

### **Pre-Lab 4: LAN Switching Part 1**

1. Review the Linux man pages for `traceroute` at [www.linuxmanpages.com](http://www.linuxmanpages.com).
2. Review LAN switching and virtual LANs at the *LAN Switching* link on the course web page.

### **Pre-Lab 4 Questions:**

1. Describe the difference between a LAN switch/bridge and a router.
2. What is the difference between an Ethernet switch and an Ethernet hub? Which is more suitable for a network with a high traffic load, a switch or a hub? Explain.

**LAB 4**

A *bridge* or *LAN switch* is a device that interconnects two or more local area networks (LANs) and forwards packets between these networks. Different from IP routers, bridges and LAN switches operate at the data link layer. I.e. bridges and LAN switches forward packets based on MAC addresses, whereas IP routers forward packets based on IP addresses.

The term *bridge* was coined in the early 1980s. Today, when referring to data link layer interconnection devices, the terms *LAN switch* or *Ethernet switch* (in the context of Ethernet) are much more common. Since many of the concepts, configuration commands, and protocols for LAN switches in this lab use the old term *bridge*, remember to keep in mind that **for the most part the terms *switch* and *bridge* are interchangeable**.

**NOTE:** Remember to reboot the PCs and to save all your files in `/root/labdata/<user>` and your floppy or usb drive.

**Network Setup for lab 4**

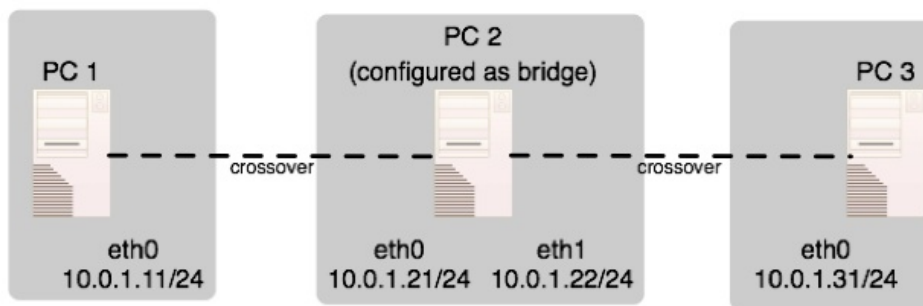
The configuration of the equipment is changed several times during the course of this lab. The IP addresses of the PCs stay relatively constant, however, so you can set them now according to the table below. Note that this lab is unusual in that both Ethernet interfaces of the machines are on the same subnet. This is a result of the Layer 2 focus of the lab.

PC	IP address of eth0	IP address of eth1
PC1	10.0.1.11/24	10.0.1.12/24
PC2	10.0.1.21/24	10.0.1.22/24
PC3	10.0.1.31/24	10.0.1.32/24
PC4	10.0.1.41/24	10.0.1.42/24

**Exercise 1 : Configuring a Linux PC as a bridge**

In this exercise you will learn how to configure a Linux PC as a bridge. Ethernet bridging functionality is integrated in all recent versions of Linux. The configuration of bridging functions in Linux is done with configuration commands and tools. In this lab, we will use the bridge configuration tool, *brctl*.

The network configuration for this exercise is shown in the figure below. PC1 and PC3 are hosts in this network, and PC2 is a bridge that connects the two together. Notice the need for crossover cables – when connecting two network interface cards directly together, a crossover cable must be used. In the lab, these cables are orange.



4.1 Topology for Lab 4, Exercise 1

**PART A: IP configuration of the PCs**

**A.1:** Connect the PCs as illustrated in Figure 4.1.

**A.2:** Set up the IP addresses of PC1, PC2 and PC3 as shown in Figure 4.1. Disable interfaces eth1 on PC1 and PC3. You can do this by typing `ifconfig eth1 down` on those PCs.

**A.3:** Since you will need to recognize the MAC addresses of the PCs frequently in this lab, use `ifconfig -a` on each PC to display them and record them in the table below.

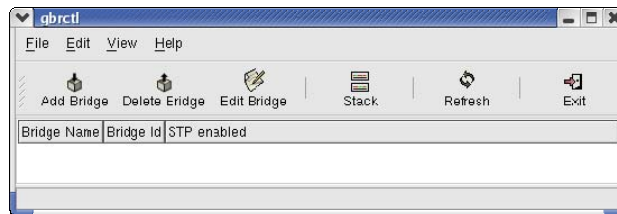
PC	MAC address of eth0	MAC address of eth1
PC1		
PC2		
PC3		
PC4		

**PART B: Configuring the Linux PC bridge**

Now it's time to configure PC2 as a bridge that forwards packets between the two hosts in Figure 4.1. The bridge configuration on the Linux PCs is done with the tool *gbrctl*. The *gbrctl* tool has a graphical user interface to configure bridging functions on a Linux PC.

