Chapter 6 Implementation and Testing

This chapter details the implementation and testing aspects of the MPLS VPN simulator. It first begins with the implementation of the MPLS VPN component classes. All the important attributes and methods are discussed.

The second section focuses on the testing of the simulator. This section will describe the testing for specific components and the overall system testing which is the topology testing.

The final section of this chapter summarizes the details of this chapter.

6.1 MPLS VPN Simulator Components Implementation

The following section discusses the implementation for each simulator components.

6.1.1 SetupMplsVPN Class

The SetupMplsVPN class is the main class for MPLS VPN configuration. It allows the user to setup the VPN details before the simulation start. The following lists all the implementation of the class.

6.1.1.1 Attributes

class SetupMplsVPN implements ActionListener, java.io.Serializable {

    private java.util.List Router; //Hold a list of components
    private java.util.List PVPN; //Hold a list of predefined VPN
    private JavaSim theSim;

    private java.util.List PEtable; //A list PE table details
    private java.util.List VRFtable; //A list of VRF table details
    private java.util.List PECEtable; //A list of PECE table details
    private java.util.List PEList; //A list of PE-Router / ATMLSR
    private java.util.List CEList; //A list of CE-Router / IPBTE
    private String TITLE;
    private long RT;
    private long RD;

}
private String iname; // Image Name

6.1.1.2 Tables

This class displays the PE-Router, VRF name and RD information.

private class PETable extends javax.swing.table.AbstractTableModel
//Hold attribute for above PE Table
private class Route implements java.io.Serializable {
    String PERouter;
    String VRFName;
    String ImageName;
    long RD;
}

This class displays the VRF name, import and export route target information.

private class VRFTable extends javax.swing.table.AbstractTableModel
//Hold attribute for above VRF Table
private class SetupVRF implements java.io.Serializable {
    String VRFName;
    long exportRT;
    long importRT;
}

This class displays the CE-VRF information.

private class PECETable extends javax.swing.table.AbstractTableModel
//Hold attribute for above PE-CE Table
private class PCE extends Route{
    String PERouter;
    String VRFName;
    String CERouter;
    int Network;
    int Subnetmask;
}

6.1.1.3 Conversion Methods

The following methods provide the conversion function.

String getStringIP(int valueip) // Format the IP to xx.xx.xx.xx
String getStringMask(int valuemask) // Format the Subnetmask to xx.xx.xx.xx
String getFormat(long a) // Format the RD/RT xxxx:xxxxxxxx

6.1.1.4 Display Dialog

The following methods display the configuration dialog box.
public void getfirstdialog(String name)
public void getseconddialog(String name)
public void getthirddialog(String name)

### 6.1.1.5 VPN Update Method

The following methods update the VPN details.

```java
//Update the VPN detail
private void updatervf(String var1, String var2, String var3, int var4, int var5, SimComponent var6)
private void updatervf(String var1, String var2, long var3, long var4, int x)
```

### 6.1.1.6 Event Update Method

This method passes the update event to ATMLSR.

```java
PEcomp.compInfo(ATMLSR.SET_VPN,null,paramlist);
```

This method passes the update event to IPBTE.

```java
thiscomp.compInfo(IPBTE.SET_COLOR,null,paramlist);
```

### 6.1.2 SimVPN Class

The SimVPN class is the class that predefined VPN.

```java
class SimVPN implements java.io.Serializable {
   //Predefined VPN
   long rd;  //route distinguisher
   String vpcolor; //the vpn site color image
   String VpnName;

   SimVPN(long a, int b, char c) {
       rd = a;
       vpcolor = getColor(b);
       VpnName = "VPN " + String.valueOf(c);
   }
}
```

### 6.1.3 SetupIP Class

The SetupIP class is the class that allows the user to setup the VPN details before the simulation start.
6.1.3.1 Attributes

class SetupIP implements ActionListener, java.io.Serializable {
    private int valueip, valuemask;
    private String TITLE;
    private JavaSim thesim;
    private java.util.List NetList;
    private java.util.List comps;
}

6.1.3.2 Verification

private boolean ipOK() //to verify the correctness of IP address
private boolean maskOK() //to verify the correctness of subnetmask

6.1.3.3 Update

This method updates the IP and subnet mask information.

private void updatevalue(int vaip, int vamask) {
    Object [] paramlist=new Object[2];
    paramlist[0] = new Integer(vaip);
    paramlist[1] = new Integer(vamask);
    thiscomp.compInfo(IPBTE.SET_IP,null,paramlist);
}

6.1.4 SimParamVPN Class

The SimParamVPN class is the class that allows the user to view the VPN details that have been setup successfully.

6.1.4.1 Attributes

class SimParamVPN extends SimParameter implements ActionListener, java.io.Serializable {
    private ATMLSR theComp;
    private transient JComponent jcomp=null; // JButton
}

6.1.4.2 Table List

This class displays the CE-VRF information.

private class CETable extends javax.swing.table.AbstractTableModel

This class displays import route target information.
private class ImportTable extends javax.swing.table.AbstractTableModel

This class displays export route target information.

private class ExportTable extends javax.swing.table.AbstractTableModel

6.1.5 IPBTE Class

The IPBTE class is the class for CE-Router. This class inherits the methods and properties of SimComponent. This class will support MPLS VPN after the configuration.

