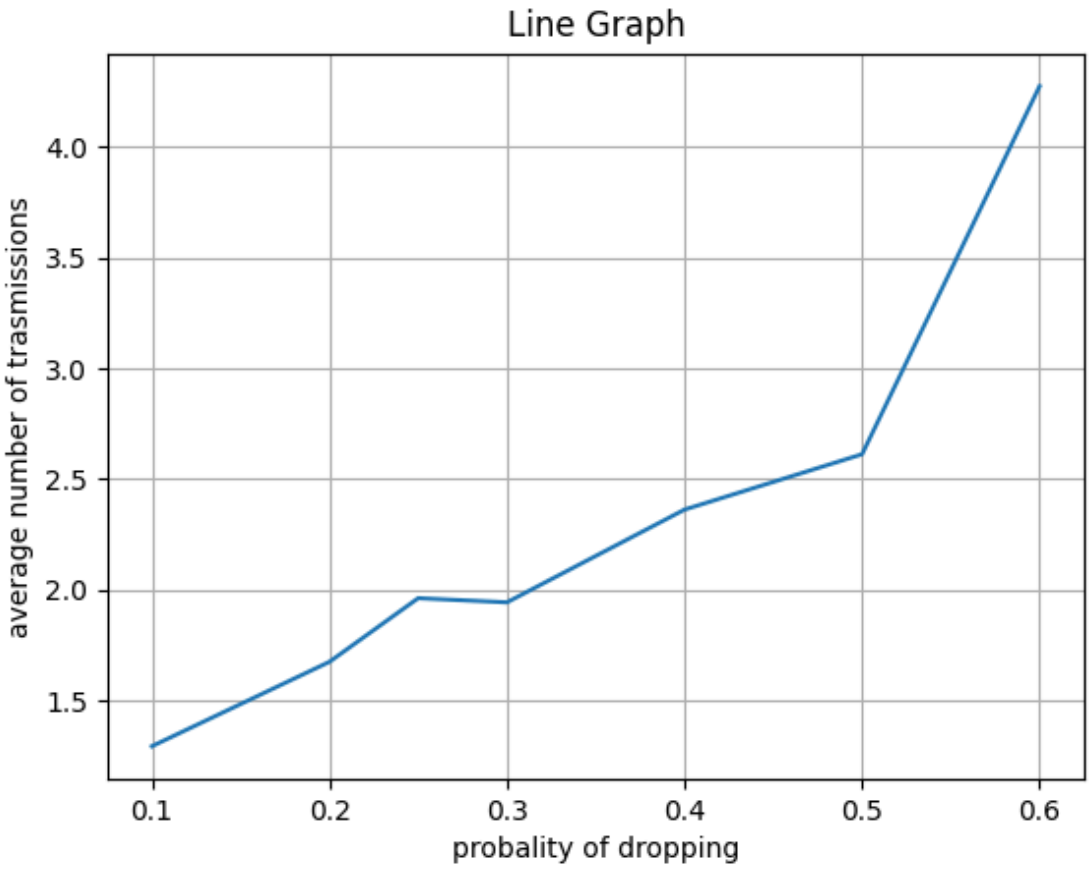
# NETWORKS LAB ASSIGNMENT REPORT

K.Hrushikesh Reddy(21CS30028)
Shivang Agrawal(21CS30048)

**The Average Number of Transmissions made for varying P :**

| Sr. No. | Probability to drop | Number of Messages Generated | Total number of transmissions | Average number of transmissions |
|---|---|---|---|---|
| 1 | 0.1 | 160 | 207 | 1.29 |
| 2 | 0.2 | 160 | 268 | 1.675 |
| 3 | 0.25 | 160 | 314 | 1.9625 |
| 4 | 0.3 | 160 | 325 | 2.03 |
| 5 | 0.4 | 160 | 378 | 2.36 |
| 6 | 0.5 | 160 | 418 | 2.6125 |
| 7 | 0.6 | 160 | 684 | 4.275 |

## Probability vs Average number of transmissions:




# Data Structures:

**1. sock_mem:The main data Structure which contains all the required fields for a single MTP socket.An array of this data structure is initialized in main() of initmsocket and shared with all the other threads.**

```
typedef struct
{
    int free; // boolean for whether or not this socket is free or not
    int pid; //Process ID of the process
    int sockfd;//UDP socket number.

    char selfip[20];
    int selfport;
```

```
    char destip[20];
    int destport;

    char send_buffer[10][1050];
    int send_len[10];
    int free_sndbuf;
    int last_snd, first_snd;

    char recieve_buffer[5][1050];
    int recieve_len[5];
    int nospace;
    int free_rcvbuf;
    int last_recv, first_recv;

    int ack_expected;
    int frame_expected;

    int time_wait;
    int next_frame; // to be allocated
    int dest_rwnd_size;
    int first_frame_sent; // this is the first frame number of the series
of frames sent latest.
    int last_ack_recv;

    int last_in_order;
    int rwnd[5];
    int last_expected;
    char temp_buf[5][1500];
    int temp_len[5];
    int next_recv;
    int size_recv;
    int ack_time; // last time ack was sent
} sock_mem;
```

