########Scripts for MPLS setup in the lab########

#########Configuration scripts commpn for all experiments############

##script to be run in all nodes before configuration##delmpls.sh##

```
#!/bin/bash

echo Deleting all mpls settings ...

echo Deleting all mpls - xc settings
xc_output="`mpls xc show | cut -c 9-75`"

#parse by line, if there are multiple lines

number_of_lines=`echo $xc_output|wc -l`

for i in `seq 1 $number_of_lines`
do
  xc_output_line="`echo $xc_output| head -$i | tail -1`"
  echo Deleting: mpls xc del $xc_output_line
  mpls xc del $xc_output

  #increment i
  i=`expr $i+1`
done

echo Deleting all mpls - ip route settings
nr_of_lines=` ip route show |grep mpls |wc -l`
for i in `seq 1 $nr_of_lines `
do
 output=`ip route show |grep mpls |head -1`
 echo Deleting: ip route del $output
 ip route del $output
 i=`expr $i+1`
 done


echo Flushing iptables chains
iptables -F

echo Deleting all mpls - nhlfe settings
nr_of_lines=` mpls nhlfe show |grep key |wc -l`
for i in `seq 1 $nr_of_lines `
do
 output=`mpls nhlfe show |grep key |cut -c 17-26 |head -1`
 echo Deleting: mpls nhlfe del key $output
 mpls nhlfe del key $output
 i=`expr $i+1`
done

echo Deleting all mpls - labelspace settings
nr_of_lines=`mpls labelspace show |grep -v "labelspace -1" |wc -l`
for i in `seq 1 $nr_of_lines `
do
```

```
 output=`mpls labelspace show |grep -v "labelspace -1"|cut -c 17-37 |head -1`
 echo Deleting: mpls labelspace set $output -1
 mpls labelspace set $output -1
 i=`expr $i+1`
done

echo Deleting all mpls - ilm settings
nr_of_lines=`mpls ilm show |grep ILM |wc -l`
for i in `seq 1 $nr_of_lines `
do
 output=`mpls ilm show |grep ILM |cut -c 10-37 |head -1`
 echo Deleting: mpls ilm del $output
 mpls ilm del $output
 i=`expr $i+1`
done
```

##Client##conf.sh##

```
#!/bin/bash

echo Configuring interfaces on Client
ifconfig eth0 down

ifconfig eth0 192.168.0.4 netmask 255.255.255.0 up
echo eth0 is 192.168.0.4

ip route add 192.168.4.2/32 via 192.168.0.1 src 192.168.0.4
```

##Server##conf.sh##

```
#!/bin/bash

echo Configuring interfaces on Server
ifconfig eth0 down

ifconfig eth0 192.168.4.5 netmask 255.255.255.0 up
echo eth0 is 192.168.4.5

ip route add 192.168.0.4/32 via 192.168.4.2 src 192.168.4.5
```

##LER1##conf.sh##

```
#!/bin/bash

echo Configuring interfaces on Ler1

ifconfig eth0 down
ifconfig eth2 down
ifconfig eth1 down

ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
ifconfig eth1 192.168.1.1 netmask 255.255.255.0 up
ifconfig eth2 192.168.2.1 netmask 255.255.255.0 up
```

```
echo eth0 is 192.168.0.1
echo eth1 is 192.168.1.1
echo eth2 is 192.168.2.1

echo Starting routing ...
echo "1" >/proc/sys/net/ipv4/ip_forward
echo 0 >/proc/sys/net/ipv4/conf/all/rp_filter
echo No MPLS debug
echo "0" >/sys/mpls/debug
```

##LSR##conf.sh##

```
#!/bin/bash

echo Configuring interfaces on Lsr

ifconfig eth0 down
ifconfig eth1 down

ifconfig eth0 192.168.3.3 netmask 255.255.255.0 up
ifconfig eth1 192.168.1.3 netmask 255.255.255.0 up

echo eth0 is 192.168.3.3
echo eth1 is 192.168.1.3

echo Starting routing ...
echo "1" >/proc/sys/net/ipv4/ip_forward
echo 0 >/proc/sys/net/ipv4/conf/all/rp_filter
echo No MPLS debug
echo "0" >/sys/mpls/debug
```

