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)

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 and let the TA know if you have any questions. After the lab, write up a lab report with your partner, including a screen shot of your results. 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**)

Labs in CprE 489 are expected to be completed with a partner. Have one partner from your team submit the lab report.

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.lenovo.com.cn

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 explain the results.

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.facebook.com
   www.microsoft.com
   www.wikipedia.com

4) Use `nslookup` to interactively find the mail exchanger for ece.iastate.edu.
5) Use `nslookup` to find the name 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 co2061-01.ece.iastate.edu, `ifconfig` returns:

```
$ /sbin/ifconfig

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 38  bytes 3264 (3.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 38  bytes 3264 (3.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

p1p1: flags=4163<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.77.1  netmask 255.255.255.0  broadcast 192.168.77.255
        inet6 fe80::9794:7b38:881d:2cb  prefixlen 64  scopeid 0x20<link>
        ether 68:05:ca:61:c2:2f  txqueuelen 1000  (Ethernet)
        RX packets 2  bytes 128 (128.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 50  bytes 8139 (7.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xf71c0000-f71e0000
```

```
p2p1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.254.1  netmask 255.255.255.0  broadcast 192.168.254.255
        ether 18:66:da:19:c6:79  txqueuelen 1000  (Ethernet)
        RX packets 973276  bytes 596283705 (568.67 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 822707  bytes 353475016 (337.1 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xf7200000-f7220000


virbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
        ether 52:54:00:7b:9f:47  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, **p2p1** 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 **p2p1** 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 p2p1**):

- **–a** commands `ifconfig` to show information about all interfaces, both active and inactive. On co2061-1, `ifconfig -a` returns results for p2p1, lo, p1p1, 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 `p2p1` 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 require both partners' machines. On partner1's computer run **iperf -s** and on partner2's run **iperf -c [address]** where "address" is the ip address that partner1 recorded earlier (the IP address of interface p2p1). Make sure to include screenshots from both computers. Summarize your observations on the bandwidth between the two hosts. Is the connection most likely 10 Mbps or 100 Mbps or 1 Gbps?

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 numerically. 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.

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 partner's computer? 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 p2p1`**'** at the shell prompt. `tcpdump` will start dumping the headers of all packets received by the network interface `p2p1` 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) 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.

tcptrace

Overview

tcptrace is a tool used to analyze output files from programs like tcpdump and Wireshark. For each connection, it keeps track of elapsed time, bytes/segments sent and received, transmissions, round trip times, window advertisements, throughput, etc. It can also produce a number of graphs for further analysis.

Usage

In order to use tcptrace to analyze an output file, packets must first be captured. For example, the following command executes tcpdump, capturing packets and writing the data to **file.dump**:

```
$ sudo /usr/sbin/tcpdump -w - > file.dump
```

The following command executes tcptrace on file.dump:

```
$ tcptrace /local/489labuser/file.dump
```

Example output of tcptrace:

```
$ tcptrace /local/489labuser/file.dump
1 arg remaining, starting with '/local/489labuser/file.dump'
Ostermann's tcptrace -- version 6.6.7 -- Thu Nov  4, 2004

5284 packets seen, 4136 TCP packets traced
elapsed wallclock time: 0:00:00.920142, 5742 pkts/sec analyzed
trace file elapsed time: 0:00:29.900456
TCP connection info:
  1: co2048-13.ece.iastate.edu:45142 - windc3.iastate.edu:389 (a2b)
2>    1<
  2: co2048-13.ece.iastate.edu:43632 - ec2-50-112-202-19.us-west-2.compute.amazonaws.com:443
(c2d)         10>    7<
  3: co2048-13.ece.iastate.edu:34310 - 72.21.91.29:emwavemsg (e2f)
12>    9<  (complete)
  4: co2048-13.ece.iastate.edu:34048 - server-52-84-143-12.yto50.r.cloudfront.net:443 (g2h)
9>    8<
  5: co2048-13.ece.iastate.edu:34268 - ord36s01-in-f142.1e100.net:443 (i2j)
39>   46<
  6: co2048-13.ece.iastate.edu:48216 - ord36s01-in-f142.1e100.net:emwavemsg (k2l)
11>    7<
  7: co2048-13.ece.iastate.edu:55222 - 209.56.124.152:443 (m2n)
636>  806<
  8: co2048-13.ece.iastate.edu:51410 - ec2-52-89-80-240.us-west-2.compute.amazonaws.com:443
(o2p)          11>    8<
```

In the above example, tcptrace is run on file.dump. The initial lines provide a brief summary of the analysis, e.g., the number of packets seen, the time taken to analyze file.dump, etc. The subsequent lines summarize the TCP connections that were traced in file.dump. In this example, the first connection was seen between machines co2048-13.ece.iastate.edu at TCP port 45142 and windc3.iastate.edu at TCP port 389. Similarly, the fifth connection was seen between machines co2048-13.ece.iastate.edu at TCP port 34268, and ord36s01-in-f142.1e100.net at TCP port 443 (FTP).

tcptrace uses a labeling scheme to refer to the individual connections traced. In the above example, the first connection is labeled a2b. For this connection, 2 packets were seen in the a2b direction (co2048-13.ece.iastate.edu → windc3.iastate.edu) and 1 packet was seen in the b2a direction (windc3.iastate.edu → co2048-13.ece.iastate.edu).

Connections are reported as complete if an entire TCP connection was traced, i.e., SYN and FIN segments opened and closed a connection, respectively. Connections may also be reported as reset if the connection was closed with an RST segment or as unidirectional if traffic was seen flowing in only one direction.

`tcptrace` can produce detailed statistics of TCP connections from dump files when given the **`-l`** (long output) option. For example:

