# CIS 549: Wireless Communications for Mobile Networks and Internet of Things

# **Project #2: Network Packet Manipulation and Packet Trace Analysis**

Instructor: Bongho Kim (kimbong@seas.upenn.edu)

The goal of this project is for you to be able to write your own packet analysis tool, and be able to manipulate the packets you capture with it.

You will need to download the “Project2\_package.zip” file from Coursera. This zip file includes a packet trace file and the source code (pcap-analysis.c) for Exercise #1, Exercise #2, and Exercise #3.

# **What to Submit** Submit one zip file containing the **source code (pcap-analysis.c)** for Problems #1 and #2, and the output files for Problems #1, #2, #3, and #4 as instructed in each problem below and **an analysis report in a PDF format** for Problem #4. Create a folder named “Prj2”, add all files listed above for the submission. Zip this folder and submit the “Prj2.zip” file.

**Do not change** the file name “pcap-analysis.c”. You need to update this file to answer the problems in this project, maintaining the same file name. The **execution file name must be** “pcap-analysis.”

The source code “pcap-analysis.c” contains the code for Exercises 1-3 that will allow you to review and understand how to parse the pcap file, and complete Problems #1 and #2.

Exercises 1-3 are not a part of submission. Refer to the codes used in these exercises; it would be enough to parse other protocol header fields with the ethernet and IP protocol header structure information taught in the class.

**Note:** Problem #3 and #4 are not programming problems, so there are no source code files or execution files to submit for these problems.

**Tip:** How to compile the source file, pcap-analysis.c. Type the following command line in the terminal under the directory where the source file is located.

|  |
| --- |
| $ gcc pcap-analysis.c  -o pcap-analysis -lm |

**Contents of the sample trace file:**

* It includes multiple PING tests.
* There are four servers (10.168.207.106, 10.168.207.107, 10.168.207.108, and 10.168.207.109), and one client (192.11.68.196). The client downloads the same set of files from all four servers multiple times. The download file sizes are around 100KB, 500KB, 1MB and 2 MB.
* **Note:** Reading the source code for the Exercises 1-3 will help you to parse the packet capture file.

**Exercise #1** (Source Code is Given)

Extract “ping” delay measurement results from the provided packet trace file, and review the source code.

* **Format:**   
  prog\_name option\_selection input\_pcap\_filename
* **Command line:**

|  |
| --- |
| $ ./pcap-analysis ping-delay prj2-Input.pcap |

**Exercise #2** (Source Code is Given)

The provided pcap file has incorrect packet length recorded. You need to correct the packet length in the pcap file and regenerate a new pcap file using the provided code. You must review the source code to understand how to handle the pcap file.

* **Format:**   
  prog\_name option\_selection input\_pcap\_filename  output\_pcap\_filename
* **Command line:**

|  |
| --- |
| $ ./pcap-analysis fix-length prj2-Input.pcap lengthFixed.pcap |

This output file (lengthFixed.pcap) will be used for Problem #1.

**Exercise #3** (Source Code is Given)

Change the IP address from “192.11.68.196” to “192.11.68.1”

* **Format:**   
  prog\_name option\_selection input\_pcap\_filename  output\_pcap\_filename
* **Command line:**

|  |
| --- |
| $ ./pcap-analysis ip-address-change  lengthFixed.pcap ipChanged.pcap |

**Problem #1**

Modify the port number for the host IP address of “192.11.68.196” in the provided packet trace (lengthFixed.pcap). This pcap file contains multiple TCP sessions. You may check with Wireshark to see how many TCP sessions are in the pcap file. Your code should be able to find the individual TCP session and replace the port number used by the host (192.11.68.196) with a port number from 5000, and increment by 1 per TCP session. In the IP packet header, check the “Protocol Type” field. If the value is “6” then the IP packet payload is a TCP packet.

You will need to implement a function to do this job using the example code. The program should be executable with the following command in a terminal. After the replacement, you may double check using Wireshark again if port number has been correctly modified.

* **Format**:   
  prog\_name option\_selection  input\_pcap\_filename output\_pcap\_filename

The “option\_selection” must be “tcp-port-change“ as shown in the command line below. The validation will be done using the

same command below. So **make sure** your submitted files can run with this command.

