– A Simple Network Analyzer –
– Decoding Ethernet frames and IP datagrams –

Objectives

The main objective of this assignment is to gain basic understanding of network activities and network packet formats using an existing packet sniffer program (Wireshark). A packet sniffer puts your network interface card (NIC) in promiscuous mode and monitors the network traffic passing your computer.

What To Do:

1. Download and install the packet sniffer Wireshark (unless you've already done so) from

   http://www.wireshark.org/download.html

   Follow all directions and make sure to download the appropriate binary package for your machine. Each package comes with the latest stable release of WinPcap, which is required for live packet capture. Installation should be straightforward. Using Wireshark is also fairly straightforward.

   Familiarize yourself with the Wireshark program by studying the relevant documentation available on the Internet at

   http://www.wireshark.org/docs/

2. Find out your IP address. To do this, type ipconfig /all at the command prompt in Windows, or ifconfig –a at the shell prompt in Unix. You will get a bunch of configuration information, one piece of which is your IP address.

3. Start a Wireshark capture session. Make sure you select a correct interface card. Type "ip.addr == your_ip and http" in the "Filter" box, where your_ip is the IP address of your PC. This will control the volume of the frames on your screen (more frames may be captured, but they are not displayed). Read the online manual for instructions if you want to try more advanced capture filters. Click Start.

4. Use your favorite browser and visit a few website to generate HTTP traffic. After capturing a few HTTP packets, stop the capture session.

5. Save the captured packets in a file (call it http.pcap, for example). To do so, select "File" then "Save as". In the window that opens, select Wireshark/ tcpdump/ …libpcap(*.pcap, …) format for the file type – this is the format that we will be decoding. Make sure to save the file in a folder where you can find it again. It is easy to save it in the wrong location if you do not pay attention.

6. Write a C or Java program that reads Ethernet frames from a libpcap file (such as the http.pcap file you saved in an earlier step) and prints out information from the Ethernet header of each

frame. If the Ethernet frame contains an IP packet (specified by Ethernet type field value 0x0800), then print the IP header (only the fields described below).

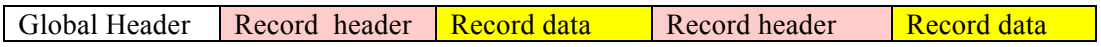
Your program should display the fields in the header as shown below:

```
ETHER: ----- Ether Header -----
ETHER:
ETHER: Packet 1
ETHER: Packet size = 210 bytes
ETHER: Destination MAC = 08:00:20:01:3d:94
ETHER: Source MAC = 08:00:69:01:5f:0e
ETHER: Ethertype = 0800 (IP)
ETHER:
IP: ----- IP Header -----
IP:
IP: Version = 4
IP: Header length = 20 bytes
IP: Time to live = 255 seconds/hops
IP: Source IP address = 128.10.26.116
IP: Destination IP address = 128.10.3.100

/* more similar records here */

/* Statistics */
Number of records processed: 112
Average packet length:     827 bytes
```

To write such a program, you will need to understand the format of a libpcap file. The libpcap file format is the main capture file format used in TcpDump/WinDump, Wireshark/TShark, snort, and many other networking tools. The file starts with a global header consisting of 24 bytes, which should be skipped. Then, a series of variable-length records follows. Each record starts with a 16 byte header containing information about the time the frame has been captured, the number of bytes in the frame that have been captured, and the actual number of bytes in the frame (the captured frame may have been cut short). The record header is followed by the Ethernet frame itself (that is, Ethernet header and Ethernet data). The file has the following layout:



In this assignment you will process the Record header and the Record data (which consists of the Ethernet header and the Ethernet data). The record header contains the following information:

RecHeader:

Timestamp	– an unsigned 64 bit integer
Captured data length	– an unsigned 32 bit integer
Frame length	– an unsigned 32 bit integer

- Timestamp is the time when the NIC driver sees the packet – ignore it
- Captured data length – you'll use this to determine where the next record starts
- Frame length is the length of Ethernet header + Ethernet data. Usually has the same value as the captured data length, however packet sniffers can be set to truncate your packets, case in which the two values differ.

