CS 638 Lab 3: Local Area Networks

Joe Chabarek, Mike Blodgett and Paul Barford
University of Wisconsin –Madison
jpchaba,mblodget,pb@cs.wisc.edu

Local Area Networks (LANs) are one of the most important computer communications technologies. There are millions of local area networks all over the world that connect together end hosts deployed in a relatively confined area and enable them to communicate. LANs are realized at the data link layer (layer 2) of the protocol stack. While a variety of protocols are available at this layer, Ethernet is the most widely used. The devices that enable LANs - hubs, switches and bridges - come in all shapes and sizes, from small home devices with 4 ports to huge systems with hundreds of ports. Many of these devices enable LANs to connect to other LANs but more importantly, they all enable local computers to uplink to the Internet and thereby communicate with computers all over the world. This third lab is designed to introduce you to LAN concepts and to expose you to and give you experience with a widely LAN device. In other words, this is the first lab in which you will not be working exclusively with end hosts!

1 Overview and Objectives

In practice, many aspects of modern local area networks are quite simple since most devices are designed to be plug and play. A typical example are the small devices that are widely used in private homes. However, things get tricky when there is a need to interconnect multiple LANs (commonly the case in large buildings) or when there are special needs for management or security that require special features on more sophisticated LAN devices to be used.

Lab 3 is divided into two parts. The first part will introduce you to *bridging*, which can be thought of as a basic mechanism for enabling two hosts to communicate at layer 2. Bridging is the term that is also used for connecting multiple LANs together and therefore is an important concept. In part 1, you will build a simple bridge on a Linux PC with multiple network interface cards. The second part of the lab will give you experience with some of the configuration and management features of a commonly used Ethernet switch - a Cisco 2950 or 3500XL. This relatively inexpensive switch enables small LANs (e.g., 24 hosts) to be built using Cat 5 twisted pair cabling, and has some very useful management features that enable network administrators to create customized configurations specific to their own environment and policy needs.

Upon completion of the lab exercised below, students will be able to:

1. *Understand the basic concepts of the Ethernet LAN protocol.*
2. *Configure a LAN bridge on a Linux PC*
3. *Understanding how forwarding loops occur and what symptoms appear in the network when they exist*
4. *Understand the Spanning Tree Protocol and why it is important in LANs*
5. *Access, configure and evaluate a standard Ethernet LAN switch*

2 Ethernet Basics

In any LAN, multiple hosts, each with a specific type of Network Interface Card, are connected together via a medium (e.g., copper, fiber or the air) that is organized in some topology (e.g., a bus, star, ring - in wireless host organization is random). Ethernet is a widely-used example of a Shared Access Network and the focus of this lab.

Ethernet was developed by Robert Metcalfe and others at Xerox PARC in the mid-1970's. It was based on shared access concepts developed in the 1960's for packet-radio networks called Aloha. Ethernet has been successfully mapped on top of many different physical technologies (i.e., increasingly higher bandwidth transceivers) over the years and is now the dominant LAN protocol. Examples of Ethernet standards include:

1. IEEE 802.3 (CSMA/CD – Ethernet) standard, originally 2Mbps
2. IEEE 802.3u standard for 100Mbps Ethernet
3. IEEE 802.3z standard for 1000Mbps Ethernet

Details of these standards can be found at http://grouper.ieee.org/groups/802/3/ .

Prior to the late 1990's, before the wide use of Ethernet switches, Ethernets were shared, bus-based networks. The consequence of bus-based LANs is that a message sent by one host can become corrupted if another host begins sending at approximately the same time. To mitigate the effects of multiple senders, many Media Access Control (MAC) protocols have been developed in the hope of making LAN communication on shared mediums efficient. In addition to conflict resolution, the MAC protocols are designed to promote fairness and high performance. Ethernet is one example of contention-based multiple access where nodes contend equally for bandwidth, and recover with high probability when a collision occurs.

Like all layer 2 protocols, each packet transmitted by Ethernet is encapsulated into a frame with a well defined format and well defined length called the Maximum Transmit Unit (MTU). MTU is set to the properties of the physical medium and the standards which define connectivity. The Ethernet frame format is shown in Figure 1. An Ethernet frame begins with a 7 byte preamble used to synchronize the receiver before actual data is sent. Destination and source addresses followed flags and by the payload (supplied by upper layer protocols) appear next. The frame concludes with a CRC checksum used to detect bit-level errors. Each Ethernet adapter has a unique 48-bit "hardware" address (e.g., 00:0a:95:f2:29:dd). In order to promote unique addressing (although duplicates do occur), each manufacturer is given their own address range.

