Networks Assignment 5 Report.

P value	Transmissions Needed	Message Send	Avg. no. of Transmission
0.1	38	15	2.53
0.15	39	15	2.6
0.2	49	15	3.27
0.25	97	15	6.47
0.3	76	15	5.06
0.4	83	15	5.53
0.45	87	15	5.8
0.5	170	15	11.3

NOTE*** The drop message uses probabilistic approach to drop a message and that probability is generated randomly.

Data Structures and Functions(briefly).

1. Defines

```
#define ACK 0   // Message type
#define MSG 1
#define MX_SEND_BUFFER 10 // Sender buffer size
#define MX_RECV_BUFFER 5  // Receiver buffer size
#define MX_SOCKETS 25 // MAX number of sockets
#define T 5 // Timeout in seconds
#define P 0.2 // Drop probability
#define MX_MSGS 10// Similar to MX_SEND_BUFFER (used, so don't have to type Long Name (MX_SEND_BUFFER)
```

2. Shared Memory

```
typedef struct {
   int is_allocated; // 0 for not allocated, 1 for allocated
   pid_t process_id; // Id of the Process which requested the socket
   int udp_socket_id;  // UDP socket id
   struct sockaddr_in other_end_addr; // Address of the other end
   char send_buffer[MX_SEND_BUFFER][1024]; // Adjusted for sender
buffer size
   char recv_buffer[MX_RECV_BUFFER][1024]; // Adjusted for receiver
buffer size
   time_t last_sent_time[MX_SEND_BUFFER]; // Time when each packet was
last sent

   struct {
       int size;
       int sequence_num[5]; // update
       int start_seq_num;//buffer start index
       int next_seq_num; // Next sequence number(index) to be used
       int w_s; //index of window start
       int w_e; //index of window end
       int acked[MX_SEND_BUFFER]; // Acknowledgment status for each
packet in the sender buffer
   } swnd;

   struct {
       int size; // size of buffer filled.
       int next_ind; //Next Free index in the receiver buffer
       int expect_seq_num; // Next expected sequence number for
in-order receipt
       int read_seq_num; // Sequence number of the next message to be
read by application
       int received[MX_RECV_BUFFER]; // Receipt status for each packet
in the receiver buffer
       int recv_seq_num[MX_RECV_BUFFER]; //NOt needed but can be used
to store the sequence number or receivec packets
   } rwnd;

} MTPSocketInfo;

typedef struct {
   MTPSocketInfo sockets[MX_SOCKETS];
} SharedMemorySegment;
```

3. MTP Packet

```
typedef struct {
   int type; // 0 for message, 1 for ACK
   int seq_num; // seq_num of the message
```

```
    int r_buff_size; // Buffer size of receiver (used only in case of
ACK)
    char data[1024 - sizeof(int) * 2];
} MTPPacket;
```

4. Sock_info

```
typedef struct{
    int sock_id; //Socket id
    char *IP; // IP address for binding
    int port; // POrt number for binding
    int errno_val; // Error number.
}SOCK_INFO;
```

5. `int m_socket(int,int,int);`

Used to create Socket and return the socket descriptor to the user file

6. `void initialize_all_variables();`

Used for initialization of all necessary semaphore, shared memory etc.

7. `int m_bind(int,char*, int,char*, int);`

Used to bind the socket to the IP address and port and also stores the IP and Port of the other end machine in the shared Memory

8. `int m_sendto(int, char*, size_t);`

Used to send the message, basically stores the message into the send buffer of shared memory of that socket

9. `int m_recvfrom(int, char*, size_t);`

Used to receive a message, return a message if it is present in the recv buffer of the shared memory.

10. `int drop_message();`

Returns 1 or 0 if random generated probability is less than P and vice versa

11. `void cleanup_resources()`

Use to clean up resources whenever SIGINT signal is invoked(ctrl + C)

12. `void *S_thread(void *arg)`

    Used to send the message over the socket

13. `void *R_thread(void *arg)`

    Used to receive the message, it listens on all the created socket

14. `void *G_thread(void* arg)`

    A garbage collector which closes the socket when the associated process is terminated or executed.

Gautam Kumar
Sanchit Yelwale