DRAGON PROJECT

DRAGON VLSR Implementation Guide

DRAGON TEAM

University of Maryland Mid-Atlantic Crossroads (UMD MAX)

University of Southern California ISI (USC ISI)

George Mason University (GMU)

Document ID: DRAGON-VLSR-Implementation-v02
Version: DRAFT

http://cnl.gmu.edu/dragon

November 2005

Table of Contents

1.0 Introduction 4
1.1 Background 4
1.2 Sample Configuration 5
2.0 DRAGON Software Suite Components 6
2.1 VLSR 6
2.2 DRAGON OSPF 6
2.3 DRAGON KOM-RSVP 6
2.4 DRAGON ZEBRA 7
3.0 Setting up the Environment for DRAGON Software 8
3.1 Preparation before Installation 8
3.1.1 Hardware Requirements 8
3.1.2 Software Requirements 9
4.0 DRAGON Software Installation Guide 12
5.0 Configure the Switch 13
6.0 Setting up the GRE tunnel 14
7.0 Configuration Guide 16
7.1 OSPF Configuration Guide 16
7.2 RSVP configuration Guide 18
7.3 DRAGON Configuration Guide 18
7.4 VLSR Configuration Guide 19
7.5 ZEBRA Configuration Guide 19
8.0 Running the Daemons 21
8.1 ZEBRA Daemon 21
8.2 OSPF Daemon 21
8.3 DRAGON Daemon 21
8.4 RSVP Daemon 21
9.0 Example of DRAGON Command Line Interface (CLI) 22
10.0 Conclusions 23
Appendix 24
Appendix A: Acronyms and Abbreviations 25
Appendix B: References 26

Summary

This document provides a description of VLSR software implementation as a part of the DRAGON software suite. It includes a discussion of the software components, installation instructions, and configuration guides. A sample configuration has been provided in the description so as to facilitate better understanding of the implementation and installation. The Software has been implemented on Linux 2.4 (Red Hat v9) and the code has been successfully compiled under gcc 3.4.4.

1.0 Introduction

This section provides a brief background on DRAGON (Dynamic Resource Allocation in GMPLS Optical Networks) project and general use of DRAGON software.

1.1 Background

The DRAGON project, funded by the National Science Foundation (NSF), is concerned with the research and development of dynamic, deterministic, and manageable end-to-end network transport services for high-end e-Science applications.

For its implementation, DRAGON deploys the IP network infrastructure and creates a Generalized Multi-Protocol Label Switching (GMPLS) capable optical core network to allow dynamic provisioning of deterministic network paths in direct response to end-user requests, spanning multiple administrative domains. Optical transport and switching equipment acting as Label Switching Routers (LSRs) provide deterministic network resources at the packet, wavelength, and fiber cross-connect levels. The all-optical capabilities is based on connection and resource management mechanisms defined in GMPLS and the network models many of the functions of an inter-regional, national, or even global wavelength based Research and Education (R&E) networks.

The basic objective of this document is to describe the installation and use of DRAGON software in order to set up the capability in a network to dynamically provision dedicated paths across the network for high end application such as high definition video or high volume low latency scientific data flows. This is accomplished through the Virtual Label Switching Router (VLSR), as described later in some detail in this document. Figure 1 depicts the general network topology where a number of VLSRs, constructed using Ethernet switches and DRAGON software, provide traffic engineered paths. Typically on a GMPLS-based topology, the dynamically provisioned path is on an edge-node to edge-node. However, if the DRAGON software is running at the end systems, the path can be established on an end-to-end basis.