```
$ tcptrace -l  file.dump
1 arg remaining, starting with 'file.dump'
Ostermann's tcptrace -- version 6.6.7 -- Thu Nov  4, 2004

5284 packets seen, 4136 TCP packets traced
elapsed wallclock time: 0:00:00.644451, 8199 pkts/sec analyzed
trace file elapsed time: 0:00:29.900456
TCP connection info:
85 TCP connections traced:
TCP connection 1:
   host a:        co2048-13.ece.iastate.edu:45142
   host b:        windc3.iastate.edu:389
   complete conn: no   (SYNs: 0)  (FINs: 0)
   first packet:  Wed Sep  7 10:50:38.538181 2016
   last packet:   Wed Sep  7 10:50:38.538889 2016
   elapsed time:  0:00:00.000708
   total packets: 3
   filename:      file.dump
   a->b:                     b->a:
     total packets:            2           total packets:             1
     ack pkts sent:            2           ack pkts sent:             1
     pure acks sent:           1           pure acks sent:            0
     sack pkts sent:           0           sack pkts sent:            0
     dsack pkts sent:          0           dsack pkts sent:           0
     max sack blks/ack:        0           max sack blks/ack:         0
     unique bytes sent:      668           unique bytes sent:       332
     actual data pkts:         1           actual data pkts:          1
     actual data bytes:      668           actual data bytes:       332
     rexmt data pkts:          0           rexmt data pkts:           0
     rexmt data bytes:         0           rexmt data bytes:          0
     zwnd probe pkts:          0           zwnd probe pkts:           0
     zwnd probe bytes:         0           zwnd probe bytes:          0
     outoforder pkts:          0           outoforder pkts:           0
     pushed data pkts:         1           pushed data pkts:          1
     SYN/FIN pkts sent:      0/0           SYN/FIN pkts sent:       0/0
     req 1323 ws/ts:         N/Y           req 1323 ws/ts:          N/Y
     urgent data pkts:         0 pkts      urgent data pkts:          0 pkts
     urgent data bytes:        0 bytes     urgent data bytes:         0 bytes
     mss requested:            0 bytes     mss requested:             0 bytes
     max segm size:          668 bytes     max segm size:           332 bytes
     min segm size:          668 bytes     min segm size:           332 bytes
     avg segm size:          667 bytes     avg segm size:           331 bytes
     max win adv:           1424 bytes     max win adv:             254 bytes
     min win adv:           1424 bytes     min win adv:             254 bytes
     zero win adv:             0 times     zero win adv:              0 times
     avg win adv:           1424 bytes     avg win adv:             254 bytes
     initial window:         668 bytes     initial window:            0 bytes
     initial window:           1 pkts      initial window:            0 pkts
     ttl stream length:       NA           ttl stream length:        NA
     missed data:             NA           missed data:              NA
     truncated data:           0 bytes     truncated data:            0 bytes
     truncated packets:        0 pkts      truncated packets:         0 pkts
     data xmit time:       0.000 secs      data xmit time:        0.000 secs
     idletime max:       29899.7 ms        idletime max:        29899.8 ms
