Documentazione Progetto

Alice Orlandini (578118)

DESCRIZIONE GENERALE

L'applicazione che fornisce il servizio di messaggistica instantanea è stata realizzata sfruttando il paradigma **client-server** per:
- la registrazione e l'autenticazione dell'utente;
- la ricezione di notifiche e di messaggi ricevuti mentre l'utente è offline;
- l'invio di messaggi ad un utente offline;
- la disconnessione dal servizio.

Il paradigma **peer-to-peer** è invece stato utilizzato per:
- lo scambio di messaggi tra due o più utenti online;
- l'invio di file.

La gestione dell'I/O avviene tramite **I/O Multiplexing** sia per il device che il server. Infatti, nel set dei monitorati vengono inseriti lo standard input e i descrittori dei socket in modo da poter prelevare e gestire opportunamente i descrittori pronti.

Infine, tutti i socket utilizzati sono di tipo **non-bloccante** e il protocollo di trasporto utilizzato è **TCP**. Tale scelta è stata presa considerando il requisito principale dell'applicazione: l'affidabilità, ovvero la garanzia che tutti i messaggi arrivino al destinatario integri e nell'ordine corretto.

NETWORK

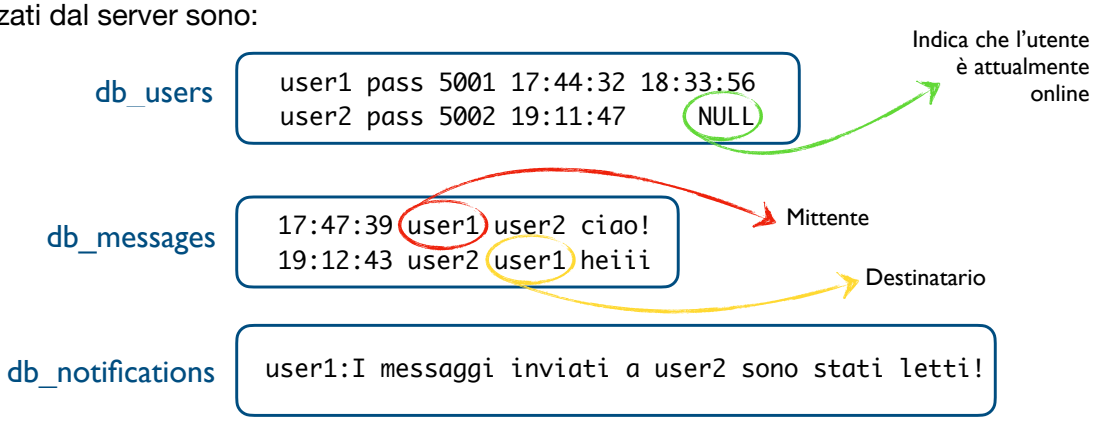
Network è una libreria che viene inclusa sia dal client che dal server. Al suo interno sono contenute le funzioni per l'inizializzazione e la chiusura dei socket e quelle per l'invio e la ricezione di messaggi e file.

Per le funzioni di invio, per prima cosa si invia la lunghezza del messaggio e, solo in seguito, il messaggio vero e proprio. Allo stesso modo, nelle funzioni di ricezione prima si riceve la lunghezza e poi il messaggio.

SERVER

Il server viene lanciato con il comando `./serv [port]`. All'inizio dell'esecuzione verrà inizializzato un socket di ascolto sulla porta indicata come parametro oppure sulla porta *4242* nel caso non sia stata specificata una porta in fase di avvio.
Successivamente, avremo un processo *padre*, il cui compito sarà quello di gestire i comandi ricevuti tramite standard input e di ricevere le richieste di connessione da parte dei device, e più processi *figli* che, utilizzando un socket di comunicazione, gestiranno le richieste dei device fino alla loro disconnessione.

I file utilizzati dal server sono:



Una struttura dati creata dal server quando un device si collega è la **lista dei messaggi pendenti** contenente tutti i messaggi a lui inviati mentre era offline. Questa è di fatto un'ottimizzazione perché permette di evitare di leggere ogni volta il file dei messaggi pendenti quando l'utente invoca il comando show su utenti diversi.

DEVICE

Il device viene lanciato con il comando `./dev <port>`.

