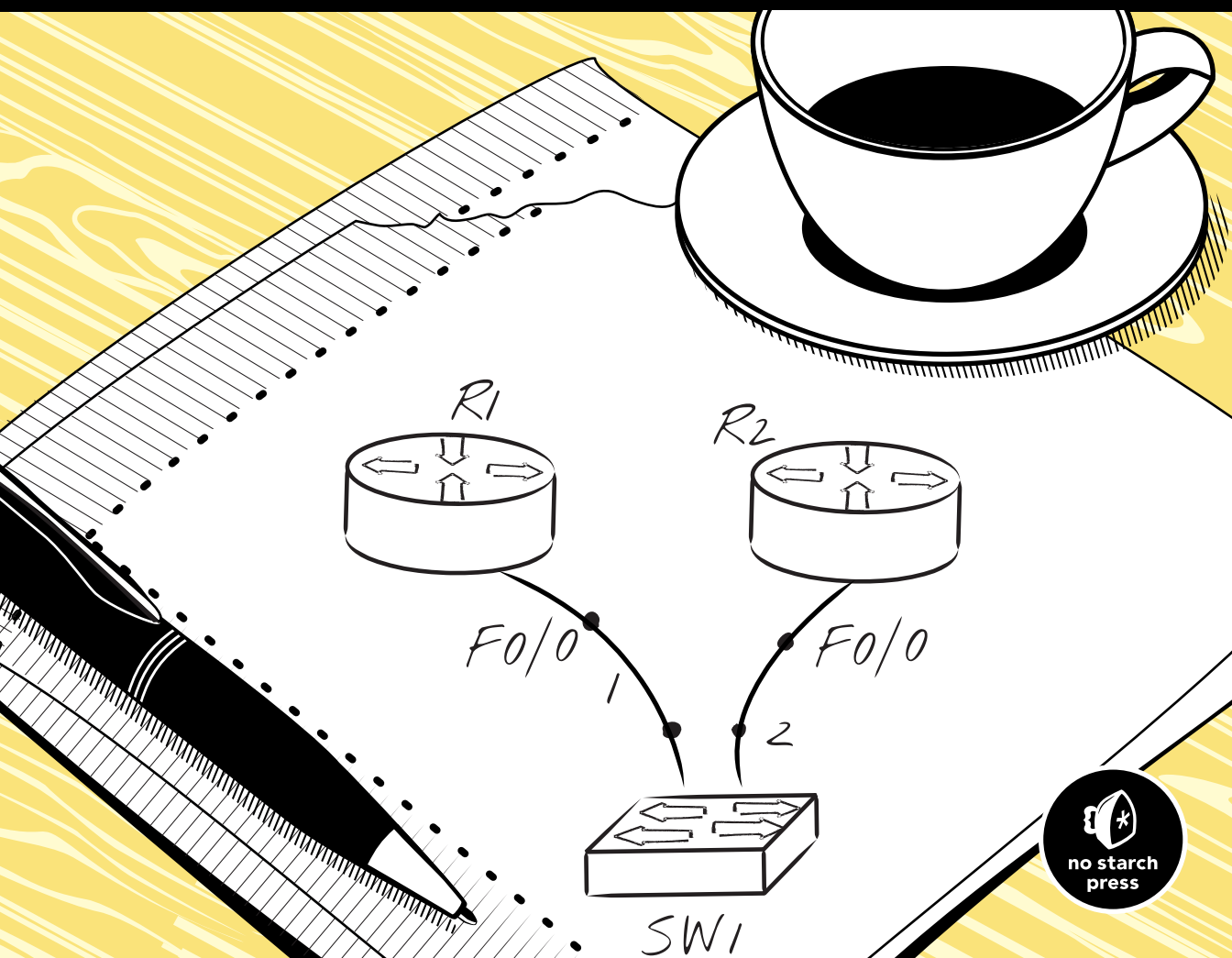
THE BOOK OF GNS3

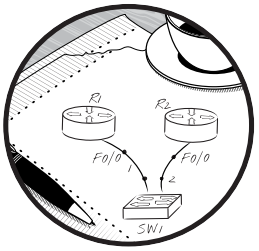
JASON C. NEUMANN



no starch press

7

DEVICE NODES, LIVE SWITCHES, AND THE INTERNET



In this chapter, I'll demonstrate how GNS3 device nodes are used with Cisco IOS routers. GNS3 provides a Hub node and various Switch nodes, including an Ethernet switch, EtherSwitch router, ATM switch, and Frame Relay switch. In addition, we'll take a look at how to create your own Frame Relay switch using a Cisco IOS router.

We'll also explore a very powerful feature known as a *Cloud node*. A Cloud node is used to expand your networks beyond the GNS3 program. With Cloud nodes, you'll soon be able to connect your GNS3 projects to live Cisco switches and access the Internet using GNS3 routers.

Built-in Device Nodes

Built-in device nodes *simulate* the features of a specific device type (like a switch). They're easy to configure, and can be useful if you need to save time and PC resources or if you just want to get something done without knowing all the details of the underlying technology.

If you create a topology that uses VLANs, you can drag a GNS3 Ethernet switch node to your workspace and use a simple menu to quickly create VLANs or VLAN trunks. Of course, if you're studying for a Cisco exam that involves switching, you need to know how to configure actual Cisco IOS switches.

Node Configurator

By now, you already know that the Node configurator can be used to configure the features of a single device node, but it can also be used to modify multiple devices at the same time. As your projects grow and you use more devices, this feature can save you a lot of time.

To open multiple devices at the same time, use your mouse to select those devices in your workspace, and then right-click any device and choose **Configure**. The selected devices display in the Node configurator, as shown in Figure 7-1.

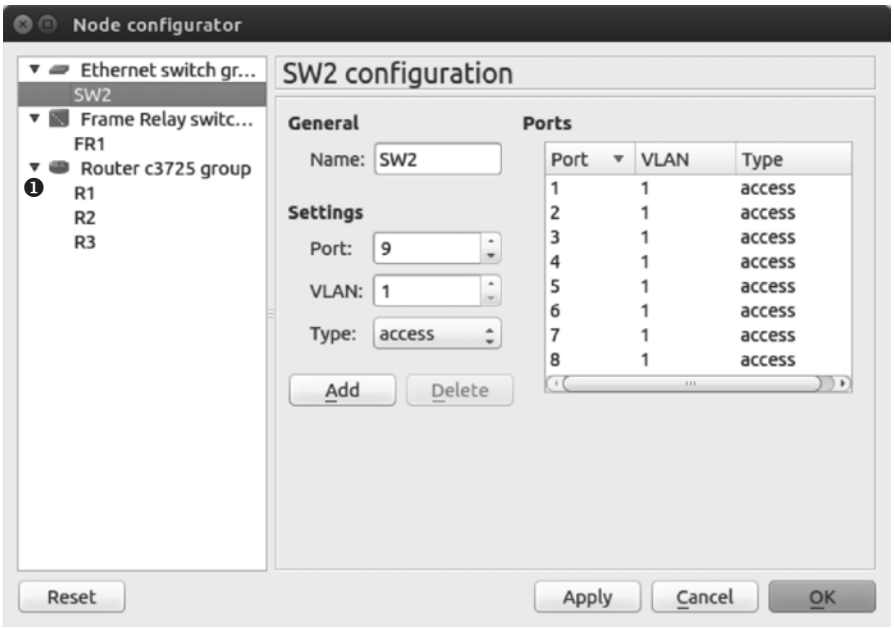


Figure 7-1: Modifying multiple devices using Node configurator

To configure a single device, select that device from the column on the right. Configure each device and click **Apply**. When you're finished configuring all the devices, click **OK** to complete the process.

To modify multiple devices, use the SHIFT key to select several at once from the column on the left and then configure them in the same manner as you would a single device. Let's say you have ten 7200 series routers, and you want to add the same network module to slot 0 on all 10 routers. Use the SHIFT key to select all the routers, add the module to slot 0, and then click **Apply** and **OK** to make the change to all 10 devices.

You can also select an entire group of routers by clicking their *group name*. In the previous figure, you could configure R1, R2, and R3 at the same time by selecting the group name *Router c3725 group* ❶.

Ethernet Hub

GNS3 provides an Ethernet hub (see Figure 7-2) as a tool that networking instructors can use to teach students about the perils of Ethernet loops, excessive broadcasts, and multiport repeating.

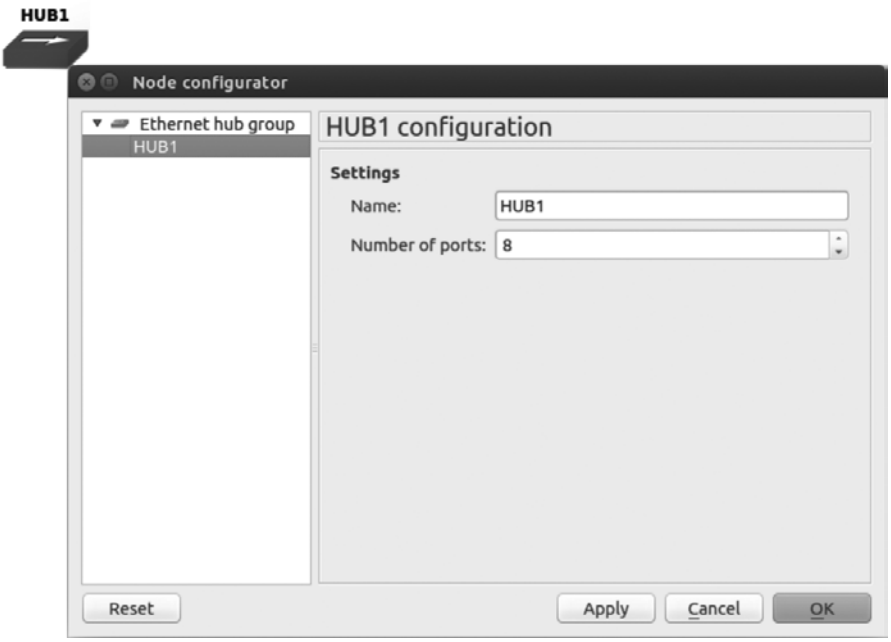


Figure 7-2: Ethernet hub Node configurator

As a general rule, you should stick with GNS3 switches and avoid including hubs in your projects. But if you do need to use an Ethernet hub, you can change the number of available Ethernet ports by pulling up the Node configurator and selecting your hub from the list on the left. The default value is eight ports, so if you want a different amount, enter that number in the Settings field on the right.

