

Exercise #1: Designing a GMPLS Control Plane for Ethernet Data Planes

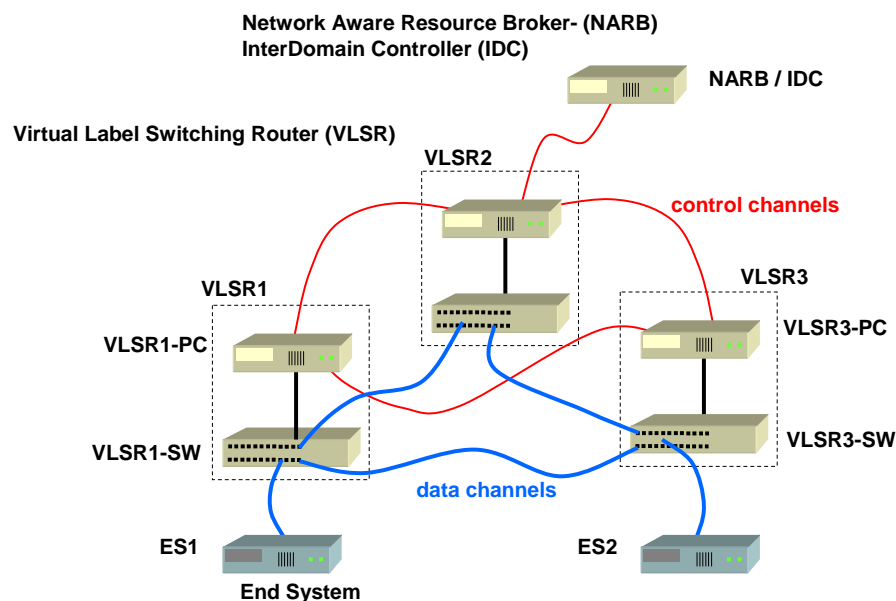
Objective: In this exercise, we will design, build, configure, and test a GMPLS control plane for a dynamic network capable of providing services across an Ethernet data plane.

There will be 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 pod architecture is organized to reflect a wide area network with multiple regional networks attached. The Green pod is in the position of the central wide area network in this configuration. The control plane could easily be configured in alternate topologies, such as full or partial mesh configurations. This specific topology is utilized here to facilitate classroom instruction, and also reflects current deployments.

The networks constructed in each exercise will form the basis for the 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.

Pod Architecture Overview

We will be building a network consisting of three Ethernet switches acting as network elements (see diagram below). Two PCs will act as End Systems. The team will develop a network diagram incorporating route diversity, where the end systems attach 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. All 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. 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 management and control, and eth1 for data plane.

(Note: Each PC has dual 1gigE integrated ethernet ports: eth0 and eth1. We use one physical interface for management and control and a separate interface for 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. We have not assigned separate loopback addresses in this workshop in order to simplify the overall addressing. Instead, we use same address for the loopback and management address.

For these exercises we use the 192.168.0.0/16 subnet for management addressing. Appendix A (Management Plane Configuration) shows the details of the management plane architecture and configuration. The management plane is already configured and in place on the workshop pods, so there are no class tasks required for this.

There are three main areas which must be addressed when building a dynamic network:

Data Plane – Exercise 1, Step 1

Control Plane Infrastructure (control communication channels) – Exercise 1, Step 2

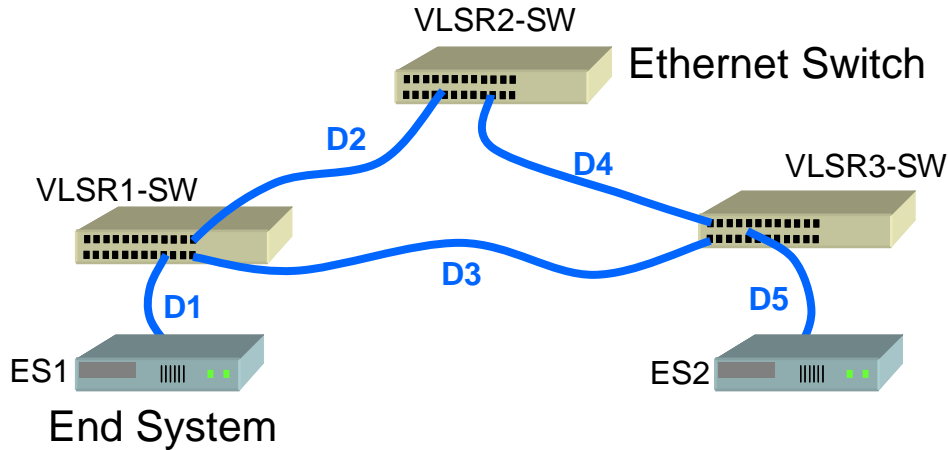
Control Plane Software (DRAGON) configuration – Exercise 1, Steps 3-6

Exercise 1 includes six steps as indicated above which will guide us through the process of building and configuring each of these. Upon completion of Exercise 1, we will have configured and tested the dynamic provision of Ethernet VLAN based network services using the DRAGON GMPLS based control plane.

Appendix B (Detailed Pod Architecture and Configuration) contains the details of the eventual architecture and configuration for the workshop pods. Reference will be made to the information in this Appendix as part of many of the steps throughout this workshop.

Step 1: Data Plane Layout and Build

In this step we will create the physical data plane connections between the network elements. The generic data plane configuration for a pod is as shown below.



Data Plane via Cat5 Patch Cable

Each group will need to visit the pod rack and connect the specific data plane for their individual pods. It is recommended that you reference the information in Appendix B, and fill out the table below to facilitate cable connections on the pod racks. Each team may also want to fill out the worksheet in Appendix C for Exercise 1 steps 1-3. For the workshop configuration 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.

ID	Network Element	Interface/Port	Network Element	Interface/Port
D2 (example)	VLSR1-SW	port 4	VLSR2-SW	port 3
D1				
D2				
D3				
D4				
D5				

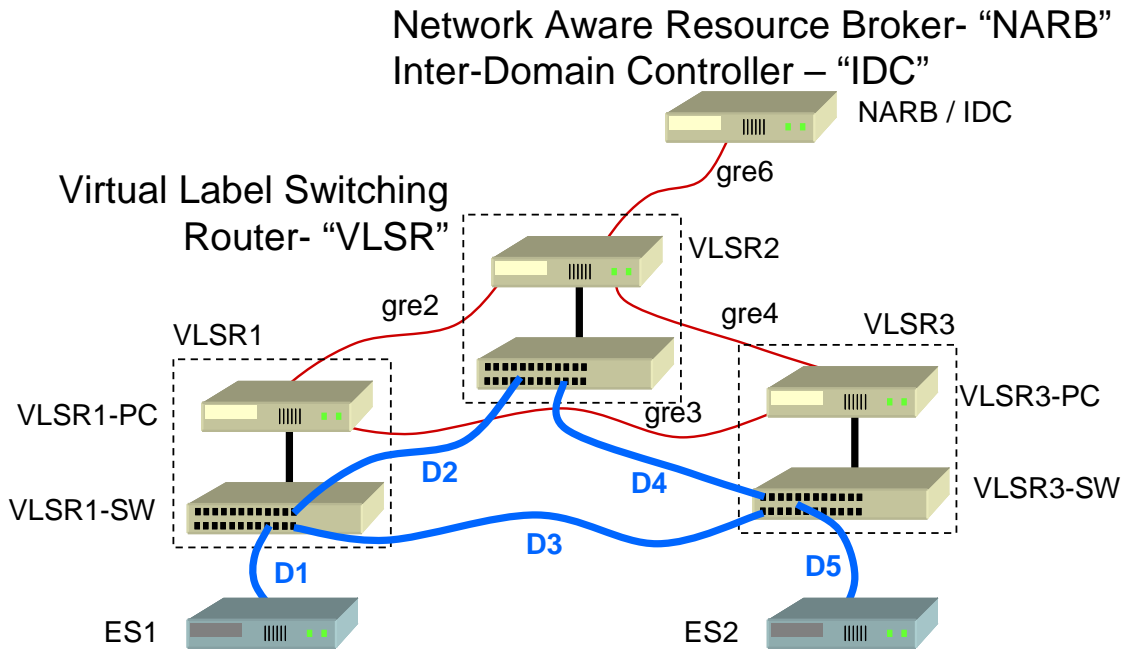
→Attendees should now go to the lab and install the appropriate Ethernet cabling for the desired data plane connectivity←

Step 2: Design and Build the Control Plane Infrastructure (control communication channels) 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 referred to as 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 devices. In GMPLS, these control plane devices may not be physically adjacent to each other. 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. The generic control plane channel configuration for a pod is as shown below. The red lines indicate the control channels along with the control plane, layered on top of the data plane from the previous exercise.



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 do not need to setup GRE tunnels between VLSRs and ESs. We use the management addresses as the endpoints for building all GRE tunnels.

The 10.0.0.0/8 subnet is utilized for control plane addressing. In a real network, the intra-domain control plane links do not strictly need to be public. All inter-domain communications are accomplished via Web Service based communication between IDCs. Therefore, the only address which strictly needs to be routed and reachable is the IDC Web Service entry point.

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 links 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-gre#
Blue: blue-gre#
Yellow: ylw-gre#
Green: grn-gre#

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 configuration problems.)

In this step attendees should make a list of control channels to be configured in their respective pod. This information will be utilized as part of the configurations in step 4. Each control link should include the associated data link, the GRE tunnel # to be used, the IP addresses of the tunnel endpoints, and the control 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. Appendix B provides detailed information which can be used to build this list.

GRE Name	Local Network Element	Local Ctrl Addr	Local Tunnel Addr	Remote Network Element	Remote Ctrl Addr	Remote Tunnel Addr
red-gre2 (example)	vlsr1-pc	10.1.2.1	192.168.1.4	vlsr2-pc	10.1.2.2	192.168.1.6

→Attendees should now create a list of control plane links for the control channels in their pod←

Step 3: Assign Traffic Engineering (TE) Addresses

The TE-addresses are “identifiers” for the data links in the dynamic network. While they are formatted like regular IP addresses, they are not IP addresses in the context of being able to use them as addresses for sending or receiving of data. TE-addresses are strictly utilized by the control plane to identify, compute paths, and signal network provisioning across the dynamic network. TE-Addresses provide a means to associate specific data plane interfaces with their control links. Since the TE addresses are not used for IP routing across the interfaces, in order to send data over a dynamically allocated circuit, the user will need to allocate and assign a data plane IP address to the interface associated with the newly created circuit. The TE address should not be used for this purpose.

