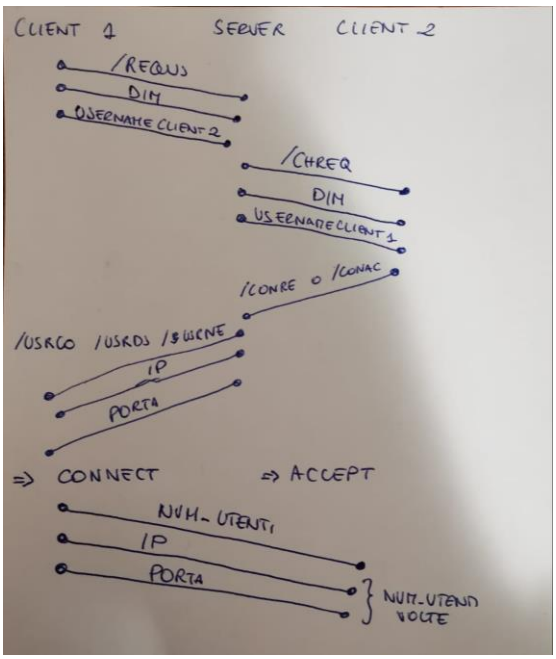
PROGETTO RETI INFORMATICHE 2021/2022

Il mio progetto fa uso di una serie di messaggi scambiati tra client e server, tutti di lunghezza prefissata, che permettono di gestire le varie richieste. Di seguito i messaggi utilizzati.

```
31  // comunicazioni col server e coi client
32  char REQUS[]="/REQUS\0"; // richiesta ip/porta user
33  char USRCO[]="/USRCO\0"; // user connesso
34  char USRNE[]="/USRNE\0"; // user non esistente
35  char USRDS[]="/USRDS\0"; // user disconnesso
36  char LOGIN[]="/LOGIN\0"; // richiesta di login
37  char SIGIN[]="/SIGIN\0"; // richiesta di signin
38  char CHREQ[]="/CHREQ\0"; // richiesta di chat
39  char CONRE[]="/CONRE\0"; // richiesta di chat refused
40  char CONAC[]="/CONAC\0"; // richiesta di chat accepted
41  char SERDO[]="/SERDO\0"; // server down
42  char CHCLS[]="/CHCLS\0"; // chat closed
43  char CHSRV[]="/CHSRV\0"; // notifico al server che vado in chat con lui perchè l'utente è disconnesso
44  char CHSRC[]="/CHSRC\0"; // notifico al server che ho chiuso la chat con lui
45  char CHHNG[]="/CHHNG\0"; // richiesta dei messaggi pendenti
46  char SHHNG[]="/SHHNG\0"; // show hanging
47  char NOTIF[]="/NOTIF\0"; // notifica che lo user ha letto i miei messaggi
48  char NOTSI[]="/NOTSI\0"; // l'user che ha fatto login ha dei messaggi pendenti
49  char NOTNO[]="/NOTNO\0"; // l'user che ha fatto login non ha messaggi pendenti
```

Il pregio nell'utilizzare questa scelta è che non c'è bisogno di prevedere un header per ogni messaggio inviato o di utilizzare formati particolari. Il difetto è che per ogni richiesta deve essere inviato un messaggio preventivo.

- IMPLEMENTAZIONE DELLA CHAT



Viene richiesto client2 con cui chattare. Client1 invia /REQUS al server, invia la dimensione del client2, e invia client2. Si pone in attesa di una risposta dal server. Il server (se l'username esiste ed è online, non in chat) invia /CHREQ a client2, invia la dimensione del client1 e client1. A questo punto se client2 accetta invia /CONAC al server, altrimenti invia /CONRE. Nel caso in cui invii /CONAC, il server invia a client1 /USRCO, ip e porta di client2. Client1 va in connect e client2 in accept. Quindi client1 invia a client2 il numero di utenti attualmente in chat, e per ognuno di essi invia ip e porta del socket di ascolto. Client2 farà la connect con ognuno di loro.