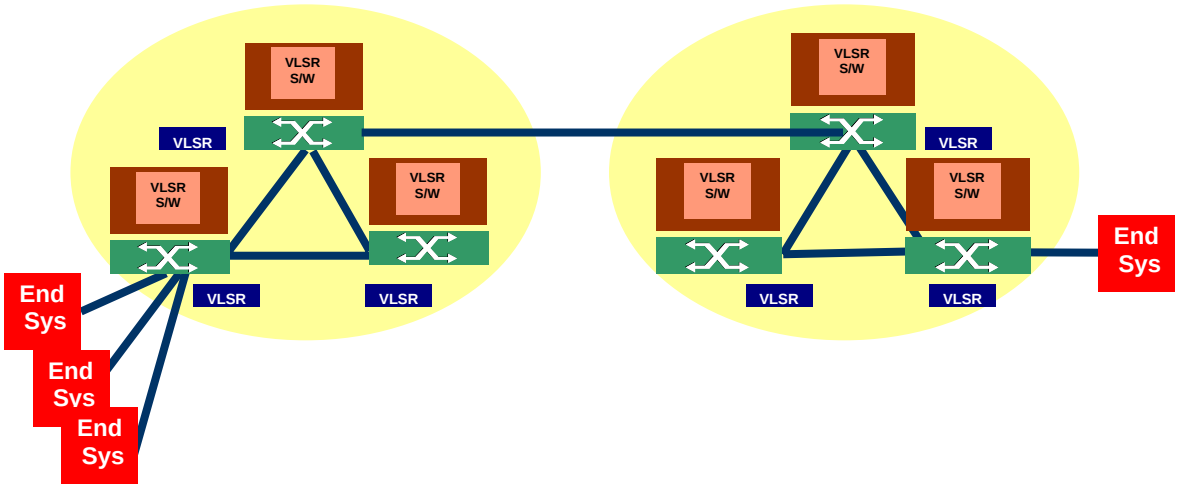

Figure 1: General Diagram for VLSR Deployment

1.2 Sample Configuration

We will describe the installation and use of DRAGON software by illustrating the implementation on a sample case. The sample configuration establishes the Label Switched Paths (LSPs) between GMPLS-enable hosts through a special host without the GMPLS capability, providing end-to-end GMPLS-based services. This special host is referred to as the Virtual Label Switching Router (VLSR). Here we install the DRAGON software on three machines where one of these machines will be the control machine (VLSR) which will have a control over the switch. The basic block diagram is depicted in Figure 2.

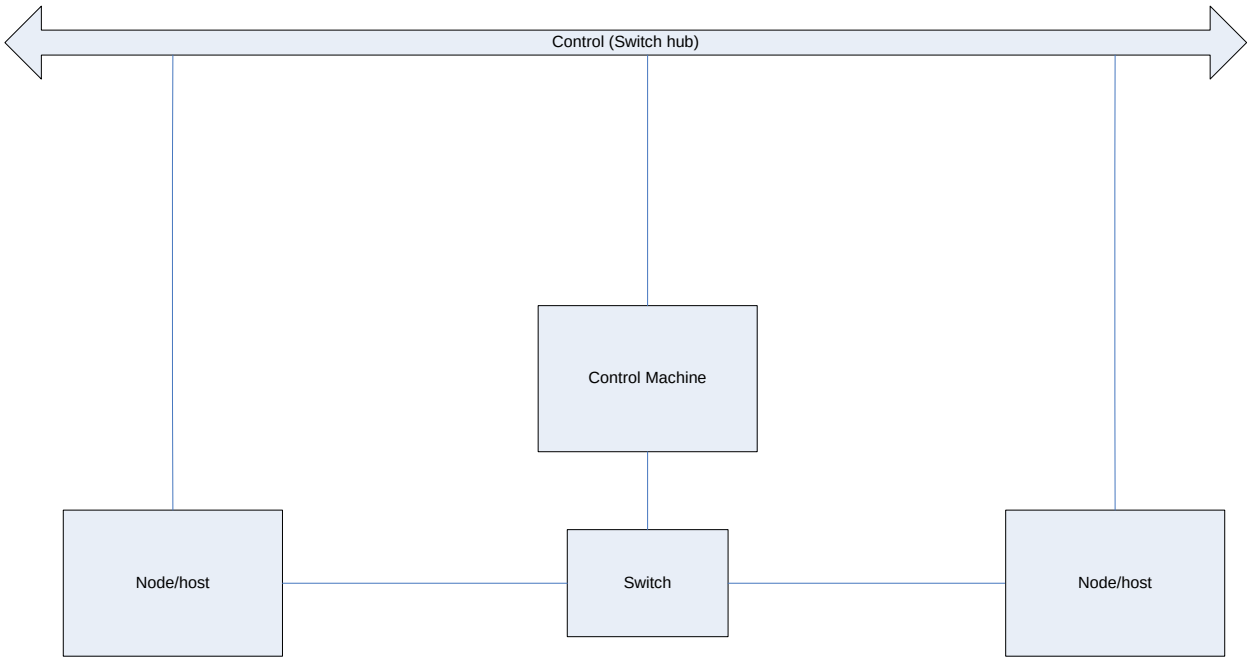

Figure 1: Basic block diagram of VLSR Configuration

The VLSR controls an Ethernet switch (which supports SNMP v1 and v2) and makes it capable of label switching. This is made possible with the help of Net-SNMP and DRAGON software installed on the VLSR. Meanwhile, GRE tunnels are set up from the hosts to the VLSR and from VLSR to the hosts. In this way the data plane and the control plane are separated as illustrated in the Figure above. The LSPs can then be set up between any two hosts through the VLSR.

2.0 DRAGON Software Suite Components

The relevant DRAGON Software Suite consists of VLSR, DRAGON OSPF, DRAGON KOM-RSVP, and DRAGON Zebra. These components are described below.