**B.1:** Start *gbrctl* by typing the following command in a terminal window on PC2:  
 PC2% `gbrctl`

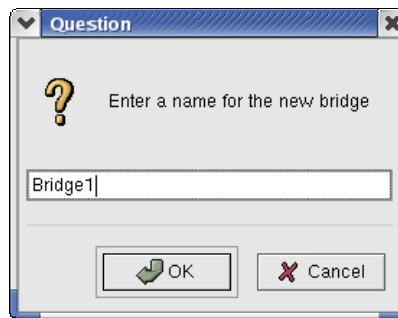
The command displays the main *gbrctl* window as shown in Figure 6.2, which is used to configure the bridging functions. The *gbrctl* tool is terminated by selecting *File* and then *Exit*.



4.2 Main *gbrctl* window

**B.2:** It is possible to configure multiple independently operating bridges on the same PC (provided there are enough network interface cards on the PC to do so). Each bridge is assigned a name and is associated with a set of interfaces. Here, you configure one bridge on PC2 and assign the bridge the name *Bridge1*.

To start the configuration of PC2, select *Add Bridge* in the *gbrctl* main window in Figure 4.2. A window pops up as shown in Figure 4.3, which asks for a name for the new bridge. Enter the name *Bridge1*. When you click OK, the newly created bridge is displayed in the *gbrctl* main window.

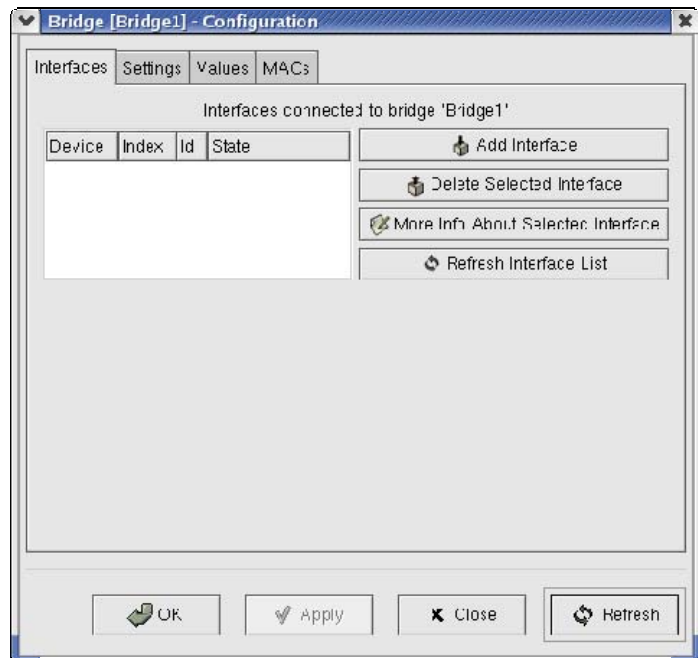


4.3 Prompt to add new bridge

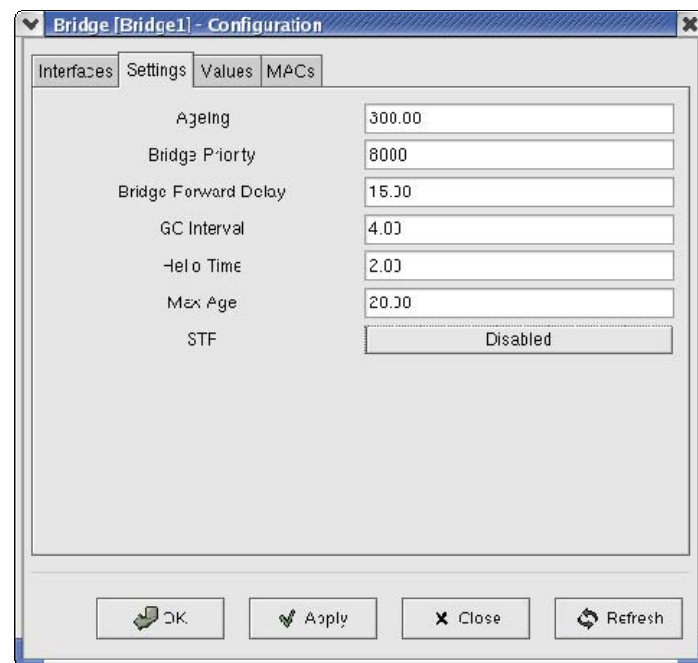
**B.3:** After the bridge is created, it is configured in the following steps:

