Lab 05: Link Layer Basics and Bridging

OSL, Thu Feb 19, 2015

Objective:

1. Get familiar with new networking commands applicable at link layer : ifconfig, ethtool, netstat and arp
2. Get familiarized with VNX, a virtualization tool used to simulate linux based networking scenarios.
3. Use VNX to understand Ethernet bridging.

General instructions:

1. This lab is to be done in **groups of two students**
2. Read the exercise fully before starting to experiment.
3. As you proceed with the lab instructions below, for each exercise, note down the answers to the exercise along with any interesting observations in a file "rollno1_rollno2_lab05.txt".

Reference:

1. Man pages of the commands
2. Video/slides of 'Ethernet Switching'
3. Bridging commands reference:
   http://www.linuxfoundation.org/collaborate/workgroups/networking/bridge
   (Look at the bridge commands under Module loading. Note not all commands listed in the above are relevant. Relevant portions are sections 5.2 till 5.6.1 and 5.8)

Lab Instructions:

Exercise 1: Play Time

Play around with ifconfig, ethtool, netstat and arp commands.
At the end of this exercise, you should have basic understanding of what these commands do.

Exercise 2: Simple stuff [11 Marks]

Now answer the following based on what you learnt in exercise 1. All questions below are related to the Ethernet interface of your machine. **Also, specify the commands you used to determine the answers.**

1. What is the MAC, IPv4, IPv6 address of the Ethernet interface of your machine?
2. What is the MTU size of the interface? What does MTU mean?
3. What Ethernet mode (half/full), speed and over what cable type is your Ethernet card operating?
4. Since the interface was up, how many packets did it transmit and how many did it receive? Were any packets dropped?
5. For part 4, do you see a difference in statistics collected via ifconfig and via netstat? If so, why is this difference there? Answer is simpler than you think.
6. How many packets suffered collisions? Can you justify the number?
7. Since the interface was up, how many bytes did it transmit and how many did it receive?
8. Who is the manufacturer of the Ethernet card? (Google for this based on first 24 bits of the

MAC address)

9. Type 'ifconfig -a'. What is this interface lo? Why doesn't it have a Mac address? Before you go browsing for what it is, design an experiment via 'ping and tcpdump' to figure out what it is. Mention the experiment (commands used), observations and conclusions.
10. What is the MAC address of the default gateway (router)? Find out via network commands not tcpdump.

Exercise 3: Play time with VNX

Funda: Setting up bridging (or for that matter doing any network configuration) on normal linux hosts requires *root* permission. To circumvent this problem, we will be using VNX (Virtual Networks over linuX). This is what the tool developers write abovt VNX.

"**VNX** is a general purpose open-source virtualization tool designed to help building virtual network testbeds automatically. It allows the definition and automatic deployment of network scenarios made of virtual machines of different types (Linux, Windows, FreeBSD, Olive or Dynamips routers, etc) interconnected following a user-defined topology, possibly connected to external networks."

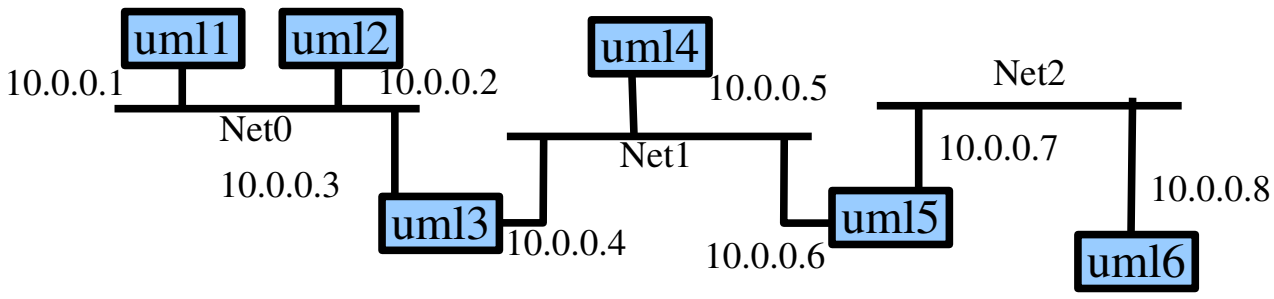
VNX is made of two main parts:

- an XML language that allows describing the virtual network scenario (VNX specification language)
- the VNX program, that parses the scenario description and builds and manages the virtual scenario over a Linux machine

So you will be working on real Linux, it does not matter to us that the real Linux is running as a virtual machine. Another advantage of using VNX is that you can create any number of virtual machines on a single physical machine (subject to memory availability). So you can create an entire *network of Linux machines* in a single physical machine!

