## Introduzione

Server: un processo si occupa di gestire i comandi del terminale, un processo fa da listener (serve le richieste che arrivano dai vari client), un processo fa da “controllore” verso i client che risultano online (per gestire le disconnessioni improvvise da parte di questi ultimi). Il **protocollo di comunicazione utilizzato è TCP**.

## Gestione della chat

I messaggi della chat sono “impacchettati” nel seguente formato:

| message (const) | from | to | timestamp_sent | timestamp_received | message_text |
|---|---|---|---|---|---|

Vengono spediti in chiaro all’host destinatario, quindi come plaintext.

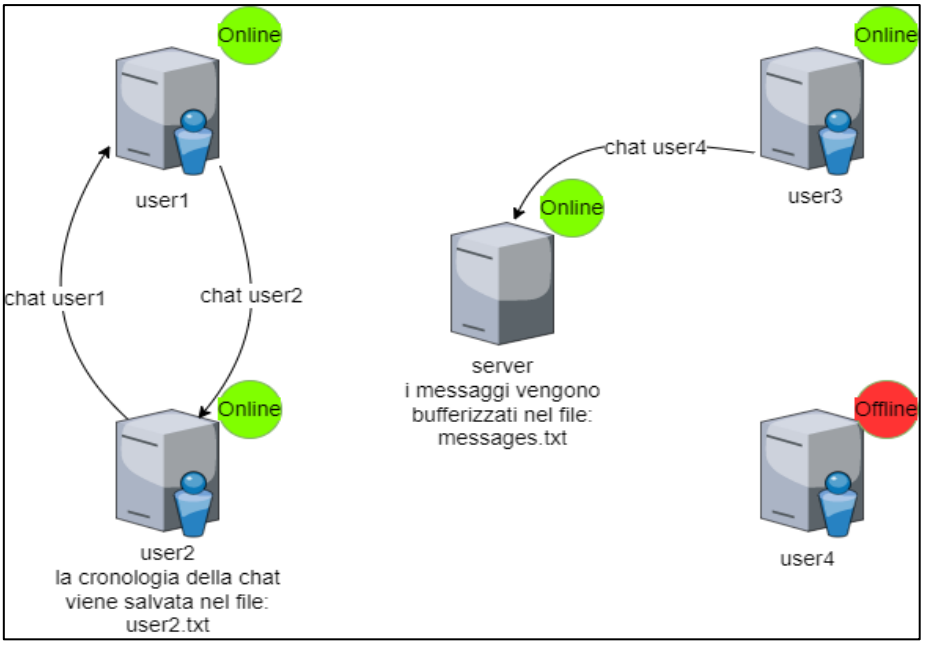


*Figura 1*: la comunicazione fra user1 e user2 è diretta, entrambi gli utenti sono online, user3 comunica con il server, i messaggi per user4 vengono bufferizzati all’interno del file **messages.txt** del server. Quando user4 torna Online tramite il comando “show user3” scarica i messaggi.

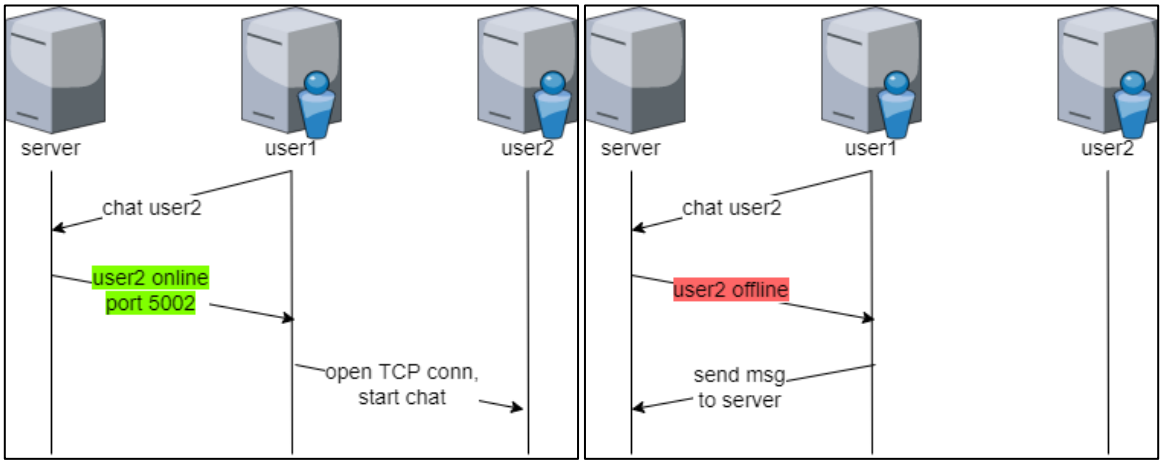


*Figure 2, 3*: chat, il server invia al device lo stato (e porta) del device destinatario.

