

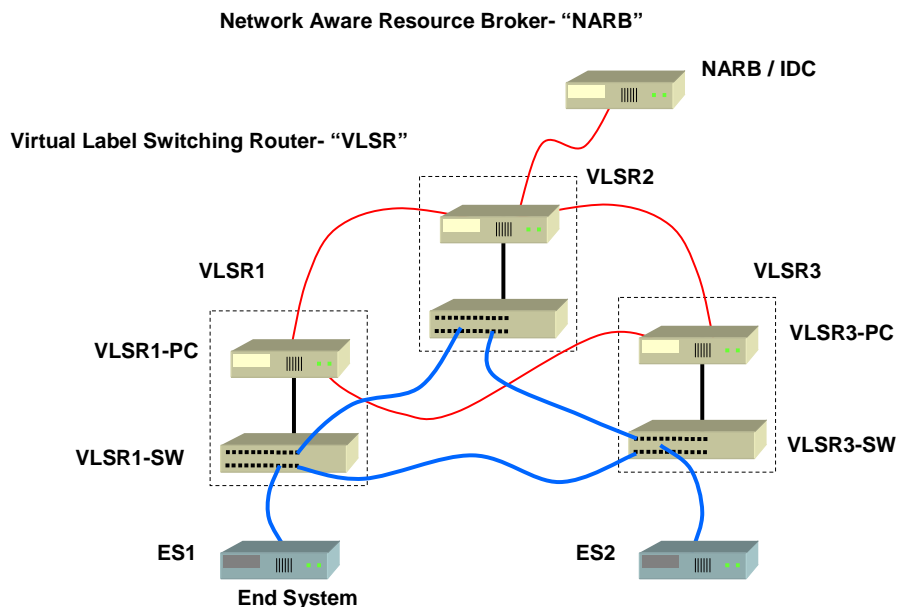
## Exercise #1: Designing a GMPLS Control Plane with VLSR and NARB

Objective: In this exercise, we will layout and configure a GMPLS control plane for a small network.

There are four engineering teams. Each team will be responsible for one network domain, or “pod”. There are four pod domains: Red, Blue, Yellow, and Green with ASNs of 1, 2, 3, and 4 respectively. The networks constructed in each exercise will form the basis for subsequent exercises. Each network pod consists of three Virtual Label Switching Routers (VLSRs), two End Systems (ES), and one machine hosting both the Network Aware Resource Broker (NARB) and Inter-Domain Controller (IDC) software.

### Step 1: Layout a diagram showing the desired data plane topology for the network.

We will be building a network consisting of three Ethernet switches (see diagram below) acting as network elements. Two PCs will act as End Systems. The team will develop a network diagram incorporating route diversity, where the end systems attached to the network, and all the interface and/or port assignments required to support the network.



Each device in the network will require a “management” IP address. For End Systems, this would be a normal IP address for that host. For the network elements, management addresses are used primarily to access the device for configuration. Management interfaces should have publicly routable IPv4 addresses. In this exercise, we will use the management address where a loopback address might be traditionally used for router IDs and such. In general, we use these

management addresses wherever a publicly reachable address for the device would normally be required.

In addition to these management interfaces, each device should have a separate “data plane” interface over which the dynamic circuits will be delivered. The End Systems provided in the workshop lab all have dual gigE ports, eth0 and eth1. In this exercise eth0 will be used for mgmt and control, and eth1 for data plane.

(Note: Each PC has dual 1gigE integrated ethernet ports: eth0 and eth1. We use separate physical interfaces for management+control and data plane. We have arbitrarily designated eth0 for management and control, and eth1 as the data plane interface. The standards do allow for control, management, and data to be provisioned over the same physical interface, but the DRAGON implementation does not yet support this feature.)

In a real network, network elements have a separate “management” address and “loopback” address assigned to each device. The management addresses are used by the network operations to access the devices for configuration and monitoring and are often deliberately not visible to general users. The loopback addresses are used to associate a publicly reachable address with each device that will always be UP. The GMPLS control channels are provisioned over GRE tunnels between network elements. In these exercises the GRE tunnels use the management addresses as the tunnel endpoints. In a real network, the loopback addresses would be more appropriate for tunnel endpoints as the management addresses may not be visible. We have not assigned loopbacks in this workshop in order to simplify the overall addressing, and so we use the management addresses instead.

For these exercises we use the 192.168.0.0/16 subnet for management addressing. We use the 10.0.0.0/8 subnet for control plane addressing. In a real network, the intra-domain control plane links do not strictly need to be public unless they are advertised as part of the inter-domain topology distribution between NARBs. In the black box scenario (minimum internal topology exposed), only the border links need be public addresses as they will be visible to the peering network.

The TE-addresses are largely a matter of convenience. TE-Addresses are not actual IP addresses in the normal IP sense – they are simply a means to associate specific data plane interfaces with their control links. We use the 11.0.0.0/8 net for this purpose in order to provide a numbering scheme congruent with the control plane addressing.

Create a list of Data Plane links. Number them  $D_1$ , to  $D_n$ . Identify the device and port/interface for each end of the data link.

<i>Cid</i>	<i>Device</i>	<i>Int/Port</i>	<i>Device</i>	<i>Int/Port</i>
<b>D1</b>	<b>ES1</b>	<b>eth1</b>	<b>VLSR1-SW</b>	<b>port3</b>
<b>D2</b>	<b>VLSR1-SW</b>	<b>port4</b>	<b>VLSR2-SW</b>	<b>port3</b>
<b>D3</b>				

Go to the lab and install the appropriate Ethernet cabling for the desired data plane connectivity.

**Step 2: Design the control plane architecture to support the GMPLS operations.**

Since each of the Ethernet switches will act as a dynamically provisioned switching element, each switch will need a control PC associated with it to run the OSPF-TE and RSVP-TE protocols. These switch-PC pairs are VLSRs. The control PC is responsible for running the routing and signaling protocols and reconfiguring the associated Ethernet switch as necessary.

Every networked device participating in the dynamically switched network will require a “control link” to be established with neighboring such devices. In GMPLS, these control plane devices may not be physically adjacent to each other. So in order to allow for intervening network infrastructure, we establish GRE tunnels between the neighboring control plane speakers, and assign ip addresses to these tunnel interfaces for the control links.

With respect to the VLSRs, all control plane links are established with the VLSR-PC (not the VLSR-SW). The team should layout the control plane links that will cover the data plane designed in step 1. Each control plane link will require a GRE tunnel be established between logically adjacent network elements (VLSRs and/or NARB/IDC). Since ESs are not running routing and signaling protocols, we don’t need to setup GRE tunnels between VLSRs and ESs. We use the management addresses as the endpoints for building the tunnels.

Each control link will require a /30 subnet be assigned from the appropriate CIDR blocks:

- Red.....10.1.0.0/16
- Blue.....10.2.0.0/16
- Yellow...10.3.0.0/16
- Green....10.4.0.0/16

Tip: In order to keep things manageable, number all the control link uniquely within the domain, and congruently with the data plane links. Then use this numbering scheme to identify which GRE interface to use for each control link. The intra-domain GRE names will include the color of the pod as follows:

Red: RED-GREx  
 Blue: BLUE-GREx  
 Yellow: YLW-GREx  
 Green: GRN-GREx

The GRE number can also be used as part of the control plane subnet as follows:  
 10.<asn>.<gre#>.h

Red: RED-GRE2: 10.1.2.0/30  
 Blue: BLUE-GRE5: 10.2.5.0/30  
 Green: GRN-GRE6: 10.4.6.0/30

(Note: The GRE interface names defined at each end of the control link do *\*not\** need to be the same. However, doing so will reduce confusion and help in debugging config problems.)

Create a list of control plane links. Each control link should include the associated data link, the GRE tunnel # to be used, the IP addresses of the tunnel endpoints, and the ctrl plane IP addresses to be assigned to the GRE interface. You may wish to build this list for each device in the network (rather than by link) in order to facilitate the set up process later:

Red Pod:

Cid	GRE	Device	Local		Remote		
			Ctrl Addr	Tun Addr	Device	Ctrl Addr	Tun addr
D2	RED-GRE2	VLSR1-PC	10.1.2.1	192.168.1.4	VLSR2-PC	10.1.2.2	192.168.1.6
D3	RED-GRE3	VLSR1-PC	10.1.3.1	192.168.1.4	VLSR3-PC	10.1.3.2	192.168.1.8
D4							

### Step 3: Assign Traffic Engineering (TE) Addresses

Traffic Engineering (TE) addresses should be assigned for each data plane interface. TE addresses are not used for IP routing across the interfaces; they simply represent a numbering scheme to uniquely identify data plane interfaces. If the user wished to use IP over a dynamically allocated circuit, the user will need to allocate and assign an IP address to the interface associated with the newly created link. The TE address should not be used for this purpose. For this exercise, and to keep the addressing scheme somewhat humanly intuitive, we use the 11.0.0.0/8 net in a similar fashion to the 10.0.0.0/8 net assignments used for control links:

Red data plane link D2: 11.1.2.0/30  
 Yellow data plane link D4: 11.3.4.0/30

The choice of whether or not to use globally unique addresses depends on whether or not you expect to inter-operate with other domains. Private address ranges, such as those given in RFC1918, can be used within your own domain. However, we recommend the use of global addresses if the long-term intent is to exchange routing information with other operators/domains. For more information, refer to section 2 of RFC3945.

