS Label: 600
     MPLS Experimental Bits: 0
     MPLS Bottom Of Label Stack: 1
     MPLS TTL: 60
▷ Internet Protocol, Src: 172.16.30.30 (172.16.30.30), Dst: 172.16.20.20 (172.16.20.20)
▷ Internet Control Message Protocol
```

Received label 600

MPLS traffic analisys

- Sniffing on E5’s eth1



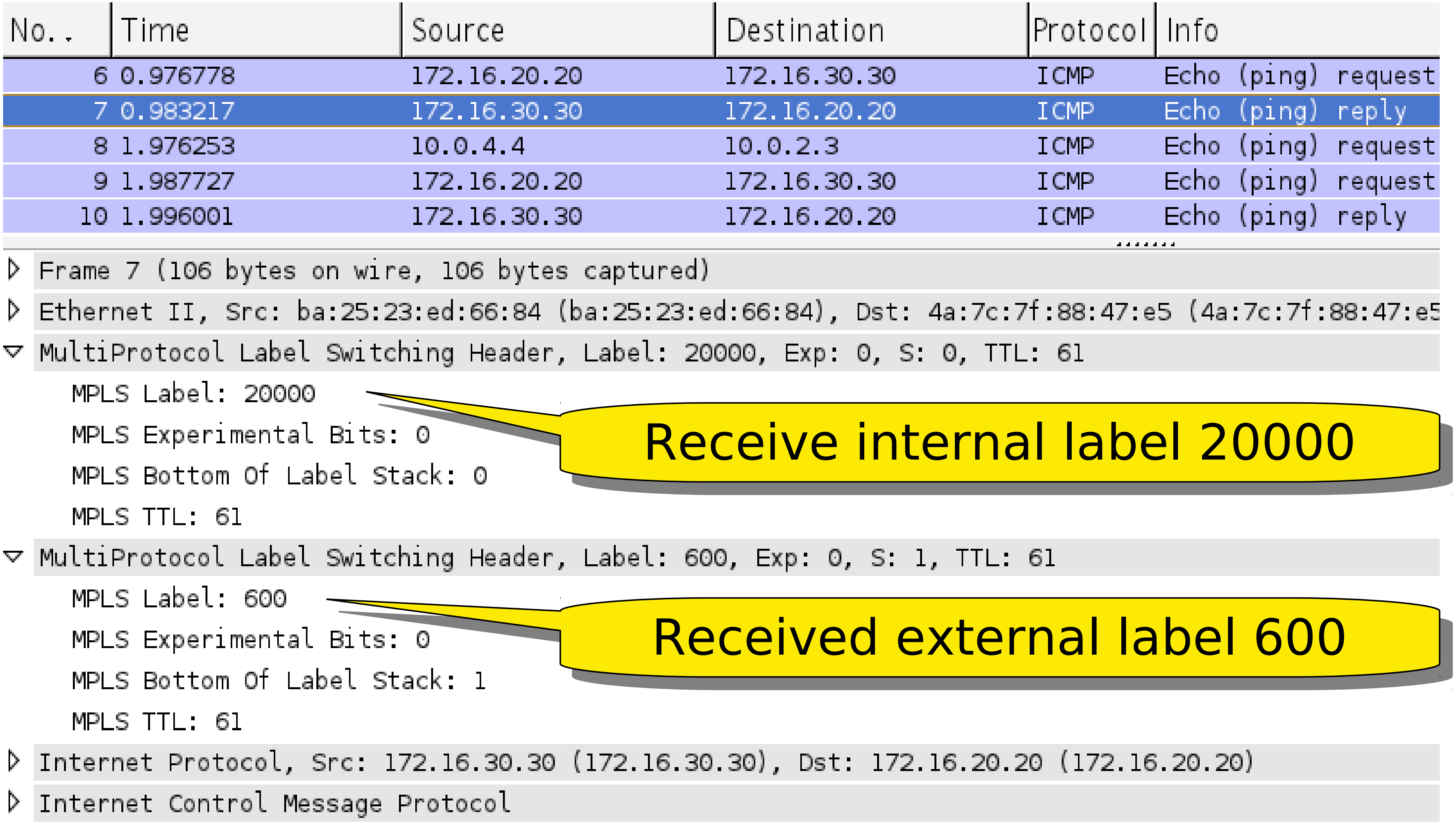
```
No..  Time       Source         Destination    Protocol  Info
    6 0.976778   172.16.20.20   172.16.30.30   ICMP      Echo (ping) request
    7 0.983217   172.16.30.30   172.16.20.20   ICMP      Echo (ping) reply
    8 1.976253   10.0.4.4       10.0.2.3       ICMP      Echo (ping) request
    9 1.987727   172.16.20.20   172.16.30.30   ICMP      Echo (ping) request
   10 1.996001   172.16.30.30   172.16.20.20   ICMP      Echo (ping) reply

▷ Frame 6 (106 bytes on wire, 106 bytes captured)
▷ Ethernet II, Src: 4a:7c:7f:88:47:e5 (4a:7c:7f:88:47:e5), Dst: ba:25:23:ed:66:84 (ba:25:23:ed:66:84
▽ MultiProtocol Label Switching Header, Label: 10000, Exp: 0, S: 0, TTL: 62
    MPLS Label: 10000
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 0
    MPLS TTL: 62
▽ MultiProtocol Label Switching Header, Label: 100, Exp: 0, S: 1, TTL: 62
    MPLS Label: 100
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 1
    MPLS TTL: 62
▷ Internet Protocol, Src: 172.16.20.20 (172.16.20.20), Dst: 172.16.30.30 (172.16.30.30)
▷ Internet Control Message Protocol
```

External label 10000

Internal label 100

MPLS traffic analisys

- Sniffing on E5’s eth1



No..	Time	Source	Destination	Protocol	Info
6	0.976778	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
7	0.983217	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply