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.

Traffic Engineering (TE) addresses should be assigned for each data plane interface. For this exercise, and to keep the addressing scheme somewhat 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

Since the inter-domain solution is based on Web Services, it makes no difference if TE addresses are based on private address space compared to globally unique addresses. The GMPLS standards do allow for configurations where inter-domain data is shared at a GMPLS level. In this configuration, use of private address space would be a bad practice. However, for the solution and configuration presented in this class, private address space for TE links numbering is fine.

In this step attendees should make a list of data plane links and associated TE addresses for the data plane links to be configured in their respective pod. This information will be utilized as part of the configurations in step 4. Each data plane link should include the data link id, the local/remote network element, local/remote TE addresses, and the associated control channel id. Appendix B provides detailed information which can be used to build this list. Please note that end systems attach at special locations to the dynamic network known as “edge” ports. These connections do not have TE addresses or associate control channels. Details on how to handle these data links and end systems will be included in the subsequent control plane software configuration steps. For now, all that is needed is to note the associated network elements.

Data Plane Link ID	Local Network Device	Local TE Addr	Remote TE Addr	Remote Network Device	Associated Control Channel ID
D2-example	vlsr1-sw	11.1.2.1	11.1.2.2	vlsr2-sw	red-gre2
D1	edge port – no TE link				
D2					
D3					
D4					
D5	edge port – no TE link				

→Attendees should now create a list of TE link addresses for the data plane links in their pod←

Step 4: Configure the VLSRs

Now we use the information gathered in the previous steps to configure the DRAGON control plane software. This will allow us to dynamically provision vlan based Ethernet “circuits” across the pods. For this exercise we will limit ourselves to “intra-domain” or “intra-pod” provisioning operations. Appendix B provides detailed information which will be helpful to understanding the configuration items to follow.

All pod PCs have been pre-loaded with Debian Linux, the software dependency packages required for the DRAGON software and/or systems support, and a recent release of the DRAGON GMPLS software suite. Detailed instructions for installation of the DRAGON software is provided as part of the DCN Software Suite release. Instructions for obtaining and installing this software are provided as part of the slides presented and in the course material.

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

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

A. Configure the GRE Tunnels (Control Plane Communication Channels)

Here we need to configure the GRE tunnels on each of the VLSRs in the pod. As described earlier, these are the control plane communication channels which the VLSRs used for routing and signaling communications. On each VLSR-PC, we need to 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.

For this step, 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 at the top of the file, as well as a call to the `setup_gre_tunnels.sh` script in `/usr/local/dragon/etc`:

```
# 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.

The script `/usr/local/dragon/etc/setup_gre_tunnels.sh` on each machine is used to configure the GRE tunnels, and is included in the startup process as shown above.

```
# 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
```

B. Edit VLSR Software Configuration Files

There are four configuration files on each VLSR that must be edited in this step:

- o `dragon.conf`: this is the configuration file for the dragon daemon which provides the user interface and provisioning entry point into the DRAGON control plane.
- o `ospfd.conf`: this is the configuration file for the OSPF daemon, which is the GMPLS OSPF-TE routing engine.
- o `RSVPD.conf`: this is the configuration file for the RSVP daemon, which is the GMPLS RSVP-TE signaling engine.
- o `zebra.conf`: this is the configuration file for the zebra daemon, which provides the interface the system kernel for routing related configurations and changes.

In each case a sample file has been provided which will be copied to the default configuration file name and then edited by the attendees. Please note that these configuration files will need to be edited on **all three VLSR** machines of your pod.

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. For now, this file only contains the hostname assignment for this VLSR and the login password. For simplicity sake, we set the password to “dragon”.

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

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

```
hostname red-vlsr1
password dragon
enable password dragon
```

Next, edit the `RSVPD.conf` file. This configuration file tells the signaling daemon which control channels (GRE tunnels) to use for signaling. In this file, 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
```

Finally, edit the `ospfd.conf` file. This is the configuration file for the GMPLS OSPF-TE routing engine. We use this file to tell the OSPF-TE routing engine the following information:

- o identify which control channels to use for establishment of OSPF-TE adjacencies (i.e., location of peering VLSRs)
- o describe the data plane (TE link) characteristics and capabilities
- o define the association between control channels and data plane links
- o define how to contact the “data plane switch fabric” which in this case is an Ethernet switch associated with each VLSR PC

It is here where we define the data plane topology over which provisioning actions can occur. In addition, this configuration file will tell OSPF which control links are associated with which data links. The resulting information exchanged by the individual OSPF-TE speakers allows for the construction of a Traffic Engineering Topology Database (TEDB) which contains a topological view of the dynamic data plane infrastructure. This provides the basis for subsequent path computation and signaling operations which will be described in subsequent sections.

An example `ospfd.conf` file is shown below. There are comments in the form of “#explanation comments here#” which are not included in the sample files and are not needed in the files each pod team will be creating. They are included here to provide some additional information regarding the configuration statements. The following text is excerpted from one VLSR. Each pod team should review this configuration example and make appropriate edits for the `ospfd.conf` files on each of their VLSRs:

```
! OSPFd sample configuration file
#here we define the hostname, passwords, and location for log files#
hostname red-vlsr1-ospf
password dragon
enable password dragon
log stdout
log file /var/log/ospfd.log
!
!
#here we define the control channels over which OSPF-TE adjacencies will be established.#
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
!
#here we define the router-id and the networks which OSPF-TE will include in its
advertisements. Please notice that the networks below are the GRE tunnel networks.#
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
#here we define the router-address for the OSPF-TE process, this is the generally the
same as for the router-id above.
ospf-te router-address 192.168.1.4
#the ospf-te statements define the data plane topology associated with this VLSR. In the
example below, the “11.1.2.1” is the TE address identified in an earlier exercise. This
is used by routing, path computation, and signaling to identify this particular link.
The “snmp switch-ip 192.168.1.3” indicates that the associate Ethernet switch is located
```

at that address and can be controlled via SNMP commands. The "switch-port 4" indicates that the specific data plane link (TE link 11.1.2.1) terminates on port 4 of the associated Ethernet switch#

#the "swcap l2sc encoding ethernet" commands indicates this is an Ethernet TE link in the data plane. GMPLS allows for other types of links like packet, sonet, lambda, fiber, etc. These types of data plane links are not covered in this workshop.#

#the "max-bw" indicates the maximum bandwidth of the data plane link being described. This typically matches the physical capabilities of the link. The values below are in bytes, so number below indicates a 1 Gbps link.#

#the "max-rsv-bw" indicates how much of the total bandwidth is available for dynamic services. For instance if someone had a 10 Gbps link and only wanted dynamic services to take up to 5 Gbps, then, that number could be reflected here.#

#"max-lsp-bw" indicates the maximum granularity for a single lsp provision. This is typically set to the same value as max-rsv-bw, so a single lsp could take all the bandwidth available for that data plane link.#

#the max-lsp-bw #" commands are intended to allow application of policies based on different administrative groups. These are not used in this manner at this time.#

#the "vlan # to #" statement indicates which vlans are available for dynamic services on this data plane link."

#the "metric" statement is a standard value to allow one link to have more (or less) preference over others#

```

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

```

#the above exit command indicates that the description of the data plane link (TE Link 11.1.2.1) associated with the control plane channel (red-gre2) has been fully described. The following statements contain similar information for the other data plane links in the topology for this VLSR. The example below is for red-gre3 without the explanation comments from above.#

```

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

```

C. Check the configuration of the covered Ethernet switches

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 set up to match the features described in the GMPLS *.conf files. Every switch is different in how they handle VLANs, spanning tree, MTUs, and in 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 data plane 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-to-VLAN mappings.

To show the port-to-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) has some peculiarities with regard to creation of new VLANs via SNMP that the DRAGON software does not yet support.

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 is 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 intra-domain service routing functions. NARB is the DRAGON path computation element (PCE) 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 NARB's intra-domain ospfd instance and any one of the VLSR ospfd 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, login to the VLSR with the adjacency to the NARB, i.e. VLSR2 in our case. Edit the `ospfd.conf` file so that the ospfd on vlsr2 will use **yourColor-gre6** to form an adjacency to the intra-domain ospfd process running on the NARB. Again, remember, there is no TE-link between the NARB and the VLSR2. Do **not** add any `ospf-te interface` or `level gmpls` configuration commands on VLSR2. You only need to add these sections to your `ospfd.conf` file on VLSR2:

```
interface yourColor-gre6
  description GRE tunnel yourColor-gre6
  ip ospf network point-to-point
router ospf
  network 10.X.6.0/30 area 0.0.0.0
```

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 dragon daemons how to contact the NARB.

Add the following lines to `/usr/local/dragon/etc/dragon.conf` on each of your VLSRs:

```
configure narb intra-domain ip-address 192.168.X.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 PCs

The final step is to start all of the DRAGON protocol agents. Use the following shell commands on the VLSR PCs to start the VLSR daemons, and then use `ps | grep` to see the DRAGON processes:

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

The `dragon.sh` command starts the `zebra`, `ospf`, `RSVP`, and `dragon` daemons and places them in the background. Note: `dragon.sh` is used to start, restart, stop, status all of the protocol daemons. Invoke `dragon.sh` without any command line arguments 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. Ideally, adjacencies are formed and link state announcements (LSAs) are 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 logging in 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 is 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.

For example, to check intra-domain topology via the 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. 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-vlsr1`, we can query `red-narb` (192.168.1.10) to see 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` files on **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 (e.g. `telnet red-vlsr[1 or 3] 2611`) without restarting the dragon daemon.

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 on both the source and destination. This is fundamentally an issue associated with flat VLANs. Unlike MPLS labels, the VLAN tag is not actually swapped at each node and 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 create 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)# set vtag any
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, run `ifconfig` on the data interfaces, 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
...
```

Please leave the ping command running while you proceed to the next step.