EtherSwitch Router

GNS3 provides two types of Dynamips switches, the *Ethernet switch node* and the *EtherSwitch router*. I discussed the Ethernet switch node in Chapter 4, so I'll only cover the EtherSwitch router here.

An EtherSwitch router is not a simulated device like an Ethernet hub or Ethernet switch node. Instead, it's a Dynamips router running Cisco IOS that's been configured with a 16-port switch module (NM-16ESW). This is the same switch module that can be installed on an actual Cisco router, and it has the same features and limitations (see Appendix C for details). Although the switch module has limited functionality, it's perfectly suited for CCNA and many CCNP studies. For more advanced switching features, you need to integrate real switches into your GNS3 projects or use Cisco IOS on Unix (IOU) switches.

NOTE *The EtherSwitch router requires that you configure a c3745 router with an IOS.*

To add additional switch ports to your EtherSwitch router, right-click the switch icon and select **Configure**, as shown in Figure 7-3.

The EtherSwitch router allows you to add additional switch modules. Adding another NM-16ESW module in slot two increases the number of switch ports to 32.

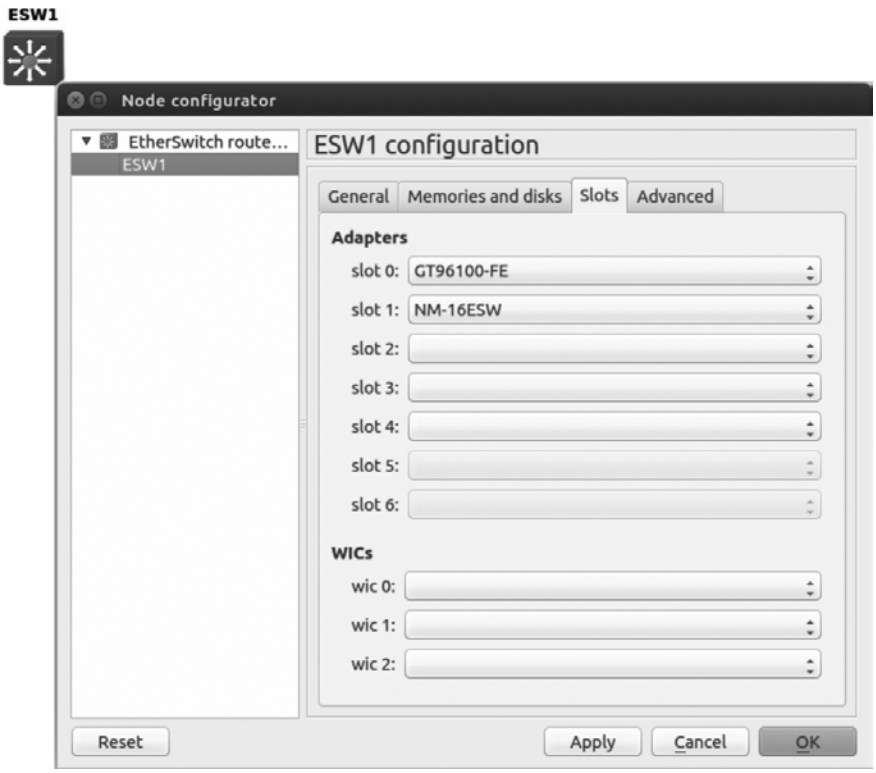


Figure 7-3: EtherSwitch router Node configurator

Frame Relay Switch

GNS3 provides a simple *Frame Relay switch node*, capable of emulating the basics of a generic Frame Relay switch. The nice thing about the GNS3 Frame Relay switch node is that it requires very little configuration. The downside is that it can sometimes be unstable.

WARNING *The GNS3 Frame Relay switch node only supports the ANSI LMI type, and "Cisco" is the default LMI type on Cisco IOS. You must use the command* `frame-relay lmi-type ansi` *on your router interfaces. Otherwise your frame cloud will not work. You can verify your LMI type using the* `show frame-relay lmi` *command (after configuring Frame Relay encapsulation).*

In Frame Relay, *Data Link Control Identifiers (DLCIs)* are used to assign frames to a *Private Virtual Circuit (PVC)* using serial port connections. To configure DLCI to serial port mappings, right-click the Frame Relay switch icon and open the Node configurator. Use the *Source* and *Destination* fields to create a mapping and click **Add**. When you're finished, click **Apply** and **OK** to complete the configuration.

The example configuration in Figure 7-4 will be used later to create a simple Frame Relay network. In the Mapping panel on the right, notice that FR1 is configured using two serial ports. Each serial port is used to link the Frame Relay switch to a router in your GNS3 project.

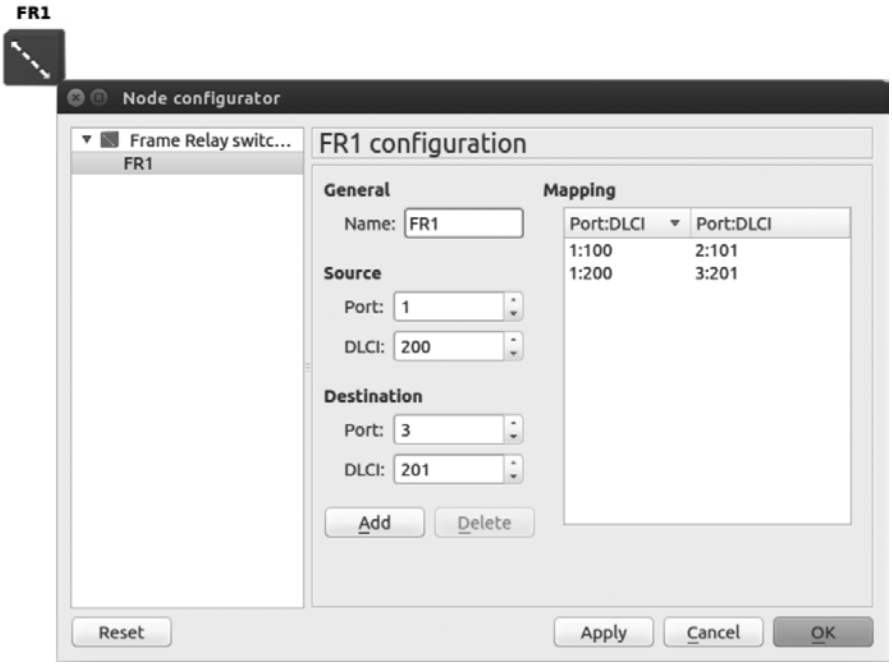


Figure 7-4: Frame Relay switch Node configurator

Port 1 has been assigned two DLCI numbers (100 and 200). Each DLCI on port 1 is mapped to a DLCI number on another serial port, and each mapping forms a Frame Relay PVC. Mapping is read from left to right, so in the first row, Port 1 DLCI 100 is mapped to Port 2 DLCI 101, and in the second row, Port 1 DLCI 200 is mapped to Port 3 DLCI 201. We'll use these mappings to configure a simple Frame Relay network.

Simple Frame Relay Hub and Spoke Configuration

There are several ways to configure a Frame Relay network, and understanding DLCI to serial port mapping is critical for understanding and configuring any of them. To better understand the mapping relationship, let's configure a simple network using the previously discussed DLCI to serial port mappings. I won't go into a lot of theory, but I'll explain enough to get you started.

We'll create our sample network using the topology in Figure 7-5. The network is divided into two subnets, 10.10.10.0 and 10.10.10.32, using the subnet mask 255.255.255.224.

This simple network is an example of a Frame Relay *hub and spoke* topology and should give you a good idea of how DLCI mappings work in a Frame Relay network. Router R2 will be on the 10.10.10.0 subnet, and router R3 will be on the 10.10.10.32 subnet. Router R1 is the "hub" in our hub and spoke topology. It will be connected to both subnets and will forward packets between them through the Frame Relay switch. This configuration will allow router R2 to ping router R3 and vice versa.

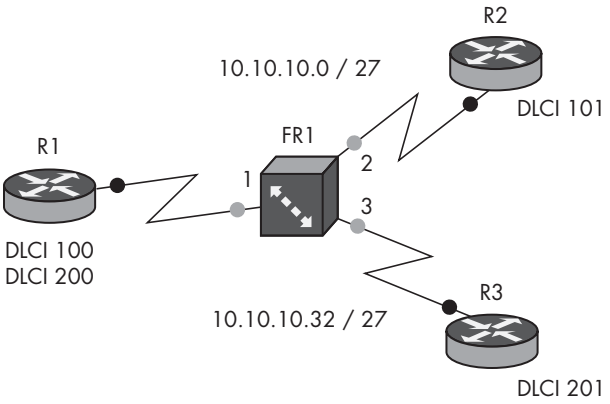


Figure 7-5: Sample hub and spoke Frame Relay network.