     throughput:          943503 Bps       throughput:           468927 Bps
```

The initial lines of output summarize a connection and any TCP port numbers used. The following lines provide a thorough list of TCP statistics, for both the forward (a2b) and reverse (b2a) directions.

RTT (round-trip time) statistics are generated when the **`-r`** option is specified along with the `-l` option.

Similarly, statistics on the estimated congestion window are generated when the **`-W`** option is specified along with the `-l` option. Since there is no direct way to determine the congestion window at the TCP sender, the outstanding unacknowledged data is used to estimate the congestion window. Four statistics produced by the **`-W`** option are explained below:

- **max owin** – The maximum outstanding unacknowledged data (in bytes) seen at any point in time in the lifetime of the connection.

- **min non-zero owin** – The minimum (non-zero) outstanding unacknowledged data (in bytes) seen.

- **avg owin** – The average outstanding unacknowledged data (in bytes), calculated from the sum of all the outstanding data byte samples (in bytes) divided by the total number of samples.

- **wavg owin** – The weighted average outstanding unacknowledged data (in bytes) seen.

For example, if the outstanding data observed was 500 bytes for the first 0.1 seconds, 1000 bytes for the next 1 second, and 2000 bytes for the last 0.1 seconds of a connection that lasted 1.2 seconds, *wavg owin* = ((500 x 0.1) + (1000 x 1) + (2000 x 0.1)) / 1.2 = 1041.67 bytes. Note that the straightforward average reported in *avg owin* would have been (500 + 1000 + 2000) / 1.2 = 2916.67 bytes, which is a value less indicative of the outstanding data observed during most of the connection's lifetime.

Other options available for tcptrace include **-n**, which causes tcptrace to skip the name resolution for each connection and display only the IP addresses used, and **-s**, which produces only short names for each IP address. More information on tcptrace can be found at https://linux.die.net/man/1/tcptrace

Exercises

12) Use `tcpdump` to capture packets and save the data to a dump file. While you are capturing, make HTTP connections to the following machines:

    www.iastate.edu
    www.google.com

    In order to generate HTTP connections, open a web browser such as Firefox on the command line with, for instance, `firefox www.google.com`
    Use `tcptrace` to analyze the output of `tcpdump`. Identify the following information:

    a. Source and destination IP addresses and port numbers of the TCP connections.
    b. Duration of TCP connections.
    c. Total number of packets sent from your machine to each server.

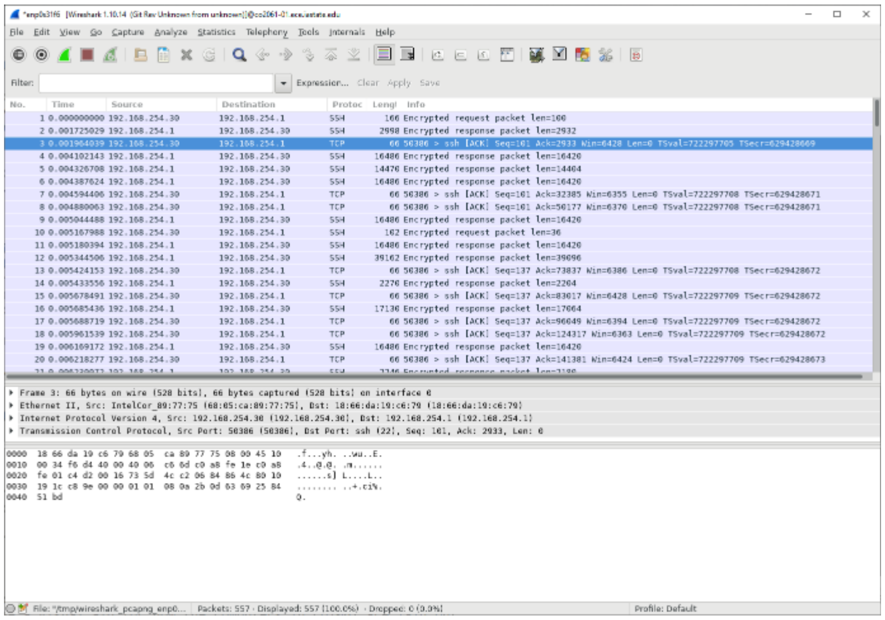
Wireshark

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.


*enp0s31f6 [Wireshark 1.10.14 (Git Rev Unknown from unknown)]@co2061-01.ece.iastate.edu
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.  Time        Source          Destination     Protoc Lengt Info
   1 0.000000000 192.168.254.30  192.168.254.1   SSH     166 Encrypted request packet len=100
   2 0.001725029 192.168.254.1   192.168.254.30  SSH    2998 Encrypted response packet len=2932