2.1 Media Access Control Algorithm

As defined in the IEEE 802.3 standard, Ethernet uses Carrier Sense Multiple Access/Collision Detect (CSMA/CD) as its MAC policy. This policy specifies behavior for each participant in the Ethernet with respect to how the medium is accessed, how collisions are detected and what to do when a collision takes place. The CSMA/CD algorithm specifies that a network device listens to the line before and during sending. If the line is idle the device will send the packet immediately. If the line is busy the device will wait until the bus is idle and transmit the packet immediately. While sending, if a collision is detected, stop sending the message and transmit a special *jam signal* so that all other senders can know that a collision has taken place. If a collision is detected, the senders will delay resending for a period of time determined using *binary exponential back-off* which enables both senders to retransmit successfully with high probability. The nth collision will cause a delay to be chosen from $Kx51.2us$ for $K = 0..2^n - 1$. If after retrying 16 times, a sender is still unsuccessful, a transmit error is reported to the upper layer and the packet is discarded.

2.2 Realities of Ethernet

Despite the utility of CSMA/CD, it is well known that shared Ethernets work best (i.e., operate with the best efficiency) under light loads. Utilization levels above 30% are considered heavy. Loads are most easily limited by restricting the number of hosts that are connected to a LAN. Most Ethernet networks are limited to at most 200 hosts even though the specification allows up to 1024. In addition, most Ethernet LANs have a relatively restricted deployment thus minimizing the round trip time (RTT) for a frame (e.g., 5-10 microsecond even though the specification allows for much longer). Ethernet has become popular because it is inexpensive, fast, and easy to administer.

3 Part 1: LAN Bridging

Due to the restrictions on size and scope of LANs such as Ethernets, *bridging* is necessary in order to couple multiple local area networks and have them appear as a single entity. Multiple LANs can be bridged as long as all devices have the same MTU, have similar address formatting, support promiscuous operation (i.e., a device can be given access to all traffic), and support address spoofing (i.e., addresses can be changed). This flexibility enables a large range of devices to be bridged in addition to standard Ethernet, such as PPP, Virtual Private Networks (VPNs), or Virtual Local Area Networks (VLANs).

3.1 Implementation

Like other LAN activities, bridging is done at layer two of the network stack. Devices that are capable of bridging include routers, switches, and specially configured general purpose computers. In this lab, we are concerned with bridging Ethernet traffic transparently i.e., so that communicating hosts are unaware that a bridging device exists between them. Said another way, the bridge will not be visible via `traceroute` since no routing is done and no

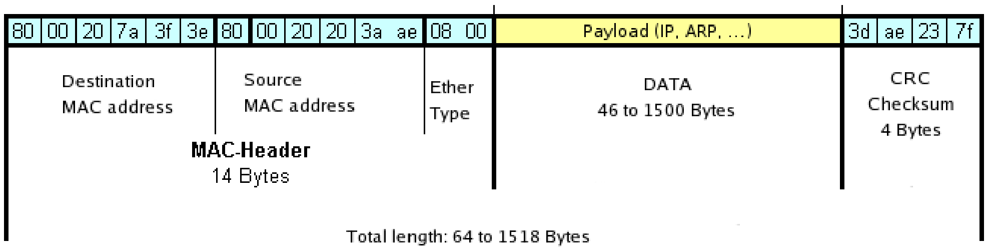

Figure 1: Frame format for Ethernet

special configuration on the sending hosts is required. A correctly configured bridge device will relay traffic transparently between multiple network interfaces (at least 2). Bridges use tables that associate MAC addresses of connected hosts with its own particular interfaces. If an incoming packet has an address that is unknown to the bridge, the packet is forwarded along *all* interfaces.

Bridges use a simple learning algorithm to determine where to forward incoming packets. In this algorithm, the MAC addresses of incoming packets are placed in the forwarding table along with the interface on which the frame was received. This enables future packets destined for these sources to be transmitted directly. There is a timeout associated with each entry in the forwarding tables that allows a bridge to handle devices that are moved. It is important to note that the terms *LAN switching* and *bridging* are interchangeable.

3.1.1 Linux PC's as LAN Bridges

It is possible to use a Linux PC with multiple network interface cards to act as a bridge. There is a tool called **brctl** which interacts with the bridge kernel module to allow any Linux machine to become a network bridge. The OSID – FC6-STD has the bridge kernel module included, but you will need to add an RPM to have the brctl tool installed. To see if the module is running use the **lsmod** command to list the modules which are currently loaded. You will find it is not loaded at startup, but will be automatically loaded after you create a bridge with the brctl tool.