To create the project, add a *Frame Relay switch node* to your workspace and set it up the way we did in Figure 7-4. Add three routers and configure each with a serial interface card such as a WIC or NM-4T. Link the routers to the FR1 switch as shown. Be sure R1 is connected to Port 1 on FR1, R2 is connected to Port 2, and R3 is connected to Port 3. After linking the devices and configuring the Port:DLCI mapping on the FR1 switch, configure the hub router R1, as shown below.

```
   R1(config)# interface Serial0/0
❶ R1(config-if)# encapsulation frame-relay
❷ R1(config-if)# frame-relay lmi-type ansi
❸ R1(config-if)# clock rate 64000
   R1(config-if)# no shutdown
   R1(config-if)# no ip address
```

On the serial interface, enable Frame Relay encapsulation ❶, and set the LMI type to ANSI ❷. Although not required for GNS3, I've also set the clock rate ❸ because, depending on your IOS, the command may be required on an actual Frame Relay switch.

Next, configure point-to-point interfaces and DLCIs.

```
❶ R1(config-if)# interface Serial0/0.100 point-to-point
❷ R1(config-subif)# frame-relay interface-dlci 100
❸ R1(config-subif)# ip address 10.10.10.1 255.255.255.224
❹ R1(config-subif)# interface Serial0/0.200 point-to-point
❺ R1(config-subif)# frame-relay interface-dlci 200
❻ R1(config-subif)# ip address 10.10.10.33 255.255.255.224
```

Configure a point-to-point subinterface ❶ on serial interface Serial0/0.100 using DLCI 100 ❷ and assign the interface an IP address from the *first subnet* ❸ (10.10.10.0/27).

Complete the configuration by adding a second point-to-point subinterface ❹ in the same manner, but use DLCI 200 ❺, and assign the interface an IP addresses from the *second subnet* ❻ (10.10.10.32/27). In this example, I'll use the first valid address from the subnet, which is 10.10.10.33.

NOTE *It's considered a Cisco best practice to use the DLCI number for your subinterface number. `Serial0/0.100` is an example of a subinterface for DLCI 100.*

The following listing contains all of the commands you need to configure router R2.

```
   R2(config)# interface Serial0/0
❶ R2(config-if)# encapsulation frame-relay
❷ R2(config-if)# frame-relay lmi-type ansi
❸ R2(config-if)# clock rate 64000
   R2(config-if)# no shutdown
   R2(config-if)# no ip address
❹ R2(config-if)# interface Serial0/0.101 point-to-point
❺ R2(config-subif)# frame-relay interface-dlci 101
❻ R2(config-fr-dlci)# ip address 10.10.10.2 255.255.255.224
   R2(config-subif)# exit
❼ R2(config)# ip route 0.0.0.0 0.0.0.0 10.10.10.1
```

To configure R2, go under the serial interface and enable Frame Relay encapsulation ❶, then set the LMI type to ANSI ❷, set the clock rate ❸, and bring up the interface. Configure a point-to-point subinterface ❹ using DLCI 101 ❺ and assign the subinterface an IP address from the first

subnet ❻ (10.10.10.0 /27 in this example). Lastly, set the router's default gateway ❼ using the IP address configured under interface DLCI 100 of router R1 (IP address 10.10.10.1). Because R1 is the "hub" of our Frame Relay hub and spoke topology, it's used as the default gateway for our two subnets so that data can be routed between routers R2 and R3.

Finally, add a configuration to router R3 to complete the project.

```
R3(config)# interface Serial0/0
R3(config-if)# encapsulation frame-relay
R3(config-if)# frame-relay lmi-type ansi
R3(config-if)# clock rate 64000
R3(config-if)# no shutdown
R3(config-if)# no ip address
R3(config-if)# interface Serial0/0.201 point-to-point
❶ R3(config-subif)# frame-relay interface-dlci 201
❷ R3(config-fr-dlci)# ip address 10.10.10.34 255.255.255.224
R3(config-subif)# exit
❸ R3(config)# ip route 0.0.0.0 0.0.0.0 10.10.10.33
```

Router R3 is configured in nearly the same way as R2, but it uses DLCI 201 ❶ and an IP address ❷ from the *second subnet*. Also, you need to set the default gateway ❸ using the IP address configured under interface DLCI 200 of router R1 (IP address 10.10.10.33).

That's it! All three routers should now be able to ping each other. In a nutshell, each router encapsulates data frames and identifies them with a Frame Relay DLCI number as they leave their serial interfaces. When the Frame Relay switch receives data from a router, the data frames are forwarded to other routers through the switch based on the DLCI to serial port mapping. Because R1 is configured as a Frame Relay hub and knows about both subnets, it can forward data between the two subnets using the two PVCs. In this example, one PVC is made up of DLCI 100 mapped to DLCI 101, and the other is made up of DLCI 200 mapped to DLCI 201.

To verify that your Frame Relay circuit is active, enter the `show frame-relay pvc` command on each of your routers.

```
R1# show frame-relay pvc

PVC Statistics for interface Serial0/0 (Frame Relay DTE)

                Active      Inactive      Deleted       Static
  Local           2❶            0             0            0
  Switched        0             0             0            0
  Unused          0             0             0            0
```

If you've set everything up correctly, your PVCs should be displayed under `Active` ❶.

Creating a Frame Relay Switch Using IOS

As handy as the Frame Relay switch node is, sometime you might need to create your own Frame Relay switch using an IOS router. Maybe you want to use a different LMI type (like Cisco or q933a) or perhaps your studies require knowing the details of an actual Cisco Frame Relay switch. In any case, an IOS switch is fairly easy to set up.

The following listing creates an IOS Frame Relay switch using DLCI mappings identical to the GNS3 Frame Relay switch node you configured earlier. It may look intimidating at first glance, but it's not; you just need to understand how Frame Relay `connect` commands are used to configure the DLCI to serial port mappings.

```
❶ FRSW(config)# frame-relay switching
  FRSW(config)# interface Serial0/0
  FRSW(config-if)# description Serial connection to Router R1 (Hub)
  FRSW(config-if)# no shutdown
  FRSW(config-if)# no ip address
❷ FRSW(config-if)# encapsulation frame-relay
❸ FRSW(config-if)# clock rate 64000
❹ FRSW(config-if)# frame-relay lmi-type ansi
❺ FRSW(config-if)# frame-relay intf-type dce
  FRSW(config-if)# interface Serial0/1
  FRSW(config-if)# description Serial connection to Router R2 (Spoke)
  FRSW(config-if)# no shutdown
  FRSW(config-if)# no ip address
  FRSW(config-if)# encapsulation frame-relay
  FRSW(config-if)# clock rate 64000
  FRSW(config-if)# frame-relay lmi-type ansi
  FRSW(config-if)# frame-relay intf-type dce
  FRSW(config-if)# interface Serial0/2
  FRSW(config-if)# description Serial connection to Router R3 (Spoke)
  FRSW(config-if)# no shutdown
  FRSW(config-if)# no ip address
  FRSW(config-if)# encapsulation frame-relay
  FRSW(config-if)# clock rate 64000
  FRSW(config-if)# frame-relay lmi-type ansi
  FRSW(config-if)# frame-relay intf-type dce
  FRSW(config-if)# exit
❻ FRSW(config)# connect PVC1 Serial0/0 100 Serial0/1 101
❼ FRSW(config)# connect PVC2 Serial0/0 200 Serial0/2 201
```

When configuring a Frame Relay switch, you must first enable Frame Relay switching with the `frame-relay switching` command ❶. You'll also configure Frame Relay encapsulation on each serial interface with the `encapsulation frame-relay` command ❷. Then, use the `clock rate` command ❸ to set up clocking, choose an LMI type with `frame-relay lmi-type` ❹, and set the interface type to DCE using the `frame-relay intf-type dce` command ❺. After bringing up the interfaces, you're ready to define your PVCs.

NOTE *On some IOS versions, the* `clock rate` *command may need to be entered as* `clockrate`.

DLCI to port mapping is configured using `connect <connection-name> [<interface> <dlci>] [<interface> <dlci>]` commands. The last two commands in the configuration above define a connection mapping between two Frame Relay PVCs. The command `connect PVC1 Serial0/0 100 Serial0/1 101` ❻ defines a PVC between routers R1 and R2 and is used for creating our first subnet. The source interface is Serial0/0 and the source DCLI is 100. The PVC is completed using interface Serial0/1, and DLCI 101 configured on router R2.

The command `connect PVC2 Serial0/0 200 Serial0/2 201` ❼ uses the same syntax to create a second PVC between routers R1 and R3 (used for the second subnet).

That's all there is to creating your own Frame Relay switch. When configuring a Frame Relay switch with Cisco IOS, the LMI type can be set to `cisco`, `ansi`, or `q933a`, but you must be consistent on all routers participating in the Frame Relay network.

ATM Switch

GNS3 provides an easy way to configure an *Asynchronous Transfer Mode (ATM)* switch. ATM is similar to Frame Relay in that it's a layer 2 protocol that maps physical ports to logical circuits.

To configure a VPI/VCI to port mapping, right-click the ATM switch icon and select **Configure**, as shown in Figure 7-6. Here, I've configured a simple virtual circuit using two ports on the ATM1 switch node.

