COMPUTER NETWORKS ASSIGNMENT 4
HEMANT, 22CS30029

1.Results

Table 1: Average Transmissions per Message for Different Values of P

P	Average Transmission per message
0.05	1.000000
0.10	1.000021
0.15	1.072414
0.20	1.293103
0.25	1.257143
0.30	1.327273
0.35	1.492260
0.40	1.646664
0.45	1.695421
0.50	1.857143

2.Data Structures Used

2.1 ksocket.h

1. MessageHeader

```
typedef struct {
    char msg_type;
    int seq_no;
} MessageHeader;
```

- Represents the header of a message.
- msg_type: Indicates type ('D' for Data, 'A' for Acknowledgment, 'P' for Probe).
- seq_no: Stores the sequence number.

2. Message

```
typedef struct {
    MessageHeader header;
    char msg[MAX_MSG_SIZE];
} Message;
```

- Contains the message header and body.

3. Send Message Structure

```
typedef struct {
    int ack_no;
    time_t time;
    int sent;
    Message message;
} send_msg;
```

- Stores information about sent messages.
- ack_no: Stores acknowledgment number.
- sent: Flag for message transmission.

4. Receive Message Structure

```
typedef struct {
    int ack_no;
    char message[MAX_MSG_SIZE];
} recv_msg;
```

- Stores received message details.

5. Sender Window

```
typedef struct {
    int window_size;
    int window_start_index;
    int last_seq_no;
    send_msg send_buff[SENDER_MSG_BUFFER];
} swnd;
```

- Manages sender-side transmission.

6. Receiver Window

```
typedef struct {
    int window_size;
    int index_to_read;
    int next_seq_no;
    int index_to_write;
    int nospace;
    recv_msg recv_buff[RECEIVER_MSG_BUFFER];
} rwnd;
```

- Manages receiver-side message buffering.

7. Socket Entry

```
typedef struct {
    int socket_alloted;
    pid_t process_id;
    int udp_socket_id;
    struct sockaddr_in destination_addr;
    swnd send_window;
    rwnd recv_window;
} KTPSocketEntry;
```

- Stores socket-related details.

8. Socket Info

```
typedef struct {
    int sock_id;
    unsigned long IP;
    unsigned short port;
    int errno_val;
} SOCK_INFO;
```

- Manages socket metadata.

2.2 initksocket.c

1. Variables Accessed
- shmid_SM: Shared memory ID for KTPSocketEntry.
- shmid_sock_info: Shared memory ID for SOCK_INFO.
- Sem1, Sem2, SM_mutex: Semaphores for synchronization.
- num_messages, num_transmissions: Transmission statistics.

2. Data Structure Defined
- Persistence Timer

```
typedef struct {
    int flag;
    time_t last_time;
    int ack_seq_no;
} Persistence_Timer;
```

- Handles zero-window deadlock situations.

3.Functions

3.1 ksocket.c

1. cleanup: Frees shared memory and semaphores before exiting.

2. dropMessage: Simulates message loss based on probability P.
3. k_socket: Creates a KTP socket and assigns it an available slot.
4. k_bind: Binds a KTP socket to given source and destination addresses.
5. k_sendto: Sends a message, updating sender buffers and window size.
6. k_recvfrom: Receives a message, updating receiver buffers and window size.
7. k_close: Closes the KTP socket and marks its slot as available.

3.2 initksocket.c

1. cleanup_on_exit: Computes statistics and cleans up shared resources.
2. signal_handler: Handles termination signals.
3. R Thread: Listens for messages, updates receiver window, and sends ACKs.
4. S Thread: Manages message retransmission and timeout handling.
5. G Thread: Garbage collector for cleaning up closed sockets.

---

4 Conclusion

This document provides an overview of the implementation and results for KGP Transport Protocol (KTP). The protocol was successfully implemented using shared memory and semaphores, ensuring proper message transmission and reception. The results show the impact of message drop probability on transmission efficiency, validating the effectiveness of our approach.