Listed below is a typical sequence of commands that can be issued while using this tool:

1. *ifconfig eth1 0.0.0.0* Sets the IP of eth1 to be 0.0.0.0
2. *ifconfig eth2 0.0.0.0* Sets the IP of eth2 to be 0.0.0.0
3. *brctl addbr bridge0* This creates a logical bridge which will contain the interfaces for your bridge
4. *brctl addif bridge0 eth1* Instructs eth1 to be part of bridge bridge0
5. *brctl addif bridge0 eth2* Instructs eth2 to be part of bridge bridge0
6. *brctl show* Gives summary Information about all bridges on this PC
7. *brctl setageing bridge0 10* Length of time (sec) the MAC forwarding table will keep an entry
8. *brctl stp bridge0 on* Enables the Spanning Tree Protocol
9. *brctl setbridgeprio bridge0 10* Sets bridge0 priority to 10 in the Spanning Tree
10. *ifconfig bridge0 up* The bridge is now active
11. *brctl showmacs* Gives MAC forwarding table

3.2 Forwarding Loops

When there are multiple bridges in a network there is a possibility that packets/frames could cycle between bridges. Remember that a packet encountered on a bridge that has been sent to an unknown host is flooded across all interfaces. If there are a set of bridges which form a cycle that do not have the MAC address of the destination host in the forwarding table, a forwarding loop occurs. This has serious consequences. First, the packets cycle around the LAN infinitely. Second, each time one of the cycling packets encounters one of the bridges, and the bridge still does not know the location of the destination host, it will flood the packet onto all of its interfaces. This is known as a broadcast storm.

3.3 Spanning Tree Protocol

The Spanning Tree Network Protocol (STP) provides a loop free topology for any bridged LAN. STP is defined in the IEEE Standard 802.1D. Spanning tree is based on an algorithm invented by Radia Perlman while working for Digital Equipment Corporation. As the name suggests, STP finds a tree topology within the mesh formed by a series of bridged LANs and disables the links that are not part of that tree. The starting point for a spanning tree is a *root node*. In LANs, a designated (usually high capacity) switch is configured as the root, although a root election process based on MAC address and priority number will take place by default.

Spanning tree protocol information is carried in special data frames called Bridge Protocol Data Units (BPDUs). BPDUs are exchanged regularly (every 2 seconds by default) and enable Ethernet switches to keep track of network changes and activate or disable ports as required. When a device is first attached to a switch port, it will not immediately start to forward data. It will instead go through a number of states while it processes BPDUs and determines the topology of the network. When a host is attached the port will always go into forwarding mode, albeit after a delay of about 50 seconds while it goes through the listening and learning states (see below). However, if instead another switch is connected, the port may remain in blocking mode if it is determined that it would cause a loop in the network. Topology Change Notification (TCN) BPDU's are used to inform other switches of port changes. TCN's are injected into the network by a non-root switch and propagated to the root. Upon receiving a TCN frame, the root switch will set a topology change flag in it's normal BPDU's. This flag is propagated to all other switches to instruct them to rapidly age out their forwarding table entries.

In order to configure a spanning tree with the brctl tool, a root bridge is designated. Each bridge in the LAN is given a priority, the bridge with the lowest priority is the root bridge and is configured as follows:

1. *brctl stp bridge0 on* Turns on the STP
2. *brctl setbridgeprio bridge0 1* Sets the bridge priority
3. *brctl stp bridge0 off* Turns off the STP

Note that the STP is enabled by default on most hardware switches, while disabled by default for a Linux host bridge. If you want to experiment with forwarding loops, you have to turn off the STP.