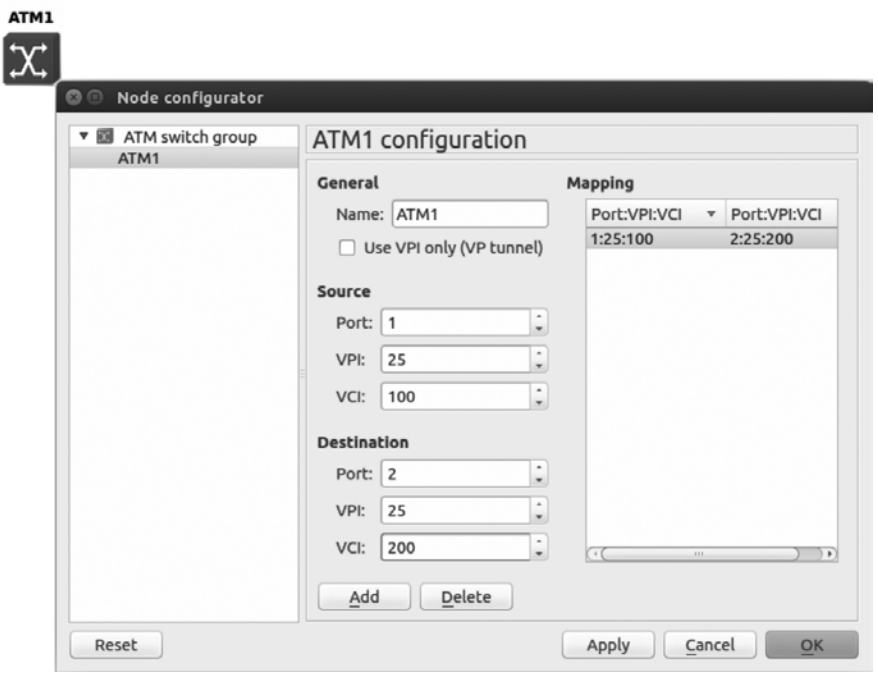


Figure 7-6: ATM Node configurator

Use the Source and Destination fields to create your port mapping, and then click **Add**. When you're finished with all the port mapping, click **Apply** and **OK** to complete the configuration.

Now let's run through a quick example of creating a simple point-to-point WAN connection using an ATM switch. Start by creating the topology shown in Figure 7-7.

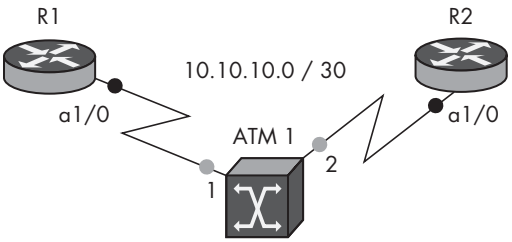


Figure 7-7: Sample ATM network

Add an ATM switch to your workspace and configure it using the information from Figure 7-6. Next, add two routers and create a link between each router and the switch. You need to use *7200 series routers* that are each configured with an *ATM port adapter* (PA-A1) in slot 1. Create a link from *a1/0* on each router to the ATM switch, as shown in the figure, and then enter the following commands to configure ATM on router R1.

```
R1(config)# interface ATM1/0
R1(config)# no shutdown
R1(config)# interface ATM1/0.100 point-to-point
R1(config-subif)# ip address 10.10.10.1 255.255.255.252
R1(config-subif)# pvc 25/100
R1(config-if-atm-vc)# protocol ip 10.10.10.2 broadcast
R1(config-if-atm-vc)# encapsulation aal5snap
```

Next, apply a similar configuration to router R2.

```
R2(config)# interface ATM1/0
R2(config)# no shutdown
R2(config)# interface ATM1/0.200 point-to-point
R2(config-subif)# ip address 10.10.10.2 255.255.255.252
R2(config-subif)# pvc 25/200
R2(config-if-atm-vc)# protocol ip 10.10.10.1 broadcast
R2(config-if-atm-vc)# encapsulation aal5snap
```

To verify that your ATM circuit is up, enter the `show atm pvc` command in the listing below.

```
R1# show atm pvc
```

If the PVC status displays "UP," then the two routers should now be able to ping each other.

Cloud Nodes

The Cloud node is a highly configurable device node that doesn't simulate a particular piece of hardware. Instead, it provides a wide range of *Network Input/Output (NIO)* connection options that allow GNS3 virtual devices to communicate with other programs or real hardware, like your PC's Ethernet adapter.

You connect to a Cloud node by creating a standard link from a GNS3 device (like a router) to the Cloud. Once this is done, any data leaving a virtual interface passes through the Cloud node's NIO connection to a destination outside GNS3, like a physical Ethernet adapter. Keep in mind that throughput limitations presented in GNS3 also apply to the virtual interface connected to the Cloud mode, meaning those limitations will affect your overall performance.

To configure an NIO connection (shown in Figure 7-8), right-click the Cloud icon and select **Configure**.

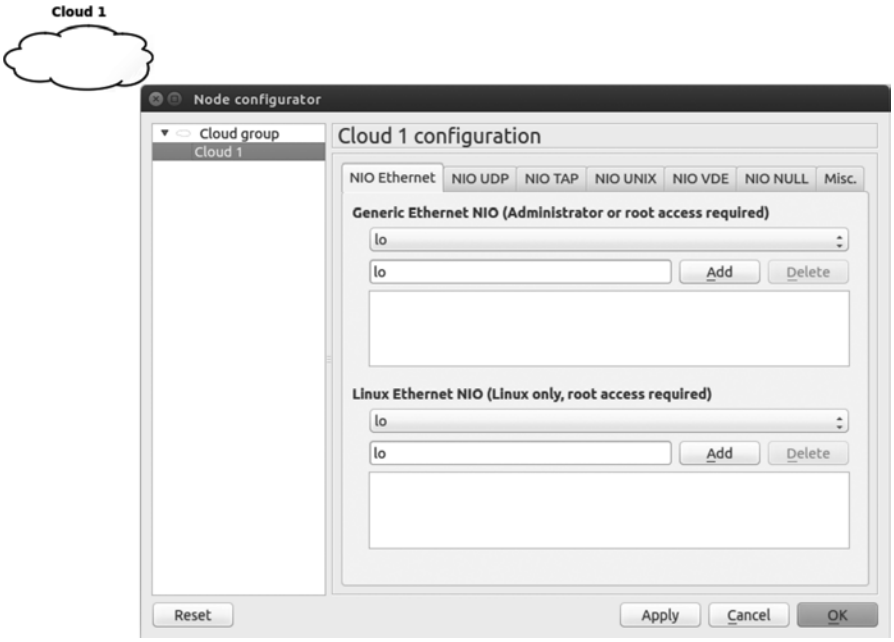


Figure 7-8: Cloud Node configurator

The Node configurator provides six different connection options:

NIO Ethernet Creates a connection to one or more physical or logical interfaces in your PC. The NIO Ethernet configuration lets you set up both a generic NIO and a Linux-only NIO.

NIO UDP Creates a UDP socket in order to form a tunnel/link between GNS3 and other programs.

NIO TAP Creates a connection to a virtual TAP interface. TAP interfaces are often bridged to physical interfaces in your PC.

NIO UNIX Creates a UNIX Socket connection between GNS3 and some other application.

NIO VDE Creates a link between GNS3 and a Virtual Distributed Ethernet device.

NIO NULL Creates a link between GNS3 and a NULL device to form a dummy link.

To configure a connection, select the NIO connection type, choose the options you want, and select **Add**. You can configure more than one connection per Cloud node, allowing you to link multiple GNS3 devices, just as you can use multiple interfaces on a GNS3 switch or router.

On Linux systems, you may want to use a generic NIO connection, which may work more reliably than a Linux-specific NIO. One reason is that the Linux-specific NIO may strip incoming VLAN tags. If you're having difficulty with a connection on Linux, try both to see which works better.

Connecting GNS3 Devices to Physical Hardware

The ability to connect projects to physical hardware is what transforms GNS3 from a diamond in the rough to the Great Star of Africa. Using the Cloud node, you can establish trunk links with live Cisco switches and even access the Internet from your GNS3 devices. This gives GNS3 nearly limitless networking possibilities. Connecting GNS3 to real devices is easier on some systems than others, but it should work on all major operating systems.

Dynamips Permissions

Before connecting GNS3 devices to a physical Ethernet adapter, you may need to make a few changes on your PC. If your Cloud node is configured using *NIO Ethernet* on a Windows system, you should run GNS3 using administrator privileges by right-clicking the GNS3 icon and selecting **Run as Administrator**. To make this option permanent, right-click the GNS3 icon and choose **Properties**. Select the **Compatibility** tab and place a check next to **Run this program as an administrator**.

On Unix-based systems, you need to elevate the permissions of Dynamips before using *NIO Ethernet* or *NIO TAP* connections. If you skip this step, you'll have to run GNS3 using the root account. Otherwise, NIO connections will fail and GNS3 will display an error message in the GNS3 Console window.

To set the correct Dynamips permissions on Mac OS X, use the following commands.

```
$ sudo chown root /Applications/GNS3.app/Contents/Resources/dynamips*
$ sudo chmod 4755 /Applications/GNS3.app/Contents/Resources/dynamips*
```

Setting permissions on most Linux distributions works the same way; just replace the file path above with the correct location for your `dynamips` file. If you're running a Debian-based Linux system like Ubuntu, use the `setcap` command instead because it's more secure.

```
$ sudo apt-get install libcap2
$ sudo setcap cap_net_raw,cap_net_admin+eip /usr/local/bin/dynamips*
```

After changing the Dynamips permissions, you can run GNS3 as a regular user, but Dynamips will be treated as though it's being run by the root account.

Preparing Your PC for a Bridge

Some operating systems don't allow GNS3 to communicate directly with your PC's Ethernet hardware, and Wi-Fi adapters usually don't work, either. Before using GNS3 with an Ethernet adapter on those systems, you might need to install additional software to make it work.

A common solution is to install a *virtual interface driver* and use a *bridge* to associate it with your PC's physical Ethernet adapter. GNS3 then passes network data to the virtual interface, which hands it off to the physical Ethernet interface via the bridge. On Unix-based systems, virtual interfaces are often provided using *TUN/TAP* drivers. On Windows, you'll use a loopback adapter that's bridged to your physical Ethernet adapter.