Before continuing, go over the diagram with the workshop instructor and/or the other attendees. There are many ways to design a GMPLS network – the objective is to understand the key components and the process.

#### Step 4: Configure the VLSRs

Now we use a standard network configuration for these workshop exercises. See the “Pod Topology” at the end of Exercise #1. This diagram provides a physical layout in each pod and the excel sheets after the diagram provide detailed addressing scheme for a small GMPLS network. All pod PCs have been pre-loaded with Linux (Debian 3.1), the software dependency packages required for the DRAGON software and/or systems support, and a recent release of the DRAGON GMPLS software suite.

You can login to the machines via SSH using the standard port, 22, in your pod. The username is **root**, password **rootme**. Or, you can optionally login with username **user1...user16** with the password **Workshop!** and then run `sudo` for root access.

We will configure the three VLSRs to reflect this network diagram. To configure a VLSR, there are several steps:

1. On each VLSR-PC, we create the appropriate GRE tunnels and then assign control plane addresses to those GRE interfaces. We have modified the Debian startup process to invoke the `/etc/rc.local` script.

All GRE tunnels in the pod are already up and running. However, it will be a good idea to get familiar with tunnel set up by following the steps as shown below:

Check `/etc/rc.local`. You should find the following `modprobe` commands should be at the top of the file, as well as a call to the `setup_gre_tunnel.sh` script in `/usr/local/dragon/etc`:

```
#!/bin/sh
# needed for GRE tunnel support, e.g. 'ip tunnel'
/sbin/modprobe ip_gre
# needed for tagged VLAN support, e.g. 'vconfig'
/sbin/modprobe 8021q
# run the script that set up the gre_tunnels
/usr/local/dragon/etc/setup_gre_tunnels.sh
```

Note: In the first exercise, circuit setup will be in port-to-port/untagged mode, so the support for tagged VLAN (the ‘8021q’ Linux module) is not necessary, but we will add the support here for exercises that come later.

We provide a script at `/usr/local/dragon/etc/setup_gre_tunnels.sh` on each machine to configure the GRE tunnels.

```
# GRE between red-vlsr1 and red-vlsr2
ip tunnel del red-gre2
ip tunnel add red-gre2 mode gre remote 192.168.1.6 local 192.168.1.4 ttl 255
ip link set red-gre2 up
ip addr add 10.1.2.1/30 dev red-gre2
ip route add 10.1.2.2/32 dev red-gre2
```

2. Then, edit the configuration files for the DRAGON software on **all three VLSR** machines. In the directory `/usr/local/dragon/etc`, copy the sample conf files to working files. Below is an example on `red_vlsr1_pc`:

```
red_vlsr1_pc:~# cd /usr/local/dragon/etc
red_vlsr1_pc:/usr/local/dragon/etc# cp RSVPD.conf.sample RSVPD.conf
red_vlsr1_pc:/usr/local/dragon/etc# cp zebra.conf.sample zebra.conf
red_vlsr1_pc:/usr/local/dragon/etc# cp ospfd.conf.sample ospfd.conf
red_vlsr1_pc:/usr/local/dragon/etc# cp dragon.conf.sample dragon.conf
```

Edit the `dragon.conf` file and the `zebra.conf` file. For now, these files only contain the hostname assignment for this VLSR and the login password. For simplicity sake, we set the password to “dragon”.

Next, edit the `RSVPD.conf` file. In this file, for now, all we need to do is to define each GRE interface RSVP will be using for signaling. For example, `red_vlsr1_pc` will have two GRE interfaces: `red-gre2` (to VLSR2) and `red-gre3` (to VLSR3). Add the following commands to `RSVPD.conf`:

```
interface red-gre2 tc none mpls
interface red-gre3 tc none mpls
```

Next, edit the `dragon.conf` file. For now, all we want to do is to change the hostname to reflect the name of the VLSR.

```
hostname red-vlsr1-pc
password dragon
```

Finally, edit the `ospfd.conf` file. OSPF must be told which control links to listen to and/or flood updates, and it must also be given switchport mappings in order to perform path computation and manage the link state associated with each link.

It is in the `ospfd.conf` file where we specify the switch ip address and capacity associated with each class of service priority. The following text is excerpted from one VLSR. The student should be able to review this configuration and make any appropriate changes for the other control links on this VLSR:

```

! *- ospf *-
!
! OSPFd sample configuration file
!
hostname red-vlsr1-ospf
password dragon
enable password dragon
log stdout
log file /var/log/ospfd.log
!
!
interface red-gre2
description GRE tunnel between red-vlsr1 and red-vlsr2
ip ospf network point-to-point
!
interface red-gre3
description GRE tunnel between red-vlsr1 and red-vlsr3
ip ospf network point-to-point
!
router ospf
ospf router-id 192.168.1.4
network 10.1.2.2/30 area 0.0.0.0
network 10.1.3.2/30 area 0.0.0.0
ospf-te router-address 192.168.1.4
ospf-te interface red-gre2
    level gmpls
    data-interface ip 11.1.2.1 protocol snmp switch-ip 192.168.1.3 switch-port 4
    swcap l2sc encoding ethernet
    max-bw 125000000
    max-rsv-bw 125000000
    max-lsp-bw 0 125000000
    max-lsp-bw 1 125000000
    max-lsp-bw 2 125000000
    max-lsp-bw 3 125000000
    max-lsp-bw 4 125000000
    max-lsp-bw 5 125000000
    max-lsp-bw 6 125000000
    max-lsp-bw 7 125000000
    vlan 100 to 200
    metric 10
    exit
ospf-te interface red-gre3
    level gmpls
    data-interface ip 11.1.3.1 protocol snmp switch-ip 192.168.1.3 switch-port 5
    swcap l2sc encoding ethernet
    max-bw 125000000
    max-rsv-bw 125000000
    max-lsp-bw 0 125000000
    max-lsp-bw 1 125000000
    max-lsp-bw 2 125000000
    max-lsp-bw 3 125000000
    max-lsp-bw 4 125000000
    max-lsp-bw 5 125000000
    max-lsp-bw 6 125000000
    max-lsp-bw 7 125000000
    vlan 100 to 200
    metric 10
    exit

```

3. Login to the covered Ethernet switch to check the configuration.

After SSH'ing to a VLSR PC, execute “telnet <switch\_ip>” and login with the username **admin** and password **admin**.

The switch IP addresses are 192.168.x.3, 192.168.x.5, and 192.168.x.7 for the switches covered by VLSR1, VLSR2 and VLSR3, respectively.

Now that the control plane software is configured, the switch must be configured to match the features configured in the GMPLS \*.conf files. Every switch is different in how they handle VLANs, spanning tree, MTUs, even whether or not they support SNMP. For the workshop, we have provided Dell PowerConnect 5324 Ethernet switches. We need to setup the proper management IP address, set the SNMP read/write community string, and initialize ports that will be under VLSR control. We want to disable spanning tree protocol and filter BPDU packets on all the VLSR ports in order to prevent STP flooding. For these Dell switches in the workshop configuration, we want ports 1 and 2 to be in VLAN #1 – this VLAN will serve as a management interface. Ports g3 through g24 will be available for switching.

Note that we have already pre-loaded the switches with the correct configuration. The following CLI commands were used to initialize the Dell PowerConnect 5324 switches for their role in these workshop networks – other switches will differ slightly in their initial configuration:

```
red_vlsr1_sw# configure
red_vlsr1_sw(config)# vlan database
red_vlsr1_sw(config-vlan)# vlan 100-200
red_vlsr1_sw(config-vlan)# exit
red_vlsr1_sw(config)# no spanning-tree
red_vlsr1_sw(config)# spanning-tree bpdu filtering
red_vlsr1_sw(config)# port jumbo-frame
red_vlsr1_sw(config)# interface range ethernet g(3-24)
red_vlsr1_sw(config-if)# switchport mode general
red_vlsr1_sw(config-if)# spanning-tree disable
red_vlsr1_sw(config-if)# switchport general pvid 2
red_vlsr1_sw(config-if)# exit
red_vlsr1_sw(config)# interface range ethernet g(1-2)
red_vlsr1_sw(config-if)# switchport mode general
red_vlsr1_sw(config-if)# switchport general pvid 1
red_vlsr1_sw(config-if)# exit
red_vlsr1_sw(config)#
red_vlsr1_sw(config)# logging console debugging
red_vlsr1_sw(config)# enable password level 15 admin
red_vlsr1_sw(config)# username admin password admin level 15
red_vlsr1_sw(config)# snmp-server community dragon rw
red_vlsr1_sw(config)# exit
red_vlsr1_sw#
```

The teams should login to the VLSR switches to peruse the configuration and to display the port and vlan mappings:

To show the port and VLAN assignments, execute the following command on the switch:

```
red_vlsr1_sw# show vlan
```

Vlan	Name	Ports	Type	Authorization
1	1	g(1-2),ch(1-8)	other	Required
100	100		permanent	Required
101	101		permanent	Required
102	102		permanent	Required
103	103		permanent	Required
...	...		permanent	Required
200	200		permanent	Required

We have VLANs 100-200 created and ready to be used – they are empty and should have no ports as members. The Dell PowerConnect switches require that empty VLANs exist in-advance because their particular implementation of RFC2674 (Q-BRIDGE-MIB) does not support creation of new VLANs via SNMP.