**2. SEMS:A struct which stores the semids of all the semaphores used throughout initmsocket and msocket.**

```
typedef struct
{
```

```
    int semid1;
    int semid2;
    int semid3;
    struct sembuf pop, vop;

} SEMS;
```

**3. sock_info:A struct which stores the protocol values through which intimsocket is notified about the action desired on the given MTP socket.**

```
typedef struct
{
    int sockfd;
    int msockfd;
    char ip[20]; // src ip
    int port;
    int err_no;
} sock_info;
```

## Functions in msocket.c :

**1.m_socket(): Function in msocket.c to create an mtp socket which in turn creates an UDO socket in the main function of intimsocket.Returns mtp_sockfd on success.**

**2.m_bind():Function which specifies mtp_sockfd which is returned by the above m_socket() function and also ip address and port number of self and destination. Returns 0 on success.**

**3.m_sendto(): Takes mtp_sockfd,a char array which is the message to be sent,length of message ,destination ip address and destination ip port.This function writes the message into send buffer and S thread later sends them using UDP socket.Returns the number of Bytes written**

**4.m_recvfrom():Takes mtp_sockfd,a char array which is the message to be sent,length of message ,source ip address and source ip port.This function retrieves the message from the receive buffer. Returns the number of Bytes of message received.**

**5.m_close(): Implements the close system call for closing an MTP socket.**

## Functions in initmsocket.c :-

**1.R_main(): This is the thread function which is created by the main() in init_msocket and is responsible for getting messages received in the UDP sockets linked to each of the active MTP sockets and writing them into their designated Receive buffers.It is also responsible for sending acks whenever a stream of messages come based on the last in order message received.It is responsible for receiving the ACK messages and updating send window appropriately. It also sends ACKs whenever the Receive window size becomes non zero to notify the sender that it can start sending again.**

**2.S_main(): This is another thread created by main function.This checks the send window of each of the active MTP sockets and sends them to the receiver based on the rwnd_size notified by the receiver and sets the timeout value of socket to T.It then sleeps for a time of T/2 after which it wakes up and send the messages of timed out sockets.**

**3.g_main():This periodically(Period=5*T) checks the shared memory for sockets which are open but their corresponding application process has exited adn closes the sockets.**

**4.Exit_main():This function waits for keyboard interrupt to initmsocket application and upon receiving 0, cleans all the memories and exits the whole application.**

**5.sighandler():Created to do the same functionality as Exit_main but upon receiving Ctrl-C interrupt from keyboard.**

**6.drop_message():This function is invoked by any MTP socket which receives a message to deliberately try and drop it with some probability.**

**7.Main(): Initializes shared memory,creates UDP sockets and binds UDP socket on demand.**

## Additional Descriptions :

**1.Send Window is maintained using the variables**

- **i)next_frame which is the frame number to be allocated to the next message coming in mtp_send.**
- **ii)first_frame_sent:The starting frame number of the Send window.**
- **iii)Size of Send window is min(messages present in Send buffer,dest_rwnd_size,5)**

**2.Receive Window variables:**

- **i) last_in_order: it is the frame number of the last frame which was received in order**
- **ii)rwnd[5]: Array to store frame number of expected frames in window**
- **iii)last_expected: Frame number which is the last expected in window**
- **iv)next_recv: pointer to the position where the new frame number should be inserted while expanding the window**
- **v)size_recv: Current receive window size**
- **vi)ack_time: To store the last time the ack was sent**

**3. Macros in msocket.h**

- **i) T - Time out for S thread for sending frames**
- **ii) N - Maximum number of sockets**
- **iii) PR- Probability for dropping a received message**
- **iv) SOCK_MTP - value 3 ,only allowed socket type**

## Make file commands

**Run in the following order**

**1)make -f makelibmsocket.mak makelib**

**2)make -f makeinitmsocket.mak initmsocket**

**3)make -f makeuser.mak user1**

**4)make -f makeuser.mak user2**

**5)./user1 with appropriate arguments for running user1**

**6)./user2 with appropriate arguments for running user2**

**7)For cleaning:**

**make -f makeclean.mak clean**