4 Part 1: Tasks and Questions

Use the Schooner "GUI-Editor" to construct the topology shown on the pre-lab page. Use the FC6-STD OSID for each node. For each node that you want to act as a bridge, you will need add the bridge-utils RPM using the same procedures you did in Lab 2. The testbed software doesn't currently have a way to directly describe what we would like to do, and will try an assign IP addresses to your nodes for our test topology such that it is two separate layer 3 networks. For this experiment you will again have to assign addresses in the GUI or manually change the addresses on the nodes.

```
node0 - Node on the left - 192.168.0.10
node1 - Node on the left - 192.168.0.11
node2 - Center Node - Interface 1 - None
                    - Interface 2 - None
node3 - Right node -  192.168.0.13
```

Once the topology is complete, do the following:

1. Configure the PC between the two LANs to be a bridge using the brctl command. After the bridge is established set link priorities on all the interfaces and be sure that you can `ping` between hosts. Be sure to note how `ifconfig` views the software bridge.
2. Next, we are interested in exploring what a forwarding loop will do to a network. Load the topology that you developed in the pre-lab, be sure to disable STP. Observe the traffic generation with ethereal. Use what you have learned in previous labs to characterize how the network is affected.
3. USING THE SAME EXPERIMENTAL SETUP, take down the bridges to eliminate the looping traffic. Set priorities for all of your bridges and enable STP. Bring your bridges back up. In your notebooks, be sure to record the priorities for the bridges in your topology. Now, we are interested in how the spanning tree is structured. Develop a technique for discovering the structure of the spanning tree. Note that MAC address tables can be cleared by setting the timeout of an address to 0.

5 Part 2: Ethernet LAN Switching

While the exercise above is useful for understanding the conceptual aspects of LANs, LANs are almost always implemented using a specialized piece of hardware that is the focal point for interconnecting end hosts. The two standard instances of these devices are *hubs* and *switches* - each which implement the Ethernet protocol. Hubs are small devices commonly used in home environments. They enable a small number of hosts to be interconnected (e.g., 4 or

8) with Cat 5 cabling and often provide wireless connectivity as well. Hubs differ from switches in that they provide a *shared medium.* In contrast, there are a wide variety of Ethernet switch devices, but all enable one-to-one communication between hosts (i.e., no shared medium is needed). The focus of part 2 of this lab is on configuration and management of a common Ethernet switch device.

Switches come in a number of different sizes and capabilities. Most manufacturers have a number of model lines that correspond to different market segments, and as capabilities go up so does the price. The first big split is between an unmanaged and managed switch. Your typical switch used in residential application is an unmanaged switch. A typical NAT Gateway device also common in residential applications blurs the lines a bit, but if you consider the four or eight ports on the back, it is an unmanaged switch. An unmanaged switch will work out of the box, you can connect hosts and it will forward frames and build it's forwarding tables just like you have learned. It will probably have lights on the front that may tell you if the link state is up, and possibly tell you what speed, 10Mbit or 100Mbit, the port is operating at. As of October 2007, the price for a well known manufacturer, simple 24 port desktop switch, goes for a per port price of $5.

Managed switches provide an interface into the switch. You will generally see more advanced capabilities such as VLAN's, trunking, and QOS. You can retrieve more statistics about how the switch is operating with a managed switch. These capabilities are very important in enterprise networks which strive for high reliability and availability in their networks. From the same manufacturer quoted above, the same configuration switch in a managed version has a per port cost of $15. Even among managed switches their are a number of different categories of switches, but you can generally categorize them into three different classes.

1. Regular Switch - a single device, generally anywhere from 4 - 48 ports, sometimes has uplink or higher bandwidth port, example 24 port 10/100 and 2 10/100/1000.

2. Stackable Switch - Similar to our regular switch, but has a specialized port such that multiple devices can be stacked together, either by directly linking the backplane, or by use of a special bus. Also the switches generally then operate as a single logical device.

3. Chassis Switch - Highest class of switches consisting of a chassis, with modular management cards, and interfaces. High availability features are generally available, multiple power supplies, hot pluggable support of interfaces, redundant management interfaces.

Just to give you another price figure, a high quality gigabit port on a large chassis switch, $185, and that is just for the port, if you consider the cost of the chassis, management blades, memory, power supplies, software support, it would be quite a bit higher.

For the second part of Lab2 you will be using Schooner to create an experiment that includes a Cisco 3500XL switch. Once your experiment is configured you will be able to login to the switch via the CLI or command line interface and do a number of configurations.

5.1 Setup

For our experiment, you will create the below topology. Notice that our 3500XL switch is a Node in our experiment, do NOT create it as a switch.

For node0 set a hardware type of c3500, and set an OSID of FC6-STD for the rest of the nodes. You will once again need to force the IP for a number of your nodes. The IP's for node1,node2,node3, and node4 are the ones you will be using in the experiment. You will not use (”ping”) the IP's assigned to node0, they are acting as a placeholder in the experiment, to workaround a bug in the testbed software.

```
link0 node1 192.168.0.1
link1 node2 192.168.0.2
link2 node3 192.168.0.3
link3 node4 192.168.0.4

link0 node0 192.168.0.5
link1 node0 192.168.0.6
link2 node0 192.168.0.7
link3 node0 192.168.0.8
```

Go ahead and swap in your experiment.

