Iowa State University
Department of Electrical and Computer Engineering
Cpr E 489: Computer Networking and Data Communication
Lab Experiment #1
Networking Utility Programs
(Total Points: 100)

Pre-lab Recommendations

Labs in this course will typically be conducted using the `489labuser` account. This means that you will not have access to your personal drive and any files left on your computer will remain there for anyone to see. The TA recommends that you bring a flash drive to easily bring home screenshots or files collected in the lab. However, you are welcome to use whatever file transfer method you prefer.

Objective

To demonstrate how to use network diagnostic and probing tools such as `ping`, `nslookup`, `ifconfig`, `iperf`, `traceroute`, `tcptraceroute`, `Nmap`, `tcpdump`, `tcptrace`, and `Wireshark`.

Lab Expectations

Work through the lab individually and let the TA know if you have any questions. After the lab, write up a lab report and be sure to:

a) summarize what you learned in a few paragraphs (**30 points**)
b) answer any questions asked in the exercises throughout the experiment. (**5 points for each of the 14 questions)**

Answers to questions should contain:

1) A screenshot including the command/action and its output
2) A written comment or annotation of the screenshot (as appropriate)

Problem Description

In this lab experiment, you will learn about several network utility programs. For each program, after some usage instructions, you will be asked to use what you learned in order to diagnose the network and/or configure the network for your machine.

Note:	Network probing tools, such as `tcpdump` and Wireshark, are useful for analyzing network traffic and for troubleshooting network problems. A number of privacy and security concerns are raised with the use of these tools – please use them in an ethical manner.

Warning:	During this lab, do not log into any websites or remote applications. The tools we will be using allow anyone to see the content of packets on the wire. Hence, your username and/or password may be easy to discover.

Login Information

Make sure that you are logged in to your lab computer using the following credentials (or else some commands that require "`sudo`" won't work):
Username: `489labuser`
Password: `489labuser`

ping

Overview

`ping` is a diagnostic tool used for verifying the connectivity between two hosts on a network. It sends Internet Control Message Protocol (ICMP) echo request packets (pings) to a remote host and waits for ICMP echo responses (pongs). If the connections exist and the target host is operational, an ICMP response will be received, if the host does not block ICMP requests. Additionally, `ping` also estimates the round-trip time of the `ping` packets.

Usage

`ping` is a command line application that must be run in a terminal window. It is often used without any additional options and is terminated with Ctrl-C. (Options and usage information are often documented in a program's "man page." You may find all available options for the `ping` command by typing **`man ping`** at the prompt.) Pinging www.iastate.edu yields:

```
$ ping www.iastate.edu
PING www.iastate.edu (129.186.23.166) 56(84) bytes of data.
64 bytes from webdev-pool05.its.iastate.edu (129.186.23.166): icmp_seq=1 ttl=252
time=0.643 ms
64 bytes from webdev-pool05.its.iastate.edu (129.186.23.166): icmp_seq=2 ttl=252
time=0.623 ms
64 bytes from webdev-pool05.its.iastate.edu (129.186.23.166): icmp_seq=3 ttl=252
time=0.727 ms
64 bytes from webdev-pool05.its.iastate.edu (129.186.23.166): icmp_seq=4 ttl=252
time=0.586 ms
64 bytes from webdev-pool05.its.iastate.edu (129.186.23.166): icmp_seq=5 ttl=252
time=0.798 ms
64 bytes from webdev-pool05.its.iastate.edu (129.186.23.166): icmp_seq=6 ttl=252
time=0.518 ms
64 bytes from webdev-pool05.its.iastate.edu (129.186.23.166): icmp_seq=7 ttl=252
time=0.742 ms
64 bytes from webdev-pool05.its.iastate.edu (129.186.23.166): icmp_seq=8 ttl=252
time=0.677 ms
64 bytes from webdev-pool05.its.iastate.edu (129.186.23.166): icmp_seq=9 ttl=252
time=0.704 ms
64 bytes from webdev-pool05.its.iastate.edu (129.186.23.166): icmp_seq=10 ttl=252
time=0.655 ms

--- www.iastate.edu ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9509ms
rtt min/avg/max/mdev = 0.518/0.667/0.798/0.079 ms
```

The output can be split into three sections:

- The first section, i.e., the single line starting with the word PING, shows an overview of the command.
- The second section, i.e., the lines beginning with 64 bytes, shows a running tally of the responses received.
- The third section, everything after the line --- www.iastate.edu ping statistics ---, shows a summary of the results. In this case, the results are acceptable – none of the packets were dropped, and the responses were received in a timely manner. The average round-trip time for this example was 0.667 ms.