VNX takes a configuration file as input. The configuration file specifies the topology and connections in the virtual network. VNX then creates as many virtual instances as are specified in the configuration, and connects them up according to the specified topology. That's it!

Playtime: I have provided a VNX configuration file corresponding to the below topology. Browse it. Bring up the virtual network using the command: " vnx -f lab05_bridging.xml -v --create". ***In case you want to release the simulator (do it for a good reason since setting up the machines takes time), you can do so using the command "*** *vnx -f lab05_bridging.xml -v --destroy* ***"***



When you run the above command (it may take some time to execute), you should see six xterm windows popping up; these are the six virtual machines running Linux. You can **root** login to each by using the password "xxxx". What you do on these machines will have no effect whatsoever on the physical machine's Linux or the filesystem! So the system administrators are happy!

Checkout the xterms corresponding to the umls. They are much like your regular terminals. You can use ifconfig, route, arp, vim, cat, bash scripts etc much like on our own laptops/desktops.

Play around with the terminals and check using networking commands that indeed the above topology has been created.

Exercise 4: Bridging **[19 Marks]**

Ms. Bree Edge coincidently is working on interconnecting hosts across three networks (much like the example above) and needs help. Can you use VNX to figure out how to bridge the different hosts at link layer by converting some of the hosts into bridges? While the above topology has no loops, it is always a good idea to turn on spanning tree protocol. Once you have successully bridged the hosts, you can help her setup her network.

Guidance:

1. Before experimenting, read the guidance fully as well as questions asked as part of the exercise. This will help save time when experimenting.
2. The main command that you will use is brctl. Go through the reference to understand its usage.
3. Before you even bridge, check ground reality i.e how the hosts are interconnected (reachability between the machines via ping).
4. Then choose the right hosts that can act as bridges and enable bridging on them one after another. Check reachability everytime you add a bridge.
5. Check out the learning aspect of the bridge as well (use ping to generate traffic). Experiment with the ageing timer. Setting it to zero clears out entries. By appropriately setting it you should see learning entries disappearing over time.
6. Once all hosts are interconnected, enable spanning tree protocol on each bridge step by step. Check spanning tree status at all bridges after each step.

Questions:

Answer the following questions

1. Before enabling bridging, note down all the IP addresses you could reach via ping from each of the machines: uml1, uml2, uml3, uml4, uml5, uml6. Does this make sense? Do you notice any anamoly, if so state it and the reason behind this? For debugging, use tcpdump on umls to figure out what is happening.
2. After you enabled bridging to interconnet just Net0 and Net1 (**not Net2**), what **interfaces** can uml1 reach now? And why?
3. After you enabled bridging to interconnect Net1 and Net2 (Net0 and Net1 are already connected), what **hosts** can uml1 reach?
4. Clear out all entries in the learning table at one of the bridges. Ping Uml6 from Uml1. After ping, note down in the report the learning table entries you saw. What command did you use for this and do the entries match theory?

5. After you turned on spanning tree at all bridges, specify which host acts as the root bridge and what is its ID? Who is the designated bridge for the lan segments Net0, Net1 and Net2 respectively.
6. List the entire commands you executed (in the correct order) to interconnect the hosts. The list you would pass to Ms. Bree Edge.

Extra Credit:

1. What is the default ageing timer used by the bridges before you fiddled with it? And what command/steps did you use to figure it out. [3 Marks]
2. Bring down the root bridge and document the changes you observed at the remaining bridges. [2 Marks]

Submission instructions

Upload the report file on moodle.