5.2 Switch Login

From the mail you received or from the experiment web page you can find which switch has been allocated to your experiment, it will look something like 's-c3500-13-a'. First grab the telnet rc file, it disable and autologin feature which causes some warning messages you don't need.

```
cp /proj/cs638/telnet-rc ~/.telnetrc
```

Then use telnet to login to the switch

```
> telnet s-c3500-13-a
Trying 10.0.1.73...
Connected to s-c3500-13-a.schooner.wail.wisc.edu.
Escape character is '^]'.

Username: wail
Password:

s-c3500-13-a#
```

Use your username and password from schooner to login to the switch. When you login you are automatically entered into a privileged mode. The cisco support multiple levels of access which can be used for security, but level 15 allows you to do everything.

```
s-c3500-13-a#show privilege
Current privilege level is 15
```

The show command is probably the most important, try typing śhowt́hen spacebar and then ?, you will get a large list of different things you can look at, try typing 'show int' and then hit tab. The shell will autocomplete from the available items, and a hint, many times you can abbreviate, example 'show int' works just as well as 'show interface'. Now do a 'show cl', hit tab nothing, it can't autocomplete, try hitting the ? instead of the tab, it will list you the available items with a cl prefix.

There are a number of different shortcuts and additional features that are just part of the shell. Try browsing around the show commands a bit see what you can find.

5.3 Verifying Config

First we need to verify the configuration of our switch do a 'show run' command, which will display the configuration of the switch, It's quite a bit of information, luckily you can just ask to show the config for a specific interface.

```
s-c3500-13-a#show run int FastEthernet 0/1
Building configuration...

Current configuration:
!
interface FastEthernet0/1
end

s-c3500-13-a#
```

Nothing is shown, but don't worry that just means it is in a default configuration, for a Cisco box the configuration file, is really just a set of commands that are run at boot time, that override the default configuration.

Now check for the rest of our interface 0/2, 0/3, 0/4. If there is any bits leftover from one of your class mates you will need to clear it out. The default command allows you to set a specific line,interface, or other configuration constructs back to their defaults.

```
s-c3500-13-a#config t
Enter configuration commands, one per line.  End with CNTL/Z.
s-c3500-13-a(config)#default interface FastEthernet 0/1
Building configuration...

Interface FastEthernet0/1 set to default configuration
s-c3500-13-a(config)#end
s-c3500-13-a#
```

Let's take a closer look at this command, remember when you logged in you were given a shell with privilege 15. But you can't actually make any changes from that mode, you first need to enter the configuration mode. The 'config' command will put you into configuration mode. You can load configurations from a variety of sources, but you want to actually type something in, so use 'config terminal' or abbreviate as 'config t'. After your commands are done you can exit back to your privileged shell with the 'end' command.

Now open another shell and login to your node1, which should be assigned 192.168.0.1, try pinging your other three nodes.

```
> ping -c 1 192.168.0.3
PING 192.168.0.3 (192.168.0.3): 56 data bytes
64 bytes from 192.168.0.3: icmp_seq=0 ttl=64 time=0.456 ms

--- 192.168.0.3 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.456/0.456/0.456/0.000 ms
> ping -c 1 192.168.0.4
PING 192.168.0.4 (192.168.0.4): 56 data bytes
64 bytes from 192.168.0.4: icmp_seq=0 ttl=64 time=0.427 ms

--- 192.168.0.4 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.427/0.427/0.427/0.000 ms
```

Your nodes now have connectivity via your switch. if your pings do succeed something is probably wrong, review the steps you've taken so far, and if nothing helps contact the testbed administrators.

5.4 Port Configurations

Common changes to a switch interface might be to move it to another VLAN, force the duplex/speed, or to affect the spanning-tree somehow, lets walk through some of these changes.

First to configure an interface you must enter the configuration mode, just like you did before. Cisco and most other manufacturers use a multi-tiered config structure. When you enter configuration mode, settings applied here will generally affect the switch as a whole. To configure an interface you must enter into that part of the config structure or context.

```
s-c3500-13-a#config t
Enter configuration commands, one per line.  End with CNTL/Z.
s-c3500-13-a(config)#int FastEthernet0/1
s-c3500-13-a(config-if)#end
```

The 'end' command will drop you out of the configuration mode completely, while the éxitẃill drop you back one level of the config structure, for example if we used the exit command in the above example, it would have dropped us into the global config level. Some configs for the QOS parameters can get three or four levels deep, but usually your just going into an interface config or a router process, but be aware of what context your in.