8	1.976253	10.0.4.4	10.0.2.3	ICMP	Echo (ping) request
9	1.987727	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
10	1.996001	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply

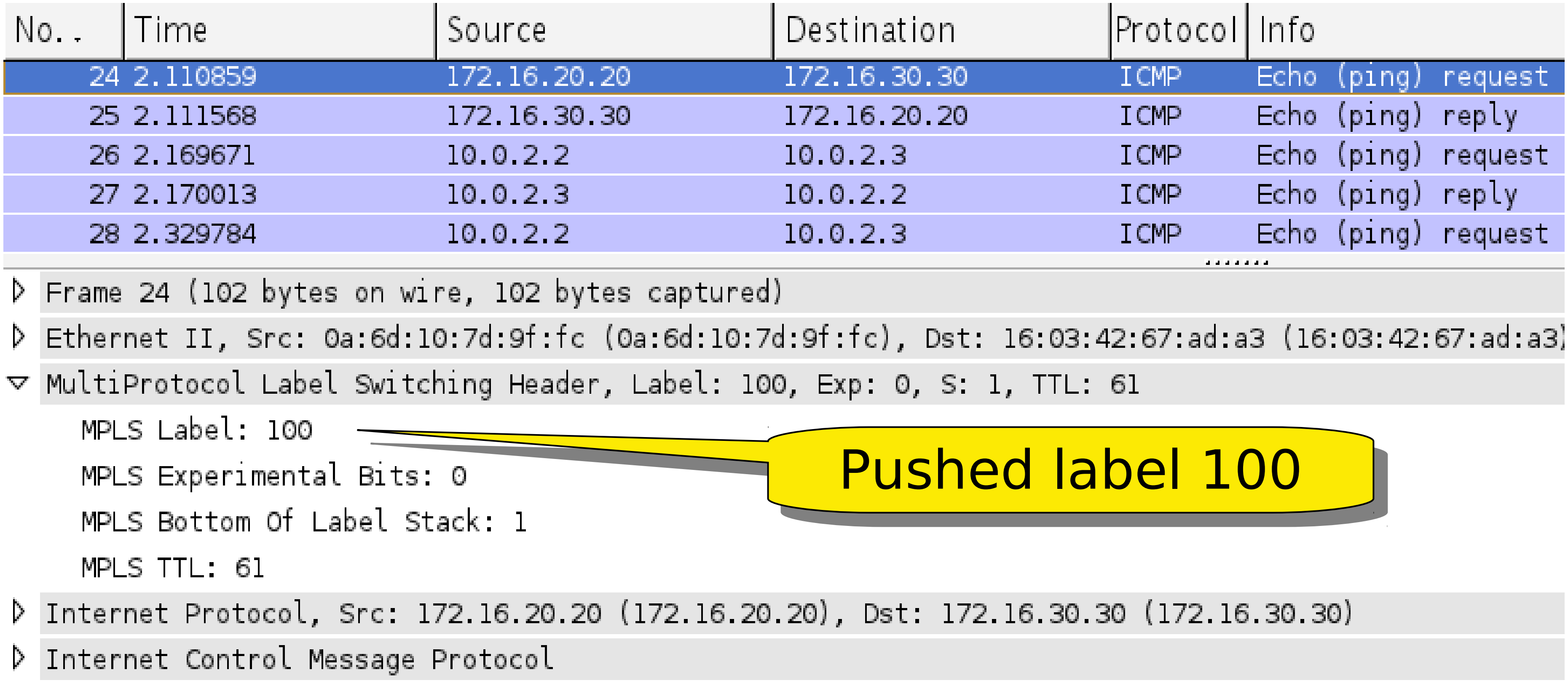
```
▷ Frame 7 (106 bytes on wire, 106 bytes captured)
▷ Ethernet II, Src: ba:25:23:ed:66:84 (ba:25:23:ed:66:84), Dst: 4a:7c:7f:88:47:e5 (4a:7c:7f:88:47:e5
▽ MultiProtocol Label Switching Header, Label: 20000, Exp: 0, S: 0, TTL: 61
    MPLS Label: 20000
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 0
    MPLS TTL: 61
▽ MultiProtocol Label Switching Header, Label: 600, Exp: 0, S: 1, TTL: 61
    MPLS Label: 600
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 1
    MPLS TTL: 61
▷ Internet Protocol, Src: 172.16.30.30 (172.16.30.30), Dst: 172.16.20.20 (172.16.20.20)
▷ Internet Control Message Protocol
```

Receive internal label 20000

Received external label 600

MPLS traffic analisys

- Sniffing on E2’s eth0



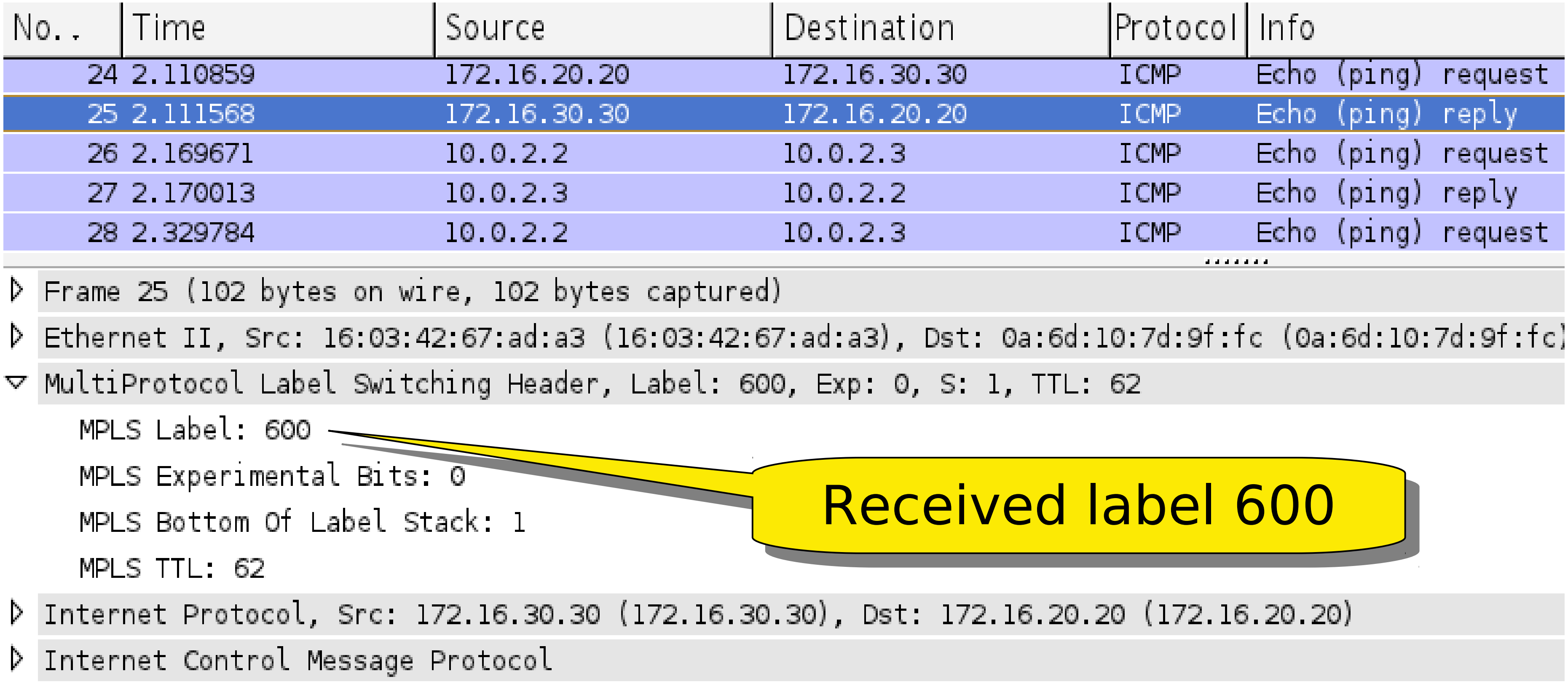
No..	Time	Source	Destination	Protocol	Info
24	2.110859	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
25	2.111568	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply
26	2.169671	10.0.2.2	10.0.2.3	ICMP	Echo (ping) request
27	2.170013	10.0.2.3	10.0.2.2	ICMP	Echo (ping) reply
28	2.329784	10.0.2.2	10.0.2.3	ICMP	Echo (ping) request

```
▷ Frame 24 (102 bytes on wire, 102 bytes captured)
▷ Ethernet II, Src: 0a:6d:10:7d:9f:fc (0a:6d:10:7d:9f:fc), Dst: 16:03:42:67:ad:a3 (16:03:42:67:ad:a3)
▽ MultiProtocol Label Switching Header, Label: 100, Exp: 0, S: 1, TTL: 61
    MPLS Label: 100