Even if your PC's Ethernet hardware works directly with GNS3, the following methods are a recommended and predictable way to connect GNS3 to the outside world.

Using a Loopback Adapter on Windows

In Windows, a *loopback adapter driver* provides a virtual network interface that can be bridged to a physical Ethernet adapter in your PC. To add a loopback adapter to Windows, go to **Control Panel ▸ System** and select **Device Manager**. Right-click your *computer name* from the list on the left, and select **Add Legacy Hardware**. Click **Next** and then select **Install the hardware that I manually select from a list (Advanced)** and choose **Network Adapters** from the list. From the Add Hardware wizard, shown in Figure 7-9, select **Microsoft** under Manufacturer, and then scroll down and select the Network Adapter named **Microsoft Loopback Adapter**. Click **Next** and **Finish** to complete the installation. You must reboot Windows after installing the Microsoft Loopback Adapter.

The Windows loopback adapter can also be installed by running the `loopback-manager.cmd` command from your GNS3 installation directory.

To create a bridge between the loopback adapter and Ethernet adapter, go to **Control Panel ▸ Network and Sharing Center** and choose **Change Adapter Settings**. Select the two adapters and right-click to bring up the menu shown in Figure 7-10. Select **Bridge connections** to create the bridge interface between the two adapters. When you're finished, restart Windows to allow the changes to take effect.

Figure 7-9: Selecting the Microsoft Loopback Adapter

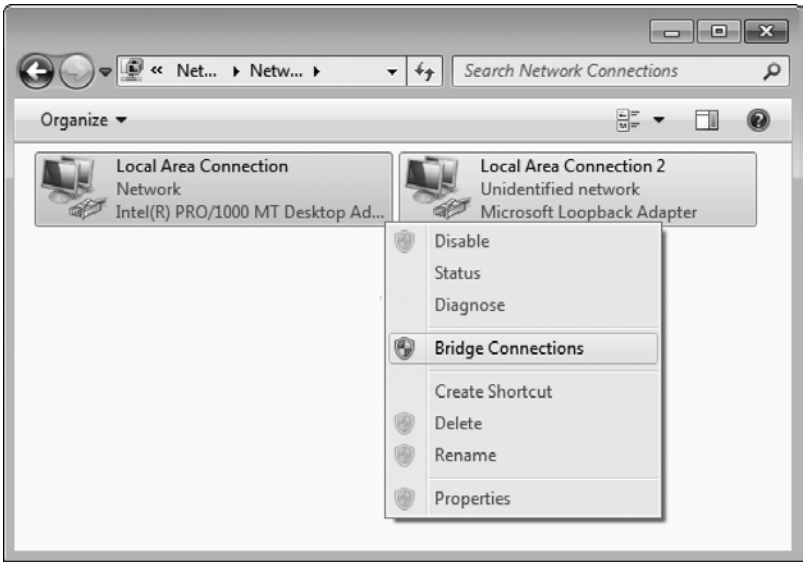


Figure 7-10: Creating a bridge using a Microsoft Loopback Adapter and an Ethernet adapter

To allow GNS3 to use your physical Ethernet adapter on a Windows system, you'll need to configure a Cloud node with the loopback adapter (see "Connecting GNS3 Devices to Physical Hardware" on page 25). It's a good idea to rename your loopback adapter, so it can be clearly identified within GNS3. In this example, I recommend changing **Local Area Connection 2** to **Loopback**.

TUN/TAP Drivers on Mac OS X

If you're using Mac OS X, you need to install a virtual interface driver before GNS3 can access a physical Ethernet Interface. To install TUN/TAP drivers on Mac OS X, download the driver software for your version of Mac OS X from SourceForge (*http://tuntaposx.sourceforge.net/*), run the installer, and follow the instructions.

The drivers should be installed under */Library/Extensions/* and should load automatically when you restart your system. To manually load the driver, change to the */Library/Extensions* directory and enter the following:

```
$ sudo kextload tap.kext
```

Before launching GNS3 on OS X, enter the following command to set permissions on your TAP interfaces. You'll have to enter the command each time you reboot your Mac.

```
$ sudo chown $(id -un):$(id -gn) /dev/tap*
```

To activate a TAP interface and bridge it to a physical Ethernet port in your PC, you must enter the following commands *after* you add a Cloud and attach a router to the nio_tap interface, configured using */dev/tap0*. If you enter the commands beforehand, they will fail and your bridge won't work. You will have to enter these commands every time you load a project that includes a cloud node.

```
$ sudo ifconfig bridge0 create
$ sudo ifconfig bridge0 addm en0
$ sudo ifconfig bridge0 addm tap0 up
```

After entering the commands from a Terminal, Mac OS X should bridge the tap0 interface to the physical Ethernet interface in your Mac, which is en0 in this example. You should use the `ifconfig` command to verify the name of your Mac's Ethernet interface and then replace en0 with that name.

NOTE *If you change the Maximum Transmission Unit (MTU) size of an interface, you may see the error "ifconfig: BRDGADD tap0: Invalid argument" when you go to create the bridge. In this case, you need to make sure that the MTU size of your physical interface matches the MTU size of the TAP interface.*

I recommend adding an annotation to your project that includes these commands as a reminder to create the bridge when you open a project. Then, after opening the project you can copy the annotation and paste it into a Terminal window to save time. Don't forget to enter those commands *every time you open a project* that bridges Ethernet to a TAP interface, or it won't work.

TUN/TAP Drivers on Ubuntu Linux

On Linux, you should be able to connect a Cloud node directly to your Ethernet interface using the NIO Ethernet tab under the Cloud configurator, but if you find that you need TUN/TAP drivers for Ubuntu, update your package manager and enter the following command to install the package.

```
$ sudo apt-get install uml-utilities
```

If you're running some other version of Linux, you may have to install a different package, but this package should work on most Debian-based distributions.

Connecting to Live Switches

GNS3 is great software, but it has some limitations. For example, the NM-16ESW switch module doesn't include all the advanced features of an actual layer 2 or layer 3 switch. If you want to work with advanced switching, you have to use additional software like Cisco IOU or use physical Cisco switches. If you're creating CCNA labs, then one live Cisco switch might be enough, but if you're creating CCNP or CCIE labs, you'll probably use multiple live switches. Often the goal is to have a GNS3 router connected to each external switch. This is tricky because most PCs have only one Ethernet adapter. Fortunately, GNS3 and IOS have this sorted out. Your options are to trunk VLANs to the switch over a single Ethernet adapter in your PC or to install multiple Ethernet adapters in your PC.

NOTE *When connecting to live switches, you can save yourself a lot of heartache by using Linux. Windows and Mac OS X are more complicated to set up and can be unreliable.*

In this section, I'll cover the following ways to connect GNS3 devices to live Cisco switches.

Standard 802.1Q Trunk This method uses a standard 802.1Q trunk to allow GNS3 devices to communicate with a live Cisco switch via your PC's Ethernet adapter.

Breakout Switch This method uses a specially configured Ethernet switch called a breakout switch to allow GNS3 devices to connect to *multiple live switches* using only a single Ethernet adapter in your PC. To create a breakout switch, you must have a second physical Ethernet switch on hand to use as the breakout switch.

The option you choose depends on how many switches you own, your PC's operating system, and what Ethernet adapters are already installed in your PC. Let's start by looking at how an 802.1Q trunk works to reach a Cisco switch.

Configuring a Standard 802.1Q Trunk

In my opinion, an 802.1Q trunk is the best way to attach live switches to your GNS3 projects. The advantages of a standard 802.1Q trunk are that it is easy to set up and works the same as connecting a switch to a physical network. The disadvantage is that your PC operating system or your PC's Ethernet drivers might not support it. Often they strip the 802.1Q tags from packets coming from the switch into your PC. Without the proper tags, GNS3 has no way of knowing which VLANs your packets belong to and in turn doesn't know where to forward the packets, which breaks your network.

If you poke around the Internet, you'll find all sorts of creative solutions that people have come up with to circumvent this problem, but they are generally platform-specific and vary from machine to machine.

NOTE *One way to prevent tag stripping on Mac OS X and Linux is to use a USB Ethernet adapter that supports 802.1Q tagging and jumbo frames. One such adapter that works well is the StartTech USB31000SW adapter, but any adapter that uses the ASIX AX88772A chipset should work.*

In this example, you'll place either an *EtherSwitch router* or *Ethernet switch node* in your workspace and configure it with the 802.1Q trunking protocol. Next, you'll add a Cloud node to your workspace and configure it using an NIO interface, and link it to the GNS3 switch. The Cloud node can either be directly connected, or bridged, to your PC's Ethernet adapter (depending on your OS). You'll then plug an Ethernet cable from your PC's Ethernet adapter into a port on the live Cisco switch. The switch port you choose also needs to be configured with 802.1Q trunking. After both the GNS3 switch and physical switch have been configured, you should be able to route GNS3-generated VLAN packets through the trunk to the live switch.

In this section, we'll create a simple project that connects our GNS3 network to a live c3550 switch using two VLANs (10 and 20). Begin by creating the topology shown in Figure 7-11. When configuring the Cloud node, choose your PCs Ethernet adapter name, found under the NIO Ethernet tab.

