Malaviya National Institute of Technology Jaipur
Computer Networks
Lab Tasks#4

Problem 1: To understand the Network layer protocol i.e. IP in detail, capture data packets in promiscuous mode by using Ethereal from the URLs (e.g. http://www.bbc.com, http://www.rediff.com, http://www.yahoo.com etc.). Apply appropriate filter to visualize only the IP segments in your capture. Capture some decent amount of packets by clicking various links on the index pages of the URLs to observe the results effectively. Answer the following questions:

a. How many bytes are in the IP header? How many bytes are in the payload *of the IP datagram*? Explain how you determined the number of payload bytes.
b. Select the first ICMP Echo Request message sent by your computer, and expand the Internet Protocol part of the packet in the packet details window. What is the IP address of your computer?
c. Within the IP packet header, what is the value in the upper layer protocol field?
d. Describe the pattern you see in the values in the Identification field of the IP datagram
e. How many IP addresses did you observe in your trace? Don't count manually. Look in Statistics menu to find it.
f. Identify, analyze and verify all the header information contained in IP datagrams? Is it in accordance to whatever discussed/given in your text book (You may refer appropriate RFC's)?
g. Did you observe any datagram fragmentation in the trace? If yes; how long is this IP datagram? How many fragmentations have been done for that datagram? Identify the first and last fragment of the fragmented IP datagram.
h. What information in the IP header indicates that the datagram has been fragmented?

Problem 2: Design and implement an *online cash register system* using TCP socket APIs. In this system client implements a simple cash register that opens a session with the server and then supplies a sequence of codes for products. The server returns the price of each one, and also keeps a running total of purchases for each client. When the client closes the session, the server returns the total cost of purchases. Server also maintains grand total cost of the purchases made so far which administrator can know by giving DISPLAY command to the server**.**

The format of the request string sent by the client is as follows:

```
<Request type> <Product-code> <Number>
```

where

- The <request type> is either 0 for item or 1 for close.
- <Product-code> is a 6-digit product code.
- <Number> is the number of items being purchased.

<Product-code> and <Number> field is meaningful only if the <request type> is 0.

Examples of client request are:

0 104204 50 → requesting cost of 50 items of product 104204
0 326788 10 → requesting cost of 10 items of product 326788
1 00000 0 → close
1 10982 9 → close
2 00910 1 → invalid

The server maintains a database containing one table with three fields. The first field is the PC (Product Code), the second field is the name of the product (assumes that it is at most 26 characters and has no white space), and the price per unit of the product, represented as an integer. Your sample database could be a simple file, which you can use for testing. For example:

104204	HPLaserJet	800
326788	Pendrive	960
723272	IBMThinkpad600E	2000
134500	HPLaserJet	700

For the close command, the server returns an integer, which is the total cost of the transaction. For the item command, the server returns:

<Response-type> <response>

Where:

- <response-type> is 0 for OK and 1 for error
- if OK, then <response> is as follows:
  - if client command was "close", then <response> contains the total
  - if client command was "item", then <response> is of the form <price> <name> where
    - <price> is the price of the requested item
    - <name> is the name of the requested item
  - if error, then <response> is as follows:
    - a null terminated string containing the error; the only possible errors are a "Protocol Error" or "PC not in database".

The two programs client and server will communicate with each other in a connection-oriented (stream) manner using sockets. Your code should provide all suitable error checking and diagnostic messages.