   3 0.001964039 192.168.254.30  192.168.254.1   TCP      66 56386 > ssh [ACK] Seq=101 Ack=2933 Win=6428 Len=0 TSval=722297705 TSecr=629428669
   4 0.004102143 192.168.254.1   192.168.254.30  SSH   16486 Encrypted response packet len=16420
   5 0.004326708 192.168.254.1   192.168.254.30  SSH   14476 Encrypted response packet len=14404
   6 0.004387624 192.168.254.1   192.168.254.30  SSH   16486 Encrypted response packet len=16420
   7 0.004594406 192.168.254.30  192.168.254.1   TCP      66 56386 > ssh [ACK] Seq=101 Ack=32385 Win=6355 Len=0 TSval=722297708 TSecr=629428671
   8 0.004880063 192.168.254.30  192.168.254.1   TCP      66 56386 > ssh [ACK] Seq=101 Ack=50177 Win=6370 Len=0 TSval=722297708 TSecr=629428671
   9 0.005044488 192.168.254.1   192.168.254.30  SSH   16486 Encrypted response packet len=16420
  10 0.005167988 192.168.254.30  192.168.254.1   SSH     102 Encrypted request packet len=36
  11 0.005180394 192.168.254.1   192.168.254.30  SSH   16486 Encrypted response packet len=16420
  12 0.005344506 192.168.254.1   192.168.254.30  SSH   39162 Encrypted response packet len=39096
  13 0.005424153 192.168.254.30  192.168.254.1   TCP      66 56386 > ssh [ACK] Seq=137 Ack=73837 Win=6386 Len=0 TSval=722297708 TSecr=629428672
  14 0.005433556 192.168.254.1   192.168.254.30  SSH    2276 Encrypted response packet len=2204
  15 0.005678491 192.168.254.30  192.168.254.1   TCP      66 56386 > ssh [ACK] Seq=137 Ack=83017 Win=6428 Len=0 TSval=722297709 TSecr=629428672
  16 0.005685436 192.168.254.1   192.168.254.30  SSH   17136 Encrypted response packet len=17064
  17 0.005688719 192.168.254.30  192.168.254.1   TCP      66 56386 > ssh [ACK] Seq=137 Ack=96649 Win=6394 Len=0 TSval=722297709 TSecr=629428672
  18 0.005961539 192.168.254.30  192.168.254.1   TCP      66 56386 > ssh [ACK] Seq=137 Ack=124317 Win=6363 Len=0 TSval=722297709 TSecr=629428672
  19 0.006169172 192.168.254.1   192.168.254.30  SSH   16486 Encrypted response packet len=16420
  20 0.006218277 192.168.254.30  192.168.254.1   TCP      66 56386 > ssh [ACK] Seq=137 Ack=141381 Win=6424 Len=0 TSval=722297709 TSecr=629428673
```

```
▸ Frame 3: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▸ Ethernet II, Src: IntelCor_89:77:75 (68:05:ca:89:77:75), Dst: 18:66:da:19:c6:79 (18:66:da:19:c6:79)
▸ Internet Protocol Version 4, Src: 192.168.254.30 (192.168.254.30), Dst: 192.168.254.1 (192.168.254.1)
▸ Transmission Control Protocol, Src Port: 56386 (56386), Dst Port: ssh (22), Seq: 101, Ack: 2933, Len: 0
```

```
0000  18 66 da 19 c6 79 68 05  ca 89 77 75 08 00 45 10   .f...yh. ..wu..E.
0010  00 34 f6 d4 40 00 40 06  c6 6d c0 a8 fe 1e c0 a8   .4..@.@. .m......
0020  fe 01 c4 d2 00 16 73 5d  4c c2 06 84 86 4c 80 10   ......s] L....L..
0030  19 1c c8 9e 00 00 01 01  08 0a 2b 0d 63 69 25 84   ........ ..+.ci%.
0040  51 bd                                              Q.
```

```
File: "/tmp/wireshark_pcapng_enp0...   Packets: 557 · Displayed: 557 (100.0%) · Dropped: 0 (0.0%)   Profile: Default
```

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 `p2p1`). 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 `p2p1` 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

13) Continuing the `tcpdump` example, your computer is still under a ping flood (ICMP request and reply packets). 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`?