

Lab 1

IGP Setup

Overview

In this lab you will:

- Enable OSPF as your IGP (with TE support)
- Save your baseline configuration for use in future labs

Part 1: Configuration Preparation

Step 1

Familiarize yourself with the L3 MPLS VPN diagram in the lab diagram handout. Each group of students will configure a PE router.

Step 2

Verify that your current configuration contains only [system] and [interface] configurations.

Step 3

Log into your PE routers. Perform the following tasks:

In order to start configuring the router, type in 'configure':

```
user@router> configure
user@router#
```

Enable OSPF by placing all (or specific) interfaces into OSPF area 0

```
user@router# set protocols ospf area 0 interface all
user@router# set protocols ospf area 0 interface fxp0 disable
user@router# set protocols ospf traffic-engineering
```

Commit your changes

```
user@router# commit
```

Step 4

Your router is now participating in OSPF with adjacent routers. Verify adjacency formation and route distribution using the following operational commands. Operational commands can be issued from within 'configuration' mode by using the 'run' command.

```
user@router# run show ospf neighbor
user@router# run show route terse
user@router# run show route terse protocol ospf
```

[optional] Your router should now have formed OSPF adjacencies with neighboring routers as shown in the lab diagram. SONET path information can be used to identify connection end-points.

```
user@router > show interfaces so-0/0/1 extensive | find "path trace"
```

Step 6

View the configuration on your PE routers. The OSPF configuration should now be:

```
lab@r1> show configuration protocols
ospf {
  traffic-engineering;
  area 0.0.0.0 {
    interface all;
    interface fxp0.0 {
      disable;
    }
  }
}
```

Lab 2

RSVP/MPLS

Overview

In this lab you will:

- Enable RSVP and MPLS configurations
- Create MPLS LSPs to all other PE routers
- Save your baseline configuration for use in future labs

Part 1: Configuration Preparation

Step 1

Note the loopback addresses of all the other PE routers. These addresses are of the type 192.168.0.X where X represents the router number. For example, if you are working on router 1, you will need to create MPLS LSPs to: 192.168.0.2, 192.168.0.3, 192.168.0.4 etc

Step 2

Enable RSVP and MPLS on all core interfaces. In production networks, you may enable this on per interface basis. However, for this lab, we will enable RSVP and MPLS on all interfaces.

```
user@router# set protocols rsvp interface all
user@router# set protocols rsvp interface fxp0 disable
user@router# set protocols mpls interface all
user@router# commit
```

Step 3

Enable MPLS packet processing on all core facing interfaces. JUNOS supports a template feature which can be used to apply common attributes to a group of objects. In this example, we will use [groups] to apply the 'family mpls' commands to all core SONET interfaces.

```
user@router# set groups SONET interfaces <so-*> unit 0 family mpls
user@router# set apply-groups SONET
user@router# commit
```

Step 4

Create MPLS LSPs to all other PEs.

```

user@router# set protocols mpls label-switched-path R1>R2 to 192.168.0.2
user@router# set protocols mpls label-switched-path R1>R3 to 192.168.0.3
user@router# ...
user@router# ...
user@router# commit

```

[optional] You may want to allow the MPLS LSPs to automatically resize themselves using the auto-bandwidth features. [group] templates can also be used to apply these attributes to each LSP

```

user@router# set protocols mpls statistics file MPLS.stats
user@router# set protocols mpls statistics auto-bandwidth
user@router# set protocols mpls label-switched-path R1>R2 auto-bandwidth

```

Using [groups]

```

lab@r1# show groups
LSP-AUTOBW {
  protocols {
    mpls {
      label-switched-path <*> {
        auto-bandwidth {
          adjust-interval 600;
          adjust-threshold 10;
          adjust-threshold-overflow-limit 3;
        }
      }
    }
  }
}

* apply using # set apply-groups LSP-AUTOBW

```

Step 5

View the state of your MPLS LSPs:

```
lab@r1# run show mpls lsp
```

Use the extensive command to get detailed information about the LSP:

```
lab@r1# run show mpls lsp extensive
```

Real-time statistics can be viewed on a per LSP basis:

```
lab@r1> monitor label-switched-path <lsp-name>
```

Real-time statistics be viewed for individual or all interfaces using:

```
lab@r1> monitor interface traffic
```

Lab 3

MPLS VPNs

Overview

In this lab you will:

- Create the framework for MPLS VPNs
- Peer your PE to the lab route-reflector
- Define a VOICE VPN

Part 1: Configuration Preparation

Step 1

This is the first step in the migration process. We will enable mBGP with a route-reflector so that the PEs can exchange ip-vpn routes without having to peer with every other PE router.

```
user@router# set protocols bgp group IBGP type internal
user@router# set protocols bgp group IBGP local-address 192.168.0.x
user@router# set protocols bgp group IBGP family inet-vpn unicast
user@router# set protocols bgp group IBGP neighbor 192.168.0.20
```

BGP configuration should now resemble the following:

```
lab@r2# show protocols bgp
group IBGP {
    type internal;
    local-address 192.168.0.2;
    family inet-vpn {
        unicast;
    }
    neighbor 192.168.0.20;
}
```

Use AS 65000 as your local-AS number:

```
set routing-options autonomous-system 65000
commit
```

Step 2

Verify that your PE has successfully peered with the route-reflector.

```
lab@r2# run show bgp summary
```

Step 3

Define the route-distinguisher-id under [routing-options] so that JUNOS will automatically generate a route-distinguisher for every VPN. Make this id the loopback of your router.

```
set routing-options route-distinguisher-id 192.168.0.x
```

