COEN 146 - Winter 2018
Lab Project 4
TFv3 - Stop and Wait for an Unreliable Channel, with Loss

Pre Lab – Week 6, Bring a pseudo-code in paper, give to the TA (10%)
Final Lab – Week 7, demo in the lab and upload by midnight on Friday (90%)

This project consists of extending protocol TFv2 to enable it to deal with loss of messages.

TFv3 implements basically the protocol rdt3.0 presented in the text book. It consists of a client and a server. Communication is unidirectional, i.e., data flows from the client to the server.

The server starts first and waits for messages. The client starts the communication. Messages have sequence or ack number 0 or 1. Before sending each message, a checksum is calculated and added to the header. After sending each message, the client starts a timer. Use select for that. If select returns zero, there is no data, and the client needs to retransmit, restart the timer, and call select again. If select returns 1, there is data, so the client calls recvfrom to receive the ACK and then processes it. If ACK is not corrupted and the ack number is right, the client can now send one more message.

The server, after receiving a message, checks its checksum. If the message is correct and the seq number is right, the server sends an ACK message (according to the seq number) to the client, and the data is ready to be written in the file. Otherwise the server repeats the last ACK message and waits to receive a message again. The server does not change much from the previous lab.

To verify your protocol, use the result of a random function to decide to send or skip a message, to decide to send or skip an ACK message (only change to the server), and to decide whether to send the right checksum or just zero. This will fake the error and loss effect.

SELECT

This is an example on how to use select (check the man page for includes):

```
// local variables needed
struct  timeval tv;     // timer
int     rv;             // select returned value

// set it up, in the beginning of the function
fd_set    readfds;
fcntl (sock, F_SETFL, O_NONBLOCK);

…

// start before calling select
FD_ZERO (&readfds);
FD_SET (sock, &readfds);

// set the timer
tv.tv_sec = 10;
tv.tv_usec = 0;
```

```
// call select
rv = select (sock + 1, &readfds, NULL, NULL, &tv);       // sock is the socket you are using

if (rv == 0)
{
        // timeout, no data
}
else if (rv == 1)
{
        // there is data to be received
}
```

TRICKY SITUATION

The server closes the file and terminates execution after the message with zero bytes arrives. If the ack sent for that last message does not make it to the client, the client will keep resending it forever to a non-responding server. To avoid that, the client will start a counter after sending a message with zero bytes, and will only resend that last message 3 times. After 3 times, it will return to the main function.