GROUP ASSIGNMENT

Develop Chat Client and Server Applications which follow the guide lines provided.

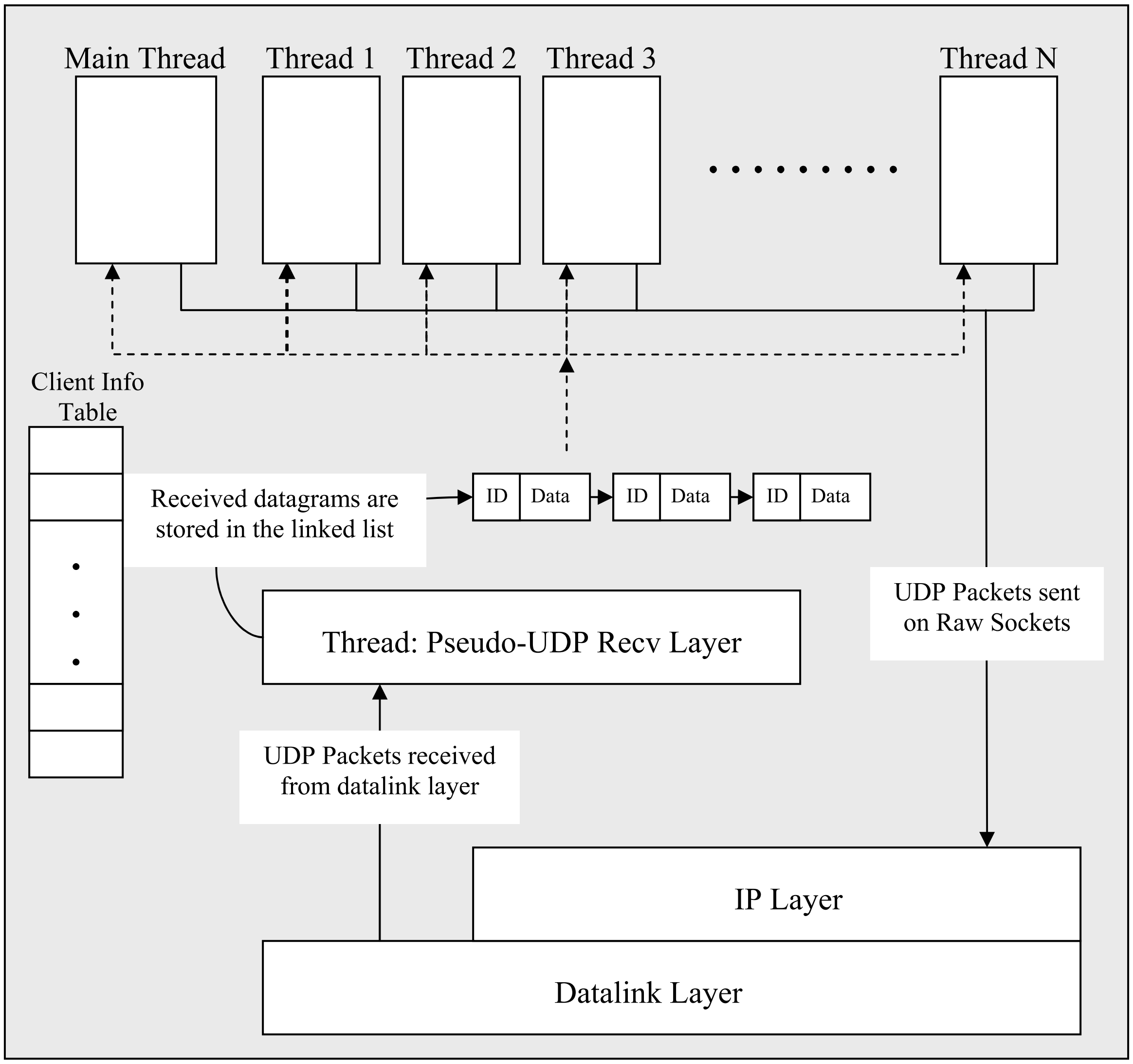
Main Thread
Thread 1
Thread 2
Thread 3
Thread N
Client Info Table
Received datagrams are stored in the linked list
ID
Data
UDP Packets sent on Raw Sockets
Thread: Pseudo-UDP Recv Layer
UDP Packets received from datalink layer
IP Layer
Datalink Layer

Figure 1: Block Diagram of Server Application

You will be using following in your server application:

1. Threads to handle each client and the Pseudo-UDP layer
2. Mutexes across the shared linked list and the user information table
3. Signal wait and broadcasting with linked list mutex. When the threads find the linked list empty or do not find a packet destined for them, they will go in signal wait state. When ever the Pseudo-UDP receiver layer will add a node in a linked list it will broadcast a signal in order to wake up any waiting threads.
4. Datalink sockets to receive the UDP packets for local IP and port 3333.
5. Raw Sockets to send your UDP packets

