15-441 Networks - Spring 2006 - Lab

GURUs: Joshua Hailpern and David Murray

Due: Friday, May 3

Welcome to the Networks Lab. What is a lab? Well it is more hands on then a homework, but unlike a project, there is very little coding. The purpose is to show you how to use "real world" tools to tackle "real world" networking issues. You should not expect to spend more than 2-3 hours on this assignment.

Please read through this lab before starting it.

Lab Overview

Most of this lab can be done on "your own schedule." Do some today, some tomorrow, some next week. However, for **QUESTION 4 and 5,** you must sign up for a 1 hour block of time to use the netclass machines 9 and 10. When you log in, please run either the `w` or `who` command to make sure you are alone on the machines; if you are not alone, please contact us. The sign up sheet will be on the door of the course secretary's office.

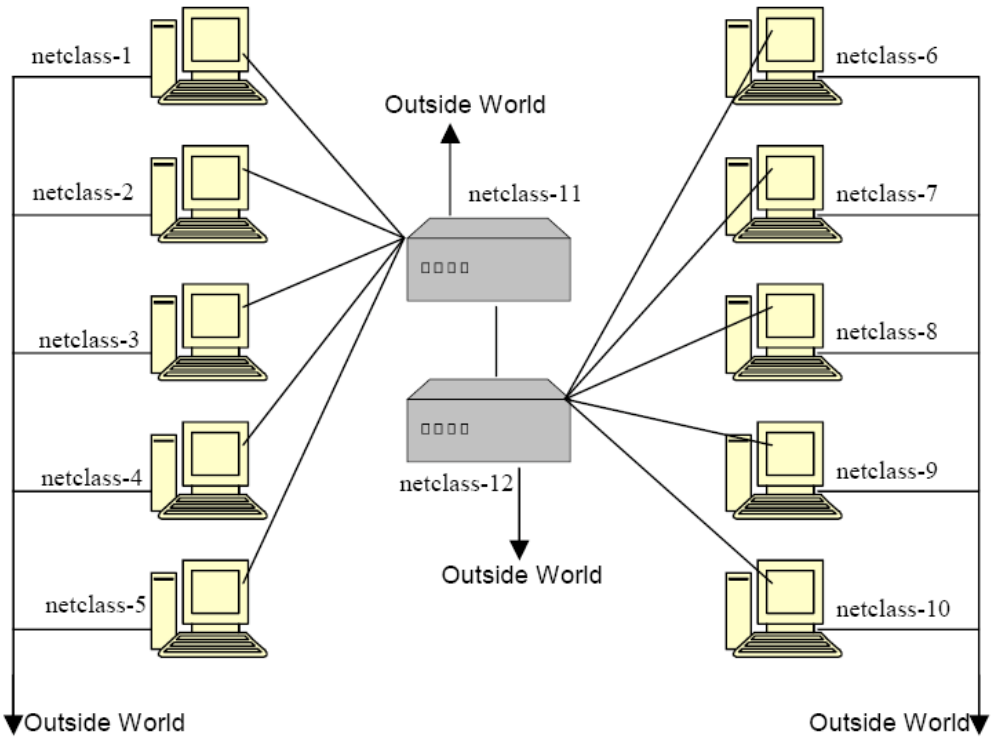
Hand-in

Please do electronic submission like we have done for previous homework assignments.

Setting Up Fish Machines

1. Create a directory in your Andrew home directory called 15-441.
2. Now you need to give access to *campusnet* so type the following:
   a. %fs sa /afs/andrew.cmu.edu/usr/<username> system:campusnet all
   b. %fs sa /afs/andrew.cmu.edu/usr/<username>/15-441 system:campusnet all
3. Create a file called *.klogin* inside the 15-441 folder which contains:
   a. *username@ANDREW.CMU.EDU*.
   b. Make sure the domain suffix is capitalized.
4. Create a another file called *.login* inside the 15-441 folder which contains:
   a. *aklog andrew.cmu.edu*
5. You cannot use unsecure telnet to connect to netclass machines. We will *only* support using SSH to connect to the netclass machines. On UNIX, the full command that you can use to login is:
   a. ssh -1 –l username@ANDREW.CMU.EDU netclass-<Y>.intro.cs.cmu.edu
   b. (<Y> ranges from 1-8; note: the first flag is "one" and the second is "ell").
6. The password is your CMU Andrew password.
7. When you are done with the lab, please make sure to perform step 2 again, but either remove all permissions or set to "l".