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 1
    MPLS TTL: 61
▷ Internet Protocol, Src: 172.16.20.20 (172.16.20.20), Dst: 172.16.30.30 (172.16.30.30)
▷ Internet Control Message Protocol
```

Pushed label 100

MPLS traffic analisys

- Sniffing on E2’s eth0



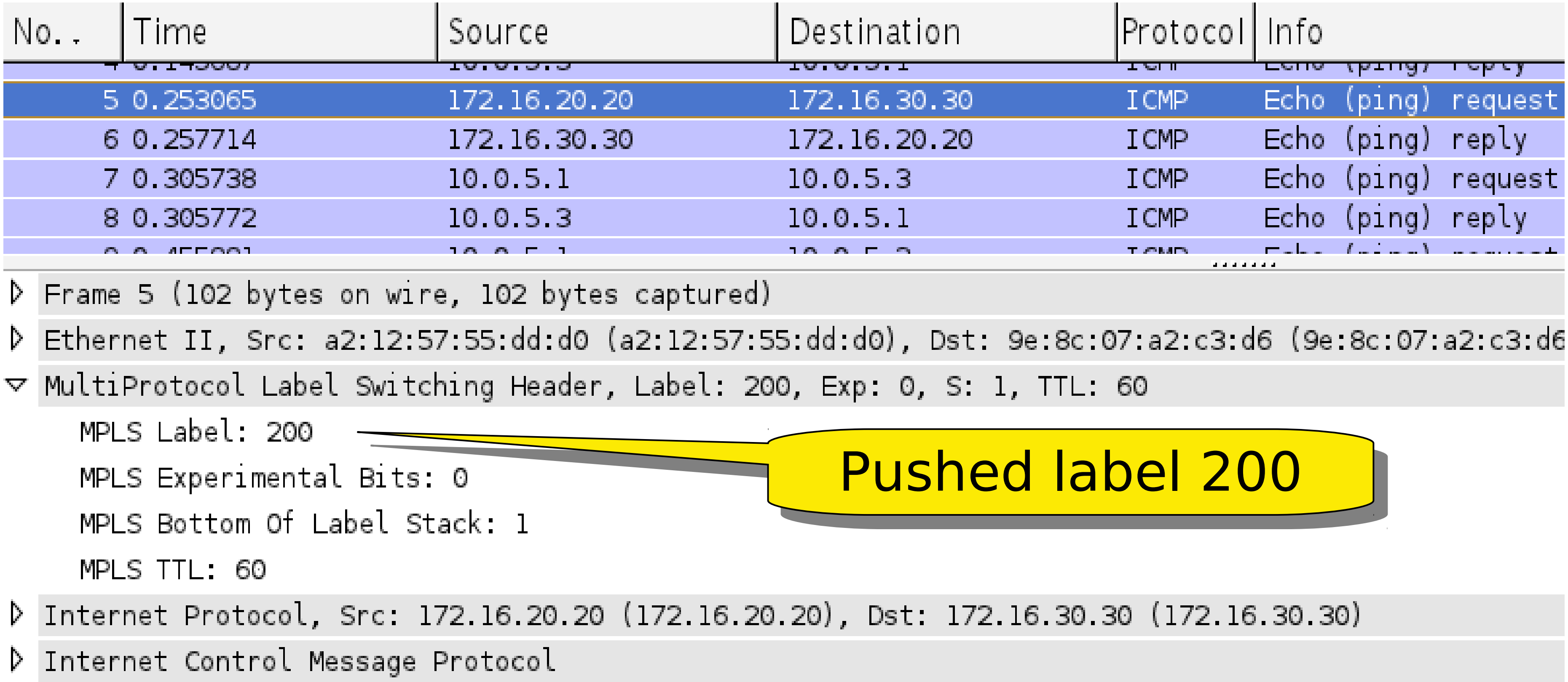
No..	Time	Source	Destination	Protocol	Info
24	2.110859	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
25	2.111568	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply
26	2.169671	10.0.2.2	10.0.2.3	ICMP	Echo (ping) request
27	2.170013	10.0.2.3	10.0.2.2	ICMP	Echo (ping) reply
28	2.329784	10.0.2.2	10.0.2.3	ICMP	Echo (ping) request

```
▷ Frame 25 (102 bytes on wire, 102 bytes captured)
▷ Ethernet II, Src: 16:03:42:67:ad:a3 (16:03:42:67:ad:a3), Dst: 0a:6d:10:7d:9f:fc (0a:6d:10:7d:9f:fc)
▽ MultiProtocol Label Switching Header, Label: 600, Exp: 0, S: 1, TTL: 62
    MPLS Label: 600
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 1
    MPLS TTL: 62
▷ Internet Protocol, Src: 172.16.30.30 (172.16.30.30), Dst: 172.16.20.20 (172.16.20.20)
▷ Internet Control Message Protocol
```

Received label 600

MPLS traffic analisys

- Sniffing on E3’s eth2



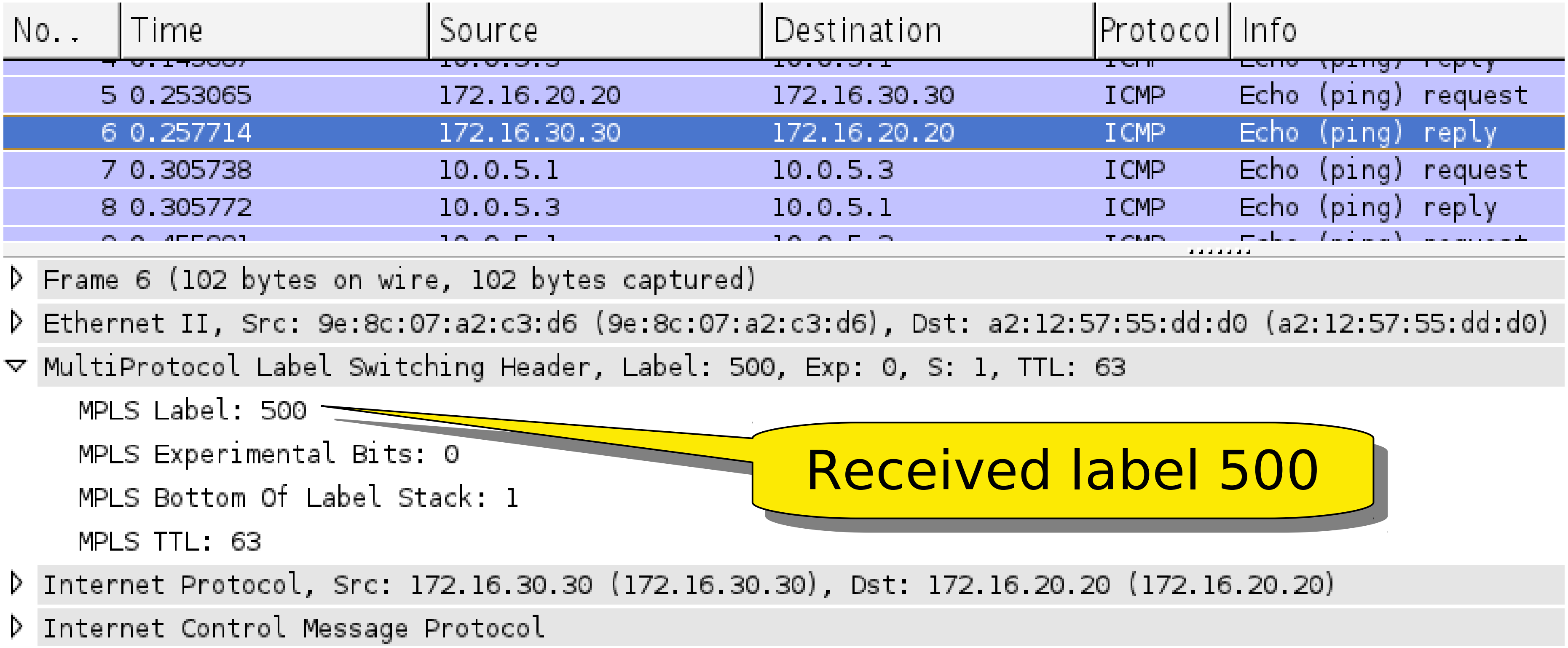
```
No..  Time       Source          Destination     Protocol  Info
    5 0.253065   172.16.20.20    172.16.30.30    ICMP      Echo (ping) request
    6 0.257714   172.16.30.30    172.16.20.20    ICMP      Echo (ping) reply
    7 0.305738   10.0.5.1        10.0.5.3        ICMP      Echo (ping) request
    8 0.305772   10.0.5.3        10.0.5.1        ICMP      Echo (ping) reply
```

```
▷ Frame 5 (102 bytes on wire, 102 bytes captured)