MESSAGE FORMAT:

```
|MESSAGE TYPE| |PAYLOAD LENGTH| |PAYLOAD VALUE|
```

- The *Message Type* (2-bytes short) for this applications is 0x0005
- The *Payload Length* (2-bytes short) will be the length of the Payload Value.
- The *Payload Value* is further subdivided into THREE fields:

  ```
  |Seq No| |Chat Msg Type| |Message|
  ```

  - Sequence number
  - Chat Message Type -> (2-bytes short) this will tell the server about the content of the message and what to do with it. There are four categories of Chat message types

    Destination User ID
    If the chat message type is some positive number, then this is peer2peer chat message. This type gives the ID of the user to whom this message should be sent.

General:

BROADCAST_MSG	0	// The chat payload message should be broadcasted to all the connected clients
ERROR	-1	// This type could be used to tell about any abnormal condition or error. The payload will contain the error message text.
QUIT	-2	// Any peer can send this message type to other before terminating the connection.

Client to Server:

CONNECT	-3	// Connection message
USER_NAME	-4	// The payload contains the user name
PASSWD	-5	// The payload contains the password for username send earlier
GET_USER_LIST	-6	// This type tells the server to send the list of connected usernames and their IDs

Server to Client:

USER_NOT_FOUND	-7	// The user is not registered. Server will terminate the connection after sending this message
USER_FOUND	-8	// User is registered. Client should now send registered password
PASSWD_INVALID	-9	// Password sent for the registered user is wrong. Server will terminate the connection after sending this message

ASSIGNED_ID	-10	// This packet contains the ID assigned to client
WELCOME_MSG	-11	// This message contains the welcome text from the server

- Chat Message Payload -> some text message; depending on the Chat Message Type.

CHAT SERVER:

1. When the application starts (main thread) it loads user list from the file and then launches a Pseudo-UDP receiver thread.
2. The Pseudo-UDP layer opens a datalink socket and starts receiving packets for UDP port number 3333.
3. When a packet is received, the Pseudo-UDP layer removes the datalink, IP and UDP headers.
4. Pseudo-UDP layer then uses the source IP address and Port number to look up the thread ID from the Client Info table. The thread pointed by this ID will be responsible to handle this packet.
    a. A new node in the linked list will be created and the data (in TLV format) will be copied to it.
    b. If information was present in the client info table then the destination thread id will be copied in this node.
    c. If information was not present in the client info table then the destination thread ID will be set to ‘0’.
    d. This Pseudo-UDP thread will also broadcast a signal in order to inform all the threads waiting for data in the linked list.
5. The main thread reads a node from the linked list with ThreadID equal to 0.
    a. After necessary reliability steps, the Chat Message Type is checked
    b. If the type is CONNECT, then a new entry in the Client Information table is made. The new thread will not be created for the client until it is authenticated (refer to assignment5 for the steps involved in client authentication).
    c. Once the username/password has been exchanged, the client is assigned a Chat ID and a new thread is created to handle this client.
    d. The main thread repeats these steps for each new packet
    e. At any stage if the main thread finds the linked list empty or finds no packet destined for it, it goes into signal wait state until the Pseudo-UDP layers broadcasts a signal after adding a new node in the linked list.
6. Each thread created by the main thread reads a packet from the linked list which has the thread ID equal to its own.
    a. The TLV header is parsed and the Chat Message Type is checked
    b. If the type is BROADCAST_MSG then the thread will send this packet to all the connected clients present in the client information table
    c. If the type is some positive number (Chat ID of a client) then this message will be forwarded only to the client specified by the Chat ID.

d. If the type is GET_USER_LIST then the thread will read the Usernames and IDs from the client information table and send it to the client

CHAT CLIENT:

1. The client opens a normal UDP socket and sends CONNECT message to the chat server using the IP address passed as command line argument
2. Conducts the login messages exchange
3. The client adds 'standard input' and 'connected socket' descriptor in the select() and starts waiting for one of them to become readable
4. If standard input becomes readable, the client read data from the user
    a. IF the first character typed is '/' character then this means that user wants some special options. A new menu should come up allowing the user to either get list of users from the server or send a message to some specific user (using the ID of that user)
        i. In case of getting the list, just send a message GET_USER_LIST to the server, the reply can be received in the main select()
        ii. Otherwise first read in the Chat ID of the destination user. Then read in the message to send. Send this to the server.
    b. ELSE send this message to the server with BROADCAST_MSG (to be broadcasted to all connected clients)
    c. Go back to select()
5. If UDP socket becomes readable, the client reads data from the socket and prints it on the screen
6. This process is continued until the user enters the escape sequence.
7. While sending/receiving data to/from socket all necessary processing for reliability must be done.