- From the *gbrctl* main window, select the bridge name *Bridge1*, and then select *Edit Bridge*. This displays the *Bridge Configuration* window, which is shown in Figure 4.4.
- The first part of the configuration is the assignment of network interfaces to the bridge. Interfaces are assigned by clicking on the *Interfaces* tab and selecting *Add Interface*. Then, type the name of the interface to be added. For PC2, add the interfaces *eth0* and *eth1*.
- The next part of the configuration sets the parameters of the spanning tree protocol (STP). Select the *Settings* tab in the *Bridge Configuration* window (see Figure 4.4). In this exercise, the spanning tree protocol is not used. Therefore, you need to disable the spanning tree protocol by toggling the button next to the STP label, so that it shows the label *Disabled*.
- When the interfaces and the spanning tree protocol parameters are configured, terminate the *gbrctl* application.
- In the last part of the bridge configuration, you activate the bridge *Bridge1* from a terminal window. On a Linux PC, each created bridge is represented as a network interface. Therefore, if you type the command `ifconfig -a` on PC2, the command shows an interface named *Bridge1*, in addition to the other interfaces, *eth0*, *eth1*, and *lo*. The bridge is activated by enabling the interface associated with the bridge. This is done with the following command:  
PC2% `ifconfig Bridge1 up`

**NOTE:** Activating a bridge disables the IP configuration of the interfaces assigned to it. Hence it is no longer possible to issue `ping` commands to these interfaces.



4.4 Bridge Configuration window (Interfaces)



4.5 Bridge Configuration window (Settings)

### PART C: Observing the bridge in operation

When the bridge configuration of PC2 is complete, PC2 forwards packets between PC1 and PC3. This exercise asks you to observe the forwarding of packets.

**C.1:** When bridging is activated on PC2, the configured IP addresses on PC2 should be disabled. To verify this, issue a `ping` command to interfaces *eth0* and *eth1* of PC2 from PC1 and PC3 respectively:

```
PC1% ping 10.0.1.21
PC3% ping 10.0.1.22
```

If PC2 is configured as a bridge correctly, these ping commands should fail.

**C.2:** Start *ethereal* on PC1 and PC3, and capture traffic on interface *eth0* on both systems.

**C.3:** Clear the ARP caches on PC1 and PC3. Note that, in Linux, each ARP entry has to be deleted separately with the command `arp -d <IPaddress>`.

**C.4:** Issue a ping command from PC1 to PC3 and save the output:

```
PC1% ping -c 1 10.0.1.31
```

- Observe that PC2 actually forwards the packets between PC1 and PC3.
- Pay attention to whether or not PC2 modifies the source and destination MAC and IP addresses?
- Think about what might be different if PC2 was configured as an IP router?

**C.5:** Run *traceroute* from PC1 to PC3 and save the output:

```
PC1% traceroute 10.0.1.31
```

- Here, you should observe that PC2 does not appear in the output of *traceroute*.
- Think about why PC2 is not visible from PC1.
- Again, think about how the output might differ were PC2 configured as an IP router.

**C.6:** Next, change the IP address of PC3 to 10.0.2.12/24. Note that PC1 and PC3 now have different IP network prefixes. Repeat Step 4 using the new address for PC3.

- Think about an explanation for why the ping command does or does not work now.