2.1 VLSR

The VLSR provides a mechanism to integrate non GMPLS equipment and network regions into the end-to-end GMPLS provisioned services. The VLSR translates standard GMPLS protocols into device specific protocols, to allow dynamic reconfiguration of non-GMPLS aware devices. The combination of a PC which runs the GMPLS based control plane software and the switch fabric is referred to as a VLSR. In this release the underlying switch fabric is an RFC 2674 [7] compliant Ethernet switch. The architecture is such that different types of switch fabrics can readily be added. In addition, methods other than RFC 2674 control could also be added. The GMPLS based control software functionality includes OSPF with GMPLS Traffic Engineering (OPSF-TE) [8] extensions and Resource Reservation Protocol-Traffic Engineering (RSVP-TE) [9] extensions.

Several switches have been evaluated for deployment in the network. In this illustration set up we will use Dell PowerConnect 5324 which has been fully tested with this release of the VLSR.

2.2 DRAGON OSPF

OSPF (Open Shortest Path First) is a link-state routing protocol developed for the IP networks. OSPF sends Link State Advertisements to all other routers within the same hierarchical area. Each OSPF router maintains an identical database describing the System topology. From this database, a routing table is calculated by constructing a shortest path tree. In other words, the OSPF routers accumulate Link State information and use the SPF algorithm to calculate the shortest path to each node. OSPF recalculates routes quickly in the face of topological changes, utilizing a minimum of routing protocol traffic. OSPF also carries the link state information for the GMPLS. It includes the OSPF TE [8]. The Constraint Shortest path First module is provided as a separate module which includes an API in the form of function calls and provides the ability to compute the traffic engineered paths based on the OSPF-TE derived State Database (LSDB).

This software is integrated into the open source routing suite software and understands how to interpret those LSDB data structures. The current CSPF implementation is limited to single LSP regions and considers the standard GMPLS-TE constraints of bandwidth availability and interface switching capability. This will be expanded in future releases to allow for multi-regions LSP path calculation and include additional constraints such as network adaptation capabilities and physical impairments for all-optical paths.

2.3 DRAGON KOM-RSVP

RSVP is a Resource Reservation Protocol. It is a signaling protocol that allows the sender and receiver in a communication to set up a reserved highway for data transmission with a specified

Quality of Service (QoS). Applications running on IP end systems can use RSVP to indicate to other nodes the nature (e.g. bandwidth) of the packet streams they want to receive.
The DRAGON project has extended the open source KOM RSVP Engine [10] from the Technical University of Darmstadt to include required GMPLS functionality. The KOM RSVP Engine provides and implementation of RSVP. The DRAGON modifications included the following:

- Addition of GMPLS Traffic Engineering Extensions for RSVP
- Extension of the RSVP API to allow control by DRAGON Zebra CLI and the DRAGON ASTB component
- Addition of functionality to allow control of RFC 2674 compliant Ethernet switches

2.4 DRAGON ZEBRA

The DRAGON project has extended the open source GNU Zebra [11] routing software package to include required GMPLS functionality. The GNU Zebra distribution is a routing protocol suite and includes multiple network protocols such as RIP, OSPF, BGP and others. The DRAGON modifications included the following:

- Addition of GMPLS TE extensions for OSPF
- Addition of capability to act as and Label Switch Router(LSR) responsible for an Ethernet switch fabric
- Addition of a DRAGON specific Command Line Interface(CLI) for LSP instantiation control
- Addition of NARB Functionality and associated CLI
- Incorporation of CSPF Routing Module
- Addition of XML interface for CLI control

3.0 Setting up the Environment for DRAGON Software

In order to set up the Environment for DRAGON software we refer to the configuration as depicted in Figure 3.