To show the entire switch configuration, run this command:

```
red_vlsr1_sw# show running-config
```

At first glance, you may realize that the running-config is different from the default configuration that shown above. Unfortunately, this is just the way Dell PowerConnect switches arrange the output of the `show running-config` command.

## Step 5: Configure the NARB

The NARB provides both intra-domain service routing functions. The path computation element (PCE) is used to create an Explicit Route Object (ERO) for the RSVP PATH message. This ERO specifies the path that the Label Switched Path (LSP) is to take with a domain. Topology information for the local domain may be provided in the `narb.conf` file, but it is not required for this workshop. This file must be edited to describe the linkage to the local OSPF module.

In this exercise, we will implement the NARB in order to provide advanced path computation capabilities on an intra-domain basis. While the intra-domain case will not take full advantage of the NARB's capabilities, its presence will satisfy requirements to provide EROs for certain types of provisioning requests...most notably the "local ID" mode that we will implement in the next step.

Procedure:

Setting up the NARB actually consists of configuring two functional entities: 1) an intra-domain OSPF listener and 2) the NARB itself. Traditionally, the NARB and all of its component daemons are run on their own host. While not strictly necessary, the path computation algorithms and link state processing in a large network could consume significant computational resources – perhaps more than a typical network element might have available.

In this exercise, we will run the NARB processes on a separate host. We must therefore allocate and set up appropriate GRE tunnels. One GRE tunnel must be established from the NARB server to one of the intra-domain VLSRs. Any VLSR will work and we have picked VLSR2 to connect to the NARB in each pod – this link will be configured in ospfd to flood LSAs to the NARB from inside the domain.

Login to the NARB PC at 192.168.x.10 via SSH.

- 1) In the directory `/usr/local/dragon/etc`, copy these sample conf files to working files. Below is an example on `red_vlsr1_pc`:

```
red-narb:~# cd /usr/local/dragon/etc
red-narb:/usr/local/dragon/etc# cp narb.conf.sample narb.conf
red-narb:/usr/local/dragon/etc# cp rce.conf.sample rce.conf
red-narb:/usr/local/dragon/etc# cp schema_combo.rsd.sample schema_combo.rsd
red-narb:/usr/local/dragon/etc# cp ospfd-intra.conf.sample ospfd-intra.conf
red-narb:/usr/local/dragon/etc# cp zebra.conf.sample zebra.conf
```

We will not be using the NARB in inter-domain mode, so just create an empty `ospfd-inter.conf` file:

```
red-narb:/usr/local/dragon/etc# touch ospfd-inter.conf
```

- 2) Edit the `/usr/local/dragon/etc/ospfd-intra.conf` file

In this file, we need to describe a passive adjacency between the intra-domain narb ospfd instance and any one of the VLSR ospf instances within the local domain. The following statements make up the red domain `ospf-intra.conf` file:

```
hostname red-narb
password dragon
enable password dragon
log stdout
log file /var/log/ospfd.log
interface red-gre6
  description GRE tunnel red-gre6
  ip ospf network point-to-point
router ospf
  ospf router-id 192.168.1.10
  network 10.1.6.0/30 area 0.0.0.0
  ospf-te router-address 192.168.1.10
```

**Note: There is no TE LINK between the NARB and VLSR.**

- 3) Edit the `narb.conf` file.

We must define the local domain ID – we use the management CIDR block, but any unsigned integer could be used (e.g. 1 for red, 3 for yellow, etc). We also need to specify the linkage to the intra-domain OSPF instances.

Below is an example for the red pod's narb.conf file:

```
domain-id { ip 192.168.1.0 }
cli { host red-narb password dragon }
intra-domain-ospfd { address localhost port 2617 originate-interface 10.1.6.2 area 0.0.0.0 }
```

- 4) Before starting the NARB, please also login to the VLSR with the adjacency to the NARB, i.e. VLSR2 in our case. Edit the file `ospfd.conf` so that the ospfd at vlsr2 can pick up `gre6` to form an adjacency to the intra-domain ospfd on the NARB. Again, remember, there is no TE-link between the NARB and the VLSR2, so you will not enter any

```
ospf-te interface <color>-gre12
level gmpls
```

- 5) Also, login to all VLSRs in your pod. Now that the NARB is configured as part of the local control plane, we need to tell the dragond daemons where to find the NARB

Add the following lines to `/usr/local/dragon/etc/dragon.conf`:

```
configure narb intra-domain ip-address 192.168.1.10 port 2609
set narb-extra-options query-with-confirmation
set narb-extra-options query-with-holding
```

- 6) Start the DRAGON software on the VLSR PC.

The final step is to start all of the DRAGON protocol agents. To start the VLSR daemons, use the following shell command on the VLSR PC:

```
red_vlsr1_pc:~# /usr/local/dragon/bin/dragon.sh start-vlsr
red_vlsr1_pc:~# ps auxwww | grep dragon
```

This command starts `zebrad`, `ospfd`, `RSVPd`, and `dragond` and places them in the background. Note: `dragon.sh` is used to start, restart, stop, status all of the protocol daemons. Invoke it without any command line parameters to get a list of options.

After starting the daemons, we want to verify that the protocols are talking properly. For example, when ospfd starts up, it will try to issue HELLO exchange on the configured GRE links. Adjacencies should be formed and link state announcements (LSAs) flooded across to neighbors to build the link state database, i.e. the intra-domain topology of the network. You can query the status of these ospf adjacencies by telnetting to the ospfd CLI port (2604): (By default, the password for logging into the dragon CLI is “dragon” unless you have overridden this in the configuration files.)

```
red_vlsr1_pc:~# telnet localhost 2604
password> *****
red_vlsr1-ospf> sh ip ospf interface
red_vlsr1-ospf> sh ip ospf-te database detail
red_vlsr1-ospf> sh ip ospf neighbor
red_vlsr1-ospf> exit
```

By default, the protocol daemons will write to log files in `/var/log`. You can inspect these files to get a more detailed understanding of the state of the protocols or interfaces.

Go back to the shell, and run

```
red_vlsr1-pc:~# less /var/log/ospfd.log
red_vlsr1-pc:~# less /var/log/RSVPD.log
```

7) Make sure the VLSRs in the local domain are all up and active. Then start the NARB with:

```
red-narb:~# /usr/local/dragon/bin/dragon.sh start-narb
...
#####
DRAGON NARB Started...
#####
NARB_OUTPUT @[2008/01/24 18:44:56] : Running without connection to OSPFd.....
NARB_OUTPUT @[2008/01/24 18:44:56] : No abstract topology generated.....
dragon-sw: started narb daemons.
red-narb:~#
```

You will see a message saying that the NARB is running without connection to OSPFd. The OSPFd that's being mentioned is the inter-domain OSPFd. Since we are not using the NARB for inter-domain provisioning, we can ignore that message.

The CLI ports associated with NARB elements are as follows:

- 2604 ospfd-intra-domain CLI
- 2626 NARB CLI
- 2688 RCE CLI

You can telnet to these processes and inspect the OSPF adjacencies and topology information.

To check intra-domain topologies at RCE CLI:

```
red-narb:~# telnet localhost 2688
rce:cli>show topology intradomain
.....Router ID Opaque LSA.....
Adv_router (192.168.1.4), Router_id (192.168.1.4)
Adv_router (192.168.1.6), Router_id (192.168.1.6)
Adv_router (192.168.1.8), Router_id (192.168.1.8)
.....TE Link Opaque LSA.....
Adv_router (192.168.1.4), Link_id (192.168.1.6), IfAddrs[11.1.2.1-11.1.2.2]
Adv_router (192.168.1.4), Link_id (192.168.1.8), IfAddrs[11.1.3.1-11.1.3.2]
Adv_router (192.168.1.6), Link_id (192.168.1.4), IfAddrs[11.1.2.2-11.1.2.1]
Adv_router (192.168.1.6), Link_id (192.168.1.8), IfAddrs[11.1.4.1-11.1.4.2]
Adv_router (192.168.1.8), Link_id (192.168.1.4), IfAddrs[11.1.3.2-11.1.3.1]
Adv_router (192.168.1.8), Link_id (192.168.1.6), IfAddrs[11.1.4.2-11.1.4.1]
.....The End.....
rce:cli>
```

## Step 6: Configure the Local-ID

The “Local ID” is a non-standard construct developed by DRAGON to facilitate circuit creation from one VLSR edge port to another VLSR edge port. This feature can be used to allow a VLSR to proxy for an otherwise RSVP unaware device – e.g. a video camera, or computational cluster, etc. The DRAGON

software requires root access to run the RSVP and OSPF protocols because of the need to access raw sockets, so it is convenient to be able to create a circuit without directly involving the end systems in the control plane. In the R&E networks, it is expected that Ethernet LSPs will often be used to link clusters in one location to clusters in another location – mapping a group of ports on one switch with a group of ports on some other switch. Since RSVP does not support VLANs as a termination point, the DRAGON software provides for the creation of a “local ID” on a local switch which can then be used as a sub-object for terminating a PATH request at the local switch. Via .conf file and/or CLI, a single [dumb] port, a group of ports, or a tagged group of ports can be set up and associated with a local ID. This local ID can then be specified in the LSP configuration at the source and destination.

In order to use local ID, there must be an active NARB available and configured in the dragon.conf files at the source and destination VLSRs. As a general rule, pointers to the NARB should be configured on all VLSRs throughout the network.