The topology of the CS441 lab is shown below: (all end with .intro.cs.cmu.edu)



The lab consists of 12 PCs as shown with 2 of them (11 and 12) configured as routers and the rest configured as endpoints. As can be seen the PCs are connected to form a LAN as well as being connected to the outside world (that's how you can telnet in). This is achieved by the use of 2 interfaces on each PC – one connected to local LAN and one to the outside world. The traffic going on in the LAN is considered private traffic – it is isolated from the outside world. In order to login to these machines use your **andrew-id** and **password** and don't log on to 11 and 12.

*Question 1: Determining Important Hardware Addresses aka *69*

The hardware addresses on our LAN are 6 byte Ethernet addresses. We want to know two hardware addresses on the LAN: the address of the machine on which you are running the Analyzer (`ethereal`) and a target machine ("netclass-11" or "netclass-12"), which is emitting packets to the other machines on the LAN and from which you will be sniffing the network traffic.

1. SSH to one of the netclass machines (1-8). Use the `ping` command to contact "netclass-12.intro.cs.cmu.edu" (you may substitute "netclass-12" for this, as you are on a machine which shares the same domain, intro.cs.cmu.edu). `ping` will cause your local host to create an entry in its ARP cache for the hardware address of the corresponding interface on the machine named "netclass-12".

   Find the IP and Hardware addresses of "netclass-12". Write the addresses in the spaces provided below:
   IP address:_____
   HW address: ____:____:____:____:____:____

2. Now we want to get the Ethernet hardware address and the IP (inet) address for *both* Ethernet interfaces on netclass-8.
   netclass-8 eth0 IP address:_____ HW address: ____:____:____:____:____:____
   netclass-8 eth1 IP address:_____ HW address: ____:____:____:____:____:____

3. Determine which Ethernet interface is used for the default route on netclass-8 (use the man pages as a guide to find the command to use) _____.

Question 2: Capturing and Viewing Packets *aka the dump*

Open up a terminal window. Since you don't have write access to the directories other than the local temporary directory, cd to "/tmp". Create a subdirectory with your username. You will create your dumpfile (the file that will contain the captured packets) in that subdirectory. Note: you can also save to your Andrew space, assuming that you have given the correct permissioned to Campusnet (see setup instructions).

You will capture the packets using *tethereal* which is in */usr/bin*. First run *tethereal –help* to see the command line options. We want to capture 500 packets from the external traffic (hint: there are 2 Ethernet cards on each PC: eth0 and eth1). You need to determine which one is dealing with the LAN traffic and which one is connected outside (which would mean you will see telnet packets when you look at the dump). Save the packets you sniff to a file (so that we can look at them with a graphical interface next).

Now start up */usr/bin/ethereal* (make sure your X-Server is running; see XWindows documentation for details). Ethereal will ask for the root password—just choose "Run unprivileged". When ethereal is started up it will have three empty panes. Go to File -> Open and open up the capture file.

Select the "+" sign in front of each of the protocol layers to get more details about each protocol header.

(a) View the captured broadcast packet data. Select the "Source" column to sort the packets by the source address. What is the most common source address? Why?

(b) List all of the protocols (ncp, sap, tcp, icmp, arp, udp, etc.) in your capture.

(c) Which protocol is most common? Why?

(d) Select a packet in the top frame that is labeled as a TCP packet. Determine the following values for this packet:

Ethernet destination address: ___:___:___:___:___:___
IP source address ___.___.___.___
IPTTL _____
TCP source port _____

(e) Select the "header length" field of the IP header. This should cause a byte in the raw data pane to be highlighted. What value does this byte have and what does it mean?