Exercises

1) Use `ping` to find the average round-trip time from your machine to each of the following machines (include the output from the third section of `ping` for verification):
   Hint: You can use "ping –c 4 hostname" to send only 4 echo requests.

www.google.com
www.cam.ac.uk
www.iastate.edu

2) A loopback address is a special IP address, 127.0.0.1, reserved by InterNIC for testing network cards. In other words, pinging the loopback address is not a test of connection, but a test of network setup. Ping 127.0.0.1 and comment on the difference in average round-trip time compared to the previous exercise.

nslookup

Overview

`nslookup` is a program used to query Internet domain servers. It has two modes: non-interactive and interactive.

- **Non-interactive mode** is used to print the name and requested information for a host or domain.
- **Interactive mode** allows the user to query name servers for information about various hosts and domains, or to print a list of hosts in a domain.

Usage

Using `nslookup` to non-interactively query for the IP address of www.iastate.edu yields:

```
$ nslookup www.iastate.edu
;; Got recursion not available from 192.168.254.254, trying next server
Server:   129.186.140.200
Address:    129.186.140.200#53

Name:    www.iastate.edu
Address: 129.186.23.166
```

Typing `nslookup` on the command line without any arguments allows you to control `nslookup` interactively. Terminate an interactive session by typing **Ctrl-C** or entering the **exit** command at the `nslookup` prompt. As an interactive session, the previous query yields:

```
$ nslookup
> set type=A
> www.iastate.edu
;; Got recursion not available from 192.168.254.254, trying next server
Server:   129.186.140.200
Address:    129.186.140.200#53

Name:    www.iastate.edu
Address: 129.186.23.166
```

By default, `nslookup` queries for **A** records, but you can use the **set type** command to change the query to one of the following:

A	the host's Internet address
CNAME	the canonical name for an alias
HINFO	the host CPU and operating system type
MINFO	the mailbox or mail list information
MX	the mail exchanger
NS	the name server for the named zone

PTR	the host name if the query is an Internet address; otherwise, a pointer to other information
SOA	the domain's "start-of-authority" information
TXT	the text information
WKS	the supported well-known services

The following example returns the name servers for google.com:

```
$ nslookup
> set type=NS
> google.com
;; Got recursion not available from 192.168.254.254, trying next server
Server:         129.186.140.200
Address:    129.186.140.200#53

Non-authoritative answer:
google.com    nameserver = ns3.google.com.
google.com    nameserver = ns2.google.com.
google.com    nameserver = ns1.google.com.
google.com    nameserver = ns4.google.com.

Authoritative answers can be found from:
ns1.google.com    internet address = 216.239.32.10
ns2.google.com    internet address = 216.239.34.10
ns3.google.com    internet address = 216.239.36.10
ns4.google.com    internet address = 216.239.38.10
```

Exercises

3) Use `nslookup` to non-interactively determine the IP addresses and aliases (canonical names) for the following machines:

   www.iastate.edu
   www.microsoft.com
   www.wikipedia.com

4) Use `nslookup` to interactively find the mail exchanger for ece.iastate.edu.
5) Use `nslookup` to interactively to find the hostname of the machine with IP address 129.186.215.40.

ifconfig

Overview

`ifconfig` is a command line tool for configuring and displaying a network's interface parameters.