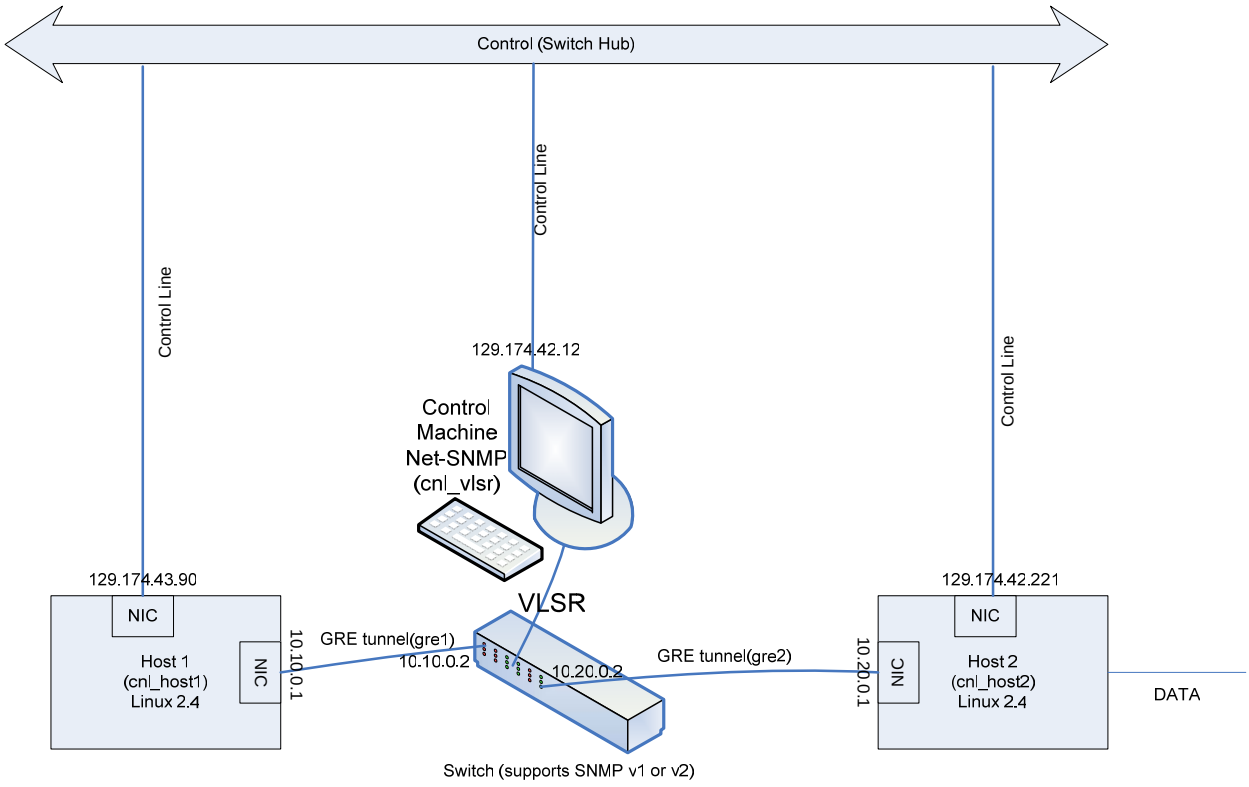

Figure 3: Sample VLSR Configuration

3.1 Preparation before Installation

The following steps must be followed to achieve proper operation of the above configuration:

3.1.1 Hardware Requirements

To download Linux is the first step to networking the three computers. But even before installing Linux we need to check the space available in the computers and upgrade the computers if necessary. The usual space and hardware requirements depend on the types of functions we require. There are 3 types of installations available:

a) **Personal Desktop**: Minimum recommended disk space for this installation is 1.7GB. With both GNOME and KDE desktop environments the installation would require 1.8GB.

b) **Workstation**: A workstation installation, including a graphical desktop environment and software development tools, requires at least 2.1GB of free space. Choosing both the GNOME and KDE desktop environments requires at least 2.2GB of free disk space.
c) **Server**: A server installation requires 850MB for a minimal installation without X (the graphical environment), at least 1.5GB of free space if all package groups other than X are installed, and at least 5.0GB to install all packages including the GNOME and KDE desktop environments.
d) **Custom**: A Custom installation requires 475MB for a minimal installation and at least 5.0GB of free space if every package is selected

Also, at least 486 MHz of processor speed and 32 or 64MB of RAM is necessary to run Linux.

After these requirements have been met, format the disks and install Linux 2.4 (Red Hat Linux v9) is installed for this project.

3.1.2 Software Requirements

a) **Verify if ssh is installed: whereis ssh**
Ssh provides with a secure way to log in to the remote machine. One can move files from one machine to another with secure communications over insecure channels. Somebody who has root access to machines on the network, or physical access to the wire, can gain unauthorized access to systems in a variety of ways. It is also possible for such a person to log all the traffic to and from a particular system, including passwords (which ssh never sends in the clear).
ssh is usually included in the Linux installation package but if not, it can be installed from ftp://ftp.net.ohio-state.edu/pub/security/ssh .

b) **GNU Compiler Collection (gcc)**
gcc stands for "GNU Compiler Collection". GCC is an integrated distribution of compilers for several major programming languages. These languages currently include C, C++, Objective-C, Objective-C++, Java, Fortran, and Ada. Although, this complier is included in the Linux installation package, the versions may vary. DRAGON software now supports the latest version 3.4.4 of gcc.

c) **Net SNMP**
DRAGON software requires Net-SNMP for its installation. The sample case referred in this document installs Net-SNMP version 5.1.1. It can be downloaded from:
http://prdownloads.sourceforge.net/net-snmp/

SNMP is a Simple Network Management Protocol which helps to provide performance information and lets us remotely manage network servers, routers and switches. SNMP helps to monitor the network equipment e.g. Routers or switches, computer equipment and even devices like UPS.

An SNMP-managed network consists of three primary components: managed devices, agents, and management systems. A managed device is a network node that contains an SNMP agent and resides on a managed network. Managed devices collect and store management information and use SNMP to make this information available to management systems that use SNMP. Managed devices include routers, access servers, switches, bridges, hubs, computer hosts, and printers.

The SNMP agent gathers data from the management information base (MIB), which is the repository for information about device parameters and network data. There are three basic versions of SNMP defined: SNMP v1, SNMP v2 and SNMP v3. Both versions 1 and 2 have a number of features in common, but SNMPv2 offers enhancements, such as additional protocol operations and additions to data types, counter size, and protocol operations. SNMP version 3 (SNMPv3) adds security and remote configuration capabilities to the previous versions.