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 of the data plane interfaces. 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 circuits using OSCARS

Upon completion of Exercise 1, we have constructed several independent networks each capable of dynamically provisioning LSPs across their respective domains. In this exercise, we will introduce a higher level set of services which make use of the capabilities constructed in Exercise 1. The DRAGON control plane provides the ability to dynamically provision across multi-layer, multi-technology network infrastructures. The OSCARS software adds to this a Web Service based AAI (Authentication, Authorization Infrastructure) and Scheduling capabilities. In addition, we will utilize the OSCARS Web Service mechanisms for Inter-Domain messaging associated with multi-domain circuit provisioning. The Inter-Domain component is covered in Exercise 3.

OSCARS is a Java application which uses X.509 certificates for user authentication, SSL Encryption for messaging, and signed SOAP messages for user data containers. Additional details regarding the OSCARS software and architecture is contained documentation referenced here [ref].

As in Exercise #1 with the DRAGON software, we have pre-installed OSCARS and its dependencies in each pod. For more information on how OSCARS was deployed in the pod environment, please refer to sections 1 through 3 of Appendix D, DCN Software Suite: OSCARS Inter-Domain Controller (IDC) Installation Guide. The task for Exercise 2 is to configure OSCARS such that it will integrate with the underlying DRAGON control plane software.

Procedure:

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 Apache Tomcat.

Step 1: 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 NARB/IDC PC at:

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

Your installation comes with a default `oscars.properties` file. Edit the default file and update the sections marked in bold below for your pod. You will need to put in the correct addresses for your VLSRs in the `pss.dragon` section.

```
# login name and password used by the OSCARS server to log into the MySQL
# database - change for your site
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 to 1 once you have SSL
# set up
aaa.secureCookie=1

#### WBUI setting ###
# set default layer to 2 or 3. Set to layer 2 if
# configured to use dragon. Set to 3 for Juniper routers.
wbui.defaultLayer=2

### pathfinder configuration
# can be either 0 (don't find path, only used for aaa testing) or 1
pathfinder.findPath=1

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

#TEDB configuration - can be terce or oscars
tedb.tedbMethod=terce

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

##perfSONAR Lookup Service configuration
# packrat.internet2.edu is a test lookup service run by Internet2
# lookup.url=http://packrat.internet2.edu:8009/perfSONAR_PS/services/LS
lookup.url=http://idc.green.pod.lan:8010/perfSONAR_PS/services/LS

#logging configuration - location of the per reservation logs
# should be ${CATALINA_HOME}/logs
# reservation logs will be put in a subdirectory: reservations
logging.rsvlogdir=/usr/local/tomcat/logs

## PSS configuration
# currently can be dragon or esnet
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.hasNarb=1
pss.dragon.setERO=1
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 that reservation notification is send to - change for your site
mail.webmaster=webmaster@red.net

# notify configuration - add your own classes here,
# make sure the system classloader can see them
notify.observer.1=net.es.oscars.notify.EmailObserver
notify.observer.2=net.es.oscars.notify.FileWriterObserver

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

Step 2: Verify the OSCARS installation

1) Start Apache Tomcat:

```
tomcat55@green-narb:~$ /usr/local/tomcat/bin/startup.sh
Using CATALINA_BASE:   /usr/local/tomcat
Using CATALINA_HOME:   /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME:        /usr/local/java5
```

Wait for the server finish starting by watching the output in

/usr/local/tomcat/logs/catalina.out:

```
tomcat55@green-narb:~$ tail -f /usr/local/tomcat/logs/catalina.out
...
Jul 16, 2008 2:54:51 PM org.apache.jk.server.JkMain start
INFO: Jk running ID=0 time=0/263 config=null
Jul 16, 2008 2:54:51 PM org.apache.catalina.storeconfig.StoreLoader load
INFO: Find registry server-registry.xml at classpath resource
Jul 16, 2008 2:54:51 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 14107 ms
```

In your web browser, visit [https://idc.\[yourColor\].pod.lan:8443](https://idc.[yourColor].pod.lan: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 Axis2 Installation

In your web browser, visit [https://idc.\[yourColor\].pod.lan:8443/axis2/axis2-admin/](https://idc.[yourColor].pod.lan:8443/axis2/axis2-admin/) and login with:

Username: admin
Password: axis2

On the left bar, click on “Available Services” and verify that you see the OSCARS and TERCE services listed on the right. For more information about the Axis2 installation, please refer to sections 8.1 and 8.2 of Appendix D.

3) Verifying OSCARS Web-Based User Interface (WBUI) Installation

In your web browser, visit [https://idc.\[yourColor\].pod.lan:8443/OSCARS](https://idc.[yourColor].pod.lan:8443/OSCARS)

You will be presented with the OSCARS login form. However, you will not be able to login just yet. We need to create a new user account before attempting to login.

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 3: Creating the First User Account in OSCARS

As shown below, execute the `idc-useradd` script located in the `/home/tomcat55/dcn-software-suite-0.3.1/idc/tools/utils/` directory to create your first OSCARS user so that you can log into the web interface.

```
tomcat55@green-narb:~$ cd /home/tomcat55/dcn-software-suite-0.3.1/idc/tools/utils
tomcat55@green-narb:~/dcn-software-suite-0.3.1/idc/tools/utils$ ./idc-useradd
* indicates a required field
Login*: oscars-admin
Password*: oscars (password will not echo to the console)
Confirm Password*: oscars (password will not echo to the console)
First Name*: OSCARS
Last Name*: Administrator
Cert Subject: [leave blank]
Cert Issuer: [leave blank]

1. Energy Sciences Network
2. Internet2
Select the user's organization (by number): 2

1. OSCARS-user
2. OSCARS-engineer
3. OSCARS-administrator
4. OSCARS-service
5. OSCARS-operator
Select the user's role(s) (numbers separated by spaces): 2 3
Personal Description: [leave blank]
Email(Primary)*: myemail@mydomain.net
Email(Secondary): [leave blank]
Phone(Primary)*: 1234567890
Phone(Secondary): [leave blank]
New user 'oscars-admin' added.
tomcat55@green-narb:~/dcn-software-suite-0.3.1/idc/tools/utils$
```

Step 4: Configure TERCE

TERCE is a web service component 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
```

Modify the `terce-ws.properties` file to have the inter-domain and intra-domain 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. The static RCE file should point to `/usr/local/tomcat/shared/classes/terce.conf/static-routes.xml`.

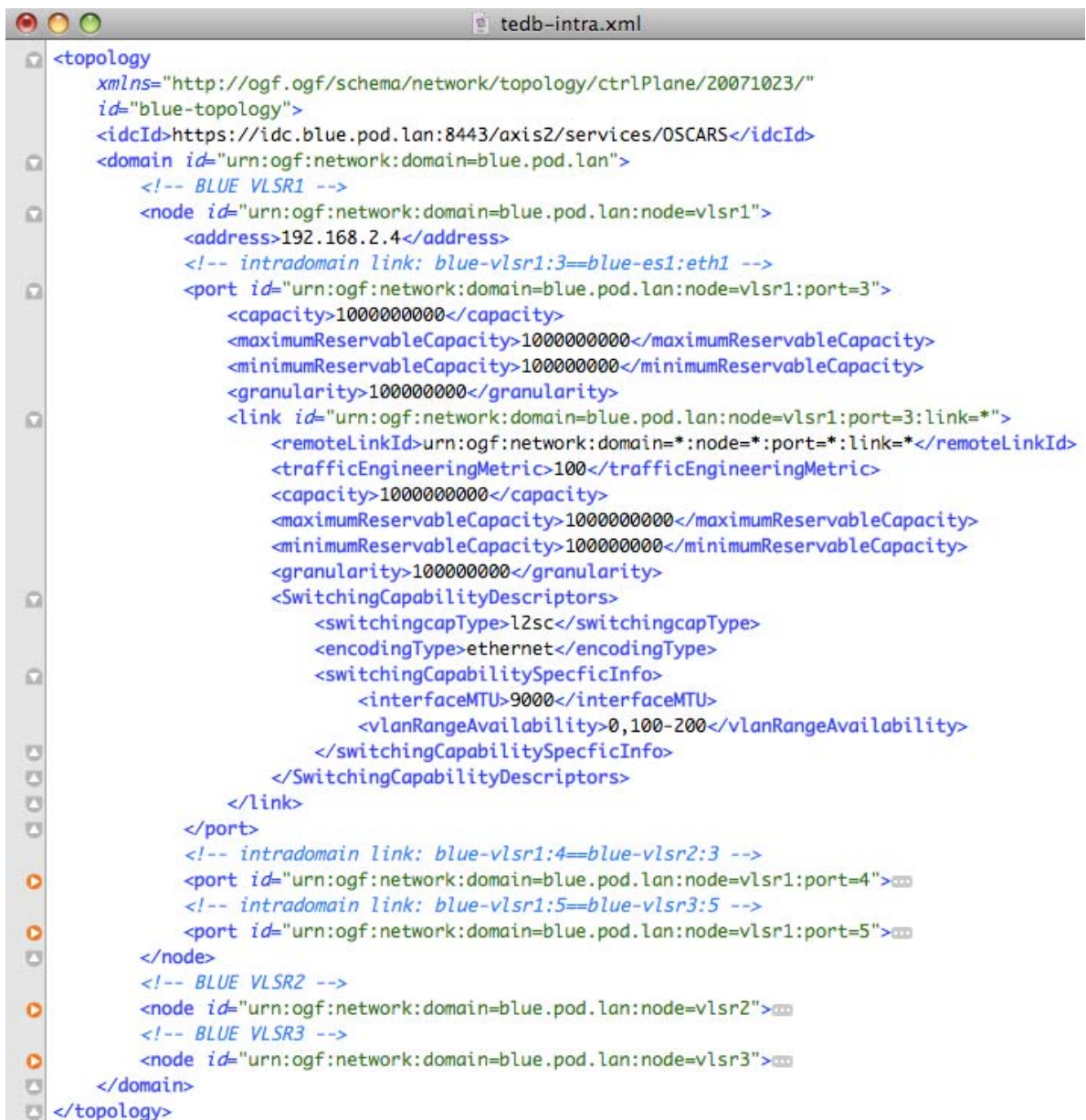
For more information about the format of the `tedb-intra.xml`, `tedb-inter.xml`, and `static-routes.xml` files, please see section 5 of Appendix D. We will be working with these files in the next two steps and examples of each file are provided.