While this is a non-standard feature incorporated into the DRAGON software, it provides a capability to link clustered systems on one switch to another set of clustered systems on another switch. The “local ID” is a terminating entity within a VLSR, allowing the signaling protocol to terminate the LSP to an internally defined VLAN port group. In future releases, this feature should most likely be implemented as a point-to-multipoint LSP, but the mechanisms for path selection and routing for such LSPs is an advanced topic and is not currently implemented.

Procedure:

1) A useful tool: `/usr/local/dragon/sbin/narb_test`

`narb_test` is a tool to query the NARB for an ERO. We use this tool often to test if a host is reachable to another host.

For example, from `red-es1`, we can query `red-narb` (192.168.1.10) if `red-vlsr1` can reach `red-vlsr3` as shown below, which VLAN tags are available, and the first available end-to-end (E2E) VLAN tag:

```
red_vlsr1_pc:~# narb_test -H 192.168.1.10 -S 192.168.1.4 -D 192.168.1.8 -V -a
NARB@[2008/01/23 02:16:47] : Request successful! ERO returned...
NARB@[2008/01/23 02:16:47] : HOP-TYPE [strict]: 11.1.3.1 [UnumIfId: 262244(4,100)]
NARB@[2008/01/23 02:16:47] : HOP-TYPE [strict]: 11.1.3.2 [UnumIfId: 262244(4,100)]
NARB@[2008/01/23 02:16:47] : E2E VLAN TAG [ 100 ]
NARB@[2008/01/23 02:16:47] : ALL E2E VLAN TAGS: 100 101 102 103 104 105 106 107 108
109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129
130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150
151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171
172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192
193 194 195 196 197 198 199 200
```

2) The local ID constructs must be defined before they can be used to establish an LSP. You can configure the local-id via the CLI, or you can define them in the `*.conf` files.

Edit the `dragon.conf` file in **VLSR1** and **VLSR3** to define several local-ids to enable port, group, and tagged group LSPs to/from VLSR1 and VLSR3. The following statements will do so:

```
set local-id port 3
set local-id group 1000 add 3
set local-id tagged-group 150 add 3
```

You have to restart dragon daemon for these statement to take into effect (on each VLSR, execute `dragon.sh restart-vlsr`). Or, you can enter these statements directly into the dragon CLI (“telnet [color]-vlsr[1,3] 2611”) without restarting dragon.

Once you have configured a number of local-ids for general use, or several specific local-ids, you are ready to login to the dragon CLI on VLSR1 (or VLSR3) and edit an LSP. When using the tagged-group local-id, the vtag must match add both source and destination (this is fundamentally an issue associated with flat VLANs – the vtag is not actually swapped at each node and so must be maintained end-to-end. There are some emerging scenarios where this constraint will be reduced or eliminated, but the standards and generally available layer2 products do not support this capability yet.)

3) Circuit creation – setting up LSPs

Finally...It is time to actually configure and provision the LSP. For this first exercise, we will be using the DRAGON CLI to instantiate the LSP. This is a two step process: 1) define and edit the LSP and its parameters, 2) “commit” the LSP to place it into service. Afterward, we’ll explore some of the ways to inspect the status of an LSP or to debug issues in the control plane protocols.

Login to the vlsr1 dragon CLI on port 2611 and configure the LSP:

```
red_vlsr1_pc:~# telnet localhost 2611
Password: *****

red-vlsr1-dragon> edit lsp test1
red-vlsr1-dragon(edit-lsp-test1)# set source ip-address 192.168.1.4 port 3
destination ip-address 192.168.1.8 port 3
red-vlsr1-dragon(edit-lsp-test1)# set bandwidth eth100M swcap 12sc encoding
ethernet gpid ethernet
red-vlsr1-dragon(edit-lsp-test1)# exit
```

```
red-vlsr1-dragon> show lsp
**LSP status summary**

Name          Status      Dir   Source (IP/LSP ID)  Destination (IP/Tunnel ID)
-----
test1         Edit        =>    192.168.1.4         192.168.1.8
                3                3
```

Now, commit the LSP to put it into service:

```
red-vlsr1-dragon> commit lsp test1
```

The LSP will be set between red-vlsr1-sw port 3 to red-vlsr3-sw port 3.

You can check the status of the LSP with:

```
red-vlsr1-dragon> show lsp
**LSP status summary**

Name          Status      Dir   Source (IP/LSP ID)  Destination (IP/Tunnel ID)
-----
test1         In service  =>    192.168.1.4         192.168.1.8
                3                3
```

You can see a bit more detail about the LSP by running the command `show lsp test1` in the DRAGON CLI.

It is important to understand that while the LSP has been set up and is operational, the interface on the End System(s) still needs some additional work. The easiest way to take advantage of the point to point layer2 path that has been established is to configure it as an IP interface. **The LSP is in port-to-port/untagged mode, so the data plane interface is eth1 on ES1 and ES2.** The teams can choose any available IP subnet for the link, ifconfig it, and then ping across it.

Login to es2:

```
red_es2_pc:~# ifconfig eth1 10.0.0.2 netmask 255.255.255.252
```

Login to es1:

```
red_es1_pc:~# ifconfig eth1 10.0.0.1 netmask 255.255.255.252
red_es1_pc:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.070 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.050 ms
...
```

As we stated before, the TE address associated with the interface is NOT useable for IP packet forwarding. It is not actually configured on any interface. It is only configured in the control plane protocols as a means of identifying the paired interfaces/ports of the data plane topology.

Finally, delete the LSP by running the following command in the DRAGON CLI:

```
red_vlsr1_pc:~# telnet localhost 2611
Password: *****

red-vlsr1-dragon> delete lsp test1
```

If you kept the ping running from the previous step, you should be able to see the pings stop when you execute the `delete lsp` command. We would also recommend that you login to your Ethernet switches and run the `show vlan` command to see the VLAN port memberships changing during the LSP setup and teardown process.

After you are done with your link, it's always a good practice to delete the IP address from your system.

```
red_es1_pc:~# ifconfig eth1 0.0.0.0
```

## Exercise #2 Intra-domain Provisioning with OSCARS

Objective 1: Configure OSCARS over the GMPLS control and data plane

Objective 2: Reserve and provision Intra-domain link using OSCARS

Discussion:

From the previous exercises, we have constructed several independent networks each capable of dynamically provisioning LSPs across their respective domains. In this exercise, we will introduce OSCARS which serve as inter-domain agent in later exercises.

In the past, we used the NARB to provide the key components for the intra- and inter- domain topology distribution and path computation. The NARB also provided a rudimentary but important capability to manage the topology that each network advertises to its peer networks. As we tried to deploy the Dynamic Circuit Network (DCN) in real network, we would like to incorporate authentication and scheduling capabilities to DCN, so we chose to use OSCARS for these purposes. While the NARB uses the OSPF LSAs to construct the real time topology in each domain and propagate that information to its peers, OSCARS currently requires you to manually generate an XML file that describes your network's topology in the Open Grid Forum (OGF) Network Measurement Working Group (NMWG) control plane topology schema. We are working on the translation between the OSPF LSAs and XML so that OSCARS can have a dynamic topology feed from the NARB directly.

In this exercise, we have already pre-installed OSCARS in each pod. Please refer to the appendix, DCN Software Suite v0.2: OSCARS inter-Domain Controller (IDC) Installation Guide, for how to install OSCARS.

**Procedure:**

**Step 1: First, we can verify the installation of different components in OSCARS**

**Please SSH to the NARB (192.168.x.10) in your domain using the following login information:**

**Username: tomcat55**

**Password: dragon**

It is very important that you follow the remaining steps while logged in as user `tomcat55` because special environment variables have been defined in `/home/tomcat55/.bash_profile` which are necessary for correct operation of tomcat.

1) Start tomcat

```
/usr/local/tomcat/bin/startup.sh
```

Then, visit <https://192.168.x.10:8443>

If installation was successful, a web page will load with the Tomcat logo and a message that reads “If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!”

## 2) Verifying IDC Installation

Visit <https://192.168.x.10:8443/axis2/axis2-admin/>, and login with

Login Name: admin

Password: axis2

## 3) Verifying IDC Web User Interface (WBUI) Installation

Make sure you can visit open URL: <https://192.168.x.10:8443/OSCARS>.

You will not be able to login yet. We will login to the OSCAR in Step 5 after we configure OSCARS.

Please note: Internet Explorer is **NOT** currently supported by OSCARS – you will not be able to get past the login page if you are using Internet Explorer.