▷ Ethernet II, Src: a2:12:57:55:dd:d0 (a2:12:57:55:dd:d0), Dst: 9e:8c:07:a2:c3:d6 (9e:8c:07:a2:c3:d6
▽ MultiProtocol Label Switching Header, Label: 200, Exp: 0, S: 1, TTL: 60
    MPLS Label: 200
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 1
    MPLS TTL: 60
▷ Internet Protocol, Src: 172.16.20.20 (172.16.20.20), Dst: 172.16.30.30 (172.16.30.30)
▷ Internet Control Message Protocol
```

Pushed label 200

MPLS traffic analisys

- Sniffing on E3’s eth2



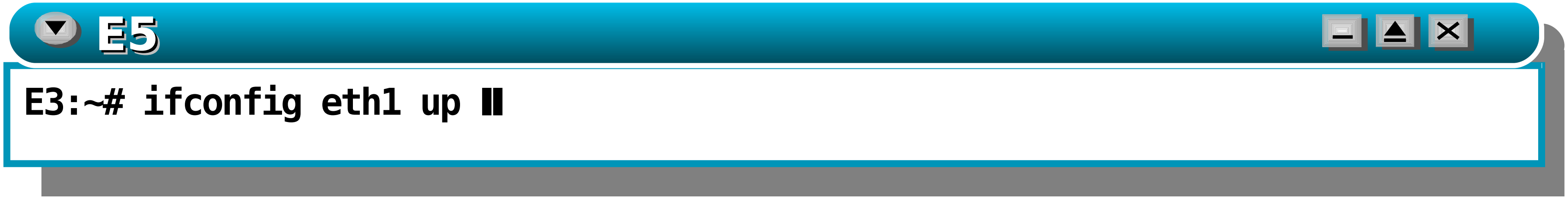
No..	Time	Source	Destination	Protocol	Info
4	0.14[illegible]	10.0.5.3	10.0.5.1	ICMP	Echo (ping) reply
5	0.253065	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
6	0.257714	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply
7	0.305738	10.0.5.1	10.0.5.3	ICMP	Echo (ping) request
8	0.305772	10.0.5.3	10.0.5.1	ICMP	Echo (ping) reply
9	0.455[illegible]	10.0.5.1	10.0.5.3	ICMP	Echo (ping) request

```
▷ Frame 6 (102 bytes on wire, 102 bytes captured)
▷ Ethernet II, Src: 9e:8c:07:a2:c3:d6 (9e:8c:07:a2:c3:d6), Dst: a2:12:57:55:dd:d0 (a2:12:57:55:dd:d0)
▽ MultiProtocol Label Switching Header, Label: 500, Exp: 0, S: 1, TTL: 63
    MPLS Label: 500
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 1
    MPLS TTL: 63
▷ Internet Protocol, Src: 172.16.30.30 (172.16.30.30), Dst: 172.16.20.20 (172.16.20.20)
▷ Internet Control Message Protocol
```

Received label 500

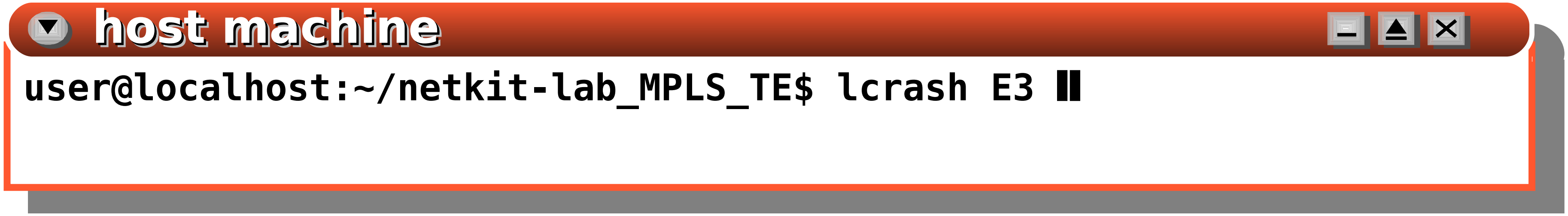
MPLS Traffic analisys

- We bring link 6 up again, you can check that traffic will switch to best



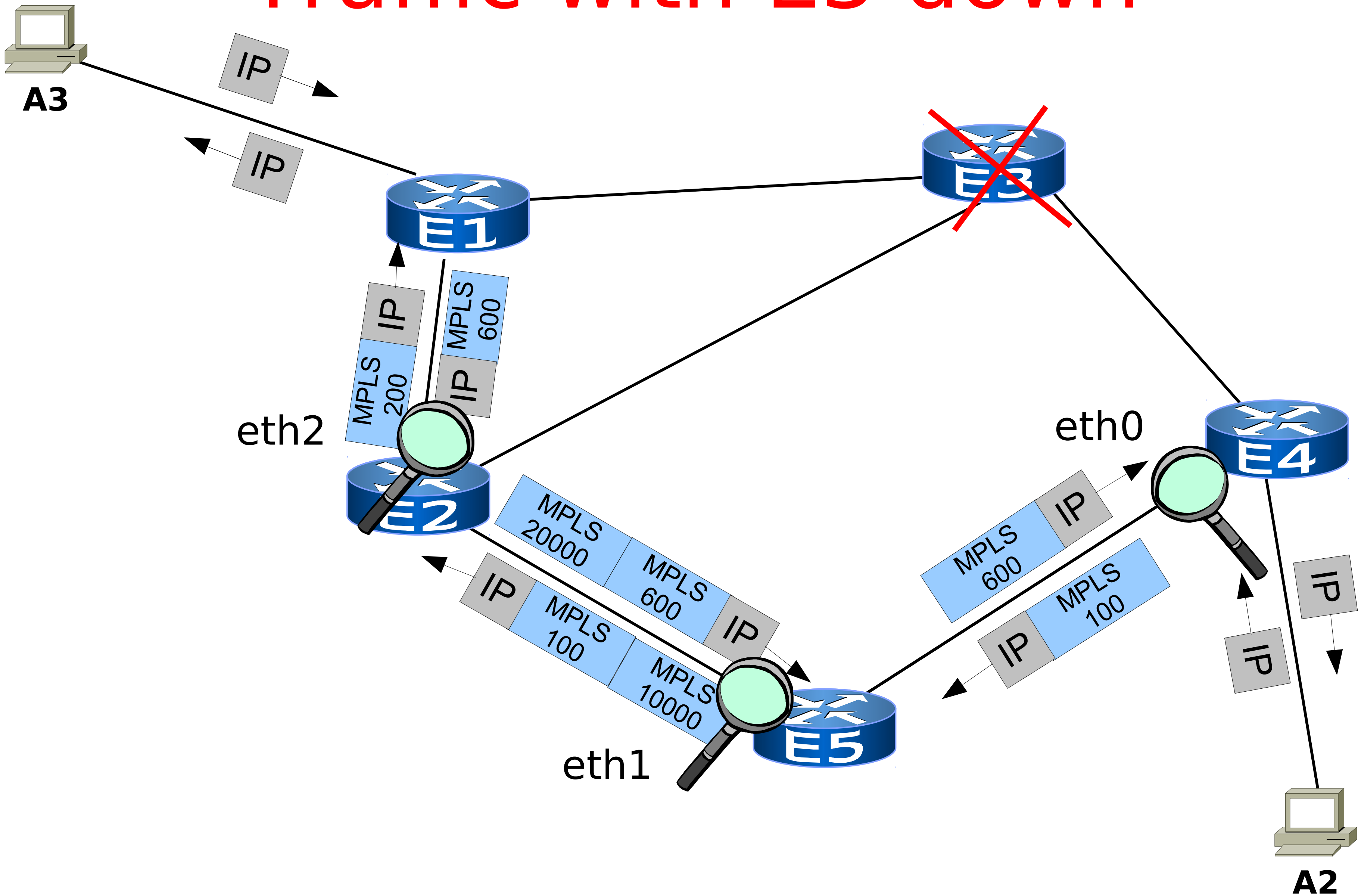
```
E3:~# ifconfig eth1 up
```

- Now we bring E3 down and see what happens