Usage

Entering `ifconfig` at the prompt (preceded by `/sbin/`) without specifying any options will provide a complete description of the current state of all active network interfaces. For example, on the lab machine with hostname co2061-20.ece.iastate.edu, `ifconfig` returns:

```
$ /sbin/ifconfig

enp0s31f6: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 74:86:e2:28:9d:d2  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
```

```
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 19  memory 0x72280000-722a0000

enp3s0f0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.254.20  netmask 255.255.255.0  broadcast 192.168.254.255
        inet6 fe80::e63d:1aff:fea0:2c42  prefixlen 64  scopeid 0x20<link>
        ether e4:3d:1a:a0:2c:42  txqueuelen 1000  (Ethernet)
        RX packets 34761626  bytes 46510978417 (43.3 GiB)
        RX errors 0  dropped 1  overruns 0  frame 0
        TX packets 22225812  bytes 15315106324 (14.2 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16

enp3s0f1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.75.20  netmask 255.255.255.0  broadcast 192.168.75.255
        inet6 fe80::e63d:1aff:fea0:2c43  prefixlen 64  scopeid 0x20<link>
        ether e4:3d:1a:a0:2c:43  txqueuelen 1000  (Ethernet)
        RX packets 51432  bytes 21292830 (20.3 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 935  bytes 78125 (76.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 17

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4624  bytes 467585 (456.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4624  bytes 467585 (456.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
        ether 52:54:00:b8:49:64  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Some important information provided by the `ifconfig` command includes:

- Each active interface is identified by its name. For instance, on this machine, **enp3s0f0** and **lo** (the loopback adapter) are both active.
- In the case of a physical network adapter, the MAC address is returned, which is preceded by the term **HWaddr**.
- The IP address of the interface is preceded by the term **inet**, the broadcast address is preceded by **broadcast**, and the subnet mask is preceded by **netmask**.
- The IPv6 address of each interface is preceded by the term **inet6** and its scope by the word **scopeid**.
- The types of activity of each interface are listed together. In the case of **enp3s0f0** above, it lists **UP BROADCAST RUNNING MULTICAST**.
- Statistics for received and transmitted packets are listed on lines beginning with **RX** and **TX**, respectively. These lines are followed by the total number of bytes received and transmitted on the device.

A number of options can be specified with the `ifconfig` command (eth0 or eth1 is what you would mostly find on other machines. **In Coover 2061, replace that with enp3s0f0**):

- **–a** commands `ifconfig` to show information about all interfaces, both active and inactive. On co2061-1, `ifconfig -a` returns results for enp3s0f0, lo, enp3s0f1, and virbr0.
- **–s** is the "short listing" option, which shows a one-line summarized listing of data about each interface. The information returned is about interface activity, and not configuration. The output will be identical to what is returned by the `netstat -i` command.
- **–v** specifies "verbose" – this option returns extra information when there are certain types of error conditions to help with troubleshooting.
- You can specify an interface. For instance, you could issue the command `ifconfig eth0` if you only wanted information about the eth0 interface, and not the loopback interface. Additionally, there are several options that require specifying the interface you wish to configure or get information about (e.g., eth[int] [addr], which is described below).
- **up** activates an interface if it is not already active. For instance, `ifconfig eth0 up` causes eth0 to be activated.
- **down** deactivates the specified interface.
- **[interface] [addr]** changes the interfaces IP address. For example, `ifconfig eth0 192.168.2.103`, will set eth0's IP address to 192.168.2.103.

Exercises

6) Use `ifconfig` to determine the IP address for interface `enp3s0f0` on your machine. HINT: Record this IP address for use in later exercises.

iperf

Overview

`iperf` is a tool to measure bandwidth between two hosts. `iperf` reports TCP and UDP bandwidth and throughput, and for UDP it additionally outputs delay jitter and datagram loss. `iperf` is useful to measure the performance of a network, which can be an indicator of hardware problems if the bandwidth is lower than expected.

Usage

Two instances of `iperf` are required to measure bandwidth: a server and client. During the bandwidth test, the client will send as many packets as possible to the server within a given time period. The bandwidth is recorded, along with loss, if any.

As an example, consider two hosts: **A with IP address 192.168.254.15** and **B with IP address 192.168.254.16** (named host-A and host-B, respectively).

On host-A, the `iperf` server will wait for clients to connect to it. The server has been started as follows:

```
$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

On host-B, the `iperf` client is started (the bandwidth test will run for 10 seconds):

```
$ iperf -c 192.168.254.15
------------------------------------------------------------
Client connecting to 192.168.254.15, TCP port 54276
```

```
TCP window size: 19.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.254.16 port 45670 connected with 192.168.254.15 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.09 GBytes   935 Mbits/sec
```

The bandwidth between the two hosts is 935 Mbits/sec. If this is a 1 Gbps switched network, it can be assumed that the network is healthy.

Exercises

7) This exercise will be completed using the loopback address. Open two terminals. On Terminal 1 run **iperf -s** and on Terminal 2 run **iperf -c 127.0.0.1**. Make sure to include legible screenshot(s) of both terminals. Summarize your observations on the bandwidth between the two hosts.

traceroute

Overview

`traceroute` allows users to determine the route that a packet takes from the local host to a remote host, as well as latency and reachability from the source to each hop. `traceroute` is generally used as a powerful debugging tool by network managers. It makes use of both UDP and ICMP. The local host first sends a UDP datagram with TTL (Time to Live) field set to 1 as well as an invalid port number to the remote host. The first gateway/router to see the datagram decreases the TTL field by one, discards the datagram since the TTL field has reached zero, and sends an ICMP Time Exceeded message back to the local host. This information allows the local host to identify the first gateway/router in the route. `traceroute` continues to identify the remaining gateways/routers between the local host and remote host by sending datagrams with successively larger TTL fields. When the datagram finally reaches the destination, the remote host returns an ICMP Port Unreachable message back to the local host because of the invalid port number deliberately set in the datagram.