## Step 2: Now, let’s move on to describe your topology

In this exercise, we are going to schedule and provision intra-domain LSP, so we will need to describe the intra-domain topology for OSCARS. We have already put a working copy of this file at

```
/usr/local/tomcat/shared/classes/terce.conf/tedb-intra.xml
```

Take a look at this file and see how it describes the topology in your pod. The following is a block in that file that describes a single node, blue-vlsr1.

```
<?xml version="1.0" encoding="UTF-8"?>
<topology
  xmlns="http://ogf.ogf/schema/network/topology/ctrlPlane/20071023/"
  id="blue-topology">
  <idcId>https://idc.blue.net:8443/axis2/services/OSCARS</idcId>
  <domain id="urn:ogf:network:domain=blue.net">
    <!-- BLUE VLSR1 -->
    <node id="urn:ogf:network:domain=blue.net:node=vlsr1">
      <address>192.168.2.4</address>
      <!-- intradomain link to blue-es1 -->
      <port id="urn:ogf:network:domain=blue.net:node=vlsr1:port=3">
        <capacity>1000000000</capacity>
        <maximumReservableCapacity>1000000000</maximumReservableCapacity>
        <minimumReservableCapacity>100000000</minimumReservableCapacity>
        <granularity>100000000</granularity>
```

```

        <link id="urn:ogf:network:domain=blue.net:node=vlsr1:port=3:link=*">
<remoteLinkId>urn:ogf:network:domain=*:node=*:port=*:link=*</remoteLinkId>
    <trafficEngineeringMetric>100</trafficEngineeringMetric>
    <capacity>1000000000</capacity>
    <maximumReservableCapacity>1000000000</maximumReservableCapacity>
    <minimumReservableCapacity>1000000000</minimumReservableCapacity>
    <granularity>100000000</granularity>
    <SwitchingCapabilityDescriptors>
        <switchingcapType>l2sc</switchingcapType>
        <encodingType>ethernet</encodingType>
        <switchingCapabilitySpecificInfo>
            <interfaceMTU>9000</interfaceMTU>
            <vlanRangeAvailability>0,100-200</vlanRangeAvailability>
        </switchingCapabilitySpecificInfo>
    </SwitchingCapabilityDescriptors>
    </link>
</port>

    <!-- intradomain link to blue-vlsr2 -->
<port id="urn:ogf:network:domain=blue.net:node=vlsr1:port=4">
    <capacity>1000000000</capacity>
    <maximumReservableCapacity>1000000000</maximumReservableCapacity>
    <minimumReservableCapacity>1000000000</minimumReservableCapacity>
    <granularity>100000000</granularity>

    <link id="urn:ogf:network:domain=blue.net:node=vlsr1:port=4:link=11.2.2.1">
<remoteLinkId>urn:ogf:network:domain=blue.net:node=vlsr2:port=3:link=11.2.2.2</remoteLinkId>
    <trafficEngineeringMetric>100</trafficEngineeringMetric>
    <capacity>1000000000</capacity>
    <maximumReservableCapacity>1000000000</maximumReservableCapacity>
    <minimumReservableCapacity>1000000000</minimumReservableCapacity>
    <granularity>100000000</granularity>
    <SwitchingCapabilityDescriptors>
        <switchingcapType>l2sc</switchingcapType>
        <encodingType>ethernet</encodingType>
        <switchingCapabilitySpecificInfo>
            <interfaceMTU>9000</interfaceMTU>
            <vlanRangeAvailability>0,100-200</vlanRangeAvailability>
        </switchingCapabilitySpecificInfo>
    </SwitchingCapabilityDescriptors>
    </link>
</port>

    <!-- intradomain link to blue-vlsr3 -->
<port id="urn:ogf:network:domain=blue.net:node=vlsr1:port=5">
    <capacity>1000000000</capacity>
    <maximumReservableCapacity>1000000000</maximumReservableCapacity>
    <minimumReservableCapacity>1000000000</minimumReservableCapacity>
    <granularity>100000000</granularity>

    <link id="urn:ogf:network:domain=blue.net:node=vlsr1:port=5:link=11.2.3.1">
<remoteLinkId>urn:ogf:network:domain=blue.net:node=vlsr3:port=5:link=11.2.3.2</remoteLinkId>
    <trafficEngineeringMetric>100</trafficEngineeringMetric>
    <capacity>1000000000</capacity>
    <maximumReservableCapacity>1000000000</maximumReservableCapacity>
    <minimumReservableCapacity>1000000000</minimumReservableCapacity>
    <granularity>100000000</granularity>
    <SwitchingCapabilityDescriptors>
        <switchingcapType>l2sc</switchingcapType>
        <encodingType>ethernet</encodingType>

```

```

        <switchingCapabilitySpecificInfo>
            <interfaceMTU>9000</interfaceMTU>
            <vlanRangeAvailability>0,100-200</vlanRangeAvailability>
        </switchingCapabilitySpecificInfo>
    </SwitchingCapabilityDescriptors>
</link>
</port>
</node>
</domain>
</topology>

```

### Step 3: Check static-routes.xml

We have already put a working copy of this file at

```
/usr/local/tomcat/shared/classes/terce.conf/static-routes.xml
```

The following is a block in that file on green-narb that describes the link between green-es1 to red-es1.

```

<staticPathEntry id="green-es1-red-es1">
  <srcEndpoint>urn:ogf:network:domain=green.pod.lan:node=vlsr1:port=3:link=*</srcEndpoint>
  <destEndpoint>urn:ogf:network:domain=red.pod.lan:node=vlsr1:port=3:link=*</destEndpoint>
  <path id="green-es1-red-es1">
    <hop id="1">
      <linkIdRef>urn:ogf:network:domain=green.pod.lan:node=vlsr1:port=3:link=*</linkIdRef>
    </hop>
    <hop id="2">
      <linkIdRef>urn:ogf:network:domain=green.pod.lan:node=vlsr1:port=7:link=11.1.11.2</linkIdRef>
    </hop>
    <hop id="3">
      <linkIdRef>urn:ogf:network:domain=red.pod.lan:node=vlsr3:port=7:link=11.1.11.1</linkIdRef>
    </hop>
    <hop id="4">
      <linkIdRef>urn:ogf:network:domain=red.pod.lan:node=vlsr1:port=3:link=*</linkIdRef>
    </hop>
  </path>
  <availableVtags></availableVtags>  <!-- deprecated: leave blank -->
</staticPathEntry>

```

### Step 4: Configure the TERCE

The TERCE is a web service that acts as the topology exchange and route computation element for OSCARS. In the future, it will act as the intermediary between OSCARS and the NARB, but only static topology description and route files are supported in this release. To ensure the TERCE properties file can locate the static topology and route files in your domain, it must be edited:

Edit `/usr/local/tomcat/shared/classes/terce.conf/terce-ws.properties`

Change the properties file to look like the following (where *location-of-your-topology-file* is replaced with the custom path to your topology file and likewise for *location-of-your-static-routes-file*).

```

#static tedb properties
tedb.type=static
tedb.static.db.interdomain=location-of-your-topology-file
tedb.static.db.intradomain=location-of-your-topology-file

```

```
#static rce properties
rce.type=static
```

```
rce.static.file=location-of-your-static-routes-file
```

For now, please have the interdomain and intradomain db files point to  
/usr/local/tomcat/shared/classes/terce.conf/tedb-inter.xml and  
/usr/local/tomcat/shared/classes/terce.conf/tedb-intra.xml, respectively.

## Step 5: Configure OSCARS

The `oscars.properties` file is the main area in which the OSCARS IDC retrieves installation-specific settings. These include settings for accessing the MySQL database, AAA, the perfSONAR Lookup Service, interacting with DRAGON, and more. The `oscars.properties` file is located on your system at:

```
/usr/local/tomcat/shared/classes/server/oscars.properties
```

Your installation comes with a default `oscars.properties` file. Please edit the default file as the following.

```
# login name and password used by the OSCARS server to log into the MySQL
# database
hibernate.connection.username=oscars
hibernate.connection.password=oscars
```

```
### OSCARS sections ###
```

```
### AAA configuration
aaa.salt=os
```

```
# cookie settings
aaa.userName=oscars
aaa.sessionName=oscarssess
# whether cookie has to be sent over SSL; change this once you have SSL
# set up
aaa.secureCookie=1
```

```
### pathfinder configuration
# can be either 0 (don't find path) or 1
pathfinder.findPath=1
```

```
# currently can be traceroute or terce
pathfinder.pathMethod=terce
```

```
#TEDB configuration
tedb.tedbMethod=terce
```

```
#TERCE configuration
terce.url=http://127.0.0.1:8080/axis2/services/TERCE
```

```
##perfSONAR Lookup Service configuration
```

```
lookup.url=http://idc.green.pod.lan:8010/perfSONAR_PS/services/LS
```

```
#logging configuration  
logging.rsvlogdir=/usr/local/tomcat/logs
```

```
## PSS configuration  
pss.method=dragon  
pss.dragon.password=dragon  
pss.dragon.ssh.portForward=1  
pss.dragon.ssh.user=oscars  
pss.dragon.ssh.key=/home/tomcat55/.ssh/id_rsa  
pss.dragon.remotePort=2611  
pss.dragon.vlsr1=127.0.0.1  
pss.dragon.vlsr1.ssh=192.168.1.4  
pss.dragon.vlsr2=127.0.0.1  
pss.dragon.vlsr2.ssh=192.168.1.6  
pss.dragon.vlsr3=127.0.0.1  
pss.dragon.vlsr3.ssh=192.168.1.8
```

```
mail.webmaster=admin@red.pod.lan
```

```
#Other properties  
mail.userAdd.subject=OSCARS user added
```

Now, restart tomcat.

```
/usr/local/tomcat/bin/shutdown.sh  
/usr/local/tomcat/bin/startup.sh
```

Then, you should be able to login to the OSCARS-admin page with

Username: oscars-admin  
Password: oscars

## Step 6: Populate the MySQL Database

The final step is to import the topology information from your XML file into the OSCARS database so that it can keep track of which resources that are being used on the network. This is done by defining your local domain in the database and running `updateTopology.sh`.

