PCAP File Analyzer

Input File: tcp-ecn-sample.pcap

Free & Open Source Library Used:

1. Jpcap (Java Library for Analysing, Capturing and Sending Network Packets)[http://netresearch.ics.uci.edu/kfujii/Jpcap/doc/index.html]

2. JFreeChart (Java Library for Charts, Graphs, Plots and Piecharts, etc.)[http://www.jfree.org/]

Programming Language: Java

Operaring System: Any OS(Linux, Windows, etc.) with Java Support.

Scope of Assignment:

The assignment is divided into **5 modules**:

1. PACKET INFORMATION Module:

It gives details about attributes like Packet Capture Date, Time and Length(Bytes).

2. ETHERNET PACKET INFORMATION Module:

It gives details about attributes like Ethernet Frame Type(ARP,IP,IPv6,LOOPBACK,PUP,REVARP,VLAN), Source and Destination MAC Address.

3. IP PACKET INFORMATION Module:

It gives details about attributes like IP Version, Type Of Service(TOS) Information, Length, ID, Fragmentation Information, TTL, Protocol & Source and Destination IP Address.

4. TCP PACKET INFORMATION Module:

It gives details about attributes like Source & Destination Ports, Sequence & Acknowledgement Numbers, Various flags, Window size.

5. HTTP HEADERS INFORMATION Module:

It shows the various attributes of HTTP header information.

Theory:

PCAP File Format

Source: http://wiki.wireshark.org/Development/LibpcapFileFormat/

Introduction:

This assignment uses the LIBPCAP Packet Format file as input and parses it using JPCAP the Java API for analyzing PCAP files.

File Format:

Global Header:

```
typedef struct pcap_hdr_s {
guint32 magic_number;   /* magic number */
guint16 version_major;  /* major version number */
guint16 version_minor;  /* minor version number */
gint32  thiszone;       /* GMT to local correction */
guint32 sigfigs;        /* accuracy of timestamps */
guint32 snaplen;        /* max length of captured packets, in octets */
guint32 network;        /* data link type */
} pcap_hdr_t;
```

magic_number: used to detect the file format itself and the byte ordering. The writing application writes 0xa1b2c3d4 with it's native byte ordering format into this field. The reading application will read either 0xa1b2c3d4 (identical) or 0xd4c3b2a1 (swapped). If the reading application reads the swapped 0xd4c3b2a1 value, it knows that all the following fields will have to be swapped too.

version_major, **version_minor**: the version number of this file format (current version is 2.4).

thiszone: the correction time in seconds between GMT (UTC) and the local timezone of the following packet header timestamps.

sigfigs: in theory, the accuracy of time stamps in the capture; in practice, all tools set it to 0.

snaplen: the "snapshot length" for the capture (typically 65535 or even more, but might be limited by the user).

network: link-layer header type, specifying the type of headers at the beginning of the packet (e.g. 1 for Ethernet, etc).

Packet Header:

```
typedef struct pcaprec_hdr_s {

        guint32 ts_sec;         /* timestamp seconds */

        guint32 ts_usec;        /* timestamp microseconds */

        guint32 incl_len;       /* number of octets of packet saved in file
                                */

        guint32 orig_len;       /* actual length of packet */

} pcaprec_hdr_t;
```

ts_sec: the date and time when this packet was captured. This value is in seconds since January 1, 1970 00:00:00 GMT; this is also known as a UN*X time_t. You can use the ANSI C *time()* function from *time.h* to get this value, but you might use a more optimized way to get this timestamp value. If this timestamp isn't based on GMT (UTC), use *thiszone* from the global header for adjustments.

ts_usec: the microseconds when this packet was captured, as an offset to *ts_sec*.

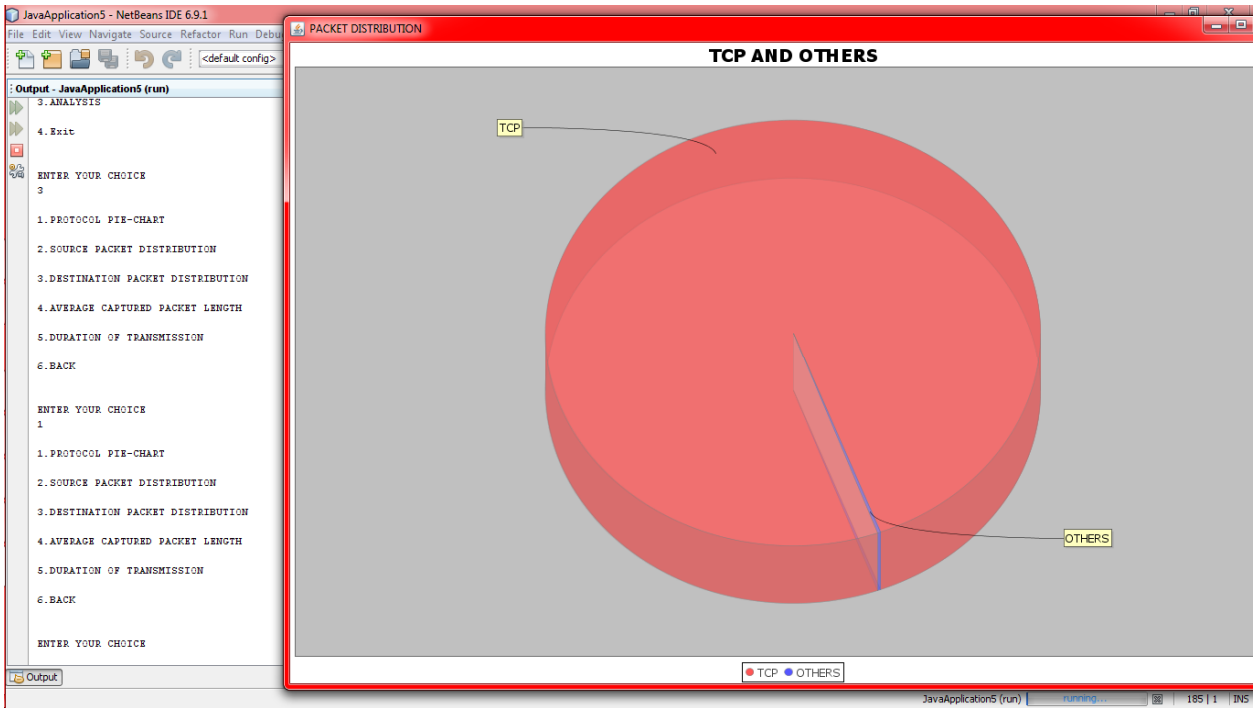
incl_len: the number of bytes of packet data actually captured and saved in the file. This value should never become larger than *orig_len* or the *snaplen* value of the global header.