**C.7:** Save the ICMP Echo Request packet(s) captured by *ethereal* on PC1 and PC3.

## PART D: Manipulating the PC bridge

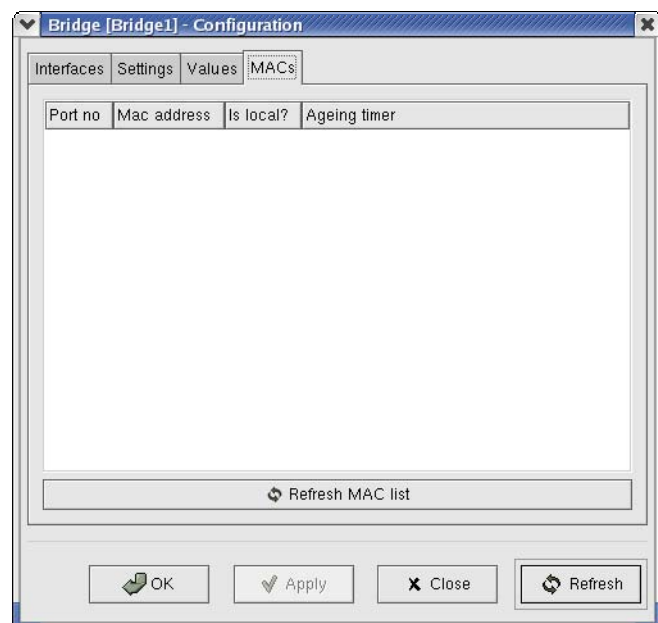
To familiarize you with some tasks related to *gbrctl* we look at

- How to display the MAC forwarding table
- How to delete the contents of the MAC forwarding table
- How to turn off the bridging functions

**D.1:** First, reset the IP address of the *eth0* interface of PC3 to 10.0.1.31/24.

**D.2:** The MAC forwarding table of a bridge plays the same role as the routing table of an IP router. To view the contents of the MAC forwarding table of *Bridge1* on PC2, perform the following steps:

- a. Start *gbrctl*
- b. Select *Bridge1* in the *gbrctl* main window and then select *Edit Bridge*
- c. Select the *MACs* tab. This displays a window as shown in Figure 4.6
- d. Click the *Refresh MAC list* button as shown at the bottom of Figure 4.6. This displays the current state of the MAC forwarding table



4.6 Bridge Configuration window

**D.3:** The *gbrctl* tool does not have a convenient way to delete the contents of the MAC forwarding table. Instead you must exploit the fact that a bridge automatically deletes the entry in the forwarding table if it has not been looked up for a certain time, which is determined by *Ageing* parameter in *gbrctl*. To delete the entries in the forwarding table, you must set the *Ageing* parameter to 0 seconds. Here are the steps to delete the entries on PC2:

- a. In the *gbrctl* main window, select the *Settings* tab and obtain a window as shown in Figure 4.5
- b. Set the *Ageing* field to 0 and click *Apply*. All entries in the MAC forwarding tables are immediately removed
- c. Select the *MACs* tab and click *Refresh MAC list*. You should now see only the MAC addresses of the local interfaces at PC2
- d. Once the entries are deleted, set the *Ageing* entry back to the original value (the default value is 300 seconds)

**D.4:** Disabling a bridge on a Linux PC is done in two steps. To disable *Bridge1* on PC2:

- a. First, deactivate the interface associated with *Bridge1*. This is done by typing  
PC2% `ifconfig Bridge1 down`
- b. Second, in the *gbrctl* main window (Figure 4.2), select *Bridge1*, and select *Delete Bridge*

You can verify that the bridge is disabled as follows:

1. Verify that PC2 is operating as a normal host. To do this, issue a `ping` command to interface *eth0* of PC2 from PC1:  
PC1% `ping -c 1 10.0.1.21`
2. Verify that PC2 does not forward packets by issuing the following `ping` command (from PC1 to PC3):  
PC1% `ping -c 1 10.0.1.31`  
The `ping` command should not be successful.

Re-enable interface PC1 and PC3. You can do this with the command `ifconfig eth1 up`.

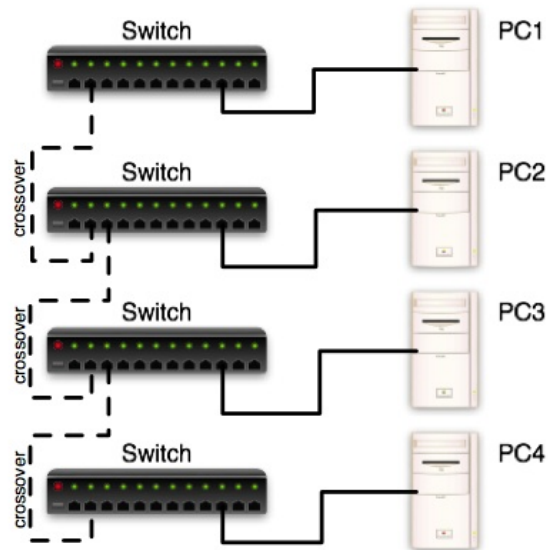
### Exercise 2 : The switch/bridge learning algorithm

Each bridge/switch has a MAC forwarding table that determines the outgoing port for a packet. When a packet arrives, the bridge looks up the destination MAC address of the packet in its MAC forwarding table and retrieves the outgoing port for this packet. If the destination MAC address is not found in the MAC forwarding table, the bridge floods the packet on all ports, with the exception of the port where the packet arrived.