Step 5: Describing the data plane topology in XML format

As of the current release, OSCARS must be provided with a static XML description of the data plane topology. The next software release will gather this information dynamically from the DRAGON NARB. We have already put a working copy of this file on each NARB at the following location:

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

Please examine this file to understand how it describes the data plane topology of your pod. The following is a screenshot of the blue pod's `tedb-intra.xml` file after being loaded into TextMate. TextMate understands the format of XML files so blocks can be folded or collapsed and the syntax is highlighted, making it easier to read.



```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- BLUE VLSR1 -->
<!-- BLUE VLSR2 -->
<!-- BLUE VLSR3 -->
```

```
<!-- BLUE VLSR1 -->
<node id="urn:ogf:network:domain=blue.pod.lan:node=vlsr1">
  <address>192.168.2.4</address>
  <!-- intradomain link: blue-vlsr1:3==blue-es1:eth1 -->
  <port id="urn:ogf:network:domain=blue.pod.lan:node=vlsr1:port=3">
    <capacity>100000000</capacity>
    <maximumReservableCapacity>100000000</maximumReservableCapacity>
    <minimumReservableCapacity>100000000</minimumReservableCapacity>
    <granularity>100000000</granularity>
    <link id="urn:ogf:network:domain=blue.pod.lan:node=vlsr1:port=3:link="*">
      <remotelinkId>urn:ogf:network:domain=*:node=*:port=*:link="*"</remotelinkId>
      <trafficEngineeringMetric>100</trafficEngineeringMetric>
      <capacity>100000000</capacity>
      <maximumReservableCapacity>100000000</maximumReservableCapacity>
      <minimumReservableCapacity>100000000</minimumReservableCapacity>
      <granularity>100000000</granularity>
      <SwitchingCapabilityDescriptors>
        <switchingcapType>l2sc</switchingcapType>
        <encodingType>ethernet</encodingType>
        <switchingCapabilitySpecficInfo>
          <interfaceMTU>9000</interfaceMTU>
          <vlanRangeAvailability>0,100-200</vlanRangeAvailability>
        </switchingCapabilitySpecficInfo>
      </SwitchingCapabilityDescriptors>
    </link>
  </port>
  <!-- intradomain link: blue-vlsr1:4==blue-vlsr2:3 -->
  <port id="urn:ogf:network:domain=blue.pod.lan:node=vlsr1:port=4">
  <!-- intradomain link: blue-vlsr1:5==blue-vlsr3:5 -->
  <port id="urn:ogf:network:domain=blue.pod.lan:node=vlsr1:port=5">
</node>
<!-- BLUE VLSR2 -->
<node id="urn:ogf:network:domain=blue.pod.lan:node=vlsr2">
<!-- BLUE VLSR3 -->
<node id="urn:ogf:network:domain=blue.pod.lan:node=vlsr3">
```

Step 6: Generate Intra-domain Routes

The TERCE component requires a `static-routes.xml` file to be placed in the `/usr/local/tomcat/shared/classes/terce.conf/` directory. This file contains an XML description of each intra-domain route.

Currently, a perl script is used to automatically generate this file by querying the underlying DRAGON NARB. It uses the `tedb-intra.xml` topology file as a guide – to determine edge ports, core links, and the IP addresses of every possible source and destination.

You will need to be logged into the NARB and the current working directory should be `/home/tomcat55/dcn-software-suite-0.3.1/idc/tools/updatedbterce/`. From this directory, you will execute the `buildStaticRoutesXML.pl` script with the arguments shown below. The output below is from the blue pod.

```
tomcat55@blue-narb:~$ cd /home/tomcat55/dcn-software-suite-0.3.1/idc/tools/updatedbterce/
tomcat55@blue-narb:~/dcn-software-suite-0.3.1/idc/tools/updatedbterce$ ./buildStaticRoutesXML.pl -v \
--xmltopo=/usr/local/tomcat/shared/classes/terce.conf/tedb-intra.xml \
--outfile=/usr/local/tomcat/shared/classes/terce.conf/static-routes.xml
buildStaticRoutesXML.pl: 2008-07-16 18:04:20 [1] domain is blue.pod.lan
buildStaticRoutesXML.pl: 2008-07-16 18:04:20 [1] address for node vlsr1 is 192.168.2.4
buildStaticRoutesXML.pl: 2008-07-16 18:04:20 [1] address for node vlsr2 is 192.168.2.6
buildStaticRoutesXML.pl: 2008-07-16 18:04:20 [1] address for node vlsr3 is 192.168.2.8
buildStaticRoutesXML.pl: 2008-07-16 18:04:20 [1] Successfully queried NARB for src=192.168.2.6,
dst=192.168.2.8
buildStaticRoutesXML.pl: 2008-07-16 18:04:20 [1] Successfully queried NARB for src=192.168.2.6,
dst=192.168.2.4
buildStaticRoutesXML.pl: 2008-07-16 18:04:20 [1] Successfully queried NARB for src=192.168.2.8,
dst=192.168.2.6
buildStaticRoutesXML.pl: 2008-07-16 18:04:20 [1] Successfully queried NARB for src=192.168.2.8,
dst=192.168.2.4
buildStaticRoutesXML.pl: 2008-07-16 18:04:20 [1] Successfully queried NARB for src=192.168.2.4,
dst=192.168.2.6
buildStaticRoutesXML.pl: 2008-07-16 18:04:20 [1] Successfully queried NARB for src=192.168.2.4,
dst=192.168.2.8
buildStaticRoutesXML.pl: 2008-07-16 18:04:20 [1] Writing out ERO for path 'vlsr1:3--vlsr3:3'
buildStaticRoutesXML.pl: 2008-07-16 18:04:20 [1] Writing out ERO for path 'vlsr1:3--vlsr2:7'
buildStaticRoutesXML.pl: 2008-07-16 18:04:20 [1] Writing out ERO for path 'vlsr3:3--vlsr1:3'
buildStaticRoutesXML.pl: 2008-07-16 18:04:20 [1] Writing out ERO for path 'vlsr3:3--vlsr2:7'
buildStaticRoutesXML.pl: 2008-07-16 18:04:20 [1] Writing out ERO for path 'vlsr2:7--vlsr1:3'
buildStaticRoutesXML.pl: 2008-07-16 18:04:20 [1] Writing out ERO for path 'vlsr2:7--vlsr3:3'
```

Please work with the course instructors if the script gives you any warnings or errors. This would indicate that you have a problem in the underlying control plane configuration from Exercise #1.

Check that the output file `/usr/local/tomcat/shared/classes/terce.conf/static-routes.xml` contains entries similar to the screenshot below. The only path expanded below is the one between blue-vlsr1 port 3 and blue-vlsr3 port 3. The other paths have been collapsed in the screenshot:

```

static-routes.xml
<staticPathDatabase xmlns="http://dragon.maxgigapop.net/DRAGON/TERCE/RCE/STATICPATHS" id="dragon-paths">
  <staticPathEntry id="vlsr1:3--vlsr3:3">
    <srcEndpoint>urn:ogf:network:domain=blue.pod.lan:node=vlsr1:port=3:link=*</srcEndpoint>
    <destEndpoint>urn:ogf:network:domain=blue.pod.lan:node=vlsr3:port=3:link=*</destEndpoint>
    <path id="vlsr1:3--vlsr3:3">
      <hop id="1">
        <linkIdRef>urn:ogf:network:domain=blue.pod.lan:node=vlsr1:port=3:link=*</linkIdRef>
      </hop>
      <hop id="2">
        <linkIdRef>urn:ogf:network:domain=blue.pod.lan:node=vlsr1:port=5:link=11.2.3.1</linkIdRef>
      </hop>
      <hop id="3">
        <linkIdRef>urn:ogf:network:domain=blue.pod.lan:node=vlsr3:port=5:link=11.2.3.2</linkIdRef>
      </hop>
      <hop id="4">
        <linkIdRef>urn:ogf:network:domain=blue.pod.lan:node=vlsr3:port=3:link=*</linkIdRef>
      </hop>
    </path>
    <availableVtags></availableVtags> <!-- deprecated: leave blank -->
  </staticPathEntry>
  <staticPathEntry id="vlsr1:3--vlsr2:7">
  <staticPathEntry id="vlsr3:3--vlsr1:3">
  <staticPathEntry id="vlsr3:3--vlsr2:7">
  <staticPathEntry id="vlsr2:7--vlsr1:3">
  <staticPathEntry id="vlsr2:7--vlsr3:3">
</staticPathDatabase>

```

Step 7: Populate the Scheduling Database

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

1) Define Your Local Domain

You must define your local domain in the scheduling database so the `updateTopology.sh` script in the next step knows which links are local. This is done by running the `idc-domainadd` command in the `/home/tomcat55/dcn-software-suite-0.3.1/idc/tools/utils` directory. The example below shows the procedure that should be run on the red pod:

```

tomcat55@red-narb:~$ cd /home/tomcat55/dcn-software-suite-0.3.1/idc/tools/utils
tomcat55@red-narb:~/dcn-software-suite-0.3.1/idc/tools/utils$ ./idc-domainadd
Topology Identifier (i.e. mydomain.net)*: red.pod.lan
IDC URL*: https://idc.red.pod.lan:8443/axis2/services/OSCARS
Descriptive Name (for display purposes)*: red-pod
Abbreviated Name (for display purposes)*: red
Is this your IDC's local domain? [y/n] y
New domain 'red.pod.lan' added.

```

2) Run `updateTopology.sh`

The `updateTopology.sh` script syncs the database with your XML topology files. Follow the procedure below to run this script:

```
tomcat55@red-narb:~$ cd ~/dcn-software-suite-0.3.1/idc/tools/updatedbterce/
tomcat55@red-narb:~/dcn-software-suite-0.3.1/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: *
Complete.
tomcat55@red-narb:~/dcn-software-suite-0.3.1/idc/tools/updatedbterce$
```

If no error is returned, your database is now populated with the topology information from your XML topology file.