Se client2 è offline, o se client2 risponde con /CONRE, client1 riceve /USRDS dal server, e inizierà una chat con client disconnesso (ovvero i messaggi vengono inviati al server e bufferizzati).

Se client2 non esiste, il server risponde con /USRNE.

Se, una volta in chat, client1 prova ad aggiungere un altro client alla chat, viene seguito lo stesso pattern.

Se un utente entra in chat con client disconnesso, ogni volta invierà al server /CHSRV, dimensione del messaggio e messaggio. Il server registra tale messaggio in una struttura dati relativa a client2, organizzata come lista di messaggi pendenti (client1, timestamp_ultimo_messaggio, buffer dei messaggi, numero di messaggi, next). Quando termina, invia al server /CHSRC.

Il server sa sempre quali utenti siano online e quali siano offline, poiché aggiorna l'entrata della sua struttura dati nel caso in cui una recv da un utente ritorni 0. Il pregio è che ovviamente non c'è bisogno di ack, e il server non deve fare tentativi per inviare il messaggio. Inoltre, è client1 a inviare direttamente il messaggio a client2 nel caso in cui client2 abbia accettato l'entrata in chat. Anche la gestione di uscita improvvisa da parte di un client è notificata dalla recv che ritorna 0. Il server aggiorna le sue strutture dati, così come i client connessi, che usciranno dalla chat.

Il protocollo utilizzato è TCP in quanto l'applicazione si occupa di scambio di messaggi che devono arrivare in ordine ed essere corretti.

Nel caso in cui il server vada down, i client che erano in chat continuano a poter chattare con gli altri utenti. Nel momento in cui chiudono la chat, l'applicazione si chiude (in quanto non potrebbero comunque iniziare nuove chat e non potrebbero usare hanging o show).

- STRUTTURE DATI DEL SERVER

```
57  struct PENDENTI{
58      char username[MAX_USERNAME];
59      char timestamp[TIME_LENGTH];
60      char buffer[MAX_CRONOLOGIA];
61      int num_msg;
62      struct PENDENTI* next;
63  };
64
65  struct NOTIFICA{
66      char username[MAX_USERNAME];
67      struct NOTIFICA* next;
68  };
69
70  struct user{
71      char username[MAX_USERNAME];
72      char password[MAX_PASSWORD];
73      uint16_t porta;
74      char timestamp_login [TIME_LENGTH];
75      char timestamp_logout [TIME_LENGTH];
76      int sd;
77      int flag_chat;
78      struct PENDENTI* chat_pendenti;
79      struct NOTIFICA* notifiche_pendenti;
80  };
81  struct user USERS[MAX_USERS];
```

Il possiede un array di strutture user, contenenti username, password, porta, timestamp di login e logout, il socket descriptor, un flag (1 se l'utente è in chat con qualcuno, 0 altrimenti), una lista di chat pendenti (ogni elemento contiene username dell'utente che ha provato a scrivermi, timestamp dell'ultimo messaggio, un buffer contenente i messaggi inviati, il numero dei messaggi e un puntatore a next) e una lista di notifiche pendenti (nel caso in cui client1 scriva a client2 disconnesso, se client1 si disconnettesse e client2 visualizzasse i messaggi pendenti, la notifica da inviare a client1 viene mantenuta dal server in questa lista).

La mia applicazione fa uso dei thread. Il client ha 3 thread: uno è quello principale che si blocca in scanf quando l'utente si trova in chat; uno è sempre in comunicazione col server, e uno si occupa di fare le recv e printare ciò che gli altri utenti scrivono.

Se client1 si trova in chat con altri utenti, e client2 fa una richiesta per entrare in chat con client1, client2 riceve dal server /USRDS come se client2 fosse disconnesso: client2 entrerà in chat con user disconnesso e client1 riceverà una notifica che qualcuno ha provato a contattarlo. In seguito, client1 troverà i messaggi persi con show client2.

Se un client si disconnette da una chat di gruppo, a tutti i client viene inviata una notifica e la chat viene chiusa.