Bridges and switches update their MAC forwarding table using what is called a *learning algorithm*, which works as follows: a bridge examines the source MAC address of each packet that arrives on a particular port and memorizes that the source address is reachable via that port. This is done by adding the source MAC address and the port to the MAC forwarding table. The next time the bridge receives a packet that has this MAC address as destination, the bridge finds the outgoing port in its forwarding table. Bridges that run this algorithm are referred to as *learning bridges*. All currently deployed Ethernet switches execute the learning algorithm.

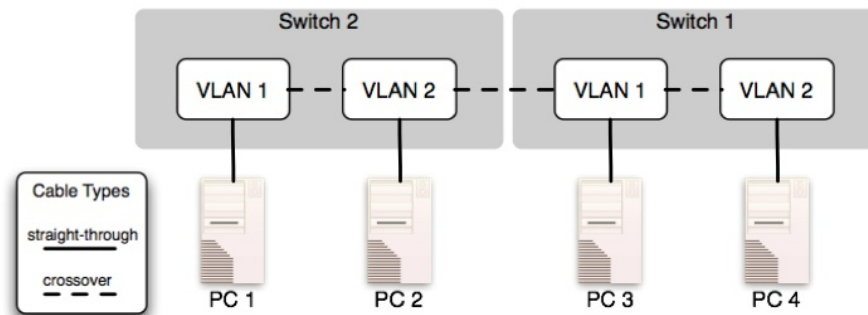
An entry in the MAC forwarding table is deleted if it is not used (looked up) for a certain amount of time. The maximum time that a MAC address can stay in the forwarding table without a lookup is determined by the *ageing* value, which is a configuration parameter.

Here you investigate the learning algorithm of bridges. The network configuration we want to set up looks as shown in Figure 4.7 .



#### 4.7 Desired network topology to investigate *learning* algorithm

By now you will have no doubt noticed that each pod in the lab only has **two** switches in it, so how can we realize the topology in Figure 4.7? You will use Virtual LANs (VLANs) to simulate multiple switches inside each switch. Each VLAN on a switch behaves much like an isolated switch. VLANs are configured entirely in software, making them very flexible. As each VLAN in a switch is effectively a separate network, packets are not forwarded between VLANs unless you explicitly connect a cable between them. Using VLANs, the topology you will realize looks as shown in the following figure. You can see how VLAN entities now replace the switch entities in Figure 4.7.



#### 4.8 VLAN topology

### **PART A: Setting up VLANs on the Catalyst 3550 (Switch 1) and Catalyst 2950 (Switch 2)**

All that is required to create a VLAN is to associate some number of ports on the switch with that VLAN. Each port on the switch can only be associated with one VLAN at a time.

Configuration of the switches is done over a serial connection between the serial port of your computer and the *console* port of the switch you are trying to configure. In later labs you will configure the routers in a similar fashion, by connecting to their respective *console* ports.

The switches have their *console* ports on the back, clearly labeled. The routers have them on the front in a more visible location. There is already a wire that runs from each computer's serial port to the back of the equipment racks, and makes its way to the front of each rack by way of the *patch panels*. You will see 4 ports on the patch panel at the bottom of each pod labeled "console". Their numbers match the number of the PC whose serial port it is

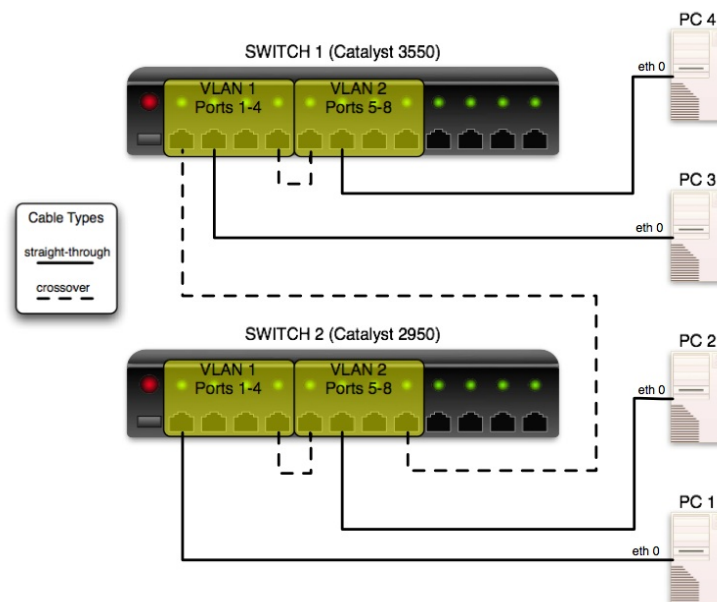
connected to. Using a straight-through Ethernet cable (the yellow ones in the lab) you can patch any computer's serial port to any of the console ports of the equipment in the rack. Once the serial connection is made, you will use the `kermit` terminal emulation program on your PC to communicate over it (see below).