Commit your changes

```
user@router# commit
```

Step 4

Create a VPN for VoIP and name it 'VOICE'. This VPN will use the target community target:65000:1 to exchange routes with other member PEs:

```
set routing-instances VOICE instance-type vrf
set routing-instances VOICE vrf-target target:65000:1
```

Create a loopback interface for this VPN. This loopback address should represent your PE number. For example, if you are using PE1, then your VPN loopback should be 11.11.11.11/32 and if you are using PE9, your VPN loopback should be 99.99.99.99/32:

```
set interfaces lo0 unit 1 family inet address 11.11.11.11/32
set routing-instances VOICE interface lo0.1
commit
```

Step 6

View the VPN configuration on your PE routers. It should resemble the following:

```
lab@r1# show routing-instances
VOICE {
    instance-type vrf;
    interface lo0.1;
    vrf-target target:65000:1;
}
```

Step 7

View the VOICE VPN routing table. Do you see routes from other member PEs ?

```
lab@r1# run show route terse table VOICE
```

Step 8

Pick a VPN route learnt from another PE and observe that this route is only visible in the VOICE.inet.0 route table. For example, route 22.22.22.22 is only visible in the VOICE.inet.0 table

```
lab@r1# run show route 22.22.22.22

VOICE.inet.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

22.22.22.22/32      *[BGP/170] 02:57:50, localpref 100, from 192.168.0.2
                   AS path: I
                   via so-0/0/1.0, label-switched-path R1>R2
```

Lab 4

Merging Route Tables

Overview

At this point, we have successfully configured a VOICE VPN on all the PE routers. Transit traffic entering a VOICE VPN will use MPLS transport to reach other PE routers. We now need to modify the configuration so that traffic in this VPN can reach routers that don't yet have MPLS capabilities. VPN traffic will have to exit the VPN if there are no MPLS paths to the desired destination. In this lab we need to accomplish the following:

- Merge the VPN route table with the global route table so that IP packets routed via the global IGP can reach destinations in the VPN.
- Allow VPN attached CEs to be reachable from non-VPN sites
- Advertise the VPN routes in the local IGP
- Allow packets to escape the VPN for those destinations that don't participate in MPLS VPNs

Part 1: Configuration Preparation

Step 1

Define a rib-group named ACCESS under routing-options. This rib-group will be referenced from the VPN so that all VPN routes will be visible in inet.0 (the global unicast routing table)

```
user@router# set routing-options rib-groups ACCESS import-rib inet.0
user@router# commit
```

Apply this configuration in the VOICE VPN. If other VPNs are defined, apply this in each VPN that needs access from IP end-nodes:

```
user@router# set routing-instances VOICE routing-options auto-export
family inet unicast rib-group ACCESS
user@router# commit
```

Local VPN routes should now be visible in your local inet.0 routing table. Check to see if your lo0.1 address is visible in inet.0

```
lab@r1# run show route terse 11.11.11.11
inet.0: 24 destinations, 27 routes (24 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

A	Destination	P	Prf	Metric 1	Metric 2	Next hop	AS path
*	11.11.11.11/32	D	0			>1o0.1	
VOICE.inet.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)							
+ = Active Route, - = Last Active, * = Both							
A	Destination	P	Prf	Metric 1	Metric 2	Next hop	AS path
*	11.11.11.11/32	D	0			>1o0.1	

Step 2

Advertise your VPN routes in the global IGP so that non-MPLS routers have access to VPN CE devices. Define a policy that calls out routes in the VPN. This policy can be further modified to be more specific. In this example, we will create a policy named VPN-ROUTES which calls all VPN routes except the default-route (if it exists).

```
set policy-options policy-statement VPN-ROUTES term t1 from instance VOICE
set policy-options policy-statement VPN-ROUTES term t1 from route-filter
0.0.0.0/0 exact
set policy-options policy-statement VPN-ROUTES term t1 then reject
set policy-options policy-statement VPN-ROUTES term t2 from instance VOICE
set policy-options policy-statement VPN-ROUTES term t2 then accept
set policy-options policy-statement VPN-ROUTES term t3 then reject
commit
```

The VPN policy should now resemble the following:

```
lab@r2# show policy-options
policy-statement VPN-ROUTES {
  term t1 {
    from {
      instance VOICE;
      route-filter 0.0.0.0/0 exact;
    }
    then reject;
  }
  term t2 {
    from instance VOICE;
    then accept;
  }
  term t3 {
    then reject;
  }
}
```

Step 3

Export this policy under OSPF.

```
set protocols ospf export VPN-ROUTES
```

Commit your changes and inspect your routing tables. As other PEs activate this configuration, you should see VPN routes appear as OSPF external routes in inet.0.

```
lab@r1> show route terse protocol ospf
```



```
lab@r1> show ospf route extern
```

Step 4

At this point, non-mpls nodes will be able to see routes to the MPLS VPN CE devices. However, the VPN CE nodes will need a way to exit the VPN if they don't see MPLS paths to these destinations. We can accomplish this using a static route with the 'next-table' keyword:

```
set routing-instances VOICE routing-options static route 0.0.0.0/0 next-  
table inet.0  
commit
```

Test this setup by logging into an IP node (ask instructor for this information) and ping your VPN loopback address from there.

This setup now allows for MPLS transport to be used between MPLS VPN PEs while also allowing IP transport between MPLS PEs and IP nodes. The next step would be to clean up all the route-merging configurations assuming that all your network services have been migrated to a VPN architecture

Step 6

Clean the configuration by removing the following:

```
delete routing-instances VOICE routing-options auto-export  
delete routing-instances VOICE routing-options static route 0.0.0.0/0  
delete routing-options rib-groups  
delete protocols ospf export VPN-ROUTES  
delete policy-options policy-statement VPN-ROUTES  
commit
```

Step 7

This concludes the migration procedure.