KTP (KGP Transport Protocol) Project Documentation

Nakul Sharma, 22CS10046

1 Overview

This project implements the **KGP Transport Protocol (KTP)**, a reliable transport layer protocol over UDP. It ensures **message delivery, flow control, and retransmissions** using shared memory and multi-threading.

2 Data Structures

2.1 ktp_message (KTP Message Format)

Defines the structure of messages exchanged between sender and receiver.

```
typedef struct {
    uint8_t type;       // DATA_MSG or ACK_MSG
    uint8_t seq_num;    // Sequence number
    uint8_t ack_num;    // Acknowledgment number
    uint8_t rwnd_size; // Receiver window size
    char data[512];     // Data payload
} ktp_message;
```

2.2 window (Sliding Window for Flow Control)

Manages the sender's and receiver's window for reliable transmission.

```
typedef struct {
    uint8_t base;              // Base sequence number
    uint8_t next_seq_num;      // Next sequence number
    uint8_t window_size;       // Window size
    struct timeval send_time[10];
    ktp_message messages[10];
} window;
```

2.3 ktp_socket (KTP Socket State)

Represents a KTP socket stored in shared memory.

```
typedef struct {
    int is_allocated;          // 1 if allocated
    int udp_socket;            // Associated UDP socket
    struct sockaddr_in src_addr, dest_addr; // Addresses
    int is_bound;              // 1 if bound
    window swnd, rwnd;         // Sender and receiver windows
    pthread_mutex_t send_mutex, recv_mutex; // Mutexes for synchronization
} ktp_socket;
```

3 Functions (One-line Descriptions)

- `k_socket()`: Creates a new KTP socket.
- `k_bind()`: Binds a KTP socket to local and remote addresses.
- `k_sendto()`: Sends a message via KTP.
- `k_recvfrom()`: Receives a message via KTP.
- `k_close()`: Closes a KTP socket.
- `send_ack_message()`: Sends an acknowledgment message.
- `process_data_message()`: Processes received data messages.
- `process_ack_message()`: Handles received ACK messages.
- `dropMessage()`: Simulates packet loss.
- `receiver_thread()`: Handles incoming messages.
- `sender_thread()`: Manages message transmission and retransmissions.
- `create_shared_memory()`: Initializes shared memory for KTP sockets.
- `cleanup_handler()`: Cleans up resources when stopping the program.

4 How to Execute the Program

```
# Compile the program
make clean && make

# Start the KTP initialization process (keep this running in a terminal)
./initksocket

# Run the sender (user1) in a new terminal
./user1 127.0.0.1 5000 127.0.0.1 6000
# Enter the file name when prompted

# Run the receiver (user2) in another terminal
./user2 127.0.0.1 6000 127.0.0.1 5000
# Enter the name for the received file

# Verify the file transfer
cmp file_to_send.txt received.txt

# Stop the KTP system (press Ctrl+C in initksocket terminal)
```