Programming Assignment #3 – Selective Reject
CPE464 – Fall 2011
Due Wednesday Nov 30, 2011

This program is due at 11:59 pm on **Wednesday Nov 30th**. If your program is late you will lose 10% per day. The last day you may work on this to receive any grade is Friday, December 2nd. **NOTE - There are only TWO (2) late days allowed with this assignment. If you do NOT turn it in by 11:59 on Friday Dec 2nd you will receive a zero**.

Extra Credit: If you turn in the program before 11:59 on Tuesday Nov 22nd you will receive 25% extra credit (You must get the multithreaded/multiprocess part of the program done to receive the extra credit.)

This assignment will continue to build on the rcopy and server programs. The major change is to implement a sliding window flow control algorithm using selective-reject ARQ. This program will use UDP and will use the Internet checksum function.

Similar to program #2, in order to induce errors you will **not** use the normal sendto(…) function. Instead you will use a sendtoErr(…) function. This is the same function you used in program #2. This function will not only drop packets but will periodically flip bits in your data You may **not** use the normal sendto(…) anywhere in either (rcopy/server) program.

In this assignment you will implement two programs in C (or C++). The first is rcopy. This program will take two file names, a remote-file name and a local-file name. The second program is the server, which will accept a request for a file from the rcopy program and transmit the file back. The client will then write the file into the local file name.

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

   rcopy remote-file local-file buffer-size error-percent window-size remote-machine remote-port

   where:

remote-file:	is the file on the remote machine to be copied
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 are in error (floating point number)
window-size	**is the size of the window in PACKETS**
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 the rcopy program.

   The server needs to handle error conditions such as non-existing files by passing back a flag to the rcopy program and then waiting for a new file request.

   The server will receive the **buffer-size** and the **window-size** from the client (rcopy).

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

server error-percent

where:

error-percent is the percent of packets that are in error (floating point number)

Requirements:

1) Do not use any code off the web or from other students. Do not look at any other code that provides a solution to this problem or parts of this problem. The work you turn in must be your entirely your own. You may make use of the code I handed out for the small socket lab earlier this quarter and the Internet Checksum code.

2) Both sides should call the sendtoErr(…) function (for data and ACK's). You should use the sendtoErr function for all transfers including the file name. You should **not** use the normal sendto(…) function in your program.

3) Your solution for sending the filename should not permanently hang a server process/thread nor should it cause rcopy to terminate just because of a few lost packets. Your solution needs to handle the loss of the filename and the loss of the filename ack (up to **10 times**).

4) In place of bind use my bindMod() function and in place of select() use my selectMod() function. See discussion in the functions.pdf document on blackboard.

5) If you resend a packet **10 times** you can assume the other side is down and terminate gracefully.

6) You must use a header in all of your packets. The header should have as a minimum a 32-bit sequence number, the internet checksum and a one byte flag field. So, your packet format for ALL packets should be seq#, checksum, flag, data/Filename/whatever

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 Normal Data packet
   b. Flag = 2 Resent **data** packet (means that seq# in this packet has been sent before and some type of error occurred. Only used for data packets. Resent filename exchange or other resent packets must use another flag.)
   c. Flag = 3 ACK packet
   d. Flag = 4 SREJ packet
   e. Flag = 6 Packet contains the file name
   f. Flag = 7 Packet contains the file name response
   g. Flags > 7 should be used for connection establishment (including the rest of the filename exchange) and connection termination/file finished or other conditions. (You decide how to use these.)

8) The command names, rcopy and server, should be used. Also the run-time parameters should be in the order given. Name your makefile: Makefile.

9) Use datagram sockets for these programs (UDP). They should be written in C (C++) and run on vogon.

10) If the remote file does not exist the rcopy program should print out an error message and terminate gracefully. The server should not terminate. It should loop waiting to process rcopy requests.

11) The only time your program should block (on the select) is when the window is closed (you have sent all the data you can.) Otherwise you should not block. Use a non-blocking select(…) (which means set your time value in your select call to 0 (zero)) to check for ACK's when the window is open.

12) You cannot buffer the entire file on either the server or rcopy. You can only maintain buffers approximately the size of your window. You cannot move back and forward in the disk file instead of buffering. The data must be buffered in memory.

13) You will not need to use the sliding window concepts for the filename exchange or after you send the last packet. Sliding windows only makes sense during the data exchange. For all packets you must use the sendtoErr (…) function.

14) Your server must be multithreading (pthreads) or multiprocessing (fork). It should create a new thread or process for every client. (You will lose 20% if this does not work.)

15) Summary
    a. You can **only** resend lost/corrupted data. You cannot resend data that was received correctly unless you need to recover from a timeout. Then you can resend a windows worth of data. Note - One lost packet should NOT result in a timeout.
    b. You must send a SREJ (NAK) on lost data (you cannot just wait until the other side times out. But you only need to send it once per lost packet – let your timeout recover from a lost SREJ.)
    c. Do **not** set timers for every packet. That would be a major pain.
    d. Your blocking select(..) may only time out if:
        i. All ACKs for a window are lost
        ii. or all data in a window is lost
        iii. or A SREJ packet is lost
        iv. (you are not required to timeout in these cases but you cannot timeout in any other situation)
    e. The server should process ACK's/SREJ (without blocking) after every send. In select() if you set the time value to 0 (zero) will check the socket for data but return immediately. See man select.
    a. On the server, when the window is closed you should do a blocking select for a time of 1-second.
    b. You cannot resend good data, if a packet arrives in the current window it is regarded as good data and should either be buffered or written to disk.

16) A run of these programs would look like:

On vogon:	**On vogon:**
> server .01	> rcopy myfile1 myfile2 1000 .01 10 vogon 1234
Server is using port 1234	Error myfile1 does not exist on vogon
	>rcopy myfile3 myfile2 1000 .01 10 vogon 1234
	>