6.1.5.1 Attributes

class IPBTE extends SimComponent implements Serializable {
   static final int GET_NET = 7; // get ip and mask
   static final int SET_COLOR = 8; // SET Vpn and Color
   cn_vpn = new SimParamBool("VPN", getName(), ctick, false, true, false);
}

6.1.5.2 Methods

This method updates the IP and subnet mask information.

case SET_IP:
    b_sourceIP.setValue((Integer)paramlist[0]).intValue(),
    (Integer)paramlist[1]).intValue());

This method sets the VPN site.

// Set VPN and the VPN site color

case SET_COLOR:
    Color c;
    cn_vpn.setValue(true);
    cn_vpn.update(theSim.now());
    c = (String) (paramlist[0]);
    this.setImageName(c);
    theSim.requestRefresh();

This method gets source network and subnet mask.

case GET_NET:
    o = new Object[2];
    o[0] = new Integer(b_sourceIP.getIP());
6.1.6 ATMLSR Class

The ATMLSR class is the class that performs as the PE-Router and P-Router. This class inherits the methods and properties of SimComponent. The following lists the attribute and methods of this class:

6.1.6.1 Attributes

class ATMLSR extends SimComponent implements Serializable {
    private SimParamVPN vpnDetail = null;
    private VRF[] newvrf = null; //hold VRF detail
    java.util.List ObjCEPE; //hold object for class CEPE
    java.util.List holdvrf; //hold object for class Forward
    java.util.List BGPTable; //hold data list for BGP
    java.util.List BGPUdpateList; //hold BGP Update message
}

6.1.6.2 Event Schedules

This event sends BGP_OPEN.

theSim.enqueue(new SimEvent(EV_GEN_BGP, this, this, 
    SimClock.Sec2Tick(0.1), null));

This event sends BGP_KEEPALIVE.

theSim.enqueue(new SimEvent(EV_SEND_BGP, this, this, 
    SimClock.Sec2Tick(0.6), null));

This event makes sure LSP has been created between PE-Router.

theSim.enqueue(new SimEvent(EV_Refresh_BGP, this, this, 
    SimClock.Sec2Tick(0.8), null));

This event sends BGP_UPDATE.

theSim.enqueue(new SimEvent(EV_UPDATE_BGP, this, this, 
    SimClock.Sec2Tick(1.2), null));

6.1.6.3 Log File

The log file is used to log the simulation process.

fname = "VpnDetail.log";
SimLog newlogl = new SimLog();
newlog1.setWriter(fname, true);
newlog1.log("Router Name : " + this.getName());
newlog1.log("Router ID : " + String.valueOf(RouteID));
//...
newlog1.close

87
6.1.6.4  Update and Retrieve Methods

This method updates VRF detail.

private void setVpn(Object [] paramlist)

This method gets VRF information.

private long getVRF_RD(int ip)
java.util.List getVRF_List(int x, int y, String varl)
String getVRF_Info(int record, int x)

6.1.6.5  Methods

Step 1: Generate the BGP_OPEN Message

Generate_BGP()

- Assign RouterID, Route Distinguisher and a list of export route target to the BGP_OPEN message

Step 2: Flood, Generate BGP_KEEPALIVE message and Install

if the PE-Router VRF import route target equal to the BGP route export route target
    Generate BGP_KEEPALIVE and Install in a temporary list
Else
    Drop the routes
End

Flood the BGP_OPEN message to other PE-Routers

Step 3: Send BGP_KEEPALIVE message back to advertising PE-Router and request for LSP if not setup yet.

Check whether LSP had been established based on source IP, destination IP and type of services.
If not found
    Establish a new one
    Request new label from downstream
End

Step 4: Advertising PE-Router generate BGP_UPDATE message and store it in a list when receiving BGP_KEEPALIVE. Prepare VPN Label.

If the destination IP match the PE-Router and is BGP_KEEPALIVE
    Prepare BGP_UPDATE message
    Assign RD, IP, Mask, VPN Label, Link and destination PE-Router ID associate to that VRF to the BGP_UPDATE message
Insert the BGP UPDATE message in a list
End

Step 5: Check whether LSP had been setup among PE-Routers

Check whether LSP had been established based on source IP, destination IP and type of services.
If not found
   Establish a new one
   Request new label from downstream
End

Step 6: Send BGP_UPDATE message and Import to the VRFs

Send BGP_Update Message
If reach the destination PE-Router
   if the PE-Router VRF import route target equal to the
     BGP_UPDATE message route export route target
   Update the Forwarding table
End

6.2 MPLS VPN Testing

The testing of the simulator is categorized into two types, which are component testing and system testing. The following section discusses the details of the testing.