### 1) Defining Your Local Domain

You must manually specify your local domain in the database so the `updateTopology.sh` script in the next section knows which links are local. This is done by logging-in to the MySQL CLI and running an INSERT command, thus:

```
tomcat55@red-narb:~$ mysql -uoscars -poscars bss  
Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A
```

```
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 19581
```

```
Server version: 5.0.32-Debian_7etch1-log Debian etch distribution
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

```
mysql> INSERT INTO domains VALUES(NULL, "blue.pod.lan", "blue",  
"https://idc.blue.pod.lan:8443/axis2/services/OSCARS", "blue", 1);  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> exit  
Bye
```

Replace 'blue.pod.lan' with the local part of you domain-id, 'blue' with a string for your domain (red/yellow/green), and add the URL to your IDC.

## 2) Run updateTopology.sh

The updateTopology.sh script syncs the database with your topology files. The three steps for running this script are:

```
tomcat55@red-narb:~$ cd ~tomcat55/dcn-software-suite-0.2/idc/tools/updatedbterce/  
tomcat55@red-narb:~/dcn-software-suite-0.2/idc/tools/updatedbterce$  
./updateTopology.sh http://127.0.0.2:8080/axis2/services/TERCE  
DOMAIN: red.pod.lan  
NODE: vlsr1  
PORT: 3  
LINK: *  
PORT: 4  
LINK: 11.1.2.1  
PORT: 5  
LINK: 11.1.3.1  
NODE: vlsr2  
PORT: 4  
LINK: 11.1.4.1  
PORT: 3  
LINK: 11.1.2.2  
NODE: vlsr3  
PORT: 4  
LINK: 11.1.4.2  
PORT: 5  
LINK: 11.1.3.2  
PORT: 3  
LINK: *  
PORT: 7  
LINK: 11.1.11.1  
Complete.  
tomcat55@red-narb:~/dcn-software-suite-0.2/idc/tools/updatedbterce$
```

If no error is returned, your database is now populated with the appropriate topology information.

## Step 7: Running the Scheduler

The scheduler is a script that automatically builds circuits for users that do not wish to use signaling and deletes expired reservation. Running the scheduler is required for the IDC to function properly. This section describes compiling and running the scheduler script.

- 1) Copy your server's keystores and axis2.xml file to the new repo:

```
tomcat55@red-narb:~$ cd ~tomcat55/dcn-software-suite-0.2/idc/tools/schedulers
tomcat55@red-narb:~/dcn-software-suite-0.2/idc/tools/schedulers$ ant
tomcat55@red-narb:~/dcn-software-suite-0.2/idc/tools/schedulers$ cp
/usr/local/tomcat/shared/classes/repo/*.jks repo
tomcat55@red-narb:~/dcn-software-suite-0.2/idc/tools/schedulers$ cp
/usr/local/tomcat/shared/classes/repo/axis2.xml repo
```

- 2) You may run the scheduler with the following command:

```
tomcat55@red-narb:~$ cd ~tomcat55/dcn-software-suite-0.2/idc/tools/schedulers
tomcat55@red-narb:~/dcn-software-suite-0.2/idc/tools/schedulers$ nohup
./scheduler.sh > ~/scheduler.log &
```

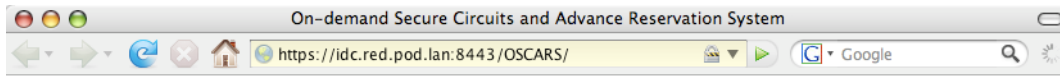
The above command will run the scheduler in the background and log DRAGON output in ~/scheduler.log.

## Step 8: Schedule and Provision the intra-domain LSP

Now, let's try to provision an intra-domain LSP via the WBUI provided by OSCARS.

- 1) Login to the Web-Based User Interface (WBUI)

Open a web browser and connect to “[https://idc.\[color\].pod.lan:8443/OSCARS/](https://idc.[color].pod.lan:8443/OSCARS/)”. Login with username **oscars-admin** and password **oscars**. Again, please note that Internet Explorer is not currently supported.



## On-demand Secure Circuits and Advance Reservation System

A collaboration between [ESnet](#), [Internet2](#), [DANTE](#), and [ISI East](#)

January 14, 2008 11:31 Please sign in.

Login Name:   
Password:

Sign in via your OSCARS login and password to access this system. To find out about OSCARS and how it works, go to the [documentation](#). To obtain an account or request more information, email one of the contacts below.

[Documentation](#) | [ESnet](#) | [Berkeley Lab](#) | [Notice to Users](#)

Contacts: [Chin Guok](#), [David Robertson](#)

- 2) Once you have logged in, create a path reservation.

**Before you start provisioning via OSCARS WBUI, please ensure that the time set on your computer is valid – you must have a valid timezone setting with the current time in that timezone. The WBUI uses JavaScript to get the local time on your machine if you leave the “Time” field blank (see below).**

Click on the tab “Create Reservation”. The “source” and “destination” fields are in the following format:

```
urn:ogf:network:domain=[color].pod.lan:node=[vlsrc1 or vlsrc3]:port=3:link=*
```

For example:

```
blue-es1 -> urn:ogf:network:domain=blue.pod.lan:node=vlsrc1:port=3:link=*
```

```
yellow-es2-> urn:ogf:network:domain=yellow.pod.lan:node=vlsrc3:port=3:link=*
```

For example, you can set up an LSP between red-es1 and red-es2 with the following filled in the “Create Reservation” form:

```
source: urn:ogf:network:domain=red.pod.lan:node=vlsrc1:port=3:link=*  
destination: urn:ogf:network:domain=red.pod.lan:node=vlsrc3:port=3:link=*  
Bandwidth: 100  
Purpose of reservation: describe what this LSP is for, i.e. testing  
VLAN: any
```

Since this is an intra-domain LSP, select the es1 and es2 in your pod as the source and destination. Then, click “Reserve bandwidth”.

January 15, 2008 10:29 Reservation creation form

Reservations **Create Reservation** Users Log out

Reserve bandwidth Reset form fields

Required inputs are bordered in green. Ranges or types of valid entries are given in parentheses after the default values, if any. If date and time fields are left blank, they are filled in with the defaults. The time zone is your local time zone.

**WARNING:** Entering a value in a red-outlined field may change default routing behavior for the selected flow.

Source	<input type="text" value="urn:ogf:network:domain=blue.pod.lan:node=vlsr1:port="/>	(Host name or IP address)
Destination	<input type="text" value="urn:ogf:network:domain=red.pod.lan:node=vlsr3:port=3"/>	(Host name or IP address)
Bandwidth (Mbps)	<input type="text" value="100"/>	(10-5000)
Purpose of reservation	<input type="text" value="lsp btw blue-es1 and red-es2"/>	(For our records)
Path	<input type="text"/>	(series of hops)
Day of year	<input type="text" value=""/>	2008-1-15
Time	<input type="text" value=""/>	10:28
Duration (Hours)	<input type="text" value=""/>	0.01 to 8760 (1 year)
Layer 2 parameters		
VLAN	<input type="text" value="any"/>	tag, or range, e.g. 3000-3100
Layer 3 parameters		
Source port	<input type="text" value=""/>	(1024-65535)
Destination port	<input type="text" value=""/>	(1024-65535)
Protocol	<input type="text" value=""/>	(0-255, or string)

You can leave all other fields as blank. This is going to create an LSP right way with your required bandwidth. Since you haven't put in the duration of this LSP, the default will have this LSP up for 5 minutes. The VLAN field must be filled in with 'any' or a value between 100-200 or else you will get an error message.

Once you click "Reserve bandwidth", you will be taken to a page that shows the summary of your reservation and the status of the LSP that you have just set up.

At the top of the page, it will indicate if your reservation is successful or not. If the reservation is successfully, you will see your LSP in "PENDING" status. If you click "Refresh" button on the top of that page, you will see the updated status of the LSP. If the IDC successfully put the LSP in service, the status will change to "ACTIVE".

January 19, 2008 13:23      Successfully got reservation details

Reservations   Create Reservation   Users   Log out

### Reservation Details

To return to the reservations list, click on the Reservations tab.

Attribute	Value
GRI	red.pod.lan-27
User	oscars-admin
Description	testing
Start time	2008-01-19 13:18
End time	2008-01-19 13:22
Created time	2008-01-19 13:18
Bandwidth	100000000
Status	ACTIVE
Source	urn:ogf:network:domain=red.pod.lan:node=vlsr1:port=3:link=*
Destination	urn:ogf:network:domain=green.pod.lan:node=vlsr1:port=3:link=*
VLAN	100
Nodes in path	urn:ogf:network:domain=red.pod.lan:node=vlsr1:port=3:link=*, desc: [ingress] VLAN: [100] urn:ogf:network:domain=red.pod.lan:node=vlsr1:port=5:link=11.1.3.1, VLAN: [100] urn:ogf:network:domain=red.pod.lan:node=vlsr3:port=5:link=11.1.3.2, VLAN: [100] urn:ogf:network:domain=red.pod.lan:node=vlsr3:port=7:link=11.1.11.1, desc: [egress] VLAN: [100]

Remember that you have put "any" on the "VLAN" field, so please check which VLAN OSCARS has assigned to your LSP. For example, the above example has VLAN 100.

If you see the status change to FAILED, please check the output of the following files and work with the instructors to debug the problem. Please check that the start time/create time are correct and that the time on your computer is valid if you left the "Time" field blank.