6 Part 2: Tasks and Questions

1. Enter the interface configuration for one of your ports, use the speed command to force the port to 100Mbs, then use the duplex command to force the port to full duplex. Can you still verify connectivity between all your nodes.

2. Use the documentation below to assign two of your ports to a different vlan. Verify you can ping in between the two nodes you moved, and can still ping between the two you didn't move. You can figure out how nodes are attached from the mail you received after swapping your experiment in.

```
Physical Lan/Link Mapping:
ID             Member         IP             MAC                 NodeID
-------------- -------------- -------------- -------------------- ---------
link0          node0:0        192.168.0.5    00:00:00:00:00:00   s-c3500-13-a
                                             1/1 <-> 1/26        r-c6500-12-b
link0          node1:0        192.168.0.1    00:02:b3:bf:27:05   pc-i2-41-n
                                             0/1 <-> 2/46        r-c6500-12-b
.....
```

   The way the testbed works, is that all devices are interconnected via a large switch. It is easiest not to think of it as a switch, but more of a virtual automated patch panel connecting wires for you. The above table says that s-c3500-13-a, port 1/1, which is actually FastEthernet0/1 on your switch, is connected to 1/26 on r-c6500-12-b, also connected is pc-i2-41-n. Look over the table and make sure you understand it.

```
http://www.cisco.com/en/US/docs/switches/lan/catalyst3550/software/
release/12.2_25_see/configuration/guide/swvlan.html#wp1036007
```

3. By default when a port on the switch comes up it will go through a number of stages in regards to spanning-tree. To the host this ends up looking like a large delay from being able to send and receive packets after the link state comes up. This delay can cause problems usually when nodes try to DHCP for an IP address. To combat this problem you can configure portfast on a port such that when the link state comes up it will immediately go into a forwarding state.

   Using the Physical Lan/Link mapping table find where one of your nodes is connected to the switch. Login to the node and start a continuous ping to another one of the nodes. Now ´bounce´the port of the machine generating the pings by using the shutdown, and then a no shutdown command on the port. then QUICKLY exit the configuration context and look at the spanning tree for the port, look at the spanning tree every few seconds and look for changes.

```
s-c3500-13-a#config t
Enter configuration commands, one per line.  End with CNTL/Z.
s-c3500-13-a(config)#int FastEthernet 0/2
s-c3500-13-a(config-if)#shutdown
s-c3500-13-a(config-if)#no shutdown
s-c3500-13-a(config-if)#end
s-c3500-13-a#show spanning-tree interface FastEthernet 0/2
```

   Now go into the same interface configuration and add the statement ´spanning-tree portfast´. Repeat the above procedure, how has the behavior changed? Can you still verify connectivity between your nodes? What problems could be introduced by making this change?

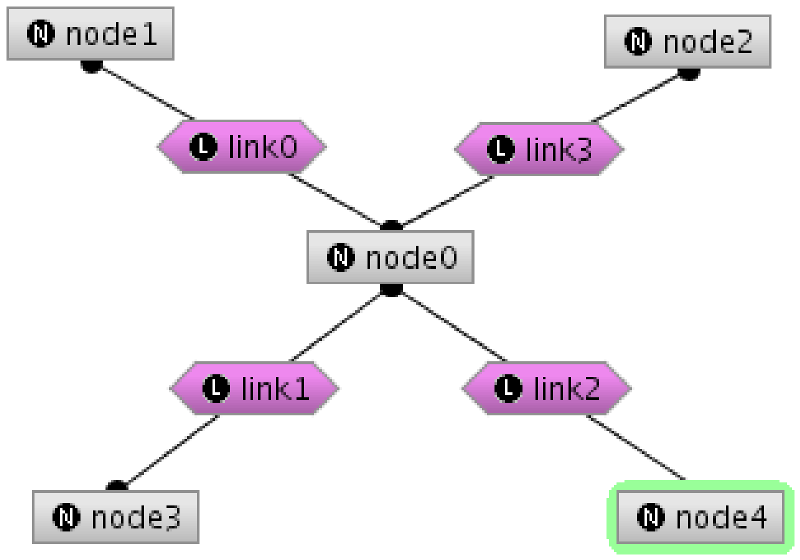

Figure 2: Lab 3 Part 2 Topology