CPE464 – Stop and Wait Design (<u>only Design</u>)

This program specification goes with the stop and wait design assignment.

For this assignment you are to read the program specification below and turn in a design for this program. See the design requirements document for what you are to turn in. This will include packet flow diagrams and state diagrams. You may work on this in groups of 3 but you must provide a picture of you all working together (with a smile). **See the design assignment document for details on what you need to turn in for this assignment.**

You are NOT going to write code for this assignment – only design it. Don't worry I have a different program for you coming up.

This program you are designing is a remote file copy command using **UDP**. This requires both a client and a server. The first program is **rcopy** (the client program). This program takes two file names, a **remote-file** name and a **local-file** name. The second program is **server**, receives the filename (i.e. the remote-file name) from the rcopy program and then transmit the data from the remote-file back to rcopy. The rcopy program will receive the packets that make up the file and write them into the local-file.

Since UDP does not have any flow control or error recover, you will implement a stop and wait protocol to provide a reliable connection between the two programs.

A few more details:

1) I will provide a function that will randomly throw away packets. Both your server and client programs should call the sendtoErr() function instead of the normal sendto() function. This function should be used for all communications between the client and server. This function will cause the client and server to drop some of the data packets and flip some bits. The probability that the server will drop a packet/flip a bit is a run time argument (error-percent).

2) In order for the file transfer to complete successfully you will need to implement a stop-and-wait flow control algorithm. In this approach every data packet is acknowledged by the receiving program prior to the sending another packet. In order to recover from lost data you will set a timer (via the select() function). See below for more details.

3) To detect bit flips you will use the Internet Checksum algorithm. If an error is detected appropriate recover measures must be taken.

4) All packets must have the following format: Sequence number (4 byte), Internet Checksum (2 bytes), Flag (1 byte) and then any data (e.g. file data, filename or whatever payload you are trying to send).

Programs:

1) **rcopy**: This program is responsible for taking the two file names as command line arguments and communicating with the server program to request the remote-file. This rcopy program will then receive the file data from the server and store it to disk using the local-file name. The program will be run as:

   ```
   > rcopy  from-remote-file to-local-file buffer-size error-percent remote-machine remote-port
   ```

from-remote-file:	is the file on the remote machine to be copied
to-local-file:	is the file copied **into** on the local machine
buffer-size:	is the number of data bytes (from the file) transmitted in a packet
error-percent	is the percent of packets that will have errors (floating point number)
remote-machine:	is the remote machine running the server
remote-port:	is the port number of the server application

2) **server**: This program is responsible for accepting requests for files from the rcopy program and transmitting the file back. This program should never terminate (unless you kill it with a ctrl-c). It should continually process requests from client (rcopy) programs.

   The server should output its port number to be used by the rcopy program. The server program will be run as:

   ```
   > server error-percent [optional-port-number]
   ```

error-percent	is the percent of packets that will be dropped (floating point number)
optional-port-number	port number for the server to use (if not present the server should use a port number assigned by the operating system)

Requirements:

1) The buffer-size parameter should be sent from rcopy to the server prior to the file transfer. The server uses this parameter to determine how much data in the file to send in each packet.
2) The server needs to handle rcopy requests for non-existing files by sending back to the rcopy program a packet with a flag indicating this error.
3) When you send a **data** packet (a packet that contains data from the file) the sequence number MUST be in network order, the sequence number should start at 0 and grow every time a NEW packet is transmitted. Retransmitted data packets should have the same sequence number as the original packet.
4) In order to prevent the server program from blocking forever the select(…) function must be used. This function allows the server to "probe" the socket for a specified period of time. If select() times out the server should assume the data was lost and retransmit the packet. For this program, use a timeout period fixed at 1 second.
5) During the **filename** transfer (so sending of the filename to the server)
   a. If the client retransmits a filename packet 10 times (so you have 10 consecutive losses of the filename) the client can assume the server is unreachable and terminate with an error message "server unreachable".
   b. rcopy should use a 1-second timeout when sending the filename and waiting on the filename ACK.

6) Timing summary:

- You should always try to resend data/filename 10 times before quitting. Your program cannot terminate a file transfer because of only 1 lost packet.
- If a program (either rcopy/server) is waiting on something other than an RR/ACK//Filename confirmation (e.g. waiting on the next data packet), it should call select() with a timeout value of 10 seconds. If no data has arrived after 10 seconds, the program can assume the other side is down.
- The program in control (the one sending the information – filename or data) should call select() with a 1-second timeout. If the select() times out the program needs to go into whatever error control is appropriate (e.g. resending a data packet). This program should terminate if it calls select() 10 times and has not heard back from the other side.
- As stated above, the other program, the program waiting on something (e.g. waiting on data), should call select() with a 10-second timeout. If the 10-second select() expires the program can assume the other side terminated and clean up appropriately.
- For example: You call select() with a 1-second timeout after sending a data packet but use a 10-second timeout after sending an ACK. If the ACK is lost the data sender will recover from this error after 1-second and resend the data.
- Your program will be very hard to implement if at any time both sides are using a 1-second timeout.

7) You must use the following flag values in your header. If there are other types of packets you want to send (meaning you need other flag values) that is ok. Please document any additional flag values in your readme file. You must use the following value:

a.	Flag = 1	Data packet
b.	Flag = 2	ACK
c.	Flag = 3	Packet contains the file name/buffer size (rcopy to server)
d.	Flag = 4	File OK on server (server to rcopy)
e.	Flag = 5	File not found on server (server to rcopy)

8) If the remote file does not exist the rcopy program should print out an error message and terminate gracefully. The server should not terminate.

9) A run of these programs would look like:

```
On unix1:                   On unix2:
> server .01 54321          > rcopy myfile1 myfile2 1000 .01 unix1 54321
Server is using port 54321  Error myfile1 does not exist on unix1
                            > rcopy  myfile3 myfile2 1000 .01 unix1 54321
```