##LER2##conf.sh##

```
#!/bin/bash

echo Configuring interfaces on Ler2

ifconfig eth0 down
ifconfig eth2 down
ifconfig eth1 down

ifconfig eth0 192.168.4.2 netmask 255.255.255.0 up
ifconfig eth1 192.168.3.2 netmask 255.255.255.0 up
ifconfig eth2 192.168.2.2 netmask 255.255.255.0 up


echo eth0 is 192.168.4.2
echo eth1 is 192.168.3.2
echo eth2 is 192.168.2.2

echo Starting routing ...
```

```
echo "1" >/proc/sys/net/ipv4/ip_forward
echo 0 >/proc/sys/net/ipv4/conf/all/rp_filter
echo No MPLS debug
echo "0" >/sys/mpls/debug
```

##########1. Basic experiment with two LERs

##LER1##ler1.sh##

```
#!/bin/bash


# Client to Server
echo add label 1000 and forward the packet to ler2 on output interface eth2
for destination Server
key_value=`mpls nhlfe add key 0 instructions push gen 1000 nexthop eth2 ipv4
192.168.2.2`
key=`echo $key_value | awk '{print $4}'`
ip route add 192.168.4.5/32 via 192.168.2.2 mpls $key

# Server to Client
echo expect and pop label 2000 from interface eth2
mpls labelspace set dev eth2 labelspace 0
mpls ilm add label gen 2000 labelspace 0
key_value=`mpls nhlfe add key 0 instructions nexthop eth0 ipv4 192.168.0.4`
key=`echo $key_value | awk '{print $4}'`
mpls xc add ilm_label gen 2000 ilm_labelspace 0 nhlfe_key $key
```

##LER2##ler2.sh##

```
#!/bin/bash

# Server to Client
echo add label 2000 and forward the packet to ler1 on output interface eth2
for destination Client
key_value=`mpls nhlfe add key 0 instructions push gen 2000 nexthop eth2 ipv4
192.168.2.1`
key=`echo $key_value | awk '{print $4}'`
ip route add 192.168.0.4/32 via 192.168.2.1 mpls $key

# Client to Server
echo expect and pop label 1000 from interface eth2
mpls labelspace set dev eth2 labelspace 0
mpls ilm add label gen 1000 labelspace 0
key_value=`mpls nhlfe add key 0 instructions nexthop eth0 ipv4 192.168.4.5`
key=`echo $key_value | awk '{print $4}'`
mpls xc add ilm_label gen 1000 ilm_labelspace 0 nhlfe_key $key
```

#########2. LER-LSR-LER with same labelspace

##LER1##ler1.sh##

```
#!/bin/bash

# Client to Server
echo add label 1000 and forward the packet to lsr on output interface eth1
for destination Server
key_value=`mpls nhlfe add key 0 instructions push gen 1000 nexthop eth1 ipv4
192.168.1.3`
key=`echo $key_value | awk '{print $4}'`
ip route add 192.168.4.5/32 via 192.168.1.3 mpls $key

# Server to Client
echo expect and pop label 200 from interface eth1
mpls labelspace set dev eth1 labelspace 0
mpls ilm add label gen 200 labelspace 0
key_value=`mpls nhlfe add key 0 instructions nexthop eth0 ipv4 192.168.0.4`
key=`echo $key_value | awk '{print $4}'`
mpls xc add ilm_label gen 200 ilm_labelspace 0 nhlfe_key $key
```

##LSR##lsr.sh##

```
#!/bin/bash

# Client to Server
echo expect and pop label 1000 from interface eth1 exchange with 2000 send to
eth0
mpls labelspace set dev eth1 labelspace 0
mpls ilm add label gen 1000 labelspace 0
key_value=`mpls nhlfe add key 0 instructions push gen 2000 nexthop eth0 ipv4
192.168.3.2`
key=`echo $key_value | awk '{print $4}'`
mpls xc add ilm_label gen 1000 ilm_labelspace 0 nhlfe_key $key

# Server to Client
echo expect and pop label 100 from interface eth0 exchange with 200 send to
eth1
mpls labelspace set dev eth0 labelspace 0
mpls ilm add label gen 100 labelspace 0
key_value=`mpls nhlfe add key 0 instructions push gen 200 nexthop eth1 ipv4
192.168.1.1`
key=`echo $key_value | awk '{print $4}'`
mpls xc add ilm_label gen 100 ilm_labelspace 0 nhlfe_key $key
```