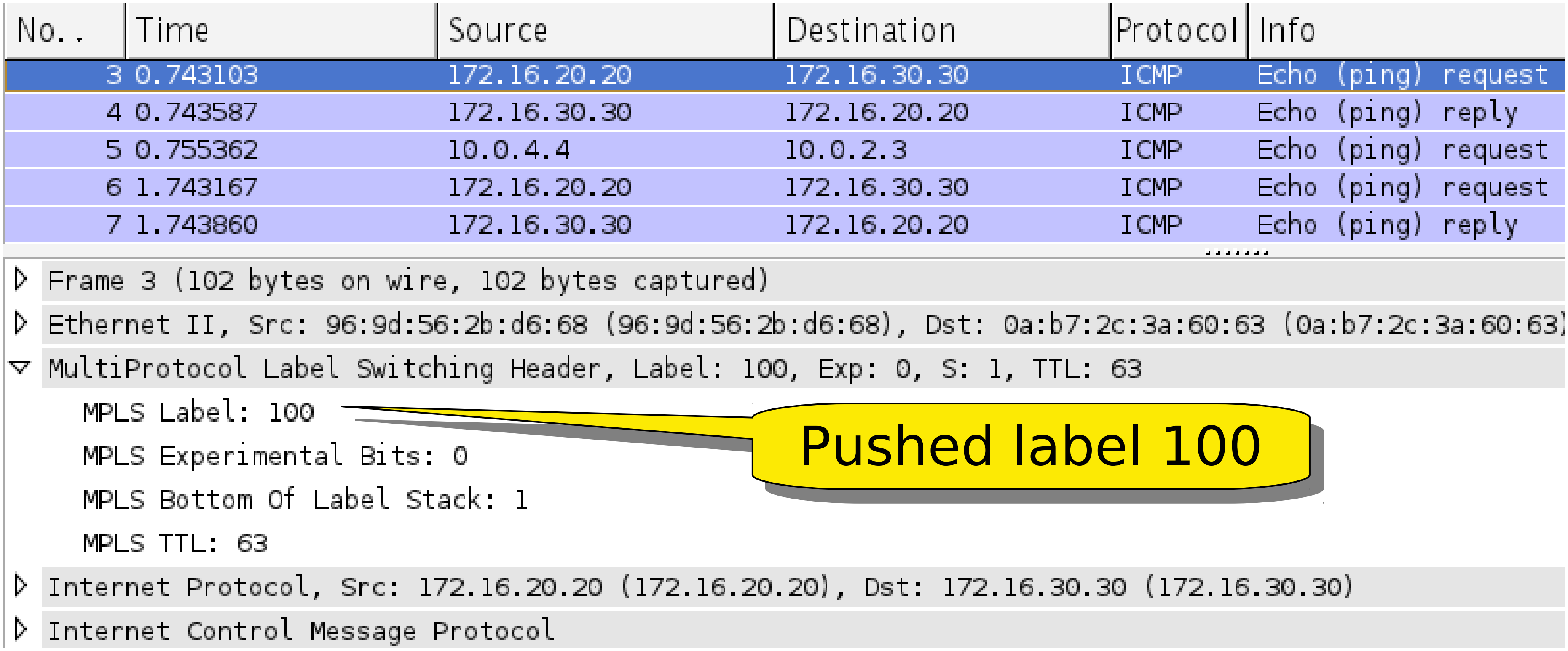
```
user@localhost:~/netkit-lab_MPLS_TE$ lcrash E3
```

Traffic with E3 down



MPLS traffic analisys

- Sniffing on E4’s eth0



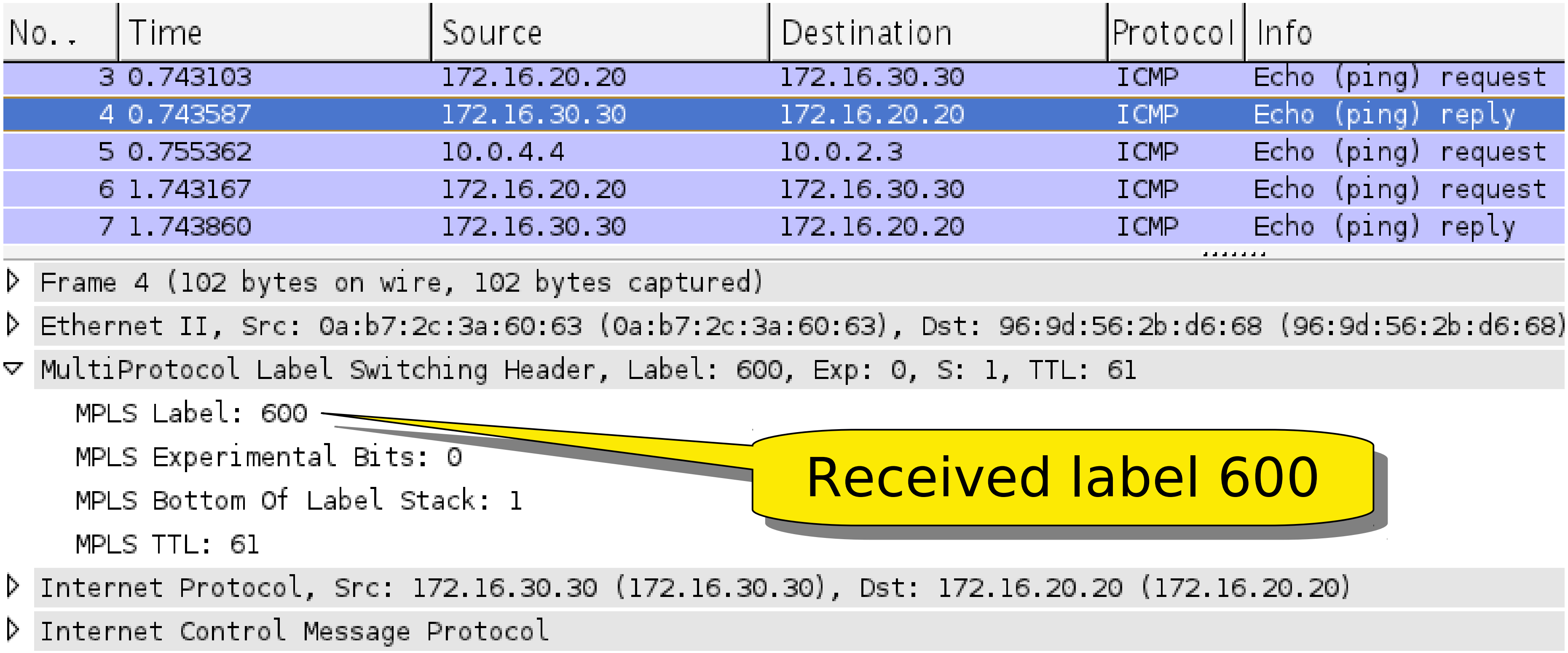
No..	Time	Source	Destination	Protocol	Info
3	0.743103	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
4	0.743587	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply
5	0.755362	10.0.4.4	10.0.2.3	ICMP	Echo (ping) request
6	1.743167	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
7	1.743860	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply

```
▷ Frame 3 (102 bytes on wire, 102 bytes captured)
▷ Ethernet II, Src: 96:9d:56:2b:d6:68 (96:9d:56:2b:d6:68), Dst: 0a:b7:2c:3a:60:63 (0a:b7:2c:3a:60:63)
▽ MultiProtocol Label Switching Header, Label: 100, Exp: 0, S: 1, TTL: 63
    MPLS Label: 100
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 1
    MPLS TTL: 63
▷ Internet Protocol, Src: 172.16.20.20 (172.16.20.20), Dst: 172.16.30.30 (172.16.30.30)
▷ Internet Control Message Protocol
```

Pushed label 100

MPLS traffic analisys

- Sniffing on E4’s eth0



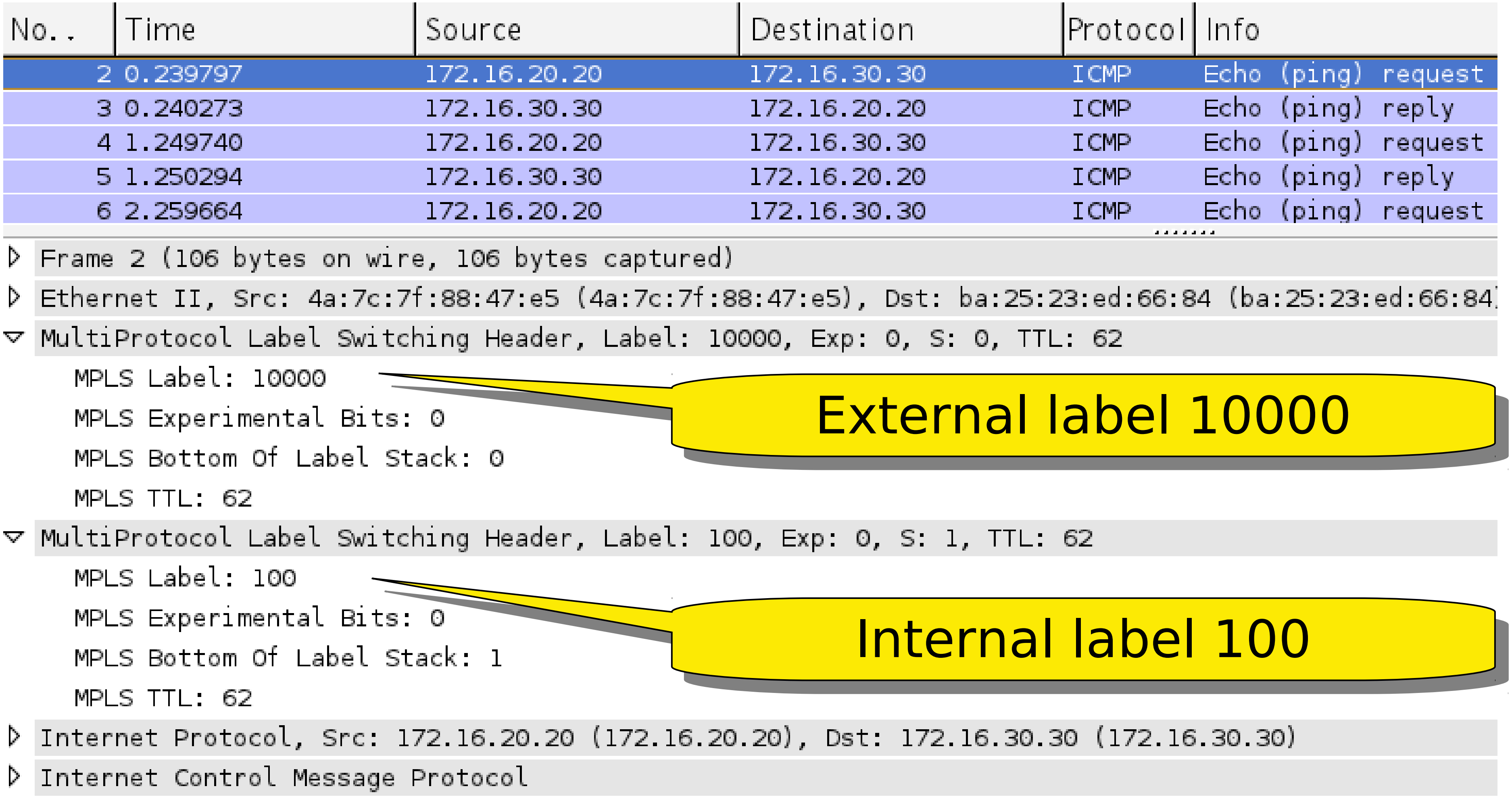
```
No..  Time      Source        Destination   Protocol  Info
   3 0.743103  172.16.20.20  172.16.30.30  ICMP      Echo (ping) request
   4 0.743587  172.16.30.30  172.16.20.20  ICMP      Echo (ping) reply
   5 0.755362  10.0.4.4      10.0.2.3      ICMP      Echo (ping) request
   6 1.743167  172.16.20.20  172.16.30.30  ICMP      Echo (ping) request
   7 1.743860  172.16.30.30  172.16.20.20  ICMP      Echo (ping) reply
```

```
▷ Frame 4 (102 bytes on wire, 102 bytes captured)