L'utente può assumere uno dei seguenti stati:

```
enum User_State { DISCONNECTED, LOGGED, CHATTING_ONLINE, CHATTING_OFFLINE };
```

All'inizio lo stato dell'utente è `DISCONNECTED` poiché non si è ancora connesso al server perciò i comandi mostrati saranno i seguenti:

- ***signup*** *srv_port username password*: si creerà una connessione con in server sulla porta specificata. Successivamente, si invieranno i dati relativi all'username e alla password per effettuare la registrazione ed il server risponderà con *"ok"* in caso di successo, *"no"* al contrario.
- ***in*** *srv_port username password*: si creerà una connessione con in server sulla porta specificata. Successivamente, si invieranno i dati relativi all'username e alla password per effettuare il login ed il server risponderà con *"ok"* in caso di successo, *"no"* in caso contrario.

A questo punto, lo stato dell'utente passerà a `LOGGED` e quindi i comandi mostrati saranno:

- ***hanging***: verrà inizialmente ricevuto il numero di utenti che hanno inviato messaggi e, nel caso in cui questo numero sia diverso da zero, verranno ricevuti gli username di tali utenti.
- ***show*** *username:* verrà inizialmente ricevuto il numero di messaggi ricevuti da parte dell'utente specificato e, nel caso in cui questo numero sia diverso da zero, verranno ricevuti i testi dei messaggi in ordine di invio.
- **out**: viene eseguita la disconnessione dal server ed il programma termina.
- ***chat*** *username*: inizialmente viene controllato che il contatto sia presente nella rubrica. Se questo controllo viene superato, viene contattato il server che risponderà con *"offline"* oppure con la porta del destinatario a seconda se quest'ultimo sia online o meno.

A questo punto lo stato dell'utente passerà a `CHATTING_ONLINE` oppure `CHATTING_OFFLINE`.

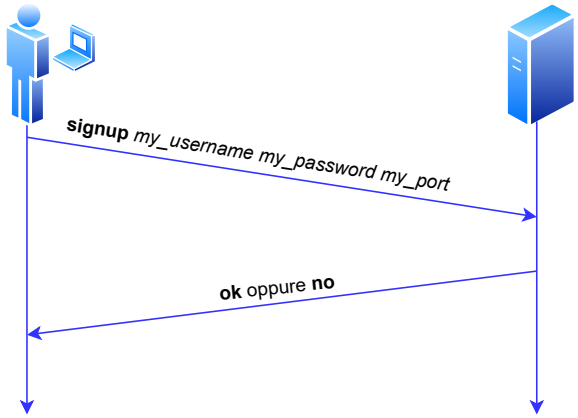
I comandi *"particolari"* invocabili durante la chat sono:

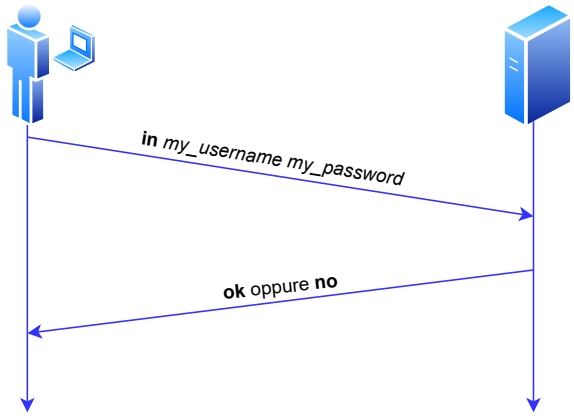
- ***u***: disponibile sono per la chat online, verrà inizialmente ricevuto il numero di utenti online e, se quest'ultimo è diverso da zero, si ricevono username e porta di tali utenti che verranno inseriti nella lista degli utenti online.
- ***a*** *username*: disponibile sono per la chat online, preleva le informazioni dell'utente specificato dalla lista degli utenti online e controlla se l'utente è già nel gruppo. In caso negativo, crea una connessione con l'utente e inserisce le sue informazioni nella lista degli utenti con cui si sta chattando. Inoltre, quando quest'ultima lista non è vuota, ogni volta che si riceve un messaggio, questo verrà inoltrato a tutti gli utenti presenti in lista (escludendo l'utente mittente).
- ***share*** *file*: disponibile sono per la chat online, divide il file in chunk e li invia a tutti i presenti nella lista degli utenti con cui si sta chattando.
- **\q**: chiude la chat disconnettendosi da tutti i device o dal server nel caso di chat offline. Dopo questo comando, lo stato dell'utente passerà a `LOGGED`.