Net-SNMP is a suite of applications used to implement SNMP v1, SNMP v2c, SNMP v3 using both IP v4 and IP v6. The suite includes Command Line application, a graphical MIB Browser, daemon application for receiving SNMP notifications, an extensible agent for responding to SNMP queries for management information and a library for developing new SNMP applications with both C and Perl APIs.
All versions of Net-SNMP are available at www.net-snmp.org/download

d) Verify if svn is installed: whereis svn

svn is a command used for subversion control which is an open source version control system. Version control is the art of managing changes to information.
Subversion is the improved version of CVS (Concurrent Version Systems) which manages files and directories in a better way than the latter. In subversion, a tree of files is kept in the central repository. The repository remembers every change that is ever made to the files and directory. This means that if an incorrect change has been made to the data, the previous data is not lost because all the work is versioned. Thus, no matter how many changes are done to a file, the original file as well all the other versions of the file can always be referenced at any time.
Subversion can be installed from the site, `http://subversion.tigris.org`

The advantages of Subversion over CVS have been enumerated as follows:

a) Directory versioning: CVS only tracks the history of individual files, but subversion implements a “virtual” versioned file system that tracks changes to whole directory trees over time. Files *and* directories are versioned.

b) True version history: Since CVS is limited to file versioning, operations such as copies and renames aren't supported in CVS. Additionally, CVS cannot replace a versioned file with some new thing of the same name without the new item inheriting the history of the old, perhaps completely unrelated file. With Subversion helps to add,

delete, copy, and rename both files and directories. And every newly added file begins with a fresh, clean history on its own.

c) Atomic commits: A collection of modifications either goes into the repository completely, or not at all. This allows developers to construct and commit changes as logical chunks, and prevents problems that can occur when only a portion of a set of changes is successfully sent to the repository.

d) Versioned metadata: Each file and directory has a set of properties—keys and their values—associated with it. Any arbitrary key/value pairs can be created or stored. Properties are versioned over time, just like file contents.

e) Efficient branching and tagging: The cost of branching and tagging need not be proportional to the project size. Subversion creates branches and tags by simply copying the project, using a mechanism similar to a hard-link. Thus these operations take only a very small, constant amount of time.

f) Hack ability: Subversion is implemented as a collection of shared C libraries with well-defined APIs. This makes Subversion extremely maintainable and usable by other applications and languages.

4.0 DRAGON Software Installation Guide

The DRAGON Software package can be installed from http://cnl.gmu.edu/dragon/software**.** The package can be unpacked by typing the following commands:

#tar –xvvf dragon-3.4.4.tar.gz

Now enter the directory where the DRAGON package has been unpacked. Finally, the following commands help to install the DRAGON software:
./do_build.sh to configure and build the package
sudo ./do_install.sh to install the dragon Software including kom-rsvp

By default, this software gets downloaded at /usr/local/dragon. But if you wish to use the normal installation method, the following takes care of installing the dragon software and kom-rsvp.

a) KOM-RSVP

The directory kom-rsvp is called,
#cd kom-rsvp

The kom-rsvp code is then configured,
#. /configure –-with-snmp=/usr/local CFLAG=-g CPPFLAG= -g

CFLAG and CPPFLAG are build options and enables debug if necessary. This command configures kom-rsvp with snmp provided net-snmp header files are installed at /usr/local else it is important to specify the correct pathname. After configuration is done, the directory is cleaned and the file dependencies are rebuilt:
 # gmake clean
 # gmake depend

Finally, the code is recompiled and the new binaries and libraries are installed for system,
 # gmake
 # gmake install

b) ZEBRA-OSPF:

The OSPF can be installed as follows:
Enter the directory zebra under the dragon directory,
 # cd zebra

Configure the protocol by typing the following command. The prefix command defines the directory where ZEBRA-OSPF should be installed.
#. /configure --prefix=/usr/local/dragon –-enable-dragon CFLAG=-g CPPFLAG=-g
Finally, compile the code and install the files,
 # make
sudo make install

5.0 Configure the Switch

The switch (which supports Net-SNMP) is now connected to all the three machines. The VLSR will control the switch and so it is important to configure it. Go through the instruction manual for the switch and follow the instructions. The IP address to the data interface between the system and switch is assigned. It is important to keep a record of it.

6.0 Setting up the GRE tunnel

Encapsulation takes packets or frames from one network system and places them inside frames from another network system. This method is sometimes called *tunneling*. Tunneling provides a means for encapsulating packets inside a routable protocol via virtual interfaces.
We need to set up 2 GRE tunnels. One from host 1 to the Control Machine (VLSR) called gre1 and the other from host 2 to Control Machine (VLSR) called gre2. The LSPs can then be set up between the two hosts through the GRE tunnel. Basically, LSPs can reach anywhere where GRE tunnels can reach. Also, all transit nodes need not support the same Label Distribution Protocols. A GRE tunnel allows any network protocol to be transmitted over a network which is running on some other protocol. However, traffic engineering is not supported over GRE tunnels. For example, reservation of bandwidth or setting up priority is not supported.