▷ Ethernet II, Src: 0a:b7:2c:3a:60:63 (0a:b7:2c:3a:60:63), Dst: 96:9d:56:2b:d6:68 (96:9d:56:2b:d6:68)
▽ MultiProtocol Label Switching Header, Label: 600, Exp: 0, S: 1, TTL: 61
    MPLS Label: 600
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 1
    MPLS TTL: 61
▷ Internet Protocol, Src: 172.16.30.30 (172.16.30.30), Dst: 172.16.20.20 (172.16.20.20)
▷ Internet Control Message Protocol
```

Received label 600

MPLS traffic analisys

- Sniffing on E5’s eth1



No..	Time	Source	Destination	Protocol	Info
2	0.239797	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
3	0.240273	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply
4	1.249740	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
5	1.250294	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply
6	2.259664	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request

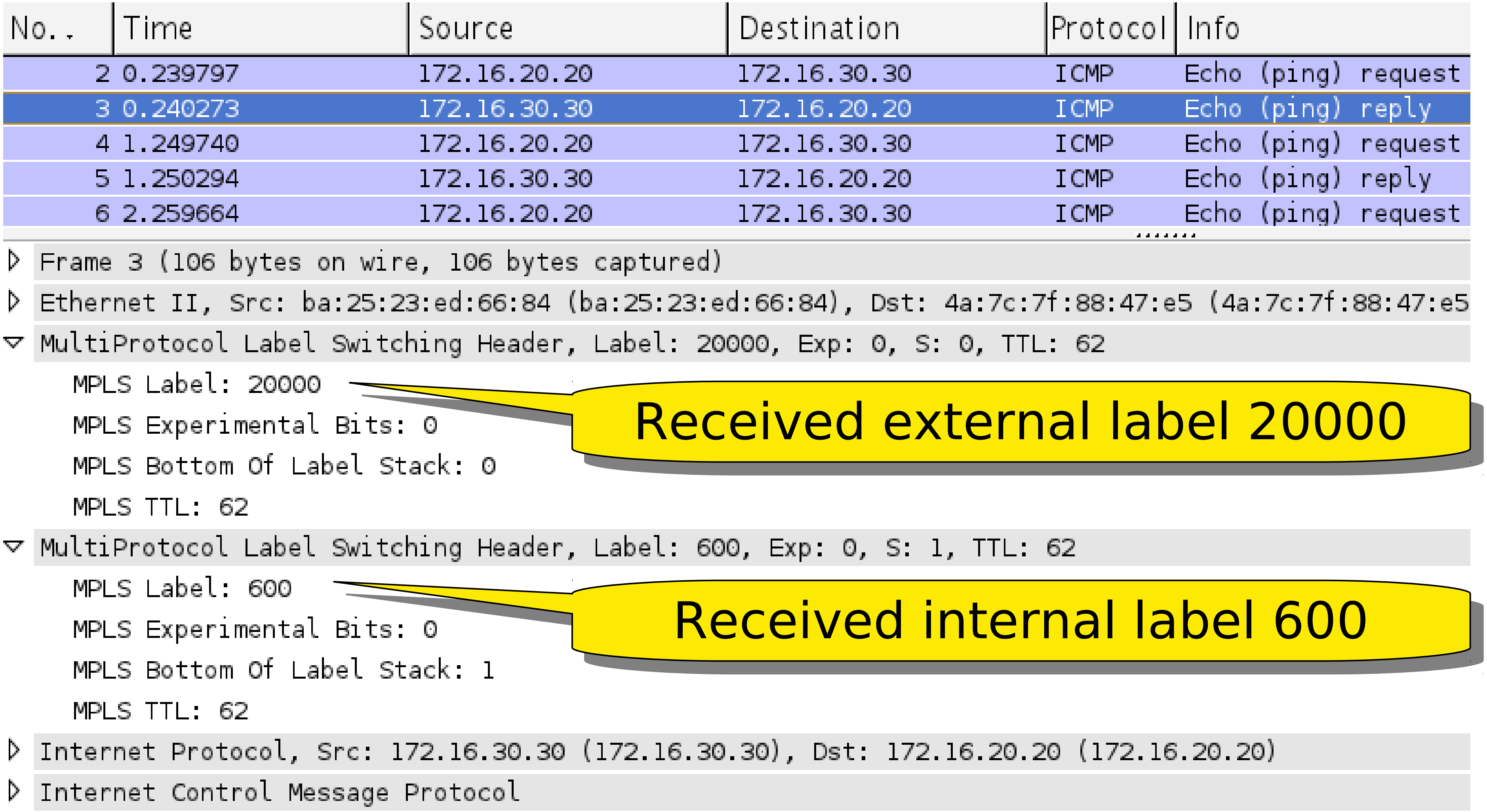
```
▷ Frame 2 (106 bytes on wire, 106 bytes captured)
▷ Ethernet II, Src: 4a:7c:7f:88:47:e5 (4a:7c:7f:88:47:e5), Dst: ba:25:23:ed:66:84 (ba:25:23:ed:66:84
▽ MultiProtocol Label Switching Header, Label: 10000, Exp: 0, S: 0, TTL: 62
    MPLS Label: 10000
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 0
    MPLS TTL: 62
▽ MultiProtocol Label Switching Header, Label: 100, Exp: 0, S: 1, TTL: 62
    MPLS Label: 100
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 1
    MPLS TTL: 62
▷ Internet Protocol, Src: 172.16.20.20 (172.16.20.20), Dst: 172.16.30.30 (172.16.30.30)
▷ Internet Control Message Protocol
```

External label 10000

Internal label 100

MPLS traffic analisys

- Sniffing on E5’s eth1



No. .	Time	Source	Destination	Protocol	Info