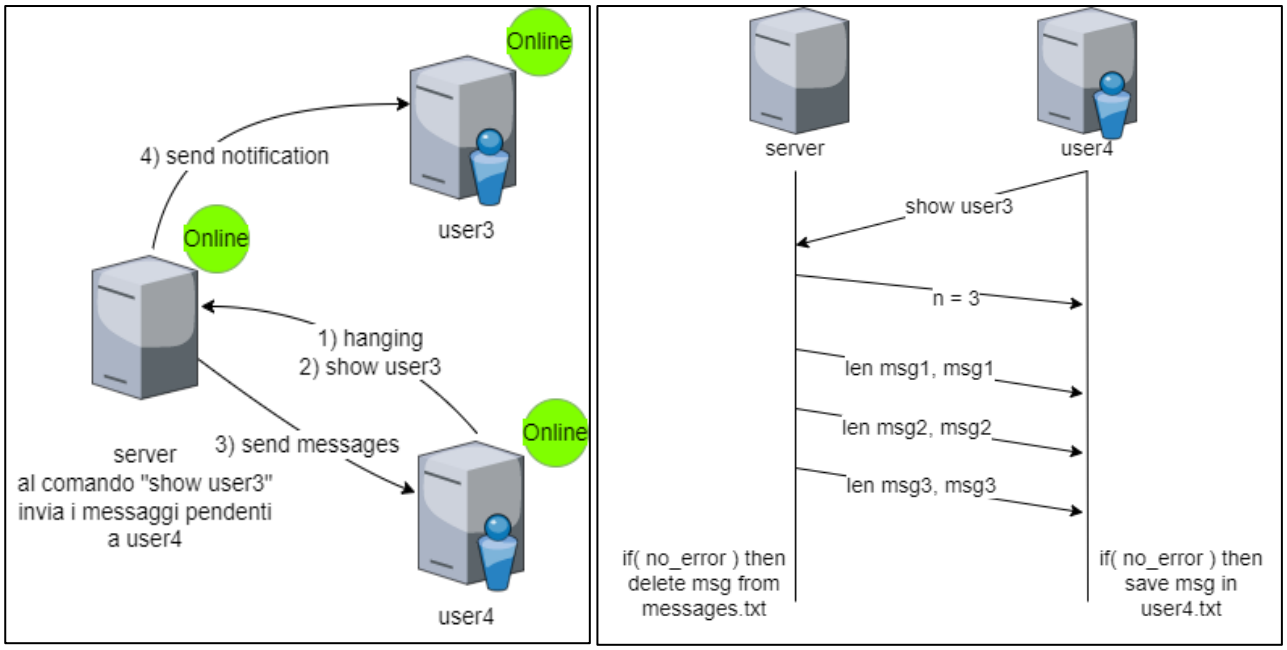


*Figure 4, 5*: protocollo di comunicazione per scaricare i messaggi pendenti, **assumendo ci siano 3 messaggi pendenti per user4**. (len msg e msg vengono inviati separatamente, per semplicità di immagine sono stati scritti insieme)

**Notifiche**

Quando un device invoca il comando “show *username*”, il server controlla lo stato di *username*:

- Se online, comunica a *username* l’avvenuta lettura dei suoi messaggi (come in figura);
- Se offline, bufferizza la notifica nel file **notifications.txt** (viene inviata quando user3 torna online e apre la chat con l’utente che ha letto i messaggi).

**Condivisione file all’interno delle chat**

Un peer stabilisce una connessione TCP con un altro peer, il peer trasmettitore invia prima la dimensione del file, poi divide e spedisce il file in chunk da 10240 byte, il peer ricevitore riceve e assembla i chunk, ricreando il file.

**Gestione delle disconnessioni improvvise dei device**

Un processo all’interno del server si occupa di controllare, periodicamente, se gli utenti che risultano online lo sono effettivamente (il valore del campo timestamp_logout è uguale a 0), stabilendo una connessione TCP con essi, e inviando un pacchetto di test (come se “pingasse” il device).

**Critica e Commento finale**

Le connessioni TCP potevano essere gestite meglio, inoltre, da un punto di vista della sicurezza, il progetto è carente, in quanto i messaggi sono spediti in chiaro.

Ho preferito puntare alla semplicità, e “separare” i device dal server, ovvero lo contattano solo quando ne hanno bisogno, in quanto l’applicazione è ibrida (peer-to-peer, client-server).