Question 3: TCP Forensics *aka CSI Networks*

Note: this question does not require you to use the 441 lab netclass machines.

You are the TCP specialist at the FBI. One day an FBI agent gives you a packet trace of a TCP connection between two machines on the Internet. The trace is believed to contain important information pertaining to national security.

This packet trace contains 113 packets, most of them IP packets. Each line in the trace is one packet, identified by its packet number (from 0 to 112). The rest of the line is a sequence of bytes, represented as hex numbers. For example, consider the first line of the trace:

```
0      45 00 00 2c f2 7b 00 00 40 06 da 71 80 02 dc 8a 80 02 d1 4f 6a b0
       00 17 43 97 0e d6 00 00 00 00 60 02 02 00 2b 13 00 00 02 04 05 b4
       00 00
```

The first number "0" (packet number 0) imples that this is the first packet received at the trace point.

The first byte of the packet is 0x45, which is 69 in decimal. As a hint, the first byte seems to indicate that this is an IPv4 packet, where the IP header contains 20 bytes. The fourth byte of the packet is 0x2c.

This trace, 441-S06-trace.txt, should be accompanied with this handout. To reduce the amount of work you have, we also provide a template code 441-S06-trace.c which parses the trace file into an array of bytes. You are free to use perl or any other tools (even by hand) to analyze this trace file.

Please answer the following questions.

(a) What is the client's IP address, the client's port number, the server's IP address, and the server's port number of this TCP connection? Based on the IP addresses of the client and server, what are their respective DNS names? You may assume that the computer which initiates the connection is the client, and the other computer is the server.
Client IP: _____
Client Port: _____
Client DNS Name: _____
Server IP: _____
Server Port: _____
Server DNS Name: _____

(b) Which Internet application (i.e. web, FTP, gopher, etc.) is running on top of this TCP connection? How do you know? (hint: you shouldn't guess based on the TCP payload).

(c) Using the TCP connection state diagram in Figure 6 of RFC 793, identify which packets (by their packet number) cause or result from the TCP state transitions of the TCP client. Your answer should be in the form: when a client receives packet X or uses request Y, its TCP state is changed from A to B, and packet Z is sent (or some other action takes places). For example, when a client receives user request OPEN, the TCP state is changed from CLOSED to SYN-SENT, and a packet Z is sent.

(d) Using the same format as used in (c), identify the state transitions with respect to the server.

(e) Recreate the "crime scene" at the client's terminal. What did the end user see on the screen? What did the end user type on the keyboard? What is he/she trying to do? We are not looking for a 100% accurate bit-by-bit transcription of the TCP payload, but you should describe in enough details what the end user has seen on the screen and typed on the keyboard.

(f) Please include any/all scripts/code/etc. you used to figure the above out. This is to server as evidence that you are indeed a qualified FBI TCP expert and did not "hire" someone else to perform this trace analysis.

Question 4: Tracepath aka network driving directions

This question will require you to use the time slot you signed up for. And please use netclass machines 9 and 10.

`tracepath`, like the more commonly known `traceroute`, is a program that allows you to determine the routers between your machine and a remote host. IP packets have a time-to-live (TTL) field, which gets decremented as the packet passes through intermediate nodes on the way to its final destination. If the time-to-live count ever reaches 0, the packet is discarded and an ICMP error message is returned to the original sender. tracepath exploits this characteristic to discover the path between your machine and a destination host by sending UDP packets with (abnormally) small time-to-live counts and then taking notice of the IP address of the host generating the ICMP error message. tracepath first sends a packet with TTL=1, then one with TTL=2, ... until the destination is reached. By then, each router along the way (each hop) would have sent back an ICMP error message. There are other details, but that is basically how `tracepath` works. On the netclass machines, `tracepath` is located in `/usr/sbin`.

1. Select a host close to CMU, such as "www.upenn.edu". Set up `tethereal` to capture packets on the external interface.

   NOTE: When *capturing* packets using `tethereal`, you may find yourself overwhelmed by SSH traffic to and from your netclass machine. You can fine-tune the kind of traffic captured by specifying a "filter expression", as documented in the `tethereal` man page.