Usage

The `traceroute` command is very flexible and has many options. The only mandatory parameter is the destination host name or IP address. An example `traceroute` command line looks like this:

```
$ traceroute -n 129.186.215.40
traceroute to 129.186.215.40 (129.186.215.40), 30 hops max, 60 byte packets
 1  129.186.5.253  0.674 ms  0.810 ms  1.040 ms
 2  129.186.254.164  0.902 ms  0.931 ms  0.968 ms
 3  129.186.215.40  0.195 ms  0.173 ms  0.175 ms
```

In this example, you can see that the packets destined for 129.186.215.40 were routed through two gateways/routers. Over three attempts at sending datagrams to 129.186.215.40, the average roundtrip time can be calculated as (0.195+0.173+0.175)/3 = 0.181 ms.

The **–n** option with `traceroute` prints the hop addresses as IP addresses rather than gateway names. Try `traceroute` without –n to see the gateway names.

Exercises

8) Perform `traceroute` from your computer to www.cmu.edu. Summarize your observations on number of hops, routes, gateways, latency, and reachability. HINT: What does an asterisk mean if it appears in the output?

tcptraceroute

Overview

`tcptraceroute` is similar to `traceroute`, but instead of using UDP and ICMP packets, `tcptraceroute` makes use of TCP SYN packets to bypass the most common firewalls and elicit responses from a wider variety of machines than `traceroute`.

Usage

The `tcptraceroute` command, like `traceroute`, is very flexible and has many options. The only mandatory parameter is the destination host name or IP address. (Note that `tcptraceroute` requires super user permissions to run, using `sudo`.) An example `tcptraceroute` command line looks like this:

```
$ sudo tcptraceroute -q 2 www.microsoft.com
traceroute to www.microsoft.com (23.222.196.57), 30 hops max, 60 byte packets
 1  gateway (192.168.254.254)  0.110 ms  0.140 ms
 2  routera-129-186-5-0.tele.iastate.edu (129.186.5.252)  1.056 ms  1.126 ms
 3  b31dmz1-vlan254.tele.iastate.edu (129.186.254.131)  0.820 ms  0.947 ms
 4  b31gb2-438.tele.iastate.edu (192.245.179.52)  0.648 ms  0.720 ms
 5  b31nat1-450.tele.iastate.edu (192.245.179.183)  0.527 ms  0.516 ms
 6  * *
 7  * *
 8  mtc-gr-01-1-te-0-0-0-17.895.northernlights.gigapop.net (146.57.253.10)  5.997
ms  5.997 ms
 9  * *
10  a23-222-196-57.deploy.static.akamaitechnologies.com (23.222.196.57)
<syn,ack>  5.032 ms  5.398 ms
```

The **`-q2`** option sends two probes per hop. The **`-s`** option could also be used but is not needed in this case since it is enabled by default.

Exercises

9) Use `tcptraceroute` to determine the route packets take to www.ed.ac.uk. What is different from the trace using `traceroute`? Why do you think this is so?

Nmap

Overview

`nmap` (Network Mapper) is a security scanner originally written by Gordon Lyon (also known by his pseudonym Fyodor Vaskovich) used to discover hosts and services on a computer network, thus creating a "map" of the network. To accomplish its goal, `nmap` sends specially crafted packets to the target host and then analyzes the responses. Many systems and network administrators also find it useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.

```
$ nmap -Pn 129.186.215.41

Starting Nmap 6.40 ( http://nmap.org ) at 2017-08-22 15:46 CDT
Nmap scan report for bones.ee.iastate.edu (129.186.215.41)
Host is up (0.00054s latency).
Not shown: 989 closed ports
PORT    STATE SERVICE
21/tcp  open  ftp
22/tcp  open  ssh
23/tcp  open  telnet
25/tcp  open  smtp
```

```
79/tcp  open  finger
110/tcp open  pop3
111/tcp open  rpcbind
143/tcp open  imap
587/tcp open  submission
700/tcp open epp

Nmap done: 1 IP address (1 host up) scanned in 5.85 seconds
```