##LER2##ler2.sh##

```
#!/bin/bash

# Server to Client
echo add label 100 and forward the packet to lsr on output interface eth1 for
destination Client
key_value=`mpls nhlfe add key 0 instructions push gen 100 nexthop eth1 ipv4
192.168.3.3`
key=`echo $key_value | awk '{print $4}'`
ip route add 192.168.0.4/32 via 192.168.3.3 mpls $key

# Client to Server
echo expect and pop label 2000 from interface eth1
mpls labelspace set dev eth2 labelspace 0
mpls ilm add label gen 2000 labelspace 0
key_value=`mpls nhlfe add key 0 instructions nexthop eth0 ipv4 192.168.4.5`
key=`echo $key_value | awk '{print $4}'`
mpls xc add ilm_label gen 2000 ilm_labelspace 0 nhlfe_key $key
```

##########3. LER-LSR-LER with different labelspace

##LER1##ler1.sh##

```
#!/bin/bash

# Client to Server
echo add label 1000 and forward the packet to lsr on output interface eth1
for destination Server
key_value=`mpls nhlfe add key 0 instructions push gen 1000 nexthop eth1 ipv4
192.168.1.3`
key=`echo $key_value | awk '{print $4}'`
ip route add 192.168.4.5/32 via 192.168.1.3 mpls $key

# Server to Client
echo expect and pop label 1000 from interface eth1
mpls labelspace set dev eth1 labelspace 0
mpls ilm add label gen 1000 labelspace 0
key_value=`mpls nhlfe add key 0 instructions nexthop eth0 ipv4 192.168.0.4`
key=`echo $key_value | awk '{print $4}'`
mpls xc add ilm_label gen 1000 ilm_labelspace 0 nhlfe_key $key
```

##LSR##lsr.sh##

```
#!/bin/bash

# Client to Server
echo expect and pop label 1000 from interface eth1 exchange with 1000 send to
eth0
mpls labelspace set dev eth1 labelspace 0
mpls ilm add label gen 1000 labelspace 0
key_value=`mpls nhlfe add key 0 instructions push gen 1000 nexthop eth0 ipv4
192.168.3.2`
key=`echo $key_value | awk '{print $4}'`
mpls xc add ilm_label gen 1000 ilm_labelspace 0 nhlfe_key $key

# Server to Client
echo expect and pop label 1000 from interface eth0 exchange with 1000 send to
eth1
mpls labelspace set dev eth0 labelspace 1
mpls ilm add label gen 1000 labelspace 1
key_value=`mpls nhlfe add key 0 instructions push gen 1000 nexthop eth1 ipv4
192.168.1.1`
key=`echo $key_value | awk '{print $4}'`
mpls xc add ilm_label gen 1000 ilm_labelspace 1 nhlfe_key $key
```

##LER2##ler2.sh##

```
#!/bin/bash

# Server to Client
echo add label 1000 and forward the packet to lsr on output interface eth1
for destination Client
key_value=`mpls nhlfe add key 0 instructions push gen 1000 nexthop eth1 ipv4
192.168.3.3`
key=`echo $key_value | awk '{print $4}'`
ip route add 192.168.0.4/32 via 192.168.3.3 mpls $key

# Client to Server
echo expect and pop label 1000 from interface eth1
mpls labelspace set dev eth2 labelspace 0
mpls ilm add label gen 1000 labelspace 0
key_value=`mpls nhlfe add key 0 instructions nexthop eth0 ipv4 192.168.4.5`
key=`echo $key_value | awk '{print $4}'`
mpls xc add ilm_label gen 1000 ilm_labelspace 0 nhlfe_key $key
```