To explain the set up of GRE tunnel between host 1 and Control Machine we have assumed the following IP addresses:

Host 1 (cnl_host1):
Network address: 129.174.43.90
Netmask: 255.255.255.0
Local address of interface gre1: 10.10.0.1

Host 2(cnl_host2):
Network address: 129.174.42.221
Netmask: 255.255.255.0
Local address of interface gre2: 10.20.0.1

Control machine (cnl_vlsr):
Network address: 129.174.42.12
Netmask: 255.255.255.0
Local address of interface gre1: 10.10.0.2
Local address of interface gre2:10.20.0.2

On the router of host 1, the following takes place:

```
/sbin/ip tunnel add gre1 mode gre remote 129.174.42.12 local
129.174.43.90 ttl 255
/sbin/ip link set gre1 up
/sbin/ip addr add 10.10.0.1 dev gre1
/sbin/ip route add 10.10.0.0/24 dev gre1
```

The explanation of each of the command lines above is as follows:
ip tunnel add gre1 mode gre remote 129.174.42.12 local 129.174.43.90 ttl 255

In this command we have added a tunnel device called ‘gre1’. This tunnel device uses the GRE protocol ‘mode gre’. The remote address is 129.174.42.12. The tunneling packets should

originate from 129.174.43.90 which is the address of the router of the host 1 at the control machine end. The TTL field of packet is set to 255.
ip link set gre1 up

This command enables the device.
ip addr add 10.10.0.1 dev gre1
The interface 'gre1' has been given an IP address 10.10.0.1.
ip route add 10.10.0.0/24 dev gre1

Now the route is set to Control machine with IP address 10.10.0.0.

a) **Host 2:** Similarly, on the switch of host 2 we will set up a GRE tunnel from host 2 to the control machine.

```
/sbin/ip tunnel add gre2 mode gre remote 129.174.42.12 local
  129.174.42.221 ttl 255
/sbin/ip link set gre2 up
/sbin/ip addr add 10.20.0.1 dev gre2
/sbin/ip route add 10.20.0.0/24 dev gre2
```

b) **VLSR**: Finally, the GRE tunnel is linked to the VLSR as follows,

```
/sbin/ip tunnel add gre1 mode gre remote 129.174.43.90 local
129.174.42.12 ttl 255
/sbin/ip link set gre1 up
/sbin/ip addr add 10.10.0.2 dev gre1
/sbin/ip route add 10.10.0.0/24 dev gre1

/sbin/ip tunnel add gre2 mode gre remote 129.174.42.221 local
129.174.42.12 ttl 255
/sbin/ip link set gre2 up
/sbin/ip addr add 10.20.0.2 dev gre2
/sbin/ip route add 10.20.0.0/24 dev gre2
```

You can save sh files of above in directory /usr/local/dragon/etc and then run the gre tunnel in all the three computers by typing sh [file name]

The following commands can be used as a **check** to see if the GRE tunnel is set right,
/sbin/insmode ip_gre
/sbin/ifconfig or /sbin/ip tunnel or /sbin/ip link.

In case, the GRE tunnel is not set right, it can be removed as follows,
#ip link set gre1 down
#ip tunnel del gre1
This removes the GRE tunnel 'gre1'.

7.0 Configuration Guide

The sample configuration guides for OSPF, RSVP, ZEBRA and DRAGON for the case illustrated in this document i.e. cnl_host1—cnl_vlsr—cnl_host2 are as follows:

7.1 OSPF Configuration Guide

i) Modify ospfd.conf on an end-host

Here, we specify hostname and password for the host, GRE tunnel interface between this host and VLSR and finally router id and switching capability. Copy this configuration file in usr/local/dragon/etc.

a) Host 1:

```
!
! zebra-ospfd configuration file for cnl_host1
!
hostname cnl_host1-ospf
password [specify password]
log stdout
!
!
interface gre10
 description GRE tunnel between cnl_host1 and cnl_narb
 ip ospf network point-to-point
interface gre1
 description GRE tunnel between cnl_host1 and cnl_vlsr
 ip ospf network point-to-point
!
!
router ospf
 ospf router-id 129.174.43.90
 network 10.1.0.0/24 area 0.0.0.0
 network 10.10.0.0/24 area 0.0.0.0
!
 ospf-te router-address 129.174.43.90
!
 ospf-te interface gre10
 exit
!
 ospf-te interface gre1
  level gmpls
   data-interface ip 10.1.10.2
  swcap lsc encoding ethernet
  exit
!
line vty
!
```