2	0.239797	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
3	0.240273	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply
4	1.249740	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request
5	1.250294	172.16.30.30	172.16.20.20	ICMP	Echo (ping) reply
6	2.259664	172.16.20.20	172.16.30.30	ICMP	Echo (ping) request

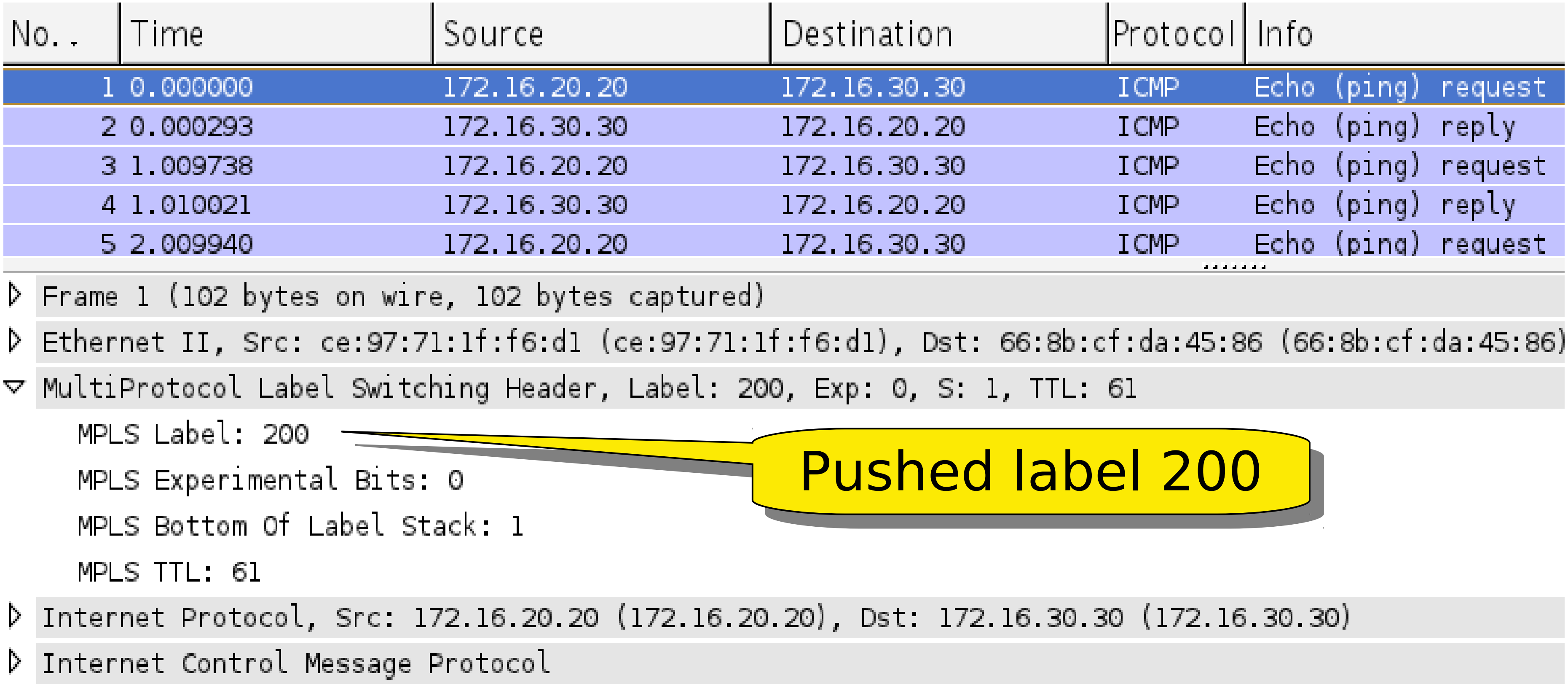
```
▷ Frame 3 (106 bytes on wire, 106 bytes captured)
▷ Ethernet II, Src: ba:25:23:ed:66:84 (ba:25:23:ed:66:84), Dst: 4a:7c:7f:88:47:e5 (4a:7c:7f:88:47:e5
▽ MultiProtocol Label Switching Header, Label: 20000, Exp: 0, S: 0, TTL: 62
    MPLS Label: 20000
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 0
    MPLS TTL: 62
▽ MultiProtocol Label Switching Header, Label: 600, Exp: 0, S: 1, TTL: 62
    MPLS Label: 600
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 1
    MPLS TTL: 62
▷ Internet Protocol, Src: 172.16.30.30 (172.16.30.30), Dst: 172.16.20.20 (172.16.20.20)
▷ Internet Control Message Protocol
```

Received external label 20000

Received internal label 600

MPLS traffic analisys

- Sniffing on E2’s eth2

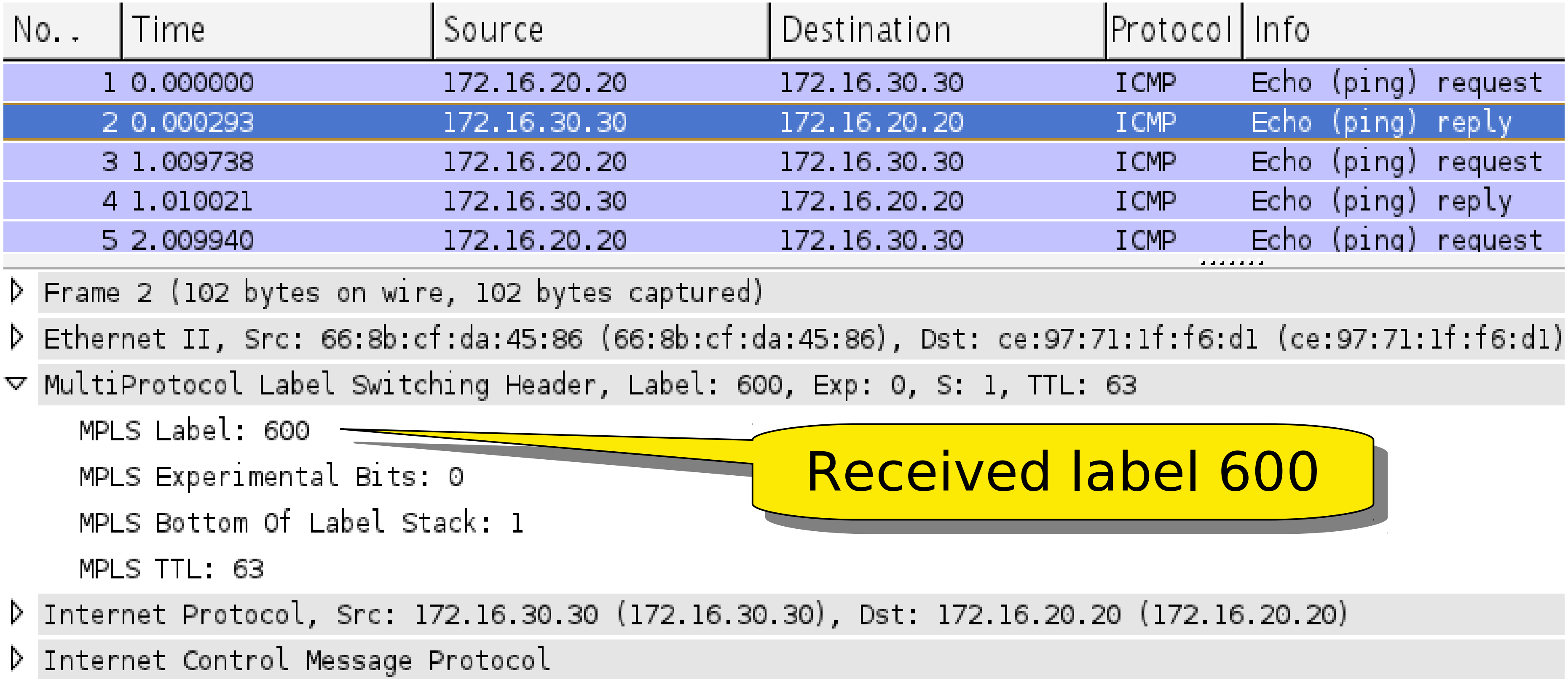


```
No..   Time        Source          Destination     Protocol  Info
     1 0.000000    172.16.20.20    172.16.30.30    ICMP      Echo (ping) request
     2 0.000293    172.16.30.30    172.16.20.20    ICMP      Echo (ping) reply
     3 1.009738    172.16.20.20    172.16.30.30    ICMP      Echo (ping) request
     4 1.010021    172.16.30.30    172.16.20.20    ICMP      Echo (ping) reply
     5 2.009940    172.16.20.20    172.16.30.30    ICMP      Echo (ping) request

▷ Frame 1 (102 bytes on wire, 102 bytes captured)
