CSC 450 Computer Networks

Programming Assignment 2
Transferring Files

Due: By midnight on **7 February 2014**

NOTE: Programming assignments are to be done in teams. Do not look at other group's code and do not allow other groups to look at your code. Doing so will result in charges of academic misconduct.

Description:
In this programming assignment you are to modify the two applications you created in programming assignment 1. The server will receive a request from the client, divide the requested file and send it via UDP packets back to the client. The client will receive these packets, subsequently logging them and printing receipt information.

Assignment Details:
The payload portion of each packet that your applications send must be of the form:

```
-----------------------------------------------------------
|    8 bit    |  32 bit   |    32 bit    | variable length |
| packet type | sequence  |    length    |     payload     |
-----------------------------------------------------------
```

Packet field details:

- Valid values for packet type are:
  - 'C' (uppercase C), meaning Client packet
  - 'S' (uppercase S), meaning Server packet
  - 'E' (uppercase E), meaning End packet
- The sequence number is unsigned and must be converted to network byte order while being placed in the packet.
  - The client fills the sequence number field with the number of the file that it is requesting. This can be 1, 2 or 3. The corresponding file names would then be `1.txt`, `2.txt` and `3.txt`.
  - For the server, the sequence number can start at any arbitrary value, as specified by the user in the parameters. The sequence value should increment with the number of payload bytes sent during an experiment. It should not include the 9 bytes required for the header in the packet layout shown above.

- The length field is unsigned and specifies the number of bytes carried in the payload of the packet. In the case of a client request (C packet type), the packet length should be 0.
- The payload data is a portion of the file that the client has requested. The server divides the file into payloads. The length field in its parameters determines the payload size (see below). The last payload can be smaller than the length parameter based on how many bytes are left. The length parameter will always be less than 50KB.

The client is invoked in the following way.

client -s <hostname> -g <server_port> -o <file_option>

hostname is the hostname of the server
server_port is the port on which the server is waiting
file_option is the number of the file that is being requested
It can be either 1, 2 or 3.

The client must print the following information for each packet that it receives, with each packet's information in a separate line:

The time at which the packet was received in millisecond granularity
The server's IP address (in decimal-dot notation)
The packet sequence number
The payload's length (in bytes)
The first 4 bytes of the payload

After the END packet is received, the client should print the following summary information about the experiment:

Total packets received from the Server
Total payload bytes received (which should add up to the file size)
The average packets per second
The duration of the experiment
This duration should start with the first packet received that contains payload and end with the END packet.

The client also should write the portions that it receives to a file with the same file name as it requested. This log file will be compared with the actual file that was sent out.

The server should be invoked in the following way:

server -p <port> -r <rate> -q <seq_no> -l <length>

port is the port on which the sender waits for requests
rate is the number of packets to be sent per second
seq_no is the initial sequence of the packet exchange
length is the length of the payload in the packets (each portion of the file)

The server must print the following information for each packet sent to the client, with each packet's information in a separate line.

- The time that the packet was sent with millisecond granularity
- The IP of the client
- The sequence number
- The first 4 bytes of the payload

Additional notes for the parameters:

- The server ports should be in the following range: 1024<*port*<65536
- For implementing the rate parameter, the sending interval should be evenly distributed. For example, when the rate is 10 packets per second the server has to send one packet at about every 100 milliseconds. The server must not send them all in a short time frame and wait for the remaining time in the second.
- Make sure that your applications are robust enough to handle user errors, including but not limited to invalid parameters and invalid input type. They should also handle parameters provided to them in any order. Any errors encountered should be handled gracefully with an error explaining problem and exit.

Deliverables:
Turn in one written report per group that describes your interpretation of the assignment, overview of your code, description of your experiments, presents a summary your results, and draws any relevant conclusions. Your report should be in the form of either a Microsoft Word or PDF file.

Also, turn in all source code that was used to implement you applications as well as a *makefile* for producing the executables. You applications should compile and run without errors.

Grading:
This programming assignment is worth 100 points and is distributed as follows:

Client	45 points
Server	45 points
Report	10 points