   Make sure you have 2 terminals open, both running SSH sessions on the *same* netclass machine (9 or 10). On one you will execute the `tracepath` command, and on the other you will run `tethereal`. Start capturing packets, and then execute `tracepath` in the other terminal. When `tracepath` finishes running, stop capturing, load up the file in Ethereal and move to the decode window.

   What host did you query and how many hops (including the final hop to the destination) were required?

   host: _____ # hops: _____

2. View the captured packet data (if it helps, sort the entries by protocol). Select one of the packets with protocol "ICMP" containing "Time-to-live exceeded" in the description field (Don't pick one of the first three).

   What is the IP address of the router that sent the error message? (Hint: Look in the first "Internet Protocol" section of the middle window.) What is the corresponding hop number shown in the `tracepath` results on your terminal window?

   IP address: _____ hop #: _____

3. The data portion of an ICMP "Time-to-live exceeded" packet includes a copy of the IP header of the "original packet". In other words, it includes a copy of the IP header from the packet that caused the router to send the "Time-to-live exceeded" reply. (In this case,

the "original packet" is the one sent by your PC when you issued the `tracepath` command.)

Refer to your text for a description of the IP header. You will notice that the TTL field immediately precedes the Protocol field. The protocol field specifies the next higher-level protocol contained in the packet (in this case, UDP which is 17 = 0x11). You can look for the 0x11 in the next byte to verify that you have the correct offset for the TTL field.

Look at the "Time-to-live exceeded" packet you selected earlier. What is the value shown for the original packet's TTL field? What is the byte offset of the original packet's TTL field?

Time-to-live (TTL) value: _____
Offset (in hex): _____

4. Look at the UDP packet sent prior to the "Time-to-live exceeded" packet you selected earlier (if you sorted the entries by protocol, click on the "No." column to sort by arrival). Look at the IP section of this packet. What is the value shown for this packet's TTL field?

Time-to-live (TTL) value: _____

Question 5: Window size and TCP Throughput aka size matters

This question will require you to use the time slot you signed up for. And please use netclass machines 9 and 10.

1. As you have seen in class, TCP's performance is affected by the size of the "window" being used (flow control window being advertised by the receiver). In this part of the homework, you will be using a utility called `nliperf` (which should already be in your path, so you can run it just by typing `nliperf`). `nliperf` runs in 2 modes, client and server. nliperf in client mode transfers as much data (over TCP) as it can to another copy of nliperf running in server mode. It calculates the throughput achieved for this transfer in a set amount of time (10 seconds by default).

   SSH to two of the machines (call them machine A & B) you will be working on. You will use nliperf to send TCP packets from one of these machines (A) to the other one (B), while experimenting with different window sizes. On A, use the `-c` flag to specify the destination host, and the `-w` flag to specify window size. On B, use the –s flag to run `nliperf` in server mode. **Note:** If you receive an error message when trying to run `nliperf` in server mode, it's possible that a previous user left an instance of it running. You can run `ps aux | grep nliperf` to see if `nliperf` is running, and by which user. If you find you're not alone, send us and the person an email.

   Measure the throughput for each of the window sizes given below.

   Note: There is a subtlety with specifying window sizes in `nliperf`, attributable to the Linux socket layer. For a window size of *X* KB, you will need to specify `-w X/2K`. For instance, use `-w 2K` to specify a window size of 4 KB. When run, `nliperf` prints the window size used.

   a. 2KB Throughput: _____ Mbits/sec
   b. 4KB Throughput: _____ Mbits/sec
   c. 8KB Throughput: _____ Mbits/sec
   d. 16KB Throughput: _____ Mbits/sec
   e. 32KB Throughput: _____ Mbits/sec
   f. 64KB Throughput: _____ Mbits/sec

2. Plot Window size (KB) vs. throughput (Mbits/sec). How are these values related? What are the implications of this?

3. What would happen if we kept increasing the window size in part 1 (512 KB, 1024 KB...)?