* **Command line:**

|  |
| --- |
| $ ./pcap-analysis tcp-port-change lengthFixed.pcap  portModified.pacp |

**Submit the output pcap file, portModified.pcap.**  The output pcap file name must be “portModified.pcap”

**Problem #2**

Analyze the individual TCP sessions and find the file download time. Do not include any TCP session transmitting less than 100 Kbytes total if exist.

**Make sure** this command line works with your submitted files. Make sure that the file names are **exactly** **the same** as mentioned in the command line below to ensure correct evaluation of your submission.

* **Command line:**

|  |
| --- |
| $ ./pcap-analysis tcp-analysis  lengthFixed.pcap tcpAnalysis.txt |

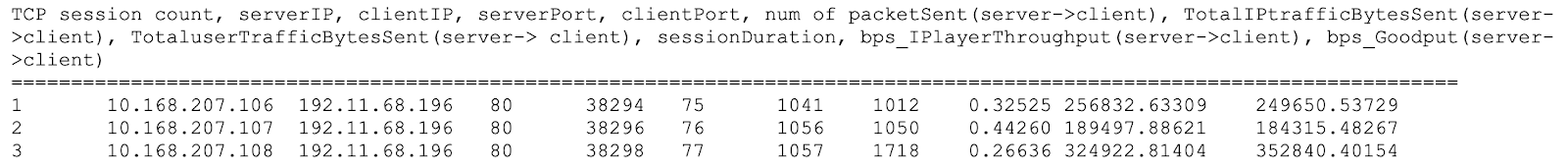
The tcpAnalysis.txt file format per row is the following:

* The first line of the file should be the titles (same as the line below this list)
* The second line onwards it displays all TCP session information i.e. one TCP session per line (same as the screenshot after the titles)

TCP session count, serverIP, clientIP, serverPort, clientPort, number of packetSent(server-> client),

TotalIPtrafficBytesSent(server-> client), TotaluserTrafficBytesSent(server-> client), sessionDuration,

bps\_IPlayerThroughput(server-> client), bps\_Goodput(server-> client)

****

**Note:**

* For a TCP session duration in this project, you may use any SYN packet (within the same TCP session) as a TCP session starting point and any FIN packet (within the same TCP session) as a TCP session ending point. All measurements (i.e. number of packets per TCP session) is in between your TCP session starting packets and the ending packets.
* bps\_IPlayerThroughput = (total transmitted byte count including IP header in a TCP session \*8) / (TCP session duration)
* bps\_ Goodput = (total transmitted user byte count excluding all protocol header in a TCP session\*8) / (TCP session duration)

**Submit the output text file, tcpAnalysis.txt, and the source code (pcap-analysis.c).**

**Problem #3**

Analyze the TCP packet trace file (use lengthFixed.pcap ) that you generated from Exercise #2. It contains multiple TCP sessions. The range of the downloaded file sizes are from around 100KB - 2 Mbytes. You should ignore the TCP sessions showing less than 100 KB size in total transmission if they exist.

Complete the analysis for all targeted TCP sessions and **report the following information:**

* **Download time (sec)**
* **Downloaded File size (Bytes)**
* **Throughput**
* **Packet loss rate**
* **RTT (round trip time)**

You may use the “tcptrace” tool for the analysis and you may want to build a simple script tool for text filtering to make your work a bit easier. You may use a text editor and manually copy and paste the desired information. Manual work is not recommended for the sake of time.

* **Command line for tcptrace:**

|  |
| --- |
| $ tcptrace  -lWr lengthFixed.pcap > tcptrace-out.txt |

**Problem #4**

Now, find the following two TCP sessions, and **analyze and compare** the TCP performance in detail including TCP sequence graph and other supporting materials.

* TCP Session #1: Client: 192.11.68.196:38341 (port number), Server #1: 10.168.207.108:80
* TCP Session #2: Client: 192.11.68.196:38344 (port number), Server #2: 10.168.207.109:80

Both tools, tcptrace and Wireshark, will be helpful for this type of analysis.

**Submit the TCP session analysis and the two TCP session comparison reports.** Your analysis and the interpretation of the outputs from the tools are an essential part of this problem. Simply adding the output from the tools is not considered an answer.