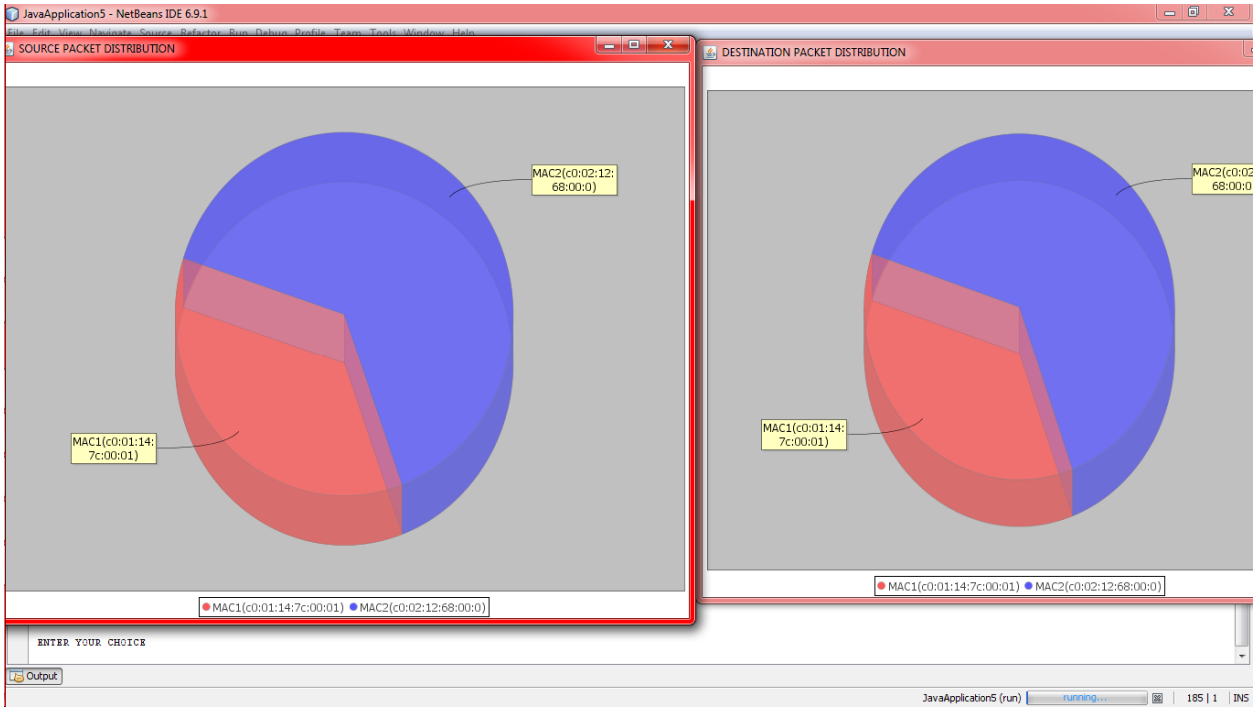
orig_len: the length of the packet as it appeared on the network when it was captured. If *incl_len* and *orig_len* differ, the actually saved packet size was limited by *snaplen*.

Packet Data:

The actual packet data will immediately follow the packet header as a data blob of *incl_len* bytes without a specific byte alignment.

Output:


JavaApplication5 - NetBeans IDE 6.9.1
Output - JavaApplication5 (run)
3.ANALYSIS
4.Exit
ENTER YOUR CHOICE
3
1.PROTOCOL PIE-CHART
2.SOURCE PACKET DISTRIBUTION
3.DESTINATION PACKET DISTRIBUTION
4.AVERAGE CAPTURED PACKET LENGTH
5.DURATION OF TRANSMISSION
6.BACK
ENTER YOUR CHOICE
1
1.PROTOCOL PIE-CHART
2.SOURCE PACKET DISTRIBUTION
3.DESTINATION PACKET DISTRIBUTION
4.AVERAGE CAPTURED PACKET LENGTH
5.DURATION OF TRANSMISSION
6.BACK
ENTER YOUR CHOICE
PACKET DISTRIBUTION
TCP AND OTHERS
TCP
OTHERS
TCP OTHERS


JavaApplication5 - NetBeans IDE 6.9.1
SOURCE PACKET DISTRIBUTION
MAC2(c0:02:12:68:00:0)
MAC1(c0:01:14:7c:00:01)
MAC1(c0:01:14:7c:00:01) MAC2(c0:02:12:68:00:0)
DESTINATION PACKET DISTRIBUTION
MAC1(c0:01:14:7c:00:01)
MAC1(c0:01:14:7c:00:01) MAC2(c0:02:12:68:00:0)
ENTER YOUR CHOICE