```
/home/tomcat55/scheduler.log
/usr/local/tomcat/logs/catalina.out
/usr/local/tomcat/logs/oscars.log
```

After 5 minutes, you should see that the status changes to “FINISHED”.

To actually make use of your LSP, you need to follow the steps described in Step 4 below. The method used in exercise 1 will not work because the edge ports are now in tagged/trunking mode – the VLSRs are expecting the end systems to be sending tagged 802.1Q frames so we must use the `vconfig` command in Linux.

3) Other ways to check the LSP status

- a. You can go to your NARB/IDC machine and look at the schedule log file to check the status of the LSP.

```
tomcat55@red-narb:~$ tail -f ~/scheduler.log
...
red-vlsr1-pc> edit lsp red.pod.lan-27
red-vlsr1-pc(edit-lsp-red.pod.lan-27)#
red-vlsr1-pc(edit-lsp-red.pod.lan-27)# set source ip-address 192.168.1.4
tagged-group 100 destination ip-address 192.168.1.8 tagged-group 100
red-vlsr1-pc(edit-lsp-red.pod.lan-27)#
red-vlsr1-pc(edit-lsp-red.pod.lan-27)# set bandwidth eth100M swcap l2sc
encoding ethernet gpid ethernet
red-vlsr1-pc(edit-lsp-red.pod.lan-27)#
red-vlsr1-pc(edit-lsp-red.pod.lan-27)# set vtag 100
red-vlsr1-pc(edit-lsp-red.pod.lan-27)#
red-vlsr1-pc(edit-lsp-red.pod.lan-27)# exit
red-vlsr1-pc>
red-vlsr1-pc> commit lsp red.pod.lan-27
red-vlsr1-pc>
red-vlsr1-pc> show lsp red.pod.lan-27
Src 192.168.1.4/100, dest 192.168.1.8/100
GRI: 1951985063-2147483653
Generic TSPEC R=eth100M, B=eth100M, P=eth100M, m=100, M=1500
Encoding ethernet, Switching l2sc, G-Pid ethernet
Ingress Local ID Type: tagged group, Value: 100
Egress Local ID Type: tagged group, Value: 100.
E2E LSP VLAN Tag: 100.
Status: Commit
red-vlsr1-pc>
red-vlsr1-pc> show lsp red.pod.lan-27
Src 192.168.1.4/100, dest 192.168.1.8/100
GRI: 1951985063-2147483653
Generic TSPEC R=eth100M, B=eth100M, P=eth100M, m=100, M=1500
Encoding ethernet, Switching l2sc, G-Pid ethernet
Ingress Local ID Type: tagged group, Value: 100
Egress Local ID Type: tagged group, Value: 100.
E2E LSP VLAN Tag: 100.
Status: In service
red-vlsr1-pc>
...
```

You will see the CLI commands that the WBUI generates based on your WBUI request.

- b. You can also go to the “source” or “destination” of the LSP to check the status at DRAGON CLI. For the example above, you can login to red-vlsr; telnet to localhost 2611 and check on the LSP.

```
red-vlsr1-pc> show lsp
**LSP status summary**

Name          Status      Dir      Source (IP/LSP ID)  Destination (IP/Tunnel ID)
-----
red.pod.lan-27
                In service <=> 192.168.1.4      192.168.1.8
                100                100

red-vlsr1-pc> show lsp red.pod.lan-27
Src 192.168.1.4/100, dest 192.168.1.8/100
GRI: 1951985063-2147483653
Generic TSPEC R=eth100M, B=eth100M, P=eth100M, m=100, M=1500
Encoding ethernet, Switching l2sc, G-Pid ethernet
Ingress Local ID Type: tagged group, Value: 100
Egress Local ID Type: tagged group, Value: 100.
E2E LSP VLAN Tag: 100.
Status: In service
```

- 4) As shown above, **the LSP is in port-to-port/tagged mode, so the data plane interface is eth1.<vlan>, not eth1 as in the previous example.** Take a look at the VLAN that you have been assigned; in the above example, it's **vlan 100**.

Login to red\_es1 as “root”:

```
red_es1_pc:~# /sbin/modprobe 8021q
red_es1_pc:~# vconfig add eth1 100
Added VLAN with VID == 100 to IF -:eth1:-
red_es1_pc:~# ifconfig eth1.100 10.0.0.1 netmask 255.255.255.252
```

Then, login to red\_es2 as “root”, and did what you do for red\_es1, except that you should put the other side of the /30 to eth1.100.

Now, you should be able to ping across your new LSP.

- 5) Schedule a reservation

Follow step 1-3 except that you can fill in “time” and “duration” for your reservation. Try to schedule an LSP 2 minutes later for 0.05 hour (i.e. 3 minutes).

## Exercise #3 Inter-domain provisioning with OSCARS

Objective: In this exercise, we will configure the OSCARS for inter-domain scheduling and provision.

Discussion:

In the previous exercise, we have scheduled and provisioned Intra-domain LSP. In this exercise, we will interconnect the four pod domains as a star:

- Green is the hub.
- Red-vlsr3 is connected to Green-vlsr1.
- Blue-vlsr2 connects to Green-vlsr2.
- Yellow-vlsr1 connects to Green-vlsr3.
- Each pod, except green, will have one inter-domain link, while green has 3 inter-domain links.

Inter-domain signaling works slightly differently than intra-domain signaling: signaling occurs end-to-end in an intra-domain LSP. When it comes to inter-domain LSP, the signaling occurs within the intra-domain basis and there is no signaling via the inter-domain link. For example, if you want to provision a link from red-es1 to blue-es2.

- The red-IDC will set up the LSP from red-sw1 port 3 to red-sw3 port 7.
- The green-IDC will setup the LSP from green-sw1 port 7 to green-sw2 port7.
- The blue-IDC will setup the LSP from blue-sw2 port 7 to blue-sw3 port 3.

**Procedure:**

### Step 1: Design and implement the inter-domain data and control plane

Refer to diagram 3.1a. Add the appropriate data circuit IDs, i.e. D11, D12 and D13, to your list of data links created in Exercise #1. Next, define the appropriate GRE tunnels for the corresponding control links. Assign IP addressing for both the TE links and the control links. Since this requires coordination with your colleagues in the neighboring pod, make sure you do so.

Go to the lab and install the appropriate Ethernet cabling for the desired data plane connectivity.

### Step 2: Configure inter-domain TE into OSPFd

On each border VLSR, we need to add the inter-domain link to the ospfd.conf. Add the “**interface gre<nn>**” configuration block. Under the “**router ospf**” block, you need to add “**passive-interface gre<nn>**” and the “**network <ctrl\_ip\_addr> area 0.0.0.0**” statement. And don’t forget to add an “**ospf-te interface gre<nn>**” block (looks like the other TE links).

Please note: we will leave the *color prefix* in the inter-domain GRE names.

Since RSVP will use this control link, we need to add the GRE interface to the RSVPD.conf file. You can look at the other interfaces to get the format.

Restart the DRAGON software at the border VLSR by

```
/usr/local/dragon/bin/dragon.sh restart-vlsr
```

### Step 3: Configure OSCARS for inter-domain

#### 1) Edit **tedb-intra.xml**

In Exercise #2, we have already edited the `tedb-intra.xml` at

```
/usr/local/tomcat/shared/classes/terce.conf/tedb-intra.xml
```

to include all intra domain nodes in the pod. Now, add the following statements to the block of each border VLSR in **tedb-intra.xml**.

For example, `blue-vlsr2`:

```
<!-- interdomain link to green-vlsr2 -->
  <port id="urn:ogf:network:domain=blue.net:node=vlsr2:port=7">
    <capacity>1000000000</capacity>
    <maximumReservableCapacity>1000000000</maximumReservableCapacity>
    <minimumReservableCapacity>1000000000</minimumReservableCapacity>
    <granularity>1000000000</granularity>

    <link
id="urn:ogf:network:domain=blue.net:node=vlsr2:port=7:link=11.2.12.1">

<remoteLinkId>urn:ogf:network:domain=green.net:node=vlsr2:port=7:link=11.2.12.2</re
moteLinkId>

      <trafficEngineeringMetric>100</trafficEngineeringMetric>
      <capacity>1000000000</capacity>

<maximumReservableCapacity>1000000000</maximumReservableCapacity>

<minimumReservableCapacity>1000000000</minimumReservableCapacity>
    <granularity>1000000000</granularity>
    <SwitchingCapabilityDescriptors>
      <switchingcapType>l2sc</switchingcapType>
      <encodingType>ethernet</encodingType>
      <switchingCapabilitySpecficInfo>
        <interfaceMTU>9000</interfaceMTU>
        <vlanRangeAvailability>100-200</vlanRangeAvailability>
      </switchingCapabilitySpecficInfo>
    </SwitchingCapabilityDescriptors>
  </link>
</port>
```

#### 2) About **tedb-inter.xml**

As the name indicates, **tedb-inter.xml** describes the topology that each pod would like to advertise to its peer. We can abstract the topology in **tedb-inter.xml** to include only the border VLSRs and the ports

that we allow other pods to access. For simplicity, we will have both **tedb-intra.xml** and **tedb-inter.xml** exactly the same. In the previous exercise, we have already set the **terce-ws.properties** files to point to **tedb-inter.xml** and **tedb-intra.xml**, so we are done with this step.

### 3) Rerun **updateTopology.sh**