### Starting a Terminal Session between a PC and the Cisco equipment with `kermit`

```
PC_% kermit                               Start the kermit terminal emulator
[/root] C-kermit> set line /dev/ttyS0      Point it to your serial port device
[/root] C-kermit> set carrier-watch off    Turn off carrier-watch (it's not used)
[/root] C-kermit> connect                  Make the connection
Press ENTER. You should see one of the following prompts:
    ▪ Switch1> ▪ Router2>
    ▪ Switch2> ▪ Router3>
    ▪ Router1> ▪ Router4>
```

**NOTE:** If you need to configure multiple switches or routers you don't need to open multiple `kermit` sessions to do it. In fact you can only have one instance of `kermit` running on each computer because the program takes ownership of the single serial port. To configure a different switch than the one you were initially connected to all you need to do is move the serial connection over to the different switch and press a key to get the new prompt – there is no need to exit `kermit` and restart it.

To help you see how to configure the switches to match the topology in Figure 4.8, the Figure has been redrawn to more closely represent your setup:



#### 4.9 VLAN topology applied to the two Switches in your pod

From the figure above, you can see that you need to set up two VLANs on each switch, with 4 ports in each VLAN. The procedure for assigning ports to a VLAN is outlined below.

### Assigning ports to a VLAN on the Catalyst 3550 and 2950 Switches

```
Switch_> enable
Password: <type in Cisco gear password>
Switch_# configure terminal
Switch_(config)# interface range Fa 0/1 - 4
```

Select a range of interfaces to which the following commands will be applied. The Fa term is the short form of FastEthernet and in this case the interfaces selected are FastEthernet 1 – 4 (The 0/ is part of each

```

interface's name and you will not need to change this 0 to anything else). Both switches have 24
FastEthernet ports/interfaces. Interface range Fa 0/5 – 8 will select the next 4 ports and so on.
Switch_(config-if-range)# switch access vlan 1
Move the interfaces (ports) selected by the previous command to vlan 1
Switch_(config-if-range)# end
Exits the “configure terminal” session. Alternatively you can press Ctrl-Z to exit the configure terminal
session.

```

**A.1:** Set up all four VLANs as shown in Figure 4.9, two VLANs per switch, 4 ports per VLAN. Don't use ports 23 or 24 on either of the switches.

**A.2:** Connect one of the PCs to each VLAN as shown in Figures 4.8 and 4.9. Before connecting the crossover cables between the VLANs, **check that the PCs cannot ping** each other (they shouldn't be able to at this stage as they are all on separate VLANs).

**A.3:** Now connect the crossover cables between the VLANs. Now all the PCs **should** be able to ping each other – verify this (wait for lights on the 2950 to turn green).

### PART B: Setting up monitoring ports

Now that your VLANs are set up, there is one more step before you can fully observe the learning algorithm. As you know (and will observe soon) once a switch finds out which port a PC can be reached on, it does not forward that PC's traffic to any other ports. A result of this is that if you are running Ethereal on PC2, even if it happens to be connected to the same switch (and even VLAN) as PC1 and PC3, you will not necessarily be able to observe the traffic between those PCs because the switch does not forward it to PC2. To circumvent this behavior, there is a feature we will use called port monitoring. Both switches in your pod support port monitoring, but there are a few differences so make sure to keep that in mind.

