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

Demo in the lab and upload by midnight on Friday (May 18)

This project consists of extending protocol TFv2 to enable it to deal with bit error and data loss.

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 (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 with the seq number to the client, and the data is ready to be written in the file. Otherwise (error detected by checksum or unexpected seq number) the server repeats the last ACK message and waits to receive a message again.

To verify your protocol, we will add unreliability (data loss and bit error) into the channel. 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.

MESSAGE FORMAT

```
typedef struct
{
    int     seq;   // sequence number for data and for Acknowledgement
    int     len;   // length of the data in byes (zero for ACKS)
    int     checksum;    // checksum calculated (by byte)
    int     fin;   // 1 for the last packet
} HEADER;

typedef struct
{
    HEADER header;
    char   data[SIZE];
} PACKET;
```

TERMINATION

For the last packet, sender sets fin = 1 to notify the receiver that the file is complete. The server closes the file and terminates execution after the message with fin=1 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 fin = 1, and will resend the last message at most 3 times. After 3 times, it will return to the main function.

CHECKSUM

The checksum must be calculated for messages from the server and client. To calculate the checksum, calculate the XOR off all the bytes (header + data) when member cksum is zero.

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
}
```