```
tomcat55@red-narb:~$ cd ~/dcn-software-suite-0.2/idc/tools/updatedbterce/
tomcat55@red-narb:~/dcn-software-suite-0.2/idc/tools/updatedbterce$
./updateTopology.sh http://127.0.0.2:8080/axis2/services/TERCE
DOMAIN: red.pod.lan
NODE: vlsr1
PORT: 3
LINK: *
PORT: 4
LINK: 11.1.2.1
PORT: 5
LINK: 11.1.3.1
NODE: vlsr2
PORT: 4
LINK: 11.1.4.1
PORT: 3
LINK: 11.1.2.2
NODE: vlsr3
PORT: 4
LINK: 11.1.4.2
PORT: 5
LINK: 11.1.3.2
PORT: 3
LINK: *
PORT: 7
LINK: 11.1.11.1
Complete.
tomcat55@red-narb:~/dcn-software-suite-0.2/idc/tools/updatedbterce$
```

## Step 4: Inter-domain configuration

**Please stop here as we would like to go through the following steps together.**

### 1) Generating a Server Certificate for Inter-Domain Requests

You must generate a certificate that your domain will pass to other domains. This certificate is stored in the following keystore file:

```
/usr/local/tomcat/shared/classes/repo/sec-client.jks.
```

These five steps will create a certificate for your domain:

1. Ssh to the NARB/IDC with username tomcat55. Generate a certificate and certificate signing request (CSR):

In our exercise, the green pod is the only CA that's signing the certificate for other pods.

```

> ssh tomcat55@red-narb
tomcat55@red-narb:~$ cd /usr/local/tomcat/shared/classes/repo/
tomcat55@red-narb:/usr/local/tomcat/shared/classes/repo$ keytool -list -keystore
sec-client.jks -alias greenCA -V -storepass password
Alias name: greenCA
Creation date: Jan 14, 2008
Entry type: trustedCertEntry

Owner: CN=idc.green.pod.lan, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=US
Issuer: CN=idc.green.pod.lan, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=US
Serial number: 89b4999737fcd0ab
Valid from: Mon Jan 14 18:51:54 EST 2008 until: Thu Jan 11 18:51:54 EST 2018
Certificate fingerprints:
    MD5: EB:FC:FD:B9:78:40:61:B1:32:D0:64:CE:37:DD:CD:29
    SHA1: A5:93:01:22:DA:FD:52:B6:F1:88:6C:64:78:97:01:B7:1E:02:BC:6C

```

The above keytool command shows that greenCA has been installed in your machine as a trusted CA.

```

tomcat55@red-narb:~ % cd /usr/local/tomcat/shared/classes/repo/
tomcat55@red-narb:/usr/local/tomcat/shared/classes/repo % keytool -genkey -alias
redidc -keystore sec-client.jks -storepass password -validity 3650
What is your first and last name?
  [Unknown]: idc.green.pod.lan
What is the name of your organizational unit?
  [Unknown]:
What is the name of your organization?
  [Unknown]:
What is the name of your City or Locality?
  [Unknown]:
What is the name of your State or Province?
  [Unknown]:
What is the two-letter country code for this unit?
  [Unknown]: US
Is CN=idc.green.pod.lan, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=US
correct?
  [no]: yes

Enter key password for <testing>
  (RETURN if same as keystore password):
tomcat55@red-narb:/usr/local/tomcat/shared/classes/repo$

```

It's important

2. Now, create a certificate signing request

```

tomcat55@red-narb:/usr/local/tomcat/shared/classes/repo % keytool -certreq -
alias redidc -keystore sec-client.jks -storepass password -file ~/redidc.csr
tomcat55@red-narb:/usr/local/tomcat/shared/classes/repo$ cat ~/redidc.csr
-----BEGIN NEW CERTIFICATE REQUEST-----
MIICbDCCAioCAQAwZzELMAkGA1UEBhMCVVMxEDAOBgNVBAGTB1Vua25vd24xEDAQBgNVBAcTB1Vu

```

```
a25vd24xEDAObgNVBAoTB1Vua25vd24xEDAObgNVBAStB1Vua25vd24xEDAObgNVBAMTB1Vua25v
d24wgG4MIIBLAYHKOZIZjgEATCCAR8CgYEA/X9TgR11EiLS30qcLuzk5/YRt1I870QAwX4/gLZR
JmlFXUAIUftZPY1Y+r/F9bow9subVWzXgTuAHTRv8mZgt2uZUKWkn5/oBHsQIsJPu6nX/rfGG/g7
V+fGqKYVDwT7g/bTxR7DAjVUE1oWkTL2dfOuK2HXKu/yIgmZndFIAccCFQCXYFCPFSMLzLKSuYKi
64QL8Fgc9QKBgQD34aCF1ps93su8q1w2uFe5eZSvu/o66oL5V0wLPQeCZ1FZV4661F1P5nEHEIGA
tEkWcSPoTCgWE7fPCTKMyKbhPBZ6ilR8jSjgo64eK7OmdZFuo38L+iE1YvH7YnoBJDvMpPG+qFGQ
iaiD3+Fa5Z8GkotmXoB7VSVkAUw7/s9JKgOBhQACgYEAiT81T1ltU0ZBuItSRaMvJNpcKFPvojo6
EDa8j1IrcNbEKssdAQtrYgD2IZwqy4mPDdk11cH1Tf+ltm77tOWmC/B79C8CEaOj70EcGhxV9CDp
HVxWmSo7CY0VnKp/ahxLtKHbiKxSIqwkCKGW80Fd6pPQxaqCjyZ+RQdtugh0bmqqADALBgcqhkjO
OAQDBQADLwAwLAIUP2mkh8i/FcwKcLsw8ack5RfVYxACFAY3WZmDDPvPKKmtfpy7xzldscLW
-----END NEW CERTIFICATE REQUEST-----
tomcat55@red-narb: /usr/local/tomcat/shared/classes/repo$
```

Email redidc.csr to the CA, who is Chris Tracy, in the class for signing your certificate. After Chris signs the certificate, he will email you back the signed certificate.

### 3. Chris should email you back the file redidc.cer

```
tomcat55@red-narb:~ % cd /usr/local/tomcat/shared/classes/repo/
tomcat55@red-narb: /usr/local/tomcat/shared/classes/repo % keytool -import -
keystore sec-client.jks -alias redidc -file redidc.cer
Certificate reply was installed in keystore
```

Also, verify they have the CA certificate of the entity that signed your certificate installed. You may view the subject of your certificate with the following command:

```
tomcat55@red-narb: /usr/local/tomcat/shared/classes/repo$ keytool -list -keystore
sec-client.jks -alias redidc -v -storepass password
Alias name: redidc
Creation date: Jan 19, 2008
Entry type: keyEntry
Certificate chain length: 2
Certificate[1]:
Owner: CN=Unknown, OU=Unknown, O=Unknown, ST=Unknown, C=US
Issuer: CN=idc.green.pod.lan, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=US
Serial number: 10000e
Valid from: Sat Jan 19 15:26:32 EST 2008 until: Tue Jan 16 15:26:32 EST 2018
Certificate fingerprints:
    MD5: 7E:AA:97:DE:E2:39:B7:67:35:34:61:1D:ED:84:13:EA
    SHA1: 4C:EC:3D:0C:0D:78:33:DF:5D:42:9B:E6:8E:49:1F:9B:A1:B3:79:EC
Certificate[2]:
Owner: CN=idc.green.pod.lan, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=US
Issuer: CN=idc.green.pod.lan, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=US
Serial number: 89b4999737fcd0ab
Valid from: Mon Jan 14 18:51:54 EST 2008 until: Thu Jan 11 18:51:54 EST 2018
Certificate fingerprints:
    MD5: EB:FC:FD:B9:78:40:61:B1:32:D0:64:CE:37:DD:CD:29
    SHA1: A5:93:01:22:DA:FD:52:B6:F1:88:6C:64:78:97:01:B7:1E:02:BC:6C
```

And you should see the display is much longer than what you saw for the green in the previous page. This output shows a chain of trust established by the green-CA signing your certificate.

4. Modify `/usr/local/tomcat/shared/classes/repo/axis2.xml` to reference your certificate by changing the `<user>` element to the alias of your certificate in the keystore (it will currently be set to “alice”):

```
...
<parameter name="OutflowSecurity">
  <action>
    <items>Timestamp Signature</items>
    <user>redidc</user>
  </action>
</parameter>
...
```

## 2) Making your IDC Aware of Other Domains

You must add an entry for each of your neighboring domains to the `bss.domains` table. This is currently a manual process done with the following commands:

```
tomcat55@red-narb:~$ mysql -u oscars -poscars bss
mysql> INSERT INTO domains VALUES(NULL, "green.pod.lan", "green",
"https://idc.green.pod.lan:8443/axis2/services/OSCARS", "green", 0);
mysql> exit
Bye
tomcat55@red-narb:~$
```

Since all pod except green will have only one neighbor, which is the green.

## Step 5: Provision the inter-domain LSP via WBUI

Follow Step 8 in previous exercise, except that you will have your “source” and “destination” from different pod. When it comes to inter-domain LSP, please coordinate with your colleagues in the other domain so that you won’t assign conflicting IP addresses on the point-to-point/tagged layer 2 links.