Note:	**Port scanning is considered one of the first steps in an attack, so perform port scans only on machines that you have been given permission to do so. This program is being introduced to you so that you can test your own machine during socket programming in order to verify that you have opened a port correctly.**

Exercises

10) Is port 22 (SSH) open on your computer's enp3s0f0 interface? Note that `nmap` accepts only the Host IP.

tcpdump

Overview

`tcpdump` is a command line tool for analyzing raw network traffic; every packet going through the network interface card is captured (i.e., `tcpdump` is a packet sniffer). This tool is commonly used by developers to debug network applications and by network administrators to log network traffic for later analysis. All network traffic received by the network interface is captured by `tcpdump`, including traffic that is not related to the system running `tcpdump`.

Usage

`tcpdump` must be started from the command line and requires root privileges to run. In order to use it, you must become the root user (using `sudo`). Start `tcpdump` by typing `'sudo /usr/sbin/tcpdump -i enp3s0f0'` at the shell prompt. `tcpdump` will start dumping the headers of all packets received by the network interface `enp3s0f0` to the terminal. Depending on the amount of traffic, the information given by `tcpdump` can quickly become overwhelming. Press `Control + C` to stop `tcpdump`.

Filters

In reality, we might only be interested in checking specific network traffic, not all of it. This is why we use the different filters available in `tcpdump`. (When running these commands with Coover 2061 computers, be sure to use the full path of `tcpdump` with `sudo`, i.e., `sudo /usr/sbin/tcpdump`.) Common filters include:

host

The host filter will filter out all traffic sent or received to a certain host. The two examples below show how to log traffic to host 129.186.1.200 and to a known machine name.

```
$ tcpdump host 129.186.1.200
$ tcpdump host ns-1.iastate.edu
```

src and dst

These work the same as host, except you can explicitly filter either source or destination traffic. For example, to log all traffic sent to your host, you can use

```
$ tcpdump dst <IP address>
```

net
This will capture an entire network segment's traffic. For example, if you wanted to capture all traffic on subnet 129.186.1.0/24:

```
$ tcpdump net 129.186.1.0/24
```

Note: /24 is a network mask for 1-Class C network in CIDR notation, i.e., 255.255.255.0.

proto
Filter based on the network protocol. Supported protocols include tcp, udp, icmp, arp, rarp and other. For example, to view all tcp traffic on the network:

```
$ tcpdump tcp
```

Note: You do not type "proto" before the protocol type.

Port
Filter based on the TCP or UDP port. For example, to view all port 80 (HTTP) traffic:

```
$ tcpdump port 80
```

src port and dst port
This works the same as port, except you can explicitly filter only source and destination ports. For example, to view all incoming port 21 (FTP) traffic:

```
$ tcpdump dst port 21
```

Combining Filters
The power of filters can be enhanced even further by combining them. By combining multiple filters, a very specific subset of network traffic can be logged. Filters are combined using the logical operators *and*, *or*, and *not*. Some examples are as follows:

View all tcp traffic from the machine 192.168.0.2 destined for port 21:

```
$ tcpdump tcp and src 192.168.0.2 and dst port 21
```

Examine all traffic originating from the 129.186.158.0 network destined for the 192.168.1 network:

```
$ tcpdump src net 129.186.158.0/23 and dst net 192.168.1.0/24
```

Examine all traffic from your host that is not the ICMP protocol:

```
$ tcpdump src <IP address> and not icmp
```

Show only ICMP traffic that is not an echo request (8) or an echo reply (0):

```
$ tcpdump 'icmp[0]!= 8 and icmp[0] !=0'
```

Exercises

11) For this exercise, work with a partner and alternate roles. Your machine is currently undergoing a ping flood attack! On partner1's computer, execute ping [address] where "address" is partner2's IP. On partner2's computer. use tcpdump and filter for ICMP packets to determine the IP address of the machine that is sending the packets. Include the IP addresses of your machine and your partner's in your report.