##########4. Protection from Link failure

##LER1##ler1.sh##

```
#!/bin/bash

echo Setting MPLS on ler1
modprobe mpls4
echo 'Server to Client'
#pop label 2000 for incoming traffic
mpls labelspace set dev eth2 labelspace 0
mpls labelspace set dev eth1 labelspace 0
mpls ilm add label gen 2000 labelspace 0
key_value=`mpls nhlfe add key 0 instructions nexthop eth0 ipv4 192.168.0.4`
key=`echo $key_value | awk '{print $4}'`
mpls xc add ilm_label gen 2000 ilm_labelspace 0 nhlfe_key $key

echo 'Client to Server default route'
#push 1000 for traffic going to ler2
key_value=`mpls nhlfe add key 0 instructions push gen 1000 nexthop eth2 ipv4
192.168.2.2`
defaultkey=`echo $key_value | awk '{print $4}'`
ip route add 192.168.4.5/32 via 192.168.2.2 mpls $defaultkey

echo 'Client to Server Indirect route'
#push 100 for traffic going to ler2 through lsr
key_value=`mpls nhlfe add key 0 instructions push gen 100 nexthop eth1 ipv4
192.168.1.3`
indirectkey=`echo $key_value | awk '{print $4}'`

#work in a loop
default_route=1
while [ "1" = "1" ] ;
do
  sleep 0.5
  #discover if the link is still up
  status=`ethtool eth2 | grep "Link detected" | grep yes`
  if [ -z "$status" ] ; then
    if [ $default_route = 1 ] ; then
     #need to switch to backup route
     default_route=0 ;
     ip route del 192.168.4.5/32 via 192.168.2.2 mpls $defaultkey
     ip route add 192.168.4.5/32 via 192.168.1.3 mpls $indirectkey
     date +%r
     echo "deleting default route"
     echo "adding backup route"
    fi
  fi

  if [ ! -z "$status" ] ; then
    if [ $default_route = 0 ] ; then
      #need to switch to default route
      default_route=1 ;
      ip route del 192.168.4.5/32 via 192.168.1.3 mpls $indirectkey
      ip route add 192.168.4.5/32 via 192.168.2.2 mpls $defaultkey
      date +%r
```

```
      echo "deleting indirect route"
      echo "adding default route"
    fi
  fi
done
```

##LSR##lsr.sh##

```
#!/bin/bash

# Client to Server
echo expect and pop label 1000 from interface eth1 exchange with 2000 send to
eth0
mpls labelspace set dev eth1 labelspace 0
mpls ilm add label gen 1000 labelspace 0
key_value=`mpls nhlfe add key 0 instructions push gen 2000 nexthop eth0 ipv4
192.168.3.2`
key=`echo $key_value | awk '{print $4}'`
mpls xc add ilm_label gen 1000 ilm_labelspace 0 nhlfe_key $key

# Server to Client
echo expect and pop label 100 from interface eth0 exchange with 200 send to
eth1
mpls labelspace set dev eth0 labelspace 0
mpls ilm add label gen 100 labelspace 0
key_value=`mpls nhlfe add key 0 instructions push gen 200 nexthop eth1 ipv4
192.168.1.1`
key=`echo $key_value | awk '{print $4}'`
mpls xc add ilm_label gen 100 ilm_labelspace 0 nhlfe_key $key
```

##LER2##ler2.sh##

```
#!/bin/bash

# Server to Client
echo add label 100 and forward the packet to lsr on output interface eth1 for
destination Client
key_value=`mpls nhlfe add key 0 instructions push gen 100 nexthop eth1 ipv4
192.168.3.3`
key=`echo $key_value | awk '{print $4}'`
ip route add 192.168.0.4/32 via 192.168.3.3 mpls $key

# Client to Server
echo expect and pop label 2000 from interface eth1
mpls labelspace set dev eth2 labelspace 0
mpls ilm add label gen 2000 labelspace 0
key_value=`mpls nhlfe add key 0 instructions nexthop eth0 ipv4 192.168.4.5`
key=`echo $key_value | awk '{print $4}'`
mpls xc add ilm_label gen 2000 ilm_labelspace 0 nhlfe_key $key
```

