CMPE 570/670 Exercise #2

Encapsulation and Packets Structure

Due Tuesday 12/11/2019

On myCourses you will find a zip file that contains the skeleton of a C-coded program to read and analyze a file of type "pcap". Files of type "pcap" store the packets captured from actual network traffic. The files start with a "pcap" file header, which includes some information about the file and the traffic capture process, and follows with the packets that have been captured from some network interface. Each packet in the pcap file starts with a header that conveys information from the Physical layer (essentially, a time stamp for the packet and the number of bytes in the packet as transmitted through the Physical layer) followed by the actual packets that have been encapsulated as they are built by passing through the different layers of the protocol stack. All headers and other information about the packets are defined in the `encap.h` header file inside the provided zip file.

In this exercise, you will analyze a stream of packets of different types to learn about their structure. To do this, you will use packet analyzer software and you will write your own code (that you will derive from the provided code skeleton). The packet analyzer software that is recommended to be used is "Wireshark". You could also use "WireEdit" but be aware that this software presents information in a more summarized format than Wireshark.

Throughout this exercise your input pcap file will be "exercise2.pcap", also included in the provided zip file.

What you need to do: answer all the following questions (except 2a, which is just a set of instructions to get code compiled and running) and create a report in pdf format. The report shall contain the answers to the questions, code that you have created and the output from the code that you have run. Please do not include all your code and the resulting output, but rather use common sense to submit only the sections that are relevant to the question you are answering.

1) Open the file exercise2.pcap with the packet analyzer software and answer the following questions:

    a) (10 pts) By examining the information in the link layer header, indicate whether the network interface from where traffic was captured was wireless (a wifi interface) or wired (an Ethernet interface).

    b) (10 pts) Create a table where you will list all the different types of packets/protocols you find in the pcap file. For each entry in the table, a second column has to indicate what are the layers that are being encapsulated in each packet type. Hint: the "Protocol" column in the Wireshark window that shows all packets (the top one) does not provide enough information to answer this question (and it may be at times misleading); a better approach is to analyze the information in the middle window (the "Packet Details" window).

c) (10 pts) Examine the ARP (Address Resolution Protocol) exchange in packets 28 and 29 and explain what is happening with this exchange. Also, indicate whether the source and destination addresses are MAC or IP addresses and explain why.

d) (20 pts) Most of the packets in the pcap file correspond to a TCP connection where a file is sent from one host to another (the file exchange is NOT using FTP, SFTP or any other standardized file exchange protocol). Identify the packets from this connection (by their packet number) and indicate the IPv4 addresses for the two hosts involved in the exchange. Use the TCP state transition diagram to record the setup and ending of the connection. For this, indicate on a state transition diagram the packet number that corresponds to each transition (use the provided powerpoint file for this). Use this work to identify next the IPv4 address for the host sending the file (the one starting the connection) and the host receiving the file.

2) The second part of this exercise involves creating your own code to parse through the packets (doing "de-encapsulating") and extract information. Note: you are free to choose the system where to work (MS Windows, Mac, Ubuntu, etc.) but keep in mind that in some cases you may need to swap the endianness of data you are reading. The provided code includes two functions to do this if needed.

a) Your first task is to compile and run the skeleton code as provided. To compile the code in a MS Windows environment, I strongly recommend the use of MinGW. Type "`gcc skeletoncode.c -o skeletoncode`" and run the code doing "`skeletoncode exercise1.pcap`". You should find a new file "`outdata.txt`", with content like the following:

```
packet: 24, type: 8
packet: 30, type: 8
packet: 31, type: 8
packet: 32, type: 8
packet: 35, type: 8
```

…… same pattern follows for packets 36 through 84 ……

```
packet: 85, type: 8
packet: 86, type: 8
packet: 87, type: 8
```

b) (10 pts) Study the function "`int isgetTCPIP(BYTE *pcktbuf, u_int *size_ip, u_int *size_tcp)`"

- What is the information coded in the `struct sniff_ethernet`? What is the function and how does it work the line "`ethernet = (struct sniff_ethernet*)(pcktbuf);`"?
- Do you see within the same function other examples of the same type of programming (type casting a pointer)?
- Explain how this function modifies pointers that effectively allow for parsing through the packet, undoing the encapsulation process done during the packet transmission.

- Enumerate from which protocols is this function separating the headers and from which layers these protocols are.

c) (10 pts) Create a new version of the skeleton code which outputs to outdata.txt the following information for each TCP packet:
   - Packet number
   - Source IPv4 address and port number.
   - Destination IPv4 address and port number.
   - Sequence number.
   - Acknowledgement number.

Show the content of outdata.txt (Note: IPv4 addresses shall be in the standard format of four 8-bit decimal numbers separated by dots).

d) (20 pts) Consider next only the packets involved in the main TCP connection mentioned in question 1c) and of these packets only those that contain fragments of the file that is being transferred. Create a new version of the skeleton code which outputs the payload size in bytes and the sequence number. Do some further calculations with the sequence numbers to calculate the payload size of each packet and verify that the results equal the payload sizes obtained from your code that parses through exercise1.pcap. Explain the procedure that you followed and show your results. Also, use your results to calculate the size (in bytes) of the file that is transferred in the TCP connection. Hint: there are some fields in the packet headers that you can use to derive the payload size after compensating for some overhead lengths.

e) (10 pts) Same as in part d), consider next only the packets involved in the main TCP connection mentioned in question 1c) and of these packets only those that contain fragments of the file that is being transferred. Create a new version of the skeleton code that outputs to outdata.txt the file that is transferred. The exchanged file is a plain text file, so you should be able to open outdata.txt and read the file (it is OK if you see some very few odd characters). What is the subject of the file?