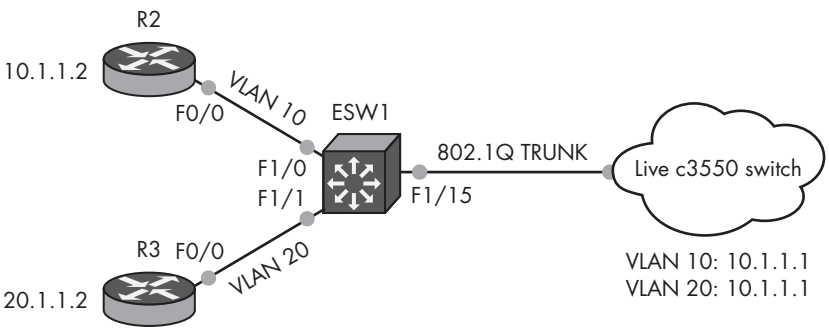


Figure 7-11: Sample topology using a standard dot1q trunk to live switch

To configure VLANs and an 802.1Q trunk using the *EtherSwitch router*, open a console on ESW1 and enter the commands from the following listing.

```
   ESW1# vlan database
❶ ESW1(vlan)# vlan 10
❷ ESW1(vlan)# vlan 20
   ESW1(vlan)# apply
   ESW1(vlan)# exit
   ESW1# configure terminal
   ESW1(config)# int f1/15
❸ ESW1(config-if)# switchport mode trunk
❹ ESW1(config-if)# switchport trunk encapsulation dot1q
   ESW1(config-if)# int f1/0
   ESW1(config-if)# switchport mode access
❺ ESW1(config-if)# switchport access vlan 10
   ESW1(config-if)# int f1/1
   ESW1(config-if)# switchport mode access
❻ ESW1(config-if)# switchport access vlan 20
```

The previous commands create VLAN 10 ❶ and 20 ❷ on the switch, configure a trunk port ❸ using the dot1q protocol ❹, and assign access ports to VLAN 10 ❺ (for router R2) and VLAN 20 ❻ (for router R3).

If instead you choose to use an *Ethernet switch node*, configure one port as an 802.1Q trunk and the others as VLAN access ports, as shown in Figure 7-12.

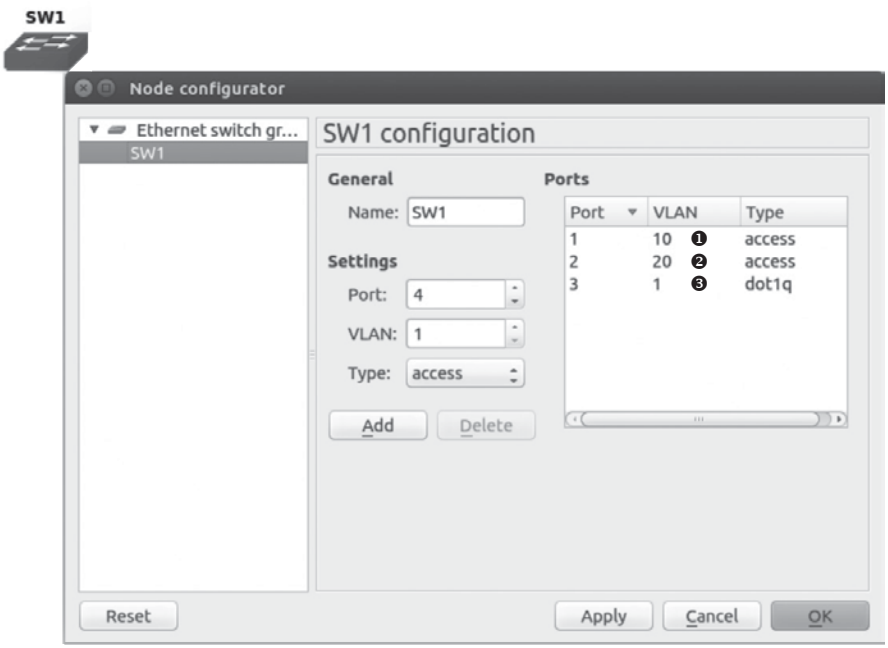


Figure 7-12: Ethernet switch node configured with a dot1q trunk port

Ports 1 and 2 are configured as *access* ports using VLAN 10 ❶ and 20 ❷, and port 3 is the *dot1q* trunk port ❸ that's connected to the Cloud node.

Next, log on to your live Cisco switch and create the same VLANs. Then, configure the 802.1Q trunk port and cable it to your PC's Ethernet adapter. The following listing is an example of how to configure a c3550 switch, using commands you've seen throughout the chapter.

```
c3550# configure-terminal
c3550(config)# ip routing
c3550(config)# interface vlan 10
c3550(config-vlan)# ip address 10.1.1.1
c3550(config-vlan)# interface vlan 20
c3550(config-vlan)# ip address 20.1.1.1
c3550(config-vlan)# exit
c3550(config)# Interface f0/1
c3550(config-if)# switchport trunk encapsulation dot1q
c3550(config-if)# switch port mode trunk
c3550(config-if)# switchport trunk allowed vlan all
c3550(config-if)# speed 100
c3550(config-if)# duplex full
```

To complete the project, configure routers R2 and R3. Log on to router R2 and configure an IP address and default gateway for VLAN 10.

```
R2(config)# interface f0/0
R2(config-if)# description Using VLAN 10
R2(config-if)# ip address 10.1.1.2 255.255.255.0
R2(config-if)# no shutdown
R2(config-if)# exit
R2(config)# ip route 0.0.0.0 0.0.0.0 10.1.1.1
```

Now, log on to router R3 and configure an IP address and default gateway for VLAN 20.

```
R3(config)# interface f0/0
R3(config-if)# description Using VLAN 20
R3(config-if)# ip address 20.1.1.2 255.255.255.0
R3(config-if)# no shutdown
R3(config-if)# exit
R3(config)# ip route 0.0.0.0 0.0.0.0 20.1.1.1
```

Test VLAN routing through the switch by entering a `ping` command from one VLAN to another.

```
R3# ping 10.1.1.2
!!!!!
```

When you're finished with the project, you can further verify the configuration using tools like CDP or Wireshark.

Creating the Elusive Breakout Switch

A *breakout switch* is another way to connect real switches to GNS3 projects, and it's common to set one up using Ubuntu Linux. Although a breakout switch does work on other systems, it's easiest to set up on Linux. As mentioned earlier, other operating systems like Windows and Mac OS X might remove VLAN information from the packets. For this reason, it's best to install Ubuntu on real hardware if you plan to create a breakout switch. If you use a virtual machine, your underlying host OS may strip the VLAN tags, and the breakout switch won't work. You can sometimes get around this by using a USB Ethernet adapter, like the StartTech USB31000SW mentioned in "Configuring a Standard 802.1Q Trunk" on page 30.

This switching method requires a minimum of two real Cisco switches; one is the breakout switch, and the other is one or more live Cisco switches that are used in your GNS3 project. The breakout switch is used to fool the live Cisco switches into thinking that each of your GNS3 routers is *directly connected* to a live switch with an Ethernet cable. In reality, only one Ethernet adapter is used in your PC to connect all your GNS3 routers to the live switches, and that adapter is connected to the breakout switch. The breakout switch is then configured to *break out all the VLANs* into individual interfaces that you plug into other live Cisco switches using Ethernet cables (one per VLAN). I call these "breakout cables." Figure 7-13 shows the physical layout of your PC and switches.

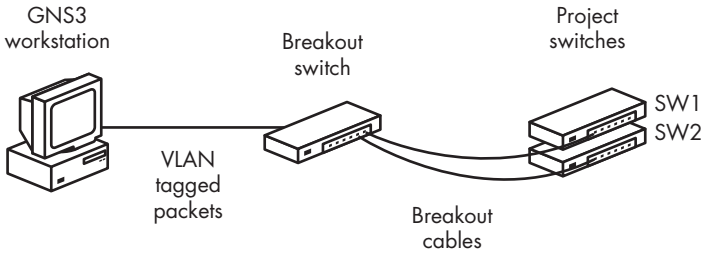


Figure 7-13: Physical layout using a breakout switch and live Cisco switches

Here's the flyby view of how it works. Breakout VLANs are configured on your Linux Ethernet adapter using the `vlan` package, and an 802.1Q trunk is configured on the breakout switch. In GNS3, you'll add one or more Cloud nodes that are configured with the Linux breakout VLANs using NIO Ethernet interfaces. In your GNS3 project, a router is connected to each of the Cloud node's VLAN interfaces (one router per interface). Packets leaving the routers go into the Cloud node, where Linux tags them with a VLAN ID and passes them to the breakout switch using your PC's Ethernet adapter. The breakout switch receives the packets on a standard 802.1Q trunk port. From there, the breakout switch uses the VLAN IDs to identify and transparently pass the packets to one (or more) live Cisco switches. What makes this so ingenious is that each GNS3 router has a separate Ethernet cable that is connected to a port on the live Cisco switch, even though your PC has only a single Ethernet adapter.

Breakout switches are fairly straightforward to set up on Linux. Start by updating your package manager and installing VLAN support on your Ubuntu system.

```
$ sudo apt-get install vlan
```

Now, enable the `8021q` Linux module using the `modprobe` command.

```
$ sudo modprobe 8021q
```

Next, increase the MTU frame size on the Ethernet interface, and create your breakout VLANs using the `vconfig` command. Be sure to substitute *eth0* with the name of your interface.

```
❶ $ sudo ifconfig eth0 mtu 1546
❷ $ sudo vconfig add eth0 10
❸ $ sudo vconfig add eth0 20
```

NOTE *Your Ethernet adapter must support frame sizes above the standard maximum of 1,500 bytes.*

The increased frame size ❶ makes room for the additional VLAN tags. Create one breakout VLAN for each router in your project. In the previous listing, I created two breakout VLANs (10 ❷ and 20 ❸) on my Linux PC's *eth0* interface.