▷ Ethernet II, Src: ce:97:71:1f:f6:d1 (ce:97:71:1f:f6:d1), Dst: 66:8b:cf:da:45:86 (66:8b:cf:da:45:86)
▽ MultiProtocol Label Switching Header, Label: 200, Exp: 0, S: 1, TTL: 61
    MPLS Label: 200
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 1
    MPLS TTL: 61
▷ Internet Protocol, Src: 172.16.20.20 (172.16.20.20), Dst: 172.16.30.30 (172.16.30.30)
▷ Internet Control Message Protocol
```

Pushed label 200

MPLS traffic analisys

- Sniffing on E2’s eth2



```
No..  Time       Source        Destination   Protocol Info
    1 0.000000   172.16.20.20  172.16.30.30  ICMP     Echo (ping) request
    2 0.000293   172.16.30.30  172.16.20.20  ICMP     Echo (ping) reply
    3 1.009738   172.16.20.20  172.16.30.30  ICMP     Echo (ping) request
    4 1.010021   172.16.30.30  172.16.20.20  ICMP     Echo (ping) reply
    5 2.009940   172.16.20.20  172.16.30.30  ICMP     Echo (ping) request
```

```
▷ Frame 2 (102 bytes on wire, 102 bytes captured)
▷ Ethernet II, Src: 66:8b:cf:da:45:86 (66:8b:cf:da:45:86), Dst: ce:97:71:1f:f6:d1 (ce:97:71:1f:f6:d1)
▽ MultiProtocol Label Switching Header, Label: 600, Exp: 0, S: 1, TTL: 63
    MPLS Label: 600
    MPLS Experimental Bits: 0
    MPLS Bottom Of Label Stack: 1
    MPLS TTL: 63
▷ Internet Protocol, Src: 172.16.30.30 (172.16.30.30), Dst: 172.16.20.20 (172.16.20.20)
▷ Internet Control Message Protocol
```

Received label 600

Proposed exercises

- Sniff on all the interfaces with wireshark
- Try to change NHLFE tables on the running lab, adding or deleting entries (use “mpls -help” inside virtual machines to see the help)
- Try to change label numbers and encapsulation