Sandy

For the trace analysis portion of the assignment, I created a packet reader using libpcap. The program, named Sandy after the ongoing enormous tropical cyclone, takes as its arguments a pcap file and a user defined BPF (Berkeley Packet Filter) string.

Running the program with a pcap file and no filter string uses “tcp” as its string to return all TCP packets (effectively all of the packets in the given traces). To save time, I assumed that the server is always 128.8.126.92; however, this can easily be turned into a command line argument or changed in the code through the macro:

```
#define SERVER_IP       "128.8.126.92".
```

Airplane

I initially chose to analyze **airplane**, the trace group captured from a satellite link during a flight to California. During the single file download, the first dozen and last dozen lines of initial output of sandy on the airplane pcap file is as follows:

```
bash-3.2$ ./sandy traces/airplane_large_out.pcap | head -n 13
WELCOME TO HURRICANE SANDY.
traces/airplane_large_out.pcap successfully opened.
iface: savefile, linktype: EN10MB (Ethernet)
Filter compiled and set.
Packet no: 1
        From: 172.19.131.96     To: 128.8.126.92
        src port: 50408
        dst port: 80

Packet no: 2
        From: 128.8.126.92      To: 172.19.131.96
        src port: 80
        dst port: 50408


bash-3.2$ ./sandy traces/airplane_large_out.pcap | tail -n 12
Packet no: 59437
        From: 128.8.126.92      To: 172.19.131.96
        src port: 80
        dst port: 50408
        Payload (1007 bytes):

Packet no: 59438
        From: 172.19.131.96     To: 128.8.126.92
        src port: 50408
        dst port: 80

Client packets sent: 22976       Server packets sent: 36462
```

(1) The client (the file receiver) is host 172.19.131.96 on port 50408. The server (the file sender) is host 128.8.126.92 on port 80.
(2) The file sender sent 36462 packets.

```
bash-3.2$ ./sandy traces/airplane_large_out.pcap "tcp[tcpflags] == tcp-ack" |
tail -n 1
Client packets sent: 22974       Server packets sent: 36073
```

22974 acknowledgement packets were sent by the file receiver. After tweaking the program to provide time elapsed:

```
bash-3.2$ ./sandy traces/airplane_large_out.pcap "tcp[tcpflags] == tcp-ack and
src net 172.19.131.96" | tail
Last result: 1348891449.521158
```

```
Result no: 22974 at time: 1348891449.521676 //in seconds
        Time since last result: 518ms
        From: 172.19.131.96     To: 128.8.126.92
        src port: 50408
        dst port: 80

Client packets sent: 22974       Server packets sent: 0
Total time elapsed: 1348891449521676 // this is in nanoseconds

bash-3.2$ ./sandy traces/airplane_large_out.pcap "tcp[tcpflags] == tcp-ack and
src net 172.19.131.96" | head
WELCOME TO HURRICANE SANDY.
Using your filter: tcp[tcpflags] == tcp-ack and src net 172.19.131.96
traces/airplane_large_out.pcap successfully opened.
iface: savefile, linktype: EN10MB (Ethernet)
Filter compiled and set.
sinceLast = 1348889774936445 + timesofar: 0
Last result: 0.000000
Result no: 1 at time: 1348889774.936445
        Time since last result: 1348889774936445ms
        From: 172.19.131.96     To: 128.8.126.92
```

The total time elapsed between acks can thus be calculated by:
(1348891449521676 / 1000000) - 1348889774.936445 = **1674.58523083** seconds. After adding cumulative ack numbers to the output, it appears that 52429222 bytes were sent, so the throughput in bytes/sec for airplane is **31308.780846 bytes/sec.**

Losses

4. Did the connection experience losses? Did these losses slow down the connection or were there enough packets in the queue to absorb the losses? You should be able to answer these questions by interpreting the sequence number plot, looking for periods of slow start, idle periods, .. etc. Compare your different traces.
5. If there were any losses, did they occur when the window was large or did the size of the window have little effect? We don't expect a statistical analysis but a visual qualitative interpretation of your plots is sufficient.

Comcast

The next trace group I analyzed was the Comcast XFinity trace group. This pcap file seemed to be missing the initial handshake SYN packet sent by the file receiver.

(1) The client (the file receiver) is host 192.168.5.162 on port 55868. The server (the file sender) is host 128.8.126.92 on port 80.

(2) The file sender sent 37266 packets. The receiver sent 19436 ACKs.
Time elapsed = (1348895351670672 / 1000000) seconds, first ack sent at 1348895319.643566 seconds.
Total time elapsed is 32.0271060467 seconds, with 52427737 bytes gives a throughput of **1636980.15436 bytes/sec**.
This was. as expected, immensely faster than the airplane connection and with fewer duplicate ACKs.

Losses

3. Based on the metrics above, compare the performance of the trace groups from which you selected the traces.
4. Did the connection experience losses? Did these losses slow down the connection or were there enough packets in the queue to absorb the losses? You should be able to answer these questions by interpreting the sequence number plot, looking for periods of slow start, idle periods, .. etc. Compare your different traces.

5. If there were any losses, did they occur when the window was large or did the size of the window have little effect? We don't expect a statistical analysis but a visual qualitative interpretation of your plots is sufficient.

VZW

The third trace group I chose to look at was the Verizon Wireless cellular network trace. I had to generalize my checks for Ethernet headers on this one because of a PPP datalink type. This capture was also incomplete, ending in an unacknowledged data packet from the server (although the other captures probably were incomplete as well).

(1) This trace was a request from 10.175.200.34 on port 57793 (the file receiver) to: 128.8.126.92 on port 80 (the file sender).

(2) The server sent 36209 packets. The receiver sent 19505 ACKs.
(1348897603351386 nanoseconds/1000000) - 1348897580.404765 seconds = 22.9466209412 seconds, 52248185 bytes
=> **2276944.61567** bytes/sec. This was faster than the airplane satellite connection, and surprisingly, much faster than the Comcast XFinity connection.