Step 8: Start the Path Scheduler

The path scheduler is a process that automatically builds circuits for users who do not wish to use signaling. Examples of circuits that do not use signaling include those which are provisioned via the web-based user interface. The path scheduler also handles deletion of expired reservations. Running the scheduler is required for the IDC to function properly.

1) Start the scheduler with the following command:

```
tomcat55@red-narb:~$ cd ~/dcn-software-suite-0.3.1/idc/tools/schedulers
tomcat55@red-narb:~/dcn-software-suite-0.3.1/idc/tools/schedulers$ nohup
./scheduler.sh > ~/scheduler.log &
```

The above command will run the scheduler in the background. The output will actually go into `/usr/local/tomcat/logs/scheduler.log`, **not** `~/scheduler.log`. For more information about the path scheduler, please see section 6.2 of Appendix D. To stop the scheduler, use the following commands:

```
tomcat55@red-narb:~$ killall scheduler.sh
tomcat55@red-narb:~$ pgrep -f PathScheduler | xargs -r kill
tomcat55@red-narb:~$ rm -f /tmp/lock.scheduler.sh
```

Step 9: Schedule and Provision the intra-domain LSP

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

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

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

Open a web browser and connect to **“[https://idc.\[yourColor\].pod.lan:8443/OSCARS/](https://idc.[yourColor].pod.lan:8443/OSCARS/)”**. Login with username **oscars-admin** and password **oscars**, or whatever username and password you chose when you ran the `idc-useradd` script in the previous step. Again, please note that Internet Explorer is **not** currently supported. You must use a Mozilla-based browser with the current OSCARS release.

On-demand Secure Circuits and Advance Reservation System

July 14, 2008 18:02

Log in to OSCARS.

Login/Logout

User Name:

Password:

LOGIN

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.

NOTE: This is the new Web-based interface based on the [Dojo](#) toolkit. Please report any problems to the administrators listed at the bottom of this page. Do not use the browser's back or forward button while in this application.

Basic OSCARS functionality is as follows:

- Clicking on the Reservations tab will bring up a default list of all pending and active reservations. You can modify the reservations listed through the input fields at the top. Clicking on a row will bring up a reservation's details.
- Clicking on the Create Reservation tab will bring up a form to create a reservation.
- Clicking on the Users tab brings up a list of OSCARS users and associated information. Clicking on a row from that list brings up that user's details. You currently need OSCARS admin privileges to view other users' information.

More documentation can be found by clicking on the link at the bottom of this page.

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

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

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

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

```
urn:ogf:network:domain=[color].pod.lan:node=[vlsr1 or vlsr3]:port=3:link=*
```

For example:

To refer to blue-es1, you would use:

```
urn:ogf:network:domain=blue.pod.lan:node=vlsr1:port=3:link=*
```

To refer to yellow-es2, you would use:

```
urn:ogf:network:domain=yellow.pod.lan:node=vlsr3:port=3:link=*
```

Recall that we configured the `lookup.url` parameter in the `oscars.properties` file. This parameter was set to http://idc.green.pod.lan:8010/perfSONAR_PS/services/LS.

To make provisioning easier, hostnames may be generated for the URNs on the edge of each network. This allows a user to enter a string such as `es1.red.pod.lan` instead of `urn:ogf:network:domain=red.pod.lan:node=vlsr1:port=3:link=*`.

This is precisely what the lookup service that is configured in the green pod is designed to do. You may use the URN format, or you may use the hostnames to refer to the source and destination nodes. The screenshot below shows an example of using hostnames for the source/destination fields, but the form could have been filled out using the URN notation. For more information about the Lookup Service, please see section 6.3 of Appendix D.

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

```
Source: urn:ogf:network:domain=green.pod.lan:node=vlsr1:port=3:link=*
Destination: urn:ogf:network:domain=green.pod.lan:node=vlsr3:port=3:link=*
Bandwidth (Mbps): 1000
Description: describe what this LSP is for, i.e. testing
VLAN: [leave blank for any]
```

The following values will also work:

```
Source: es1.green.pod.lan
Destination: es2.green.pod.lan
Bandwidth (Mbps): 1000
Description: describe what this LSP is for, i.e. testing
VLAN: [leave blank for any]
```

Once the form is filled out, click the **Create Reservation** button. An example on the green pod is given below.

July 15, 2008 18:09

Reservation creation form

Reservations	Reservation Details	Create Reservation	User List	Add User	User Profile	Login/Logout
--------------	---------------------	--------------------	-----------	----------	--------------	--------------

Required inputs are bordered in green. The source and destination can be topology identifiers, host names, or IP addresses, depending on the layer used. Click on the boxes associated with the start and end dates to bring up a calendar widget. The reservation time slot defaults to now, and now + 4 minutes, respectively, if you leave the dates and times empty.

WARNING: Entering a series of hops in the Path field may alter routing behavior for the selected flow. Hops can be topology identifiers, host names, or IP addresses, depending on the layer used. Note that the path field will expand to the number of lines occurring in the hops list.

Production circuit

Source	<input type="text" value="es1.green.pod.lan"/>
Destination	<input type="text" value="es2.green.pod.lan"/>
Path (series of hops)	<input type="text"/>
Bandwidth (Mbps)	<input type="text" value="1,000"/> (10-10000)
Description	<input type="text" value="LSP between green-es1 and green-es2"/> (For our records)
Start date	<input type="text" value="7/15/2008"/>
Start time	<input type="text" value="18:09"/>
End date	<input type="text" value="7/15/2008"/>
End time	<input type="text" value="18:13"/>

Use layer 2 parameters Use layer 3 parameters

VLAN	<input type="text" value="any"/> tag, or range, e.g. 3000-3100
Source Port	<input type="text" value="Tagged"/>
Destination Port	<input type="text" value="Tagged"/>

You may leave all other fields blank. This will create an LSP with your required bandwidth, starting as soon as possible. Since you haven't put in the end time of this LSP, the default will leave this LSP up for 5 minutes. After 5 minutes, the LSP is automatically torn down. The VLAN field can be left blank, or filled in with the keyword 'any', or with a value between 100 and 200.

Once you click "Create Reservation", 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" within about 30-60 seconds.

Your screen should look something like the following after clicking Create Reservation and Refresh button several times. If your screen indicates that the status is still PENDING, try clicking Refresh again.

July 15, 2008 18:13

Successfully got reservation details for green.pod.lan-1

Reservations	Reservation Details	Create Reservation	User List	Add User	User Profile	Login/Logout
--------------	---------------------	--------------------	-----------	----------	--------------	--------------

NEW GRI	<input type="text"/>	QUERY
---------	----------------------	-------

REFRESH	MODIFY	CANCEL	CLONE
---------	--------	--------	-------

GRI	green.pod.lan-1
Status	ACTIVE
User	oscars-admin
Description	LSP between green-es1 and green-es2
Start date	7/15/2008
Start time	18:13
End date	7/15/2008
End time	18:17
Created time	2008/07/15 18:13
Bandwidth (Mbps)	1000
Source	urn:ogf:network:domain=green.pod.lan:node=vlsr1:port=3:link=*
Destination	urn:ogf:network:domain=green.pod.lan:node=vlsr3:port=3:link=*
Intradomain hops	urn:ogf:network:domain=green.pod.lan:node=vlsr1:port=3:link=*
	urn:ogf:network:domain=green.pod.lan:node=vlsr1:port=5:link=11.4.3.1
	urn:ogf:network:domain=green.pod.lan:node=vlsr3:port=5:link=11.4.3.2
	urn:ogf:network:domain=green.pod.lan:node=vlsr3:port=3:link=*
Interdomain path	
VLAN	100
Tagged	true

In the above example, OSCARS has chosen to use VLAN 100, since we let it pick the VLAN tag instead of providing our own.

