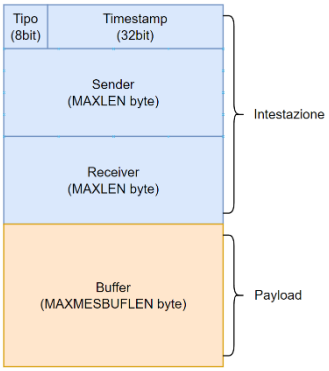
UNIVERSITÀ DI PISA

Documentazione progetto Reti Informatiche

Francesco Berti

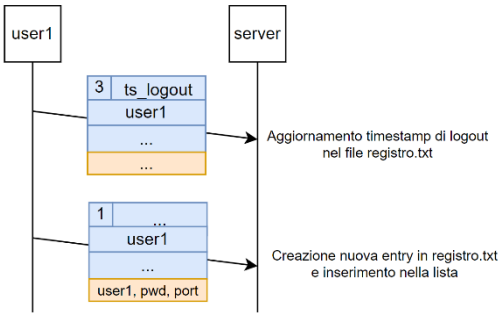
Sia client che il server fanno utilizzo di I/O multiplexing per gestire input dall'utente, richieste di connessione dai client e servire le varie richieste che questi possono effettuare, per questo motivo i socket utilizzati sono tutti bloccanti. È un approccio non scalabile in quanto le richieste sono servite una per volta quindi l'applicazione risulta meno responsive all'aumentare degli utenti ma consente di non dover gestire i problemi di sincronizzazione che si avrebbero con l'utilizzo di un server o client multiprocesso. L'applicazione fa esclusivo utilizzo di socket TCP in modo tale da garantire affidabilità ed utilizzare gli stessi socket sia per lo scambio dei file che per la comunicazione. I socket TCP hanno anche il vantaggio di comunicare la chiusura del socket da parte di un utente, quindi la sua disconnessione. Sia per i client che per il server i dati di ogni connessione attiva sono registrati all'interno di una struttura dati chiamata peer che memorizza socket descriptor, username e porta dell'utente. Le varie connessioni sono quindi organizzate in una lista di peer. Questo consente un accesso alle informazioni sulle connessioni più rapido rispetto ad una memorizzazione su file ma una minore robustezza in caso di crash del processo server. Per questo motivo viene anche usato il file registro.txt per contenere timestamp di login e logout degli utenti a cui però si accede solamente al login e logout.

L'applicazione utilizza uno specifico formato per i messaggi come descritto a fianco. Il campo *tipo* permette di specificare lo scopo del messaggio, il campo *timestamp* è usato per la gestione della visualizzazione dei messaggi, *sender* e *receiver* specificano gli username del mittente e del destinatario e infine *buffer* contiene