NOTE *Don't confuse breakout VLANs with VLANs you create in your GNS3 projects. It's important to understand that breakout VLANs should be used only by Linux and the breakout switch—they are not to be used in your GNS3 project or live Cisco switches.*

Start configuring the breakout switch by increasing the system wide MTU size. After the command is entered, you must reload the switch for the change to take effect.

```
Breakout(config)# system mtu 1546
```

After the switch reboots, log on and configure an 802.1Q trunk link as follows.

```
Breakout# configure terminal
Breakout(config)# interface FastEthernet 0/1
❶ Breakout(config-if)# switchport trunk encapsulation dot1q
❷ Breakout(config-if)# switchport mode trunk
❸ Breakout(config-if)# switchport trunk allowed vlan all
```

The interface you configure as the trunk is then connected to your PC's physical Ethernet adapter using an Ethernet cable. As shown above, dot1q encapsulation ❶ is configured on the trunk port ❷, and all VLANs ❸ are allowed through the trunk.

Next, go under each interface that you plan to connect to a live switch, and configure a breakout VLAN and a dot1q tunnel for each VLAN, as shown in the following listing.

```
Breakout(config)# vlan 10
Breakout(config-vlan)# vlan 20
Breakout(config-vlan)# exit
Breakout(config)# interface FastEthernet 0/2
Breakout(config-if)# description GNS3 R1 Physical Uplink to Live Switch SW1
❶ Breakout(config-if)# switchport access vlan 10
❷ Breakout(config-if)# switchport mode dot1q-tunnel
❸ Breakout(config-if)# l2protocol-tunnel cdp
Breakout(config-if)# interface FastEthernet 0/3
Breakout(config-if)# description GNS3 R2 Physical Uplink to Live Switch SW2
❹ Breakout(config-if)# switchport access vlan 20
Breakout(config-if)# switchport mode dot1q-tunnel
Breakout(config-if)# l2protocol-tunnel cdp
```

Above, our breakout switch's FastEthernet 0/2 interface is configured for VLAN 10 ❶, dot1q tunneling ❷, and Cisco discovery protocol tunneling ❸. FastEthernet 0/3 is configured the same way, but for VLAN 20 ❹. These are the interfaces used to connect GNS3 routers to your live Cisco switches.

We already know that *Cisco Discovery Protocol (CDP)* is used to share and gather information with directly connected Cisco equipment, often called *neighbors*. However, our live Cisco switch is not directly connected to GNS3; instead, it's connected to the breakout switch. In this instance, the only way to use CDP is by tunneling the protocol through the breakout switch to a live Cisco switch with the `l2protocol-tunnel cdp` command. (You can also tunnel STP and VTP.) This is where things get tricky because CDP tunneling does not work on all switches. If you need to use CDP, be sure to choose a breakout switch that fully supports CDP tunneling. Table 7-1 lists a few common Cisco switches and their CDP tunneling capabilities.

Table 7-1: CDP Tunneling Compatibility

Switch	CDP Tunneling Compatibility
Cisco 2950	CDP will not work in either direction. Layer 2 tunneling is not supported on this switch.
Cisco 3550	CDP works only in one direction. Neighbors cannot be seen on the switches regardless of the IOS version.
Cisco 3560	CDP works only in one direction. Neighbors cannot be seen on the switches regardless of the IOS version.
Cisco 3750	Bidirectional CDP and fully functional (layer 2 and layer 3) using IP Services image. IP Base image does not support tunneling.
Cisco 4948	Bidirectional CDP and fully functional (layer 2 and layer 3) using a minimum IP Services image.

The Cisco 3750 switch works well, both as a breakout switch and for tunneling CDP, but it's not the cheapest switch available. If you don't have the cash to spring for this model, you can choose a cheaper model that should work as a breakout switch, but you won't have complete transparency in your GNS3 project.

Before continuing, be sure that Fa0/1 on the breakout switch is connected to your PC's Ethernet port, and that Fa0/2 and Fa0/3 are connected to Ethernet ports on your live project switches. After checking those connections, start GNS3 and configure a Cloud node using the Linux VLANs you created previously, as shown in Figure 7-14.

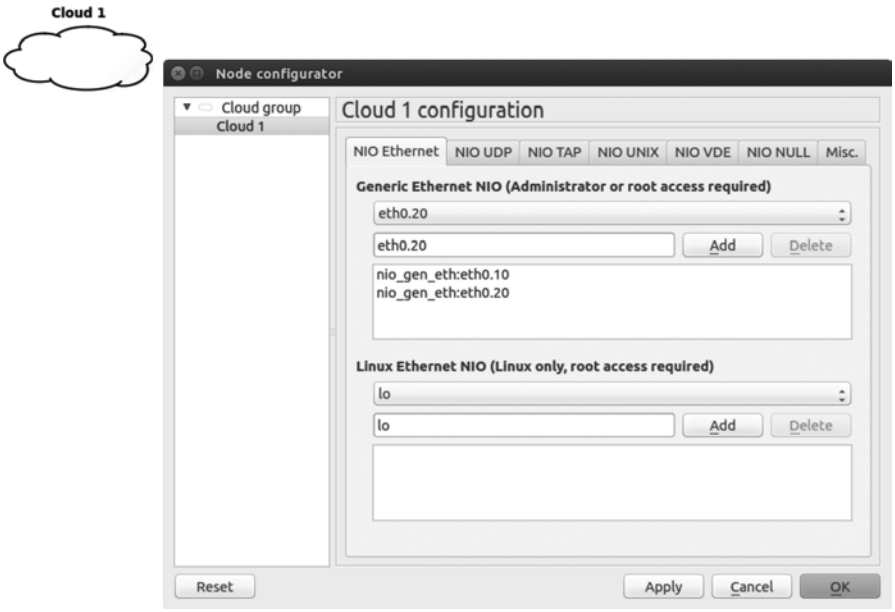


Figure 7-14: Configure a Cloud node using Linux VLANs

NOTE *Even though you're using Linux, it's important to use the "Generic Ethernet NIO" and not the Linux Ethernet NIO. Otherwise, VLAN tagging may not work.*

When you're finished adding breakout VLANs to the Cloud node, add a couple of routers to your workspace and create a link from each router to a breakout VLAN on the Cloud node. In Figure 7-15, router R1 (F0/0) is linked to VLAN 10 in the Cloud using *nio_gen_eth:eth0.10*, and R2 (F0/0) is linked to VLAN 20 using *nio_gen_eth:eth0.20*.

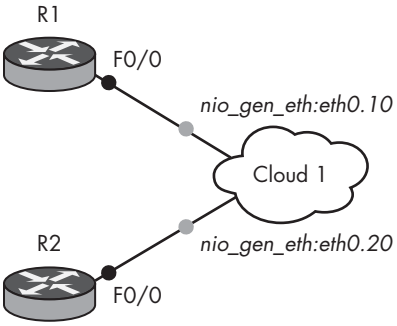


Figure 7-15: Virtual routers connected to Linux VLANs using a Cloud node

Because the breakout switch is only used to split VLANs into multiple physical ports, it requires no further configuration. In this example, router R1 is now linked to any live switch plugged into breakout switchport f0/2, and R2 should be linked to any live switch plugged into port f0/3.

NOTE *If you create large projects using many VLANs, you may want to assign only one Linux VLAN interface per Cloud node to help clarify the layout in your workspace.*

Your GNS3 routers should now be able to communicate with one or more real Cisco switches.

Optional Breakout Switch Configuration

If you're running Windows or Mac OS X, you may be able to use a GNS3 Switch node to link your project to a breakout switch, as shown in Figure 7-16. In this setup, the breakout switch is configured in the same way as above, but you'll need to make some tweaks to your PC. The Ethernet Switch node connects to a Cloud node using an NIO interface configured with a loopback adapter on Windows or a TAP interface on Mac OS X. The virtual adapter is bridged to your PC's physical Ethernet adapter. This allows Cloud 1 to connect to the breakout switch using your PC's Ethernet adapter.

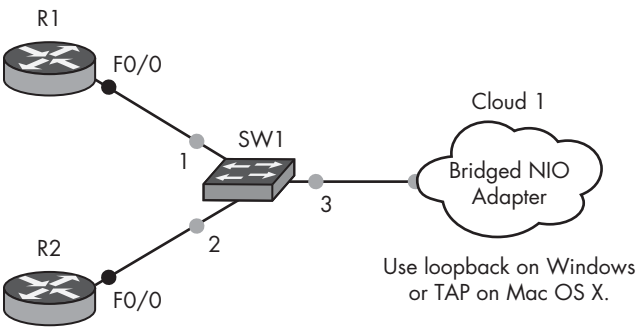


Figure 7-16: Breakout switch configuration using an Ethernet switch node

For the breakout switch to work, you should increase the MTU size on your PC's physical Ethernet adapter, but not all adapters support this feature.

If you're a Mac OS X user, the MTU size must be increased on both your physical adapter and virtual adapter. To increase the MTU size from the command line, use the following example.

```
$ sudo ifconfig en0 mtu 1546
$ sudo ifconfig tap0 mtu 1546
```

NOTE *On Mac OS X, configure the virtual and physical adapter using the same MTU size, or the bridge creation will fail.*

You may have to check your Ethernet adapter documentation to configure your adapter on Windows. On many adapters, however, the MTU setting can be found under the Advanced properties of the adapter, as shown in Figure 7-17. In this example the `Jumbo Packet` value is set to 9014 bytes on an Intel PRO/1000 MT card.