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/end times are correct and that the time on your computer is valid if you left the “Time” fields blank.

```
/usr/local/tomcat/logs/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 /usr/local/tomcat/logs/scheduler.log
2008-07-15 22:14:01,273 [INFO] green-vlsr1-pc>
2008-07-15 22:14:01,274 [INFO] edit lsp green.pod.lan-1
2008-07-15 22:14:01,275 [INFO] green-vlsr1-pc(edit-lsp-green.pod.lan-1)#
2008-07-15 22:14:01,275 [INFO] green-vlsr1-pc(edit-lsp-green.pod.lan-1)#
2008-07-15 22:14:01,277 [INFO] set source ip-address 192.168.4.4 tagged-group 100 destination ip-
address 192.168.4.8 tagged-group 100
2008-07-15 22:14:01,278 [INFO] green-vlsr1-pc(edit-lsp-green.pod.lan-1)#
2008-07-15 22:14:01,280 [INFO] green-vlsr1-pc(edit-lsp-green.pod.lan-1)#
2008-07-15 22:14:01,281 [INFO] set bandwidth gige swcap l2sc encoding ethernet gpid ethernet
2008-07-15 22:14:01,282 [INFO] green-vlsr1-pc(edit-lsp-green.pod.lan-1)#
2008-07-15 22:14:01,283 [INFO] green-vlsr1-pc(edit-lsp-green.pod.lan-1)#
2008-07-15 22:14:01,284 [INFO] set ero-hop-ipv4 strict ip-address 11.4.3.1
2008-07-15 22:14:01,284 [INFO] green-vlsr1-pc(edit-lsp-green.pod.lan-1)#
2008-07-15 22:14:01,284 [INFO] green-vlsr1-pc(edit-lsp-green.pod.lan-1)#
2008-07-15 22:14:01,287 [INFO] set ero-hop-ipv4 strict ip-address 11.4.3.2
2008-07-15 22:14:01,287 [INFO] green-vlsr1-pc(edit-lsp-green.pod.lan-1)#
2008-07-15 22:14:01,288 [INFO] green-vlsr1-pc(edit-lsp-green.pod.lan-1)#
2008-07-15 22:14:01,289 [INFO] set vtag 100
2008-07-15 22:14:01,289 [INFO] green-vlsr1-pc(edit-lsp-green.pod.lan-1)#
2008-07-15 22:14:01,289 [INFO] green-vlsr1-pc(edit-lsp-green.pod.lan-1)#
2008-07-15 22:14:01,290 [INFO] exit
2008-07-15 22:14:01,290 [INFO] green-vlsr1-pc>
2008-07-15 22:14:01,291 [INFO] green-vlsr1-pc>
2008-07-15 22:14:01,310 [INFO] commit lsp green.pod.lan-1
2008-07-15 22:14:01,310 [INFO] green-vlsr1-pc>
2008-07-15 22:14:01,310 [INFO] created lsp green.pod.lan-1
2008-07-15 22:14:01,311 [INFO] green-vlsr1-pc>
2008-07-15 22:14:01,312 [INFO] show lsp green.pod.lan-1
2008-07-15 22:14:01,312 [INFO] Src 192.168.4.4/100, dest 192.168.4.8/100
2008-07-15 22:14:01,312 [INFO] Generic TSPEC R=gige, B=gige, P=gige, m=100, M=1500
2008-07-15 22:14:01,312 [INFO] Encoding ethernet, Switching l2sc, G-Pid ethernet
2008-07-15 22:14:01,313 [INFO] Ingress Local ID Type: tagged group, Value: 100
2008-07-15 22:14:01,313 [INFO] Egress Local ID Type: tagged group, Value: 100.
2008-07-15 22:14:01,314 [INFO] E2E LSP VLAN Tag: 100.
2008-07-15 22:14:01,314 [INFO] Status: Commit
2008-07-15 22:14:01,314 [INFO] green-vlsr1-pc>
2008-07-15 22:14:06,330 [INFO] green-vlsr1-pc>
2008-07-15 22:14:06,331 [INFO] show lsp green.pod.lan-1
2008-07-15 22:14:06,331 [INFO] Src 192.168.4.4/100, dest 192.168.4.8/100
2008-07-15 22:14:06,332 [INFO] GRI: 1727633764-2147483649
2008-07-15 22:14:06,332 [INFO] Generic TSPEC R=gige, B=gige, P=gige, m=100, M=1500
2008-07-15 22:14:06,332 [INFO] Encoding ethernet, Switching l2sc, G-Pid ethernet
2008-07-15 22:14:06,333 [INFO] Ingress Local ID Type: tagged group, Value: 100
2008-07-15 22:14:06,333 [INFO] Egress Local ID Type: tagged group, Value: 100.
2008-07-15 22:14:06,334 [INFO] E2E LSP VLAN Tag: 100.
2008-07-15 22:14:06,334 [INFO] Status: In service
2008-07-15 22:14:06,334 [INFO] green-vlsr1-pc>
2008-07-15 22:14:06,338 [INFO] green.pod.lan-1 is IN SERVICE
```

You will see the CLI commands that OSCARS 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 green-vlsr1; telnet to localhost 2611 and check on the LSP.

```

green-vlsr1-pc> show lsp
                        **LSP status summary**
Name          Status      Dir    Source (IP/LSP ID)  Destination (IP/Tunnel ID)
-----
green.pod.lan-1
                In service <=>  192.168.4.4        192.168.4.8
                                100                100
green-vlsr1-pc> show lsp green.pod.lan-1
Src 192.168.4.4/100, dest 192.168.4.8/100
GRI: 1727633764-2147483649
Generic TSPEC R=gige, B=gige, P=gige, 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 green_es1 as “root”:

```

green_es1_pc:~# /sbin/modprobe 8021q
green_es1_pc:~# vconfig add eth1 100
Added VLAN with VID == 100 to IF -:eth1:-
green_es1_pc:~# ifconfig eth1.100 10.0.0.1 netmask 255.255.255.252

```

Then, login to green_es2 as “root”, and repeat what you did for green_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 “book ahead” reservation

Repeats steps 1-3 above, but fill in the start/end time fields. For example, try to “schedule” an LSP to come up 4 minutes in the future and stay active for 10 minutes.

Step 10: Provision a circuit using the example Java client

Circuits do not exclusively need to be setup using the Web-Based User Interface. They may be setup by an application, such as a Java program. An example Java client is included in the DCN Software Suite. This Java client is designed to use the Web Services interface to send a signed SOAP message to OSCARS. The client has been pre-loaded with an X.509 user certificate. In order for the client to be able to create a reservation, we must first add a user to OSCARS so that it recognizes the X.509 certificate presented by the client. Run the `idc-useradd` script with the following parameters to create this user for the example Java client:

```

tomcat55@red-narb:~$ cd /home/tomcat55/dcn-software-suite-0.3.1/idc/tools/utills
tomcat55@red-narb:~/dcn-software-suite-0.3.1/idc/tools/utills$ ./idc-useradd
* indicates a required field
Login*: alice
Password*: anyPassword (password will not echo to the console)
Confirm Password*: anyPassword (password will not echo to the console)
First Name*: Alice
Last Name*: Alice
Cert Subject: CN=Alice, OU=OASIS Interop Test Cert, O=OASIS
Cert Issuer: [leave blank]

1. Energy Sciences Network
2. Internet2
Select the user's organization (by number): 2

1. OSCARS-user
2. OSCARS-engineer
3. OSCARS-administrator
4. OSCARS-service
5. OSCARS-operator
Select the user's role(s) (numbers separated by spaces): 1
Personal Description: [leave blank]
Email(Primary)*: alice@alicedomain.net
Email(Secondary): [leave blank]
Phone(Primary)*: 1234567890
Phone(Secondary): [leave blank]
New user 'alice' added.

```

Next, setup the Java client by creating a new file in the following directory:

```

tomcat55@red-narb:~ $ cd ~/dcn-software-suite-0.3.1/idc/examples/javaClients/
tomcat55@red-narb:~/dcn-software-suite-0.3.1/idc/examples/javaClients$ emacs
myCircuit.properties

```

The **myCircuit.properties** must contain parameters as shown below. Please view the **example.12.properties** (that's 12 as in layer 2, **not** the number 12) in the same directory for a description of all of the available parameters.

```

interactive=0
url1=https://idc.red.pod.lan:8443/axis2/services/OSCARS/
gri=
layer=2
sourceEndpoint=es1.red.pod.lan
destEndpoint=es2.green.pod.lan
vtag=any
bandwidth=100
description=testing-java-client
pathSetupMode=timer-automatic
start_time=
end_time=
duration=0.5

```

Launch the example Java client by using the **createRes.sh** script. You may view the status of the reservation in the WBUI, or with the **listRes.sh** script in the **javaClients** directory.

```

tomcat55@red-narb:~/dcn-software-suite-0.3.1/idc/examples/javaClients$
./createRes.sh -pf myCircuit.properties

```

Exercise #3: Inter-domain Provisioning with OSCARS

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

Inter-domain operations include use of the IDC protocol. This protocol is a product of the DICE Control Plane Working Group. DICE is an acronym for an informal collaboration between Dante, Internet2, Canarie and ESNNet. The DICE control plane working group has met over the past 2 years to define the architecture and protocol use by an IDC. The protocol has been implemented by Internet2, ESNNet, and GEANT and currently interconnects dynamic circuit networks at all three organizations.

The IDC protocol defines messages for reserving network resources, signaling resource provisioning, gathering information about previously requested resources, and basic topology exchange. These messages are defined in a SOAP web service format. Since all messages are defined using SOAP, the protocol also utilizes a few external web service protocols and XML descriptions for features such as security and topology description.

In the previous exercise, we scheduled and provisioned intra-domain LSPs. In this exercise, we will connect 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 different 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 Appendix B. Add the appropriate data circuit IDs, i.e. D11, D12 and D13, to your list of data links created in Exercise #1.

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

Step 2: Configure OSCARS for inter-domain

In Exercise #2, the XML topology file you were given (`tedb-intra.xml`) already contained the inter-domain link. Thus, the inter-domain link has already been added to the OSCARS scheduling database, and the `static-routes.xml` file already has routes defined to your inter-domain edge port.

In this step, you will generate a certificate signing request, get it signed by the CA, install the signed certificate, tell OSCARS about the other domains, and where exactly it may reach your neighboring IDC.

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 three 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. Use the following commands to verify that the green pod CA is installed in your keystores:

```
> ssh tomcat55@red-narb
tomcat55@red-narb:~$ cd ~/dcn-software-suite-0.3.1/idc/tools/utils/
tomcat55@red-narb:dcn-software-suite-0.3.1/idc/tools/utils/$ ./idc-certview
Choose the keystore or file with the certificate you'd like to view:
  1. sec-client.jks - stores the private key and certificate used to send messages to
other IDCs
  2. sec-server.jks - stores certificates trusted to sign incoming messages
  3. ssl-keystore.jks - stores SSL certificates of other IDCs running HTTPS
  4. Open certificate file...
Enter choice: 2

Choose the certificate you wish to view:
  1. alice
  2. greenca
  3. ca
  4. root
  5. bob

Enter choice: 2

-- Certificate details

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 `idc-certview` command shows that **greenCA** has been installed in your machine as a trusted CA. The other listings are default certificates that may be used for testing but should be deleted in production environments. They will not be used in this course and can be ignored.

You will now create your own private key and certificate using the `idc-certadd` script:

```
tomcat55@red-narb:~$ cd ~/dcn-software-suite-0.3.1/idc/tools/utils/
tomcat55@red-narb:/usr/local/tomcat/shared/classes/repo % ./idc-certadd
What would you like to do?
    1. Create a new certificate my IDC will use in outgoing messages to other IDCs
    2. Import a certificate created using choice 1 that was signed by a CA
    3. Trust a CA or another IDC's certificate
Enter choice: 1

-- You have chosen to create a new certificate for sending messages to other IDCs.

Enter an alias for this certificate: redidc
How many days will this certificate be valid?: 3650
-- Using keystore password from /usr/local/tomcat/shared/classes/repo/sec-client.properties
What is your first and last name?
    [Unknown]: idc.red.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

-- Certificate created.
Would you like to generate a Certificate Signing Request (CSR) y/n? y
What filename should I give the CSR?: redidc.csr
-- Certificate Signing Request saved in file red-idc.csr
--- Please send red-idc.csr to your CA for signing.
--- You may then import your signed certificate by running idc-certadd and choosing option 2.
--- Send the following X.509 subject to your neighboring IDCs: CN=idc.red.pod.lan, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=US
```

NOTE: Do not use ~ in the path of the CSR filename, the script has been known to throw errors when this character is used. Also, if you need to generate a new CSR for any reason then you may do so with the `idc-certsignreq` script.