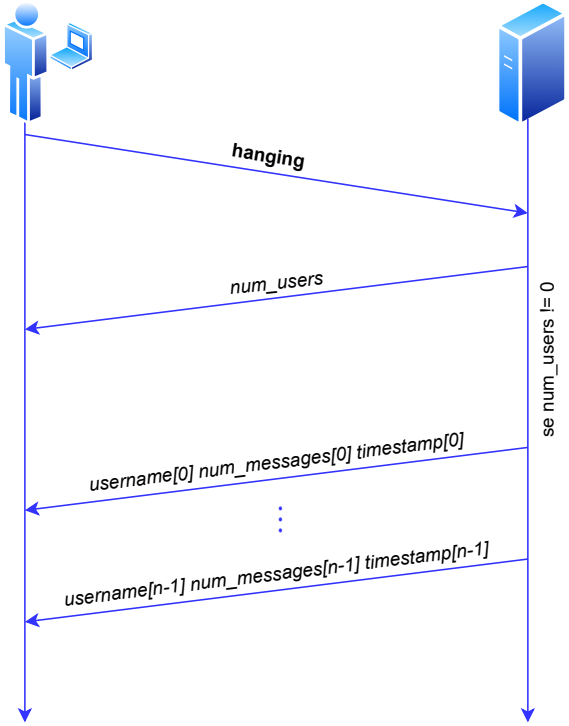
Le strutture dati utilizzate per salvare le informazioni sono le seguenti:

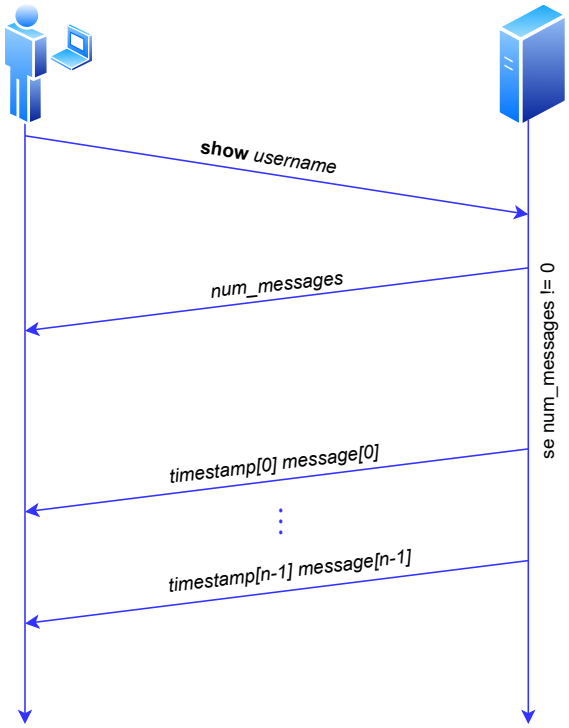
```
struct User {
    enum User_State user_state;
    char* my_username;
    char* my_password;
    char* my_port;
    char* srv_port;
    char* dst_username;
    struct usersChattingWith* users_chatting_with;
};

struct usersChattingWith {
    char* dst_username;
    struct sockaddr_in addr;
    in_port_t port;
    int p2p_sd;
    struct usersChattingWith* next;
};

struct onlineUser {
    char* username;
    char* port;
    struct onlineUser* next;
};
```