#### Setting up Port Monitoring on Switch1 (Catalyst 3550)

This switch supports two monitoring sessions – i.e. you can monitor two sets of ports.

```

Switch1> enable
Password: <type in Cisco gear password>
Switch1# configure terminal
Enter configuration mode.
Switch1(config)# no monitor session all
Begin by clearing all previously configured monitoring sessions.
Switch1(config)# monitor session 1 source vlan 1 rx
Monitor all received traffic on VLAN 1 with monitor session 1.
Switch1(config)# monitor session 1 destination interface Fa0/23
Send session 1 monitored traffic to port 23 on the switch.
Switch1(config)# monitor session 2 source vlan 2 rx
Monitor all received traffic on VLAN 2 with monitor session 2.
Switch1(config)# monitor session 2 destination interface Fa0/24
Send session 2 monitored traffic to port 24 on the switch.
Switch1(config)# end
Exit configuration mode

```

#### Setting up Port Monitoring on Switch2 (Catalyst 2950)

This switch supports only one monitoring session.

```

Switch2> enable
Password: <type in Cisco gear password>
Switch2# configure terminal
Enter configuration mode.
Switch2(config)# no monitor session all
Begin by clearing all previously configured monitoring sessions.
Switch2(config)# monitor session 1 source interface Fa0/5 – 8 rx
Monitor all received traffic on ports 5 through 8 (what you should have set up to be VLAN 2) with monitor
session 1.

```

```
Switch2(config)# monitor session 1 destination interface Fa0/24
    Send session 1 monitored traffic to port 24 on the switch.
Switch1(config)# end
    Exit configuration mode
```

Notice the difference in configuring monitoring ports on the two switches? The Catalyst 2950 does not allow “vlan” as a source for the monitoring session, so instead it must be specified as a range of ports. This means that you are responsible for making sure that the ports you monitor correspond to the VLAN you want to monitor.

**B.1:** Now that the monitoring ports are set up on the switches, plug in the PCs as follows:

- eth1 of PC2 into port 24 of Switch 2 (monitoring VLAN 2 on Switch 2)
- eth1 of PC3 into port 23 of Switch1 (monitoring VLAN 1 on Switch 1)
- eth1 of PC4 into port 24 of Switch1 (monitoring VLAN 2 on Switch 1)
- You will use `etherreal` on eth1 on all these PCs to observe the traffic between VLANs during the next part of this exercise (don't start `etherreal`, yet!).

By now you probably noticed that there are 4 VLANs and only 3 total monitoring sessions possible. Because of this we must resort to monitoring traffic between PC1 and VLAN 1 on Switch 2 using eth0 of PC1.

Before moving on, take a moment to go through this checklist:

- Are all the computers' IPs configured as shown in the table before Exercise 1 in this lab?
- There are a lot of cables on your rack by now – check that the computers are connected to the switches as illustrated in Figures 4.8 and 4.9. Don't forget to also check the 3 monitoring port connections to PC2, PC3 and PC4
- Make sure that you've written down the MAC addresses of all 4 PCs in the table at the end of Exercise 1, Part A. You will need to have these on hand to observe the learning process of the switches.

### **PART C: Exploring the learning algorithm of bridges**

In this exercise you study how bridges set up their MAC forwarding tables from the network traffic.

**C.1:** In the following steps you will have to track the changes of the MAC forwarding tables of both switches as you execute ping commands. To make this easier, make sure that each switch's console port is patched to a serial port on one of the computers and that you have `kermit` sessions open on those computers.

**C.2:** To make sure that the spanning-tree protocol is disabled for all the VLANs you are working with on the switches, follow the following steps on each switch:

#### **Disabling Spanning-tree protocol**

```
Switch_> enable
Password: <type in Cisco gear password>
Switch_# configure terminal
Switch_(config)# no spanning-tree vlan 1-2
Switch_(config)# end
```

**C.3:** You will use the following commands in the next few tasks:

#### **Commands for working with the MAC address-tables on the switches**

*Note: You must be in **privileged EXEC mode** (# prompt after having typed the `enable` command)*

```
show mac address-table dynamic
    Display the dynamically generated (learned) MAC forwarding table entries of the switch
show mac address-table dynamic vlan 1
    Display the dynamically generated (learned) MAC forwarding table entries of pertaining to vlan 1
clear mac address-table dynamic
    Clear all dynamically generated (learned) MAC forwarding table entries of the switch
```

On each switch,

1. Clear the MAC forwarding table entries
2. Display the MAC forwarding table entries (even though you cleared the table, a few will have reappeared already)
3. Since your topology has 3 crossover cables, there are 6 endpoints on these cables that will be associated with MAC addresses other than those of your PCs. These MAC addresses belong to the corresponding port of the switch on the other side of the cable. Using the `show interfaces` command on both switches, fill in the following table.

Switch Number	Port Number	Vlan Number	MAC address
1		1	
1		1	
1		2	
2		1	
2		2	
2		2	

**C.4:** Clear the ARP caches on all the PCs.

**C.5:** Start to capture traffic with *ethereal* on the eth0 interface of PC1 and the eth1 interfaces of PC2, PC3, and PC4.