2. Email **redidc.csr** to the Certificate Authority (CA), in our case the workshop instructors, to get your certificate signed. After the instructors sign your certificate signing request, they will e-mail you back the signed certificate.
3. The instructors should e-mail you back a file with a **.cer** extension. You will import the signed certificate by running **idc-certadd** again:

```
tomcat55@red-narb:~$ cd ~/dcn-software-suite-0.3.1/idc/tools/utils/
tomcat55@red-narb:/usr/local/tomcat/shared/classes/repo % ./idc-certadd
What would you like to do?
  1. Create a new certificate my IDC will use in outgoing messages to other IDCs
  2. Import a certificate created using choice 1 that was signed by a CA
  3. Trust a CA or another IDC's certificate
Enter choice: 2

-- You have chosen to import a signed certificate for talking to other domains.

Enter the filename of your signed certificate: /home/tomcat55/redidc.cer
-- Using keystore password from
/usr/local/tomcat/shared/classes/repo/sec-client.properties
-- Using certificate with the alias redidc
--- If this is not the correct alias please exit (Ctrl-C) and
modify the <user> tag in axis2.xml

-- Signed certificate imported
--- Send the following X.509 subject to your neighboring IDCs: CN=idc.red.pod.lan,
OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=US
```

If there are no errors then you may view the signed certificate with the following command:

```
tomcat55@red-narb:~$ cd ~/dcn-software-suite-0.3.1/idc/tools/utils/
tomcat55@red-narb:dcn-software-suite-0.3.1/idc/tools/utils/$ ./idc-certview
Choose the keystore or file with the certificate you'd like to view:
  1. sec-client.jks - stores the private key and certificate used to send
messages to other IDCs
  2. sec-server.jks - stores certificates trusted to sign incoming messages
  3. ssl-keystore.jks - stores SSL certificates of other IDCs running HTTPS
  4. Open certificate file...
Enter choice: 2

Choose the certificate you wish to view:
  1. alice
  2. greenca
  3. ca
  4. root
  5. bob
  6. redidc
Enter choice: 6

-- Certificate details

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

```

You should see the display is much longer than what you saw in the output of the first `idc-certview` command for the greenCA's root certificate on the previous page. This output shows that a chain of trust has been established by the greenCA signing your certificate.

2) Making your IDC Aware of Other Domains

You must add an entry for both direct neighbors and downstream domains using the `idc-domainadd` command. This will add the corresponding rows to the `domains` table in the `bss` MySQL database on your IDC. For more information about this step, see section 7.1 of Appendix D.

First, add your directly connected neighbor – the green pod. Below is an example of how this is done on the red pod:

```

tomcat55@red-narb:~$ cd /home/tomcat55/dcn-software-suite-0.3.1/idc/tools/utils
tomcat55@red-narb:~/dcn-software-suite-0.3.1/idc/tools/utils$ ./idc-domainadd
Topology Identifier (i.e. mydomain.net)*: green.pod.lan
IDC URL*: https://idc.green.pod.lan:8443/axis2/services/OSCARS
Descriptive Name (for display purposes)*: green-pod
Abbreviated Name (for display purposes)*: green
Is this your IDC's local domain? [y/n] n
New domain 'green.pod.lan' added.

```

From the red pod's perspective, entries must also be added for downstream domains. We can enter Unknown in the IDC URL field, since we may not know (or be able to reach) the IDC directly from our local domain. The red IDC will never need to directly contact the IDC in either the blue or yellow domains. These steps are currently necessary so that the local IDC will recognize the topology identifier of downstream domains during provisioning.

```
tomcat55@red-narb:~$ cd /home/tomcat55/dcn-software-suite-0.3.1/idc/tools/utis
tomcat55@red-narb:~/dcn-software-suite-0.3.1/idc/tools/utis$ ./idc-domainadd
Topology Identifier (i.e. mydomain.net)*: blue.pod.lan
IDC URL*: Unknown
Descriptive Name (for display purposes)*: blue-pod
Abbreviated Name (for display purposes)*: blue
Is this your IDC's local domain? [y/n] n
New domain 'blue.pod.lan' added.
```

```
tomcat55@red-narb:~$ cd /home/tomcat55/dcn-software-suite-0.3.1/idc/tools/utis
tomcat55@red-narb:~/dcn-software-suite-0.3.1/idc/tools/utis$ ./idc-domainadd
Topology Identifier (i.e. mydomain.net)*: yellow.pod.lan
IDC URL*: Unknown
Descriptive Name (for display purposes)*: yellow-pod
Abbreviated Name (for display purposes)*: yellow
Is this your IDC's local domain? [y/n] n
New domain 'yellow.pod.lan' added.
```

Similar to the example Java client, we must now add a user so that OSCARS will recognize requests coming from a directly connected neighbor. Run the `idc-useradd` command and add a new user for the green IDC, using **CN=idc.green.pod.lan, OU=Unknown, O=Unknown, ST=Unknown, C=US** as the certificate subject. This user will need the **OSCARS-engineer** and **OSCARS-service** roles in order to have the correct privileges to signal paths via the Web Services interface.

```
tomcat55@red-narb:~$ cd /home/tomcat55/dcn-software-suite-0.3.1/idc/tools/utis
tomcat55@red-narb:~/dcn-software-suite-0.3.1/idc/tools/utis$ ./idc-useradd
* indicates a required field
Login*: greenidc
Password*: anyPassword (password will not echo to the console)
Confirm Password*: anyPassword (password will not echo to the console)
First Name*: Green
Last Name*: IDC
Cert Subject: CN=idc.green.pod.lan, OU=Unknown, O=Unknown, ST=Unknown, C=US
Cert Issuer: [leave blank]

1. Energy Sciences Network
2. Internet2
Select the user's organization (by number): 2

1. OSCARS-user
2. OSCARS-engineer
3. OSCARS-administrator
4. OSCARS-service
5. OSCARS-operator
Select the user's role(s) (numbers separated by spaces): 2 4
Personal Description: [leave blank]
Email(Primary)*: admin@green-domain.net
Email(Secondary): [leave blank]
Phone(Primary)*: 1234567890
Phone(Secondary): [leave blank]
New user 'greenidc' added.
```

The final step to allow inter-domain provisioning is to build your inter-domain routing table. Since the edge pods have only one inter-domain link, the simplest configuration is to simply add a “default” inter-domain route.

This is done by running the `idc-route` command, as shown in the example below.

You should replace the Link ID after `-egress` with the correct Link ID for your pod. You will need to change the `domain` parameter to match your domain identifier, and the `node` parameter to match the border VLSR in your pod that has the inter-domain link to the green pod.

```
tomcat55@red-idc:~$ cd /home/tomcat55/dcn-software-suite-0.3.1/idc/tools/utils/
tomcat55@red-idc:~/dcn-software-suite-0.3.1/idc/tools/utils$ ./idc-route add -default \
  -loose -egress urn:ogf:network:domain=red.pod.lan:node=vlsr3:port=7:link=*
Route added.
```

Verify the inter-domain routing table by running `idc-route` without any arguments:

```
tomcat55@red-idc:~/dcn-software-suite-0.3.1/idc/tools/utils$ ./idc-route
IDC static routing table
ID   Source      Destination   Strict   Multi   Egress
1    default     default      N        N
urn:ogf:network:domain=red.pod.lan:node=vlsr3:port=7:link=*
```

Since the green pod has multiple inter-domain links, we do not use the default route option. Below are the commands we used to setup the inter-domain routing table on the green pod, in case you are interested in how it was configured. You should **not** run the commands below on your pod:

```
./idc-route add -loose -dest urn:ogf:network:domain=red.pod.lan \
  -egress urn:ogf:network:domain=green.pod.lan:node=vlsr1:port=7:link=*
./idc-route add -loose -dest urn:ogf:network:domain=blue.pod.lan \
  -egress urn:ogf:network:domain=green.pod.lan:node=vlsr2:port=7:link=*
./idc-route add -loose -dest urn:ogf:network:domain=yellow.pod.lan \
  -egress urn:ogf:network:domain=green.pod.lan:node=vlsr3:port=7:link=*
```

Again, the above commands were provided for informational purposes only. You do not need to run these commands on your pod. You should only have a single, default inter-domain route on your pod.

Step 5: Provision Inter-domain LSPs via WBUI or example Java client

Repeat the last step in Exercise #2, except that you will have your “source” and “destination” from a different pod. You will need to coordinate with your colleagues in the other domains so that you don’t assign a conflicting IP addresses on the point-to-point/tagged layer 2 links.

Feel free to use either the WBUI or the example Java client to create inter-domain circuits. The example below uses the WBUI, but you could just as easily use the Java client to create these circuits.

Below is an example of provisioning a circuit between red-es1 and yellow-es2. First, the create reservation form:

July 16, 2008 15:29 Reservation creation form

Reservations Reservation Details **Create Reservation** User List Add User User Profile Login/Logout

Required inputs are bordered in green. The source and destination can be topology identifiers, host names, or IP addresses, depending on the layer used. Click on the boxes associated with the start and end dates to bring up a calendar widget. The reservation time slot defaults to now, and now + 4 minutes, respectively, if you leave the dates and times empty.

WARNING: Entering a series of hops in the Path field may alter routing behavior for the selected flow. Hops can be topology identifiers, host names, or IP addresses, depending on the layer used. Note that the path field will expand to the number of lines occurring in the hops list.

