CS39006: Assignment 5 Emulating End-to-End Reliable Flow Control over Unreliable Communication Channels

Team Members:
Sampreeth R S (21CS30038)
Yash Kumar (21CS30059)

Data Structures used:

Send window:

Struct sndwnd:

int start: To store the seq no of the frame at the start of window
int end: To store the seq no of the frame at the end of window
int mid: To store the seq no of the frame till which ack has been received

Receive window:

Struct rcvwnd:

int next_expected: Next frame expected in order
int next_supplied: Next frame to be sent to the receiver

Socket handler:

Struct sh: Shared structure between the processes

int free: if the current struct is free or occupied
int pid: pid of the process whose information is stored
int sockfd: socket file descriptor
int mtpfd: mtp socket file descriptor
char sender_ip[16]: sender ip
int sender_port: sender port number
char receiver_ip[16]: receiver ip
int receiver_port: receiver port number
char sendbuf[10][1024]: buffer array for sending messages
char recvbuf[5][1024]: buffer for receiving messages
struct sndwnd sendwindow: sendwindow
struct rcvwnd receivewindow: receivewindow
struct tm timers[16]: used for timeout

int send_isfree[10]: array indicating whether a particular index in send buffer is free or not
int recv_isfree[5]: array indicating whether a particular index in receive buffer is free or not
int is_empty: no-space flag
int marked_deletion: whether marked for deletion
int is_bound: whether bind call has been made on the socket

Number of retransmissions taken:

The following table lists the number of retransmissions taken to transmit 30 messages with the various associated values of P

P	Retransmissions
0.05	7
0.1	17
0.15	15
0.2	18
0.25	26
0.3	27
0.35	24
0.4	51
0.45	57
0.5	55

Frame format:

The following header has been used along with the data to create the frame to send on the UDP socket:

- 8 bit Header Format:
  - LSB: Used to differentiate between ACK frame and data frame(set to 0 for data frame and 1 for ACK)
  - Bit 1 to 4: Sequence number.
  - Bit 6: Used in ACK frame as the no space flag

- ○ Bit 8: Used for special frame to probe whether receive buffer has been freed when the no-space flag has been set on the sender side and ACK message to clear the no-space flag is lost.
- ○ First Byte of the ACK frame is used to send the amount of empty space left in the receive buffer.

Data frame format:

Reserved	Sequence Number	0	Data
7	4	1	0

ACK frame format:

Reserved	Sequence Number	1	Empty Space
7	4	1	0

List of functions:

m_socket: find an free slot in the shared memory, create a socket for the calling process and enter the details of the socket into the free slot.

m_socket: mark the socket on which the call took place for deletion and return.

R: periodically check the sockets to receive data and process the data received.

S: periodically check the sendbuffer and send the data yet to be sent and timeout out data.

G: periodically wake up and clean the sockets marked for deletion and the sockets corresponding to dead processes.

m_sendto: for sending data over the socket using MTP. If the socket is valid and bound, the function allocates a buffer index for the outgoing message. It determines the sequence number for the message and prepares a buffer with the appropriate data. The function then inserts the message into the send buffer at the allocated index, along with its sequence number. Finally, the function returns either 0 if the operation was successful or -1 if an error occurred.

m_recvfrom: for receiving data over the socket using MTP. If the socket is valid, the function searches for an available buffer index in the receive buffer to fetch the incoming message. If a message is found in the receive buffer, the function retrieves the data from the buffer and copies it into the provided buffer. It also populates the client address information (cliaddr) with the sender's IP and port. After retrieving the message, the function returns either 0 if the operation was successful or -1 if an error occurred.