#########5. Protection from Node failure

##LER1##ler1.sh##

```
#!/bin/bash

echo Setting MPLS on ler1
modprobe mpls4
echo 'Server to Client'
#pop label 2000 for incoming traffic
mpls labelspace set dev eth2 labelspace 0
mpls labelspace set dev eth1 labelspace 0
mpls ilm add label gen 2000 labelspace 0
key_value=`mpls nhlfe add key 0 instructions nexthop eth0 ipv4 192.168.0.4`
key=`echo $key_value | awk '{print $4}'`
mpls xc add ilm_label gen 2000 ilm_labelspace 0 nhlfe_key $key

echo 'Client to Server default route'
#push 100 for traffic going to ler2 through lsr
key_value=`mpls nhlfe add key 0 instructions push gen 100 nexthop eth1 ipv4
192.168.1.3`
defaultkey=`echo $key_value | awk '{print $4}'`
ip route add 192.168.4.5/32 via 192.168.1.3 mpls $defaultkey

echo 'Client to Server indirect route'
#push 1000 for traffic going to ler2
key_value=`mpls nhlfe add key 0 instructions push gen 1000 nexthop eth2 ipv4
192.168.2.2`
indirectkey=`echo $key_value | awk '{print $4}'`

#work in a loop
default_route=1
while [ "1" = "1" ] ;
do
  sleep 1
  #discover if the link is still up
  status=`ping -c 1 192.168.1.3 >&1 | grep "100% packet loss"`
  if [ ! -z "$status" ] ; then
    if [ $default_route = 1 ] ; then
     #need to switch to backup route
     default_route=0 ;
     ip route del 192.168.4.5/32 via 192.168.1.3 mpls $defaultkey
     ip route add 192.168.4.5/32 via 192.168.2.2 mpls $indirectkey

     date +%r
     echo "deleting default route"
     echo "adding backup route"
    fi
  fi

  if [ -z "$status" ] ; then
    if [ $default_route = 0 ] ; then
      #need to switch to default route
      default_route=1 ;
      ip route del 192.168.4.5/32 via 192.168.2.2 mpls $indirectkey
      ip route add 192.168.4.5/32 via 192.168.1.3 mpls $defaultkey
```

```
      date +%r
      echo "deleting indirect route"
      echo "adding default route"
    fi
  fi
done
```

##LSR##lsr.sh##

```
#!/bin/bash

# Client to Server
echo expect and pop label 1000 from interface eth1 exchange with 2000 send to
eth0
mpls labelspace set dev eth1 labelspace 0
mpls ilm add label gen 1000 labelspace 0
key_value=`mpls nhlfe add key 0 instructions push gen 2000 nexthop eth0 ipv4
192.168.3.2`
key=`echo $key_value | awk '{print $4}'`
mpls xc add ilm_label gen 1000 ilm_labelspace 0 nhlfe_key $key

# Server to Client
echo expect and pop label 100 from interface eth0 exchange with 200 send to
eth1
mpls labelspace set dev eth0 labelspace 0
mpls ilm add label gen 100 labelspace 0
key_value=`mpls nhlfe add key 0 instructions push gen 200 nexthop eth1 ipv4
192.168.1.1`
key=`echo $key_value | awk '{print $4}'`
mpls xc add ilm_label gen 100 ilm_labelspace 0 nhlfe_key $key
```

##LER2##ler2.sh##

```
#!/bin/bash

# Server to Client
echo add label 100 and forward the packet to lsr on output interface eth1 for
destination Client
key_value=`mpls nhlfe add key 0 instructions push gen 100 nexthop eth1 ipv4
192.168.3.3`
key=`echo $key_value | awk '{print $4}'`
ip route add 192.168.0.4/32 via 192.168.3.3 mpls $key

# Client to Server
echo expect and pop label 2000 from interface eth1
mpls labelspace set dev eth2 labelspace 0
```

```
mpls ilm add label gen 2000 labelspace 0
key_value=`mpls nhlfe add key 0 instructions nexthop eth0 ipv4 192.168.4.5`
key=`echo $key_value | awk '{print $4}'`
mpls xc add ilm_label gen 2000 ilm_labelspace 0 nhlfe_key $key
```