 Production circuit

Source

Destination

Path (series of hops)

Bandwidth (Mbps) (10-10000)

Description (For our records)

Start date

Start time

End date

End time

Use layer 2 parameters Use layer 3 parameters

VLAN tag, or range, e.g. 3000-3100

Source Port ▾

Destination Port ▾

Next, the Reservation Details screen showing that the reservation was successful and that the circuit is now ACTIVE:

July 16, 2008 15:30 Successfully got reservation details for red.pod.lan-1

Reservations Reservation Details Create Reservation User List Add User User Profile Login/Logout

NEW GRI QUERY

REFRESH MODIFY CANCEL CLONE

GRI	red.pod.lan-1
Status	ACTIVE
User	oscars-admin
Description	inter-domain test from red-es1 to yellow-es2
Start date	<input type="text" value="7/16/2008"/>
Start time	<input type="text" value="15:29"/>
End date	<input type="text" value="7/16/2008"/>
End time	<input type="text" value="15:33"/>
Created time	2008/07/16 15:29
Bandwidth (Mbps)	1000
Source	urn:ogf:network:domain=red.pod.lan:node=vlsr1:port=3:link=*
Destination	urn:ogf:network:domain=yellow.pod.lan:node=vlsr3:port=3:link=*
Intradomain hops	urn:ogf:network:domain=red.pod.lan:node=vlsr1:port=3:link=*
	urn:ogf:network:domain=red.pod.lan:node=vlsr1:port=5:link=11.1.3.1
	urn:ogf:network:domain=red.pod.lan:node=vlsr3:port=5:link=11.1.3.2
	urn:ogf:network:domain=red.pod.lan:node=vlsr3:port=7:link=*
Interdomain path	
VLAN	100
Tagged	true

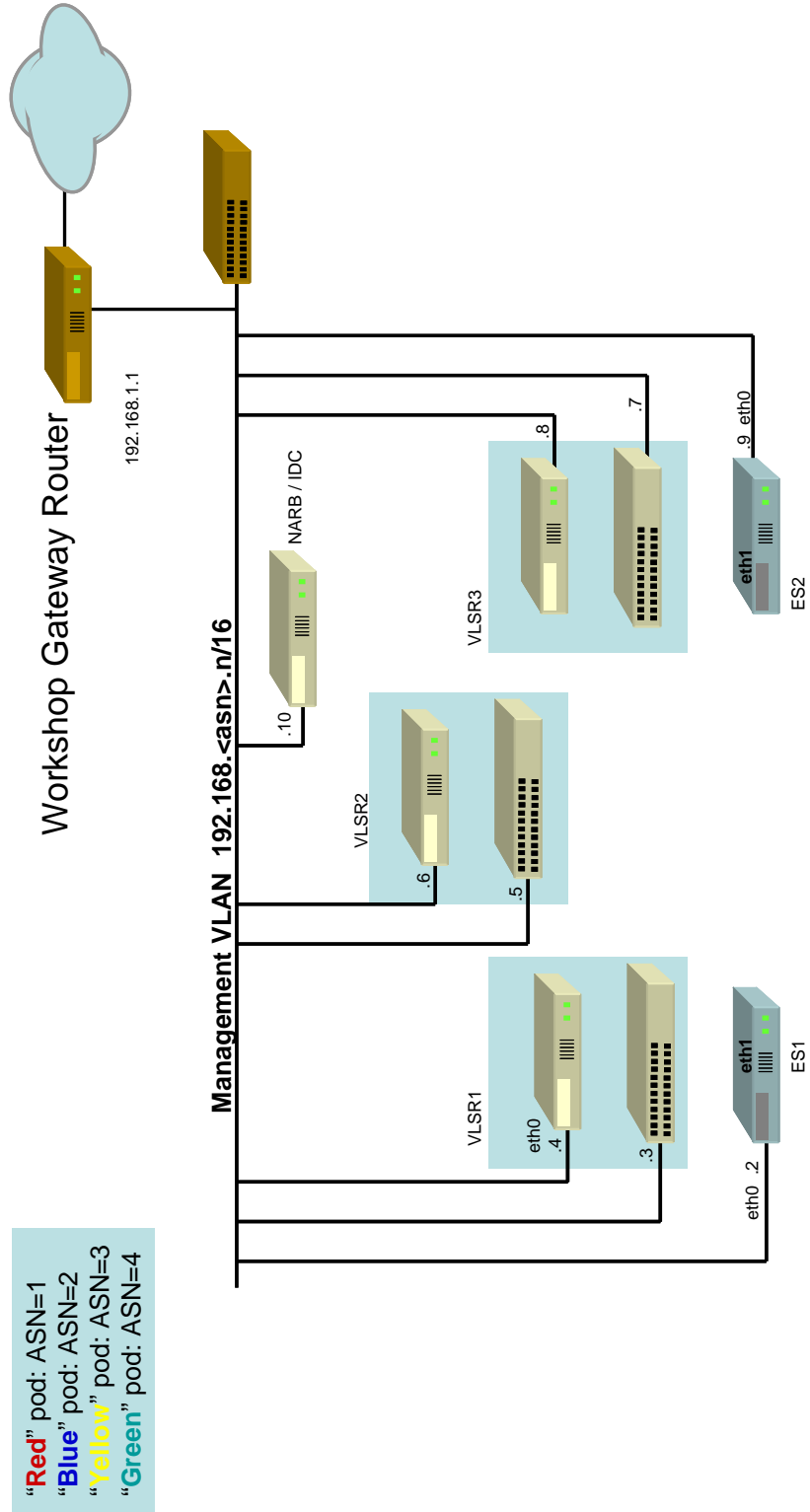
If we login to any of the six VLSRs included in this path, we can see the status of the LSP. For example, since this LSP transits green-vlsr1, we see an LSP with the same GRI if we login to the DRAGON CLI on green-vlsr1 and run `show lsp`:

```
green-vlsr1-pc> show lsp
                        **LSP status summary**
Name           Status      Dir      Source (IP/LSP ID)  Destination (IP/Tunnel ID)
-----
red.pod.lan-1
                In service <=>  192.168.4.4        192.168.4.8
                100                                100
green-vlsr1-pc> show lsp red.pod.lan-1
Src 192.168.4.4/100, dest 192.168.4.8/100
GRI: 1877564093-2147483649
Generic TSPEC R=gige, B=gige, P=gige, 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
```

Appendix A

DCN Workshop Pod Management Plane Configuration and Addressing

Pod Management Addressing



Appendix B

Detailed Pod Architecture and Configuration

Insert the following here:

-11x17 foldout of drawing with all four pods
(workshop-pods.png)

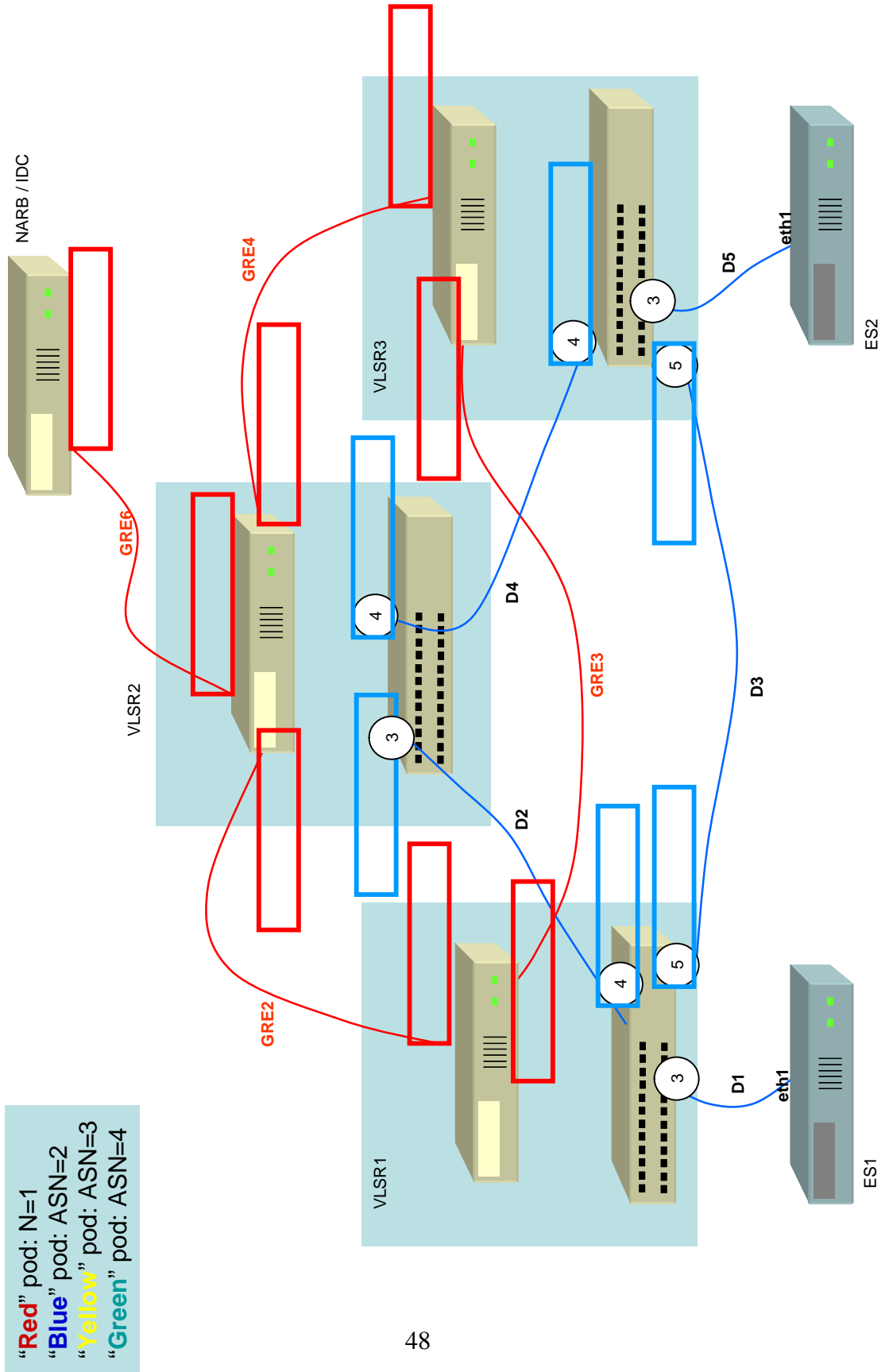
-AddressPlan-v3.xls spread sheet pages with gre and te link configurations for each pod

-Rack Layout drawing

-possibly 11x17 versions of individual pod drawings from above 11x17 foldout

Appendix C

Exercise 1 Steps 1-3 Worksheet



Appendix D

DCN Software Suite Installation Guide

<https://wiki.internet2.edu/confluence/display/DCNSS>