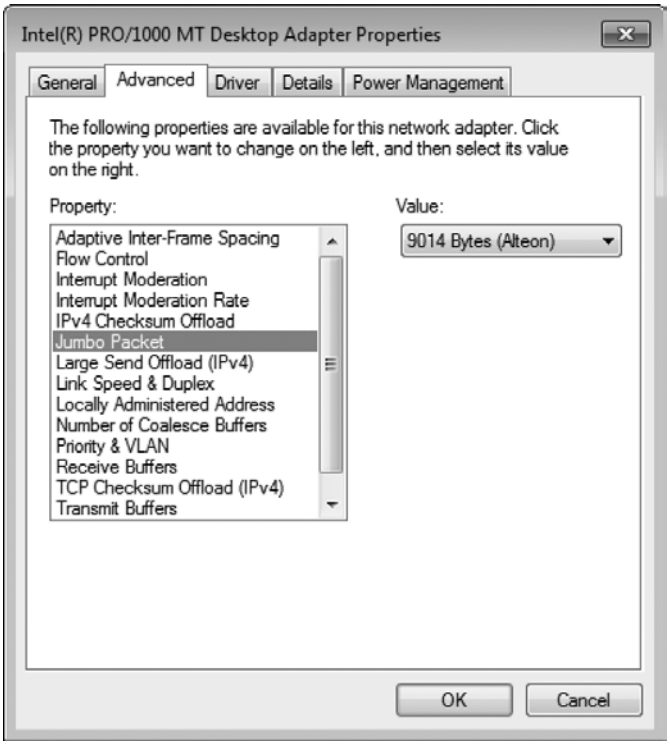


Figure 7-17: Setting the Jumbo Packet size on Windows

Next, bring up the Node configurator for the Ethernet Switch node to define VLANs and the dot1q trunk port, as shown in Figure 7-18.

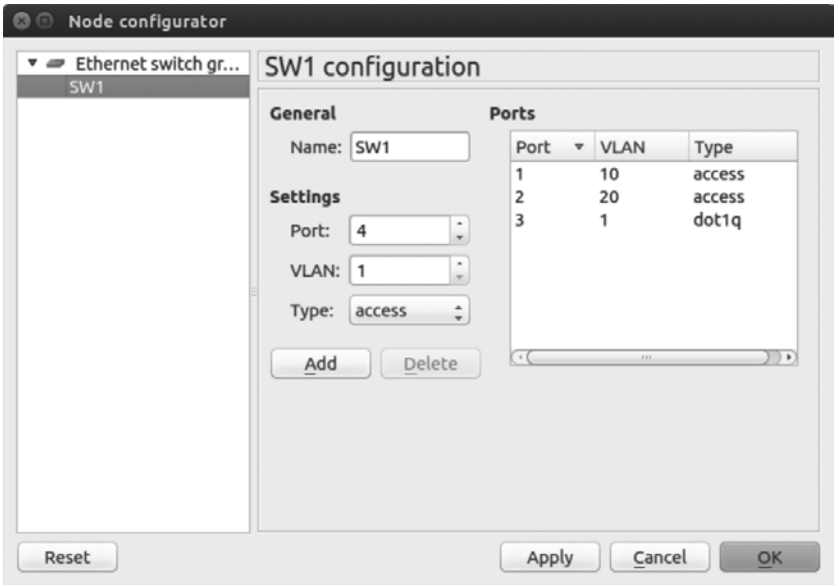


Figure 7-18: Ethernet Switch node configured for breakout switch

In this example, R1 is connected to access Port 1 (VLAN 10), R2 is connected to access Port 2 (VLAN 20), and Port 3 is the dot1q trunk to the breakout switch. After everything is configured in GNS3, you can log on and configure the breakout switch with VLANs and dot1q tunneling as previously described.

Using Multiple Adapters in Your PC

Instead of using an 802.1Q trunk or a breakout switch, you could use one physical Ethernet interface for each router in GNS3. If you have a laptop, you can use a USB hub and connect multiple USB Ethernet adapters to your PC; with a desktop, you could use the USB hub method or purchase a multiport Ethernet card. Figure 7-19 shows the physical layout of the design.

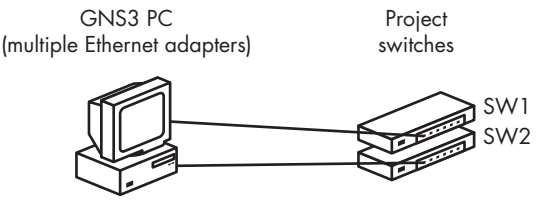


Figure 7-19: Physical layout using multiple adapters in a PC

Connect each Ethernet adapter in your PC to a physical port on a live Cisco switch. Log on to each switch and configure an access port using the `switchport mode access` command. Then, assign a unique VLAN number to the port by entering `switchport access vlan {vlan number}`, as in the following listings.

```
c3550_sw1(config)# interface f0/1
c3550_sw1(config-if)# description VLAN used for PC Ethernet Adapter 1
c3550_sw1(config-if)# switchport mode access
c3550_sw1(config-if)# switchport access vlan 10
```

```
c3550_sw2(config)# interface f0/1
c3550_sw2(config-if)# description VLAN used for PC Ethernet Adapter 2
c3550_sw2(config-if)# switchport mode access
c3550_sw2(config-if)# switchport access vlan 20
```

To connect GNS3 devices to the switches, add a Cloud node to your workspace and assign one NIO Ethernet interface for each Ethernet adapter in your PC. Connect your GNS3 routers to each NIO interface using one router per interface.

NOTE *Mac OS X users will have to create a unique bridge interface for each adapter and bridge each Ethernet interface to a unique TAP interface. Mac OS X supports up to 16 TAP devices (tap0 through tap15).*

After all the devices in your project have been configured, your GNS3 devices should be able to communicate with the live Cisco switches. Further configuration is required on the switches to enable routing between GNS3 routers and is determined by the switch block design you create.

Now let's take a look at connecting GNS3 to the Internet.

Connecting GNS3 Devices to the Internet

To connect GNS3 devices to the Internet, you need to use an Ethernet adapter in your PC. Wireless network adapters are not supported if you use them directly, though one may work if bridged to a loopback adapter or TAP interface (but don't count on it). Make the GNS3 to Ethernet connection the same way that you connect to a physical switch: add a Cloud node to your project and configure it with either an NIO TAP or NIO Ethernet interface.

Configuring Windows

On Windows systems, create a bridge using a loopback adapter and your PC's physical Ethernet adapter. Then, in GNS3, use a Cloud node configured using NIO Ethernet and select the loopback adapter. Because the loopback adapter is bridged to the physical Ethernet adapter, you can use the Cloud to connect to networks outside GNS3—including the Internet.

On Windows 8.*x*, install a loopback adapter, but do not add it to a bridge. Instead, configure Internet Connection Sharing (ICS) on your physical interface (Ethernet or Wifi). To do so, right-click the Windows Start button and select **Network Connections**. Next, right-click your physical Interface and select **Properties**. Select the **Sharing** tab, and check the

Allow other network users to connect through this computers Internet connection option. Lastly, select your loopback adapter from the Home Networking Connection drop-down, and click **OK**.

Configuring Unix-Based Systems

On Ubuntu Linux, create an NIO Ethernet connection using your PC's Ethernet interface. On Mac OS X and some Linux systems, use an NIO TAP connection configured with */dev/tap0* and bridge the TAP interface to your PC's Ethernet interface.

Creating a Simple Network

Create a project by adding a router and a Cloud node to your workspace and adding a link from the router to the Cloud. In Figure 7-20, the Cloud is configured using the Windows loopback adapter, named `Local Area Connection 2`.

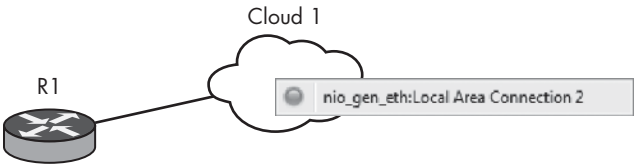


Figure 7-20: Connecting to a Cloud using a Windows loopback adapter

To test Internet connectivity, attach an Ethernet cable from your PC to an Internet device, like a cable modem, and configure an IP address on your GNS3 router. If you use DHCP to assign an IP address as I did below, you may have to wait a moment for the router to receive an IP address before you can test connectivity.

```
R1(config)# ip domain-lookup
R1(config)# ip name-server 8.8.8.8
R1(config)# interface f0/0
R1(config)# no shutdown
R1(config-if)# ip address dhcp

*Mar  1 00:01:08.875: %DHCP-6-ADDRESS_ASSIGN: Interface FastEthernet0/0
assigned
DHCP address 192.168.1.101, mask 255.255.255.0, hostname R1
```

After your router has obtained an IP address, you should be able to ping a host on the Internet. Try to ping *www.gns3.net*! You're not limited to using a router to connect to the Internet; you can also use an ASA device, Juniper router, or any other device that supports TCP/IP.

NOTE *If you have other devices behind router R1, you need to configure* Network Address Translation (NAT) *on R1 before they can route to the Internet. For more information, visit Cisco's website* (http://www.cisco.com/en/US/tech/tk648/tk361/technologies_tech_note09186a0080094e77.shtml#topic6).

Final Thoughts

In this chapter, we looked at configuring GNS3 device nodes and integrating them into your projects using Cisco IOS. They're simple to set up and well suited for large projects because they greatly reduce the load on your PC. Compared to Dynamips devices, GNS3 device nodes use almost no PC resources.

Connecting a GNS3 device to a live Cisco switch is tricky. If you choose to create a standard 802.1Q trunk to connect with a live switch, you can use either an EtherSwitch router or an Ethernet switch node, but your PC operating system and Ethernet adapter drivers must both support 802.1Q tagging. Without the proper VLAN tags, trunking won't work.

If you have an extra IOS switch, you can create a breakout switch, which is a very reliable way to integrate multiple real Cisco switches into your GNS3 projects. A breakout switch works most reliably on Linux systems but can also be configured using Windows and Mac OS X.

In the next chapter, we'll look at some more advanced features including Cisco, ASA, and IDS/IPS.

We hope that you enjoyed this
early sample of *The Book of GNS3*,
courtesy of No Starch Press.

To order the print book or ebook,
visit *nostarch.com/gns3*
and use code **GNS3** to get 30% off.