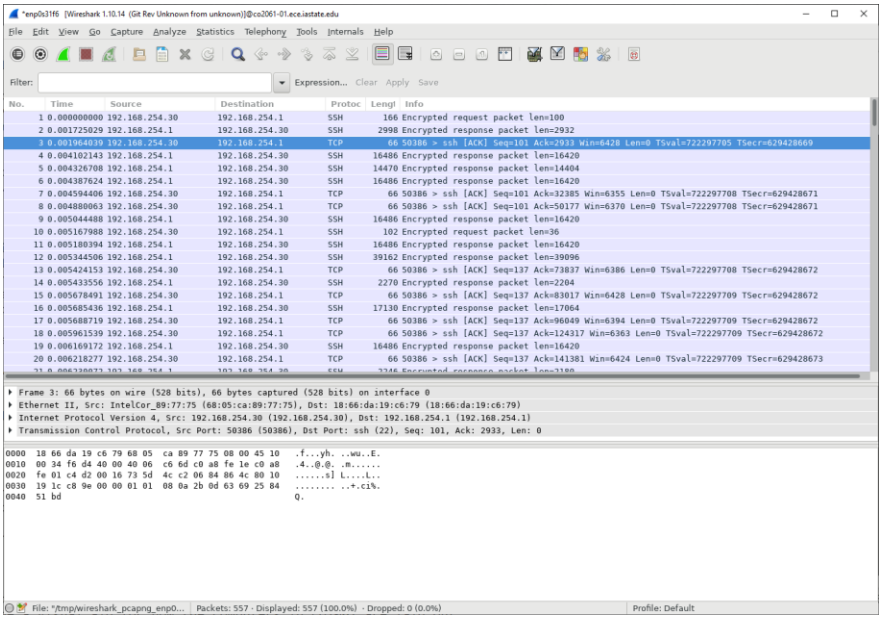
Wireshark

Overview

Wireshark, the successor to Ethereal, is an open-source network analyzer. Like `tcpdump`, it's considered a packet sniffer. Wireshark is functionally similar to `tcpdump`; however, it has a graphical user interface and many more filtering options than `tcpdump`. Wireshark is heavily used when trying to debug the network from a host perspective, and its powerful feature set makes it popular throughout industry.

Usage

Wireshark can be started by typing `wireshark` at the command line. After typing the command, you should get the GUI version of Wireshark.



First, notice that Wireshark has three panes:

- The top pane is the packet list pane. It displays a summary of each captured packet. Click a packet in this pane to display more detailed information about it in the other two panes.
- The middle pane is the tree view pane. It displays more information about the packet selected in the packet list pane. It's organized hierarchically, with the lower network layers at the top of the pane.
- The bottom pane is data view pane. It displays the packet selected in the packet list pane as hex and ASCII. Also, it highlights the data associated with the field selected in the tree view pane.

In order to use Wireshark to sniff the wire, you first need to select the device used to communicate with the network (most often eth0, but in our case we'll be using `enp3s0f0`). After you do this, Wireshark will capture all packets that go to and out of the interface card. To accomplish this, follow these steps:

- Go to **Capture → Options**. Make sure `enp3s0f0` is selected as the interface at the top of the Options window.
- Now, make sure the "**Use promiscuous mode on all interfaces**" box is checked. This will allow the network interface card to capture any packets on the wire (as opposed to just the packets destined for your machine).
- Click **Start**. After about 10 seconds, click **Stop**. Take a look at what was captured: source, protocol, time, etc. Since the results are usually plenty, you can also filter through them using

preset filters, or even your own custom filters. Click on the **Filter** button and select "**TCP only**" from the list. Now look at the capture list and see how your results are filtered.

Exercises

12) To generate HTTP connections, open a web browser such as Firefox on the command line with, for instance, "`firefox www.google.com`". Let's analyze the TCP connections between your computer and www.iastate.edu.

Start a new capture and then connect to www.iastate.edu. Let it run for about 10 seconds. Use **Statistics-> Conversations -> TCP** to see all the TCP connections between your computer and various IP addresses. For this exercise we are concerned with connections to the IP address corresponding to www.iastate.edu. (HINT: what command resolves a hostname to an IP address?). Sort the connections by **Address B** and take a **screenshot** of **only** the conversations between your IP address and the IP address of "www.iastate.edu". Then answer the following questions:

- For **each** connection, indicate the duration and the number of packets sent from your computer during that connection.
- What were the destination ports used and what protocols do they correspond to?

13) Continuing the `tcpdump` example, your computer is still under a ping flood (ICMP request and reply packets). Work with a partner and perform alternate roles. Start a new capture, and let it run for about 10 seconds.

- Determine how much data (in bytes) each ICMP packet contains.
- Determine the arrival time for each ping request packet.

14) Start a new capture, and let it run while you complete a `traceroute` and `tcptraceroute` to www.ebay.com.

- What types of packets are sent with `traceroute`?
- What types of packets are sent with `tcptraceroute`?