b) Host 2:

```
!
! zebra-ospfd configuration file for cnl_host2
!
hostname cnl_host2-ospf
```

```
password [specify password]
log stdout
!
!
interface gre2
 description GRE tunnel between cnl_host2 and cnl_vlsr
 ip ospf network point-to-point
!
!
router ospf
 ospf router-id 129.174.42.221
 network 10.20.0.0/24 area 0.0.0.0
 ospf-te router-address 129.174.42.221
 ospf-te interface gre2
      level gmpls
        data-interface ip 10.1.20.2
      swcap lsc encoding ethernet
 exit
!
line vty
!
```

ii) Modify ospfd.conf on the control machine

a. Specify hostname and password for the host
b. Specify GRE tunnel interfaces between VLSR to each of the two hosts
c. Specify router id and switching capability

Copy this configuration file in usr/local/dragon/etc

```
!
! zebra-ospfd configuration file for cnl_vlsr
!
  hostname cnl_vlsr-ospf
  password [specify here]
  log stdout
!
!
  interface gre1
  description GRE tunnel between cnl_vlsr and cnl_host1
  ip ospf network point-to-point
!
!
  interface gre2
  description GRE tunnel between cnl_vlsr and cnl_host2
  ip ospf network point-to-point
!
!
  router ospf
  ospf router-id 129.174.42.12
  network 10.10.0.0/24 area 0.0.0.0
  network 10.20.0.0/24 area 0.0.0.0
!
  ospf-te router-address 129.174.42.12
!
  ospf-te interface gre1
  level gmpls
  data-interface ip 10.1.10.1 protocol snmp switch-ip
```

```
   10.1.1.2 switch-port 9
    swcap lsc encoding ethernet
   exit
 !
   ospf-te interface gre2
   level gmpls
   data-interface ip 10.1.20.1 protocol snmp switch-ip
   10.1.1.2 switch-port 10
   swcap lsc encoding ethernet
   exit
!
   line vty
!
```

7.2 RSVP configuration Guide

This configuration file specifies the GRE tunnel interfaces of the machine. Copy this configuration file in usr/local/etc.

a) Host 1:

```
!
   interface gre1 tc none mpls
   api 4000
!
```

b) Host 2:

```
!
interface gre2 tc none mpls
   api 4000
!
```

c) VLSR:

```
!
   interface gre1 tc none mpls
   interface gre2 tc none mpls
   api 4000
   !
```

7.3 DRAGON Configuration Guide

This configuration file specifies the hostname and the password for every machine. Copy this configuration file in usr/local/dragon/etc.

a) Host 1:

```
! -*- dragon -*-
!
! DRAGON configuration file for cnl_host1
!
hostname cnl_host1-dragon
password [specify password here]
```

b) Host 2:

```
! -*- dragon -*-
!
! DRAGON configuration file for cnl_host2
```

```
!
hostname cnl_host2-dragon
password [specify password here]
```

7.4 VLSR Configuration Guide

```
! -*- dragon -*-
!
! DRAGON configuration file for cnl_vlsr
!
hostname cnl_vlsr-dragon
password dragon
```

7.5 ZEBRA Configuration Guide

Copy this configuration file in usr/local/dragon/etc

a) Host 1:

```
! -*- zebra -*-
!
! zebra configuration file for cnl_host1
!
hostname cnl_host1-zebra
password [specify password]
! enable password [specify password]
!
! Interface description.
!
interface lo
interface gre1
interface gre10
! description test of desc.
!
!interface sit0
! multicast

!
! Static default route sample.
!
!log file zebra.log
```

b) Host 2:

```
! -*- zebra -*-
!
! zebra configuration file for cnl_host2

!
hostname cnl_host2-zebra
password [specify password]
! enable password [specify password]
!
! Interface's description.
!
interface lo
interface gre2
! description test of desc.
!
```

```
!interface sit0
! multicast
!
! Static default route sample.
!
!log file zebra.log
```

c) VLSR:

```
! -*- zebra -*-
!
! zebra configuration file for cnl_vlsr

!
hostname cnl_vlsr-zebra
password dragon
! enable password dragon
!
! Interface's description.
!
interface lo
interface gre1
interface gre2
! description test of desc.
!
! interface sit0
! multicast
!
! Static default route sample.
!
!log file zebra.log
```

8.0 Running the Daemons

After we modify the configuration file, run ZEBRA, OSPFD, DRAGON and RSVPD daemons. The procedure is as follows:

8.1 ZEBRA Daemon

Go to zebra directory:
#. /zebra –d

8.2 OSPF Daemon

Go to ospf directory:
#. /ospfd -d

To check if ospfd daemon is working correctly, the procedure is as follows:
#telnet localhost 2604
#password: [enter password specified in OSPF configuration files]

#cnl_host2_ospf >list
#show ip ospf neighbor
#show ip ospf interface
#show ospf database
You can try all the options as displayed in the list on all the three machines.