6.2.1 Components Testing

Components testing are used to test the individual classes for the MPLS VPN simulator. The testing will include the feature testing as well as the interaction testing. Following are some of the testing that has been conducted:

6.2.1.1 SetupMplsVPN

The testing on SetupMplsVPN class is to ensure all the configuration inputs by the user are correctly update to the PE-Router. The value input by the users include PE-Router, VRF Name, VPN site, import and export policies, the association of CE to the relevant VRF. The
SimParamVPN class is used to test whether all the details have been updated correctly to the PE-Router. Appendix B shows the snapshot of the configuration.

6.2.1.2 SimVPN

The testing on SimVpn class is to ensure all the VPN is predefined with the VPN Site name, route distinguisher and the VPN image name. The following shows the output of this class:

<table>
<thead>
<tr>
<th>VPNA</th>
<th>257:1</th>
<th>blue.jpg</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPNB</td>
<td>257:2</td>
<td>orange.jpg</td>
</tr>
<tr>
<td>VPNC</td>
<td>257:3</td>
<td>pink.jpg</td>
</tr>
<tr>
<td>VPND</td>
<td>257:4</td>
<td>red.jpg</td>
</tr>
<tr>
<td>VPNE</td>
<td>257:5</td>
<td>green.jpg</td>
</tr>
<tr>
<td>VPNF</td>
<td>257:6</td>
<td>gold.jpg</td>
</tr>
<tr>
<td>VPNG</td>
<td>257:7</td>
<td>skyblue.jpg</td>
</tr>
<tr>
<td>VPNH</td>
<td>257:8</td>
<td>yellow.jpg</td>
</tr>
<tr>
<td>VPNI</td>
<td>257:9</td>
<td>lightgreen.jpg</td>
</tr>
<tr>
<td>VPNJ</td>
<td>257:10</td>
<td>rose.jpg</td>
</tr>
</tbody>
</table>

6.2.1.3 SetupIP

The testing on SetupIP class is to ensure that the correct source network and subnet mask is updated correctly into the correct IPBTE. Figure 6.1 and 6.2 show the correctness of the testing.

![Figure 6.1 Setting Component IP](image)

*Figure 6.1 Setting Component IP*

![Figure 6.2 Sanjose Router Source Network](image)

*Figure 6.2 Sanjose Router Source Network*
6.2.1.4 IPBTE

The testing on IPBTE class is to ensure that the IPBTE change to the particular VPN based on the configuration. In the following scenario, the IPBTE’s colour for VPNA is set to blue and the IPBTE’s colour for VPNB is set to orange. Figure 6.3 shows that Sanjose, LA1 and Lyon IPBTE are set to VPNA and SanFrancisco and York1 IPBTE are set to VPNB.

![Figure 6.3 IPBTE with Different VPN Site](image)

6.2.1.5 ATMLSR

The testing on ATMLSR class is to ensure that the class performs the functionality of PE-Router as well as P-Router. This PE-Router must support MP-iBGP. The testing of ATMLSR shows that the PE-Router is able to generate BGP_OPEN message, BGP_KEEPALIVE message and the BGP_UPDATE message. Besides that, the PE-Router is able to import route information into the correct VRF. When the destination IP not found, the ATMLSR is able to drop the packet.
6.2.2 System Testing

The major purpose of the simulator is to ensure that the MPLS VPN backbone is configured correctly and perform according to the specification based on BGP/MPLS VPN. Thus, the system testing is performed to identify the correctness of the system. The system testing is done in three different types of topologies such as the Intranet topology, the Intranet and Extranet Integration topology and the Central Services topology.

6.2.2.1 Intranet Topology

The Intranet topology allows overlapping of IP address between multiple VPNs. The testing of this topology will first build up an Intranet environment in multiple sites with overlapping of IP address between different VPNs in the simulator. Then, the simulation is started to verify that each VPN can only send data to their VPN and not to other VPNs. The detail of this testing is listed in Appendix A Case study 1.

6.2.2.2 Intranet and Extranet Integration Topology

Intranet and Extranet Integration topology shows that two separate VPN can communicate with each other when each of the VPNs imports each other’s route. The testing of this topology will first build up an Extranet environment in multiple sites with overlapping of IP address between different VPNs but not the VPN which needs to communicate together. Then, it will start the simulation and verify that the two joining VPN can communicate with one and another, but not to other VPNs. The detail of this testing is listed in Appendix A Case study 2.

6.2.2.3 Central Service Topology

Central service topology allows clients from multiple VPN to access services from a central servers located at one or more central sites. However, this topology didn’t allow clients from multiple VPN to communicate with each other. The testing of this topology will first build up an environment with multiple VPN and a central site. Then, the simulation is started to verify
that the VPN is able to access the central site but unable to communicate with other VPNs. The detail of this testing is listed in Appendix A Case study 3.

6.3 Chapter Summary

This chapter covers the idea on how the implementation process on the MPLS VPN simulator is carried out. The MPLS VPN simulator component implementations explain the attribute and method of each class. This chapter also covers the simulator components testing, system testing and the topology testing.