The Ethernet header contains the following information:

EthHeader:

MAC address of destination	– on 6 unsigned bytes (48 bits)
MAC address of source	– on 6 unsigned bytes (48 bits)
The protocol type	– on 2 unsigned bytes (16 bits)

The IP header contains the following information:

IP_Header:

Version and header length	on 1 unsigned byte
Type of service	on 1 unsigned byte
Total packet length	on 2 unsigned bytes
Datagram identifier	on 2 unsigned bytes
Flags and fragment offset	on 2 unsigned bytes
Time to live (gateway hops)	on 1 unsigned byte
Type of protocol	on 1 unsigned byte
Header checksum	on 2 unsigned bytes
Source IP address	on 4 unsigned bytes
Destination IP address	on 4 unsigned bytes
(No options)	

Implementation Guidelines

The key to success in developing a program is to build in incrementally, in stages. I suggest that you go through the following implementation stages:

Stage 1. Write a program that does the following:

- open the input data file in binary
- skip the first 24 bytes
- read the first record header
- print out the length of the first record
- load the data file in Wireshark and compare the value printed by your program with the length value given by Wireshark – should be the same

Stage 2. Extend the program from Stage 1 to do the following:

- read or skip the rest of first the record (that is, the entire Ethernet frame)
- read the second record header
- print out the length of the second record
- load the data file in Wireshark and compare the value printed by your program with the length value given by Wireshark – should be the same

Once you make sure that you correctly jump to the second record in the file, go to Stage 3 below.

Stage 3. Modify the program from Stage 2 to do the following:

- open the input data file and skip the first 24 bytes
- in a loop (until end of file is encountered) do the following:

        - read the next record header
        - print out the length of the record
    - print out the number of records in the input file
    - compare the number of records displayed by your program against the number of records given by Wireshark

A successful implementation of Stage 3 ensures that you process the records in the input file properly.

Stage 4. Extend the program from Stage 3 to read each Ethernet header following a record header. For each Ethernet header, print out:

- the MAC address of the destination node
- the MAC address of the source node
- the protocol type

If the protocol type is 0x0800 (indicating an IP datagram), then print out the following piece of information from the IP header:

- IP version
- the length of the IP header
- the Time to Live value
- the source IP address
- destination IP address

You will need to use bit operations (&, >>) to extract bits out of a byte. Examples:

b & 0x0F	value of the least significant half of b
(b >> 4) & 0x0F	value of the most significant half of b

When handling integer values, keep in mind that your computer stores them in one of two ways: *big-endian* or *little-endian*. Consider, for instance, the integer 91329, which is 00 01 64 C1 in hexadecimal. This could be stored as:

Big endian:

00	01	64	C1

Little endian:

C1	64	01	00

Each processor chooses its own format, for example Sun Spark uses big-endian and Intel Pentiums use little-endian.

Computer networks are big endian. This means that when little-endian computers (like Pentium based computers running Windows) are going to pass integers over the network, they need to convert them to big endian. Likewise, when they receive integer values over the network, they need to convert them back to their own native representation.

Integer fields that are longer than one byte must be converted from network byte order to host byte order.

Submission Guidelines

Hand in a copy of your source code along with a sample output and a short README file that explains any bugs and/or deviations in your program from the assignment specification. Include in your README file your answer to Step 5 of the section "Capturing, Saving and Inspecting Frames".

Grading

The majority of your grade for this assignment will depend upon how well your implementation works. I will run your program on a capture file created by myself. Your grade will be determined by:

- 90%: Correct Execution
- 10%: Program structure and Documentation

Use good indentation, meaningful variable names and helpful comments. Start early!

Have Fun!