**Save Data:**

**C.6:** Now, issue a set of ping commands. After each command, save the MAC forwarding table entries pertaining to VLAN 1 and VLAN 2 on each switch. Observe how far the ICMP Echo Request and Reply packets travel by looking at the ethereal windows on each PC.

```
PC1% ping -c 1 10.0.1.21
PC2% ping -c 1 10.0.1.11
PC2% ping -c 1 10.0.1.41
PC3% ping -c 1 10.0.1.21
```

- Keep in mind that you will need to use the captured data to illustrate the algorithm used by bridges to forward packets in your lab report.
- You will also need to note for each of the ping commands whether or not it results in any changes to the MAC forwarding tables and if so, what changes.

**Save Data:**

**C.7:** Stop the traffic capture on the PCs and save the *ethereal* output.

**PART D: Learning about new locations of hosts**

Learning switches and bridges adapt their MAC forwarding tables automatically when the location of a host changes. Due to the learning algorithm, the time it takes to adapt to a change depends on the network traffic and on the value of the *ageing* parameter. This is illustrated in the following part of this exercise.

**D.1:** Continue with the configuration of the previous exercise. Restart all the ethereal sessions on the PCs as they were set up previously. Applying the following display filter will get rid of the unnecessary *loopback protocol* packets:

```
eth.type != 0x9000
```

Start a continuous ping from PC1 to PC2:

```
PC1% ping 10.0.1.21
```

Now, connect PC2 to the same VLAN that PC4 is connected to. Do you expect there to be a delay before the ping replies start arriving at PC1 once PC2 moves to a new location? Observe what happens.

**Save Data:**

**D.2:** Now, without stopping the ping command from PC1 to PC2 (restart it if you stopped it already), move PC2 back to where it was connected originally. Is there a delay now? Think about why this happens (If you need to, observe the MAC forwarding tables at the switches while repeating question D.1 and D.2). You can stop the ping from PC1 to PC2 when you've made your observation. Save all the ethereal captures.

**Save Data:**

**D.3:** Restart all the ethereal captures to clear out the results from the previous question. Start a continuous ping from PC1 to PC4:

```
PC1% ping 10.0.1.41
```

Connect PC4 to the same VLAN that PC2 is connected to (should be VLAN 2 on Switch 2). Immediately ping from PC4 to PC1:

```
PC4% ping -c 3 10.0.1.11
```

This scenario is exactly the same as that in question D.2 except for this ping. What difference did the ping make in the delay associated with the replies to PC1's ping? Again save all your ethereal captures.

**D.4:** Restore the configuration of both switches with the reload command. Do **not** save the modified configuration.

**Reset switches with the "reload" command. Do not save the modified config.**

**REMEMBER TO COPY SAVED FILES FROM ALL PCS!**

**LAB REPORT****Exercise 1 Questions:**

**1.1:** (In Part C, Step 4) Do the source and destination MAC/IP addresses change when a packet traverses a bridge? Provide an explanation and include an example from the captured data. Suppose that PC2 was configured as an IP router, which differences would you observe in the Ethernet and IP headers?

**1.2:** (In Part C, Step 4) Does the bridge manipulate any of the fields in the MAC and IP headers (besides the addresses)?

**1.3:** Include the output of the `traceroute` command from Part C, Step 5. Provide an explanation why PC2 does not appear in the output of the `traceroute` command.

**1.4:** If PC2 were configured as an IP router, how would the output of `traceroute` in Part C, Step 5 differ?

**1.5:** Include the PC MAC addresses from Part A, Step 3

**Exercise 2 Questions:**

**2.1:** Use the data collected from step 6 of Exercise 2, Part C (both MAC forwarding tables and ethereal captures) to illustrate the algorithm used by switches to forward packets.

**2.2:** For each of the pings in step 6 of Part C, identify and explain changes made to the MAC forwarding tables. Include the MAC forwarding tables to document your answer.

**2.3:** What were the differences in your observations regarding the delay in ICMP Echo Replies to PC1 from PC2 in Step D.1 and D.2. Explain the reason for these differences? Hint: The switches can sense the presence or absence of a link on any one of their ports.

**2.4:** What difference did the `ping` make in the delay associated with the replies to PC1's `ping` in Step D.3? Explain.

**2.5:** Include the MAC Forwarding tables from Part C, Step 3.