8.3 DRAGON Daemon

Go to the Dragon Directory and type as follows:
./dragon –h → help file for running dragon
#./dragon –d → to start the daemon

8.4 RSVP Daemon

Go to kom-rsvp/bin directory:
./RSVPD –d

9.0 Example of DRAGON Command Line Interface (CLI)

The following example shows the use of dragon Command Line Interface (CLI) to configure and test if everything worked right. This example sets up LSPs between sender and receiver machines,

1. Show the modules and configure them one by one
    a. show module {show what modules we have}
    b. configure [MODULE_NAME] {configure the module}
2. Create a RSVP listener on the receiver side from the dragon CLI
    a. edit lsp [LSP_NAME] {give a name to lsp}
    b. set ip_src X.X.X.X port x ip_dest Y.Y.Y.Y port y {set ip address and port number for both source and destination}
    c. exit
    d. commit lsp [LSP_NAME] receiver {commit to be lsp receiver}
3. Create a RSVP sender on the sender side from the dragon CLI
    a. edit lsp [LSP_NAME] {give a name to lsp}
    b. set ip_src X.X.X.X port x ip_dest Y.Y.Y.Y port y {set ip address and port number for both source and destination}//They must be exactly the same as the receiver side
    c. set bandwidth gige_f swap lsc encoding Ethernet gpid Ethernet {set the switching capability to be lsc(label switch capability)}
    d. exit
4. Initiate the path from the sender
    a. commit lsp [LSP_NAME] {commit to be lsp sender and set up lsp}
5. Check the lsp status
    a. show lsp [LSP_NAME] {show the status of the lsp}

    The lsp status is like:

    Lsp status = In service: the path has been established
    = Commit: waiting for responses from the narb
    = Edit: has not been committed yet
    = Listening: listening for coming path requests
    = Delete: waiting for deleting confirmation from the narb

Note: If there is no narb, the “commit” command will go to local RSVP/OSPF.

10.0 Conclusions

The DRAGON software suite installation and implementation guide as described in this document is intended to provide the necessary information to successfully install the VLSR software and implement a VLSR-based Network. The illustrative example here incorporates three machines.

An updated version of this document can be downloaded from http://cnl.gmu.edu/dragon.

Appendix

Appendix A: Acronyms and Abbreviations

- DRAGON: Dynamic Resource Allocation in GMPLS Optical Networks
- VLSR: Virtual Label Switching Router
- NSF: National Science Foundation
- GMPLS: Generalized Multi-Protocol Label Switching
- GMPLS-TE: Generalized Multi-Protocol Label Switching-Traffic Engineering
- LSR: Label Switch Router
- R&E: Research and Education
- LSP: Label Switched Paths
- GRE: Generic Routing Encapsulation
- OSPF: Open Shortest Path First
- OSPF-TE: Open Shortest Path First-Traffic Engineering
- RSVP : Resource Reservation Protocol
- RSVP-TE: Resource Reservation Protocol-Traffic Engineering
- LSDB: Link State DataBase
- RFC: Request for Comments
- RIP: Routing Information Protocol
- BGP Border Gateway Protocol
- CLI: Command Line Interface
- XML: Extensible Markup Language
- CSPF: Constrained Shortest Path First
- RAM: Random Access Memory
- GNU: GNU's Not Unix
- GCC: GNU Compiler Collection
- SNMP: Simple Network Management Protocol
- API: Application Programming Interface
- CVS: Concurrent Version Systems
- SVN: SubVersioN
- SSH: Secure SHell
- KDE: K Desktop Environment
- GNOME: GNU Object Model Environment
- IP: Internet Protocol
- NARB: Network Aware Resource Broker

Appendix B: References

[1] DRAGON Team, DRAGON Architecture Description, May 2005

[2] DRAGON Team, DRAGON Network Description, May 2005

[3] DRAGON Team, DRAGON Software Suite Release 1.0

[4] RFC3630, Traffic Engineering (TE) Extensions to OSPF Version 2.

[5] An extended Quagga/Zebra OSPF daemon supporting an API for external applications http://www.tik.ee.ethz.ch/%7Ekeller/ospfapi/

[6] IETF draft-ietf-ccamp-gmpls-routing-09.txt.

[7] E.Bell, A.Smith, P.Langille, A.Rijhsinghani, K.McLoghrie, "Definitions of Managed Objects for Bridges with Traffic Classes, Multicast Filtering and Virtual LAN Extenstions", RFC 2674, August 1999.

[8] K. Kompella, Y. Rekhter, "OSPF Extensions in Support of Generalized Multi-Protocol Label Switching", draft-ietf-ccamp-ospf-gmpls-extensions-12.txt, April 2004.

[9] L. Berger, "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC 3473, January 2003.

[10]KOM RSVP Engine, Technical University of Darmstadt, www.kom.e-technik.tu-darmstadt.de/rsvp/

[11] GNU Zebra Routing Protocol Suite, www.zebra.org