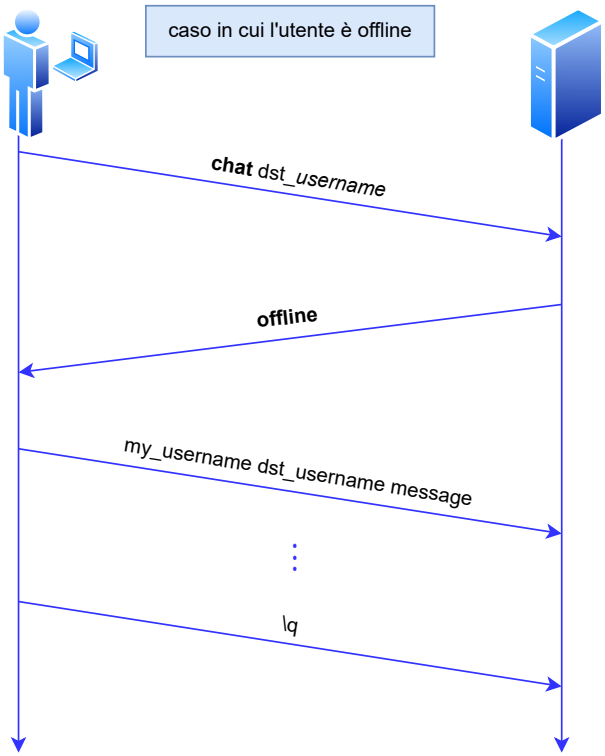
DIAGRAMMI DI COMUNICAZIONE

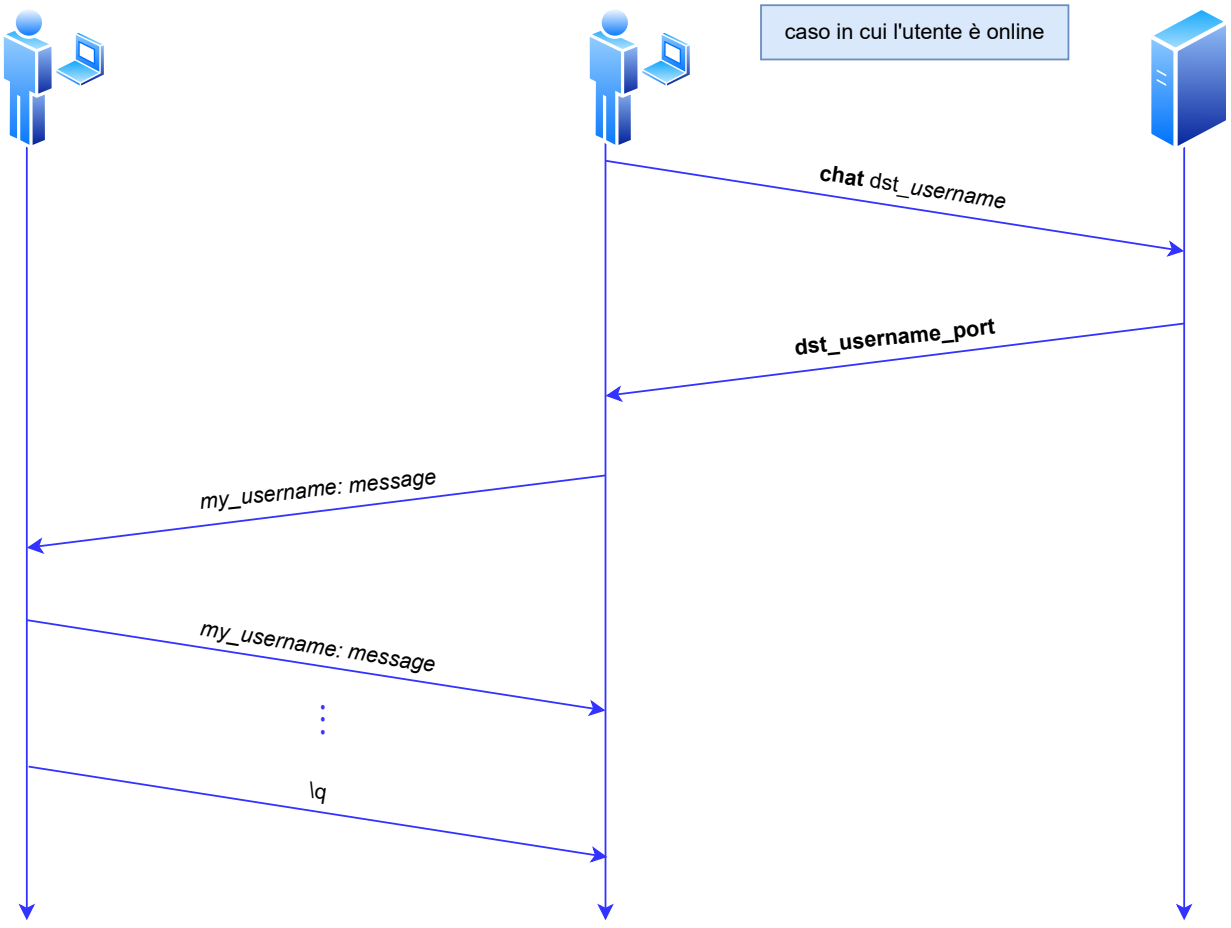


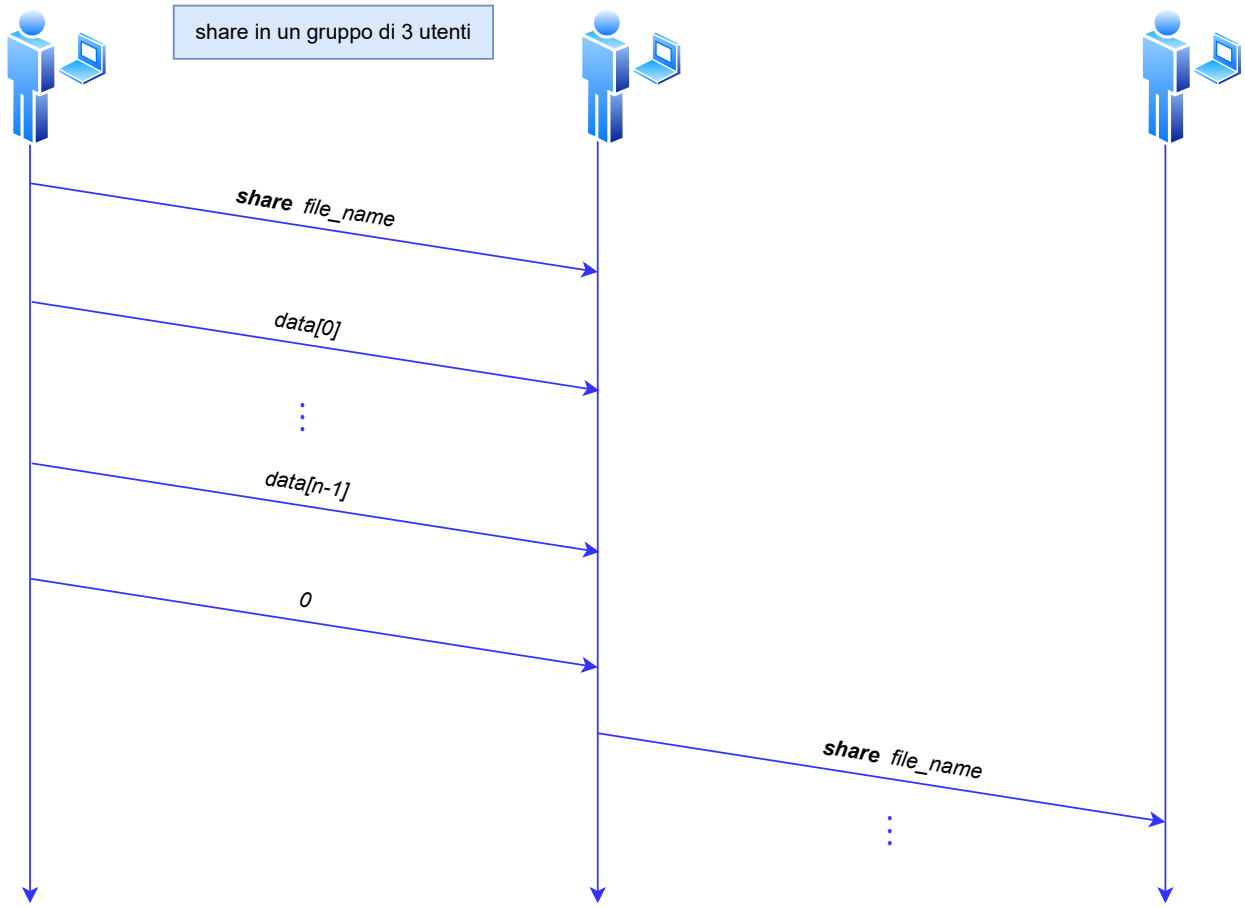







caso in cui l'utente è offline
chat dst_username
offline
my_username dst_username message
\q


caso in cui l'utente è online
chat dst_username
dst_username_port
my_username: message
my_username: message
\q


share in un gruppo di 3 utenti
share file_name
data[0]
data[n-1]
0
share file_name