Lab Assignment 4
GROUP 6: Sanyukta Deogade (19CS30016) and Soumita Hait (19CS10058)

My Reliable Protocol Socket (MRP)

Data structures used:

1) The received-message table is implemented using a circular queue.
   Fields of the circular queue –
   - MsgData* buffer: Each node in the circular queue containing the message received and other fields related to it.
     - char* msg: Stores the message.
     - int msg_len: Stores the length of the message.
     - struct sockaddr* source_addr: Local struct sockaddr which is to be filled in with the IP address and port of the originating machine.
     - socklen_t source_addr_len: Local int which is to be filled in with the length of the struct sockaddr.
   - int start: Front end of the queue from which elements are deleted.
   - int end: Rear end of the queue at which elements are entered.
   - int size: Number of elements currently contained in the queue.
   - int max_size: Maximum size of the queue.
   - pthread_mutex_t Lock: Mutex lock for the queue.
   - pthread_cond_t Empty: Condition variable that the queue is empty.
   - pthread_cond_t Full: Condition variable that the queue is full.
2) The unacknowledged-message table is implemented using an array.
   Fields of the unacknowledged-message table –
   - AckData** Table: The array storing all the unacknowledged messages and the time they were sent to the receiver.
     - char* Packet: Contains information about the data message sent.
     - int packet_len: Stores length of the packet.
     - int flags: This field is generally set to 0 for our purpose.
     - const struct sockaddr* dest_addr: Struct sockaddr which is to be filled in with the IP address and port of the destination machine.
     - socklen_t dest_addr_len: Length of the struct sockaddr.
     - struct timeval msg_time: Stores the time when the message was last sent.
   - int size: Stores the number of unacknowledged messages.
   - pthread_mutex_t Lock: Mutex lock for the table.
   - pthread_cond_t NonEmpty: Condition variable that the table is non-empty.

Lab Assignment 4
GROUP 6: Sanyukta Deogade (19CS30016) and Soumita Hait (19CS10058)

Functions used:

1) **int r_socket(int domain, int type, int protocol)**: Creates an MRP Socket and dynamically allocates memory to the received-message table and the unacknowledged-message table and creates threads R and S. Parameters same as the ***socket()*** function.
2) **int r_bind(int socketfd, const struct sockaddr *addr, socklen_t addr_len)**: Binds the socket port with an address port. Parameters same as the ***bind()*** function.
3) **int r_sendto(int socketfd, const void* Msg, size_t length, int flags, const struct sockaddr *dest_addr, socklen_t dest_addr_len)**: Sends the message with a type field (indicating whether it is a data message or an acknowledgement message), a sequence number and the data content of the message. It also adds the message sent to the unacknowledged-messages table along with the time field which contains the current time. Parameters same as the ***sendto()*** function for UDP Socket.
4) **int r_recvfrom(int socketfd, void* message, size_t length, int flags, struct sockaddr *source_addr, socklen_t *source_addr_len)**: Looks up the received-message table. If the table is empty, it blocks. Otherwise it returns the first message from the table and deletes it. Parameters same as the ***recvfrom()*** function for UDP Socket.
5) **int r_close(int socketfd)**: Closes the MRP Socket and frees the memory allocated dynamically and kills the threads R and S.
6) **int dropMessage(float p)**: Generates a random number between 0 and 1. Returns 1 if the number generated is less than some predefined quantity p, else returns 0.
7) **void* Thread_R(void* param)**: It is the function executed by the thread R. Here the socket file descriptor is passed as a parameter in the form of a void* pointer. It waits for receiving a message through the ***r_recvfrom()*** call. On receiving a message, it checks if it is a data message or an acknowledgement message. If it is a data message, it is added to the received-messages table and its corresponding acknowledgment is sent. If it is an acknowledgement message, then it removes the corresponding data message from the unacknowledged-messages table.
8) **void* Thread_S(void* param)**: It is the function executed by the thread S. Here the socket file descriptor is passed as a parameter in the form of a void* pointer. It periodically scans the unacknowledged-messages table and retransmits the data packets which have timed out. It also updates the last sent time of the message if it is re-sent.
9) **void GetMessage(MsgData* msg_Info, char* message)**: Copies the data message content from msg_Info to the char pointer message.

10) **void GetCurrentTime(struct timeval *time_struct)**: Fills the time_struct with the current time.
11) **void sigHandler(int sig)**: For signal handling.

Table for average number of transmissions made to send each character:

P	**29**	**48**	**34**	**Theoretical**	**Calculated**
0.05	31	52	38	1.108	1.090
0.1	37	55	40	1.234	1.201
0.15	39	67	43	1.384	1.335
0.2	45	75	54	1.562	1.567
0.25	52	89	61	1.778	1.813
0.3	60	101	65	2.041	2.028
0.35	64	108	75	2.367	2.221
0.4	82	130	94	2.778	2.767
0.45	92	147	110	3.306	3.157
0.5	118	197	136	4.000	4.057

P is changed in rsocket.h from 0.05 to 0.5 in steps of 0.05.
Theoretical value calculated by: 1/(1-P)^2.
Row corresponds to each value of P whereas column represents length of transmitted string.
Calculated value is the mean of (transmitted length/length of string).
Theoretical and calculated values of the average number of transmissions are almost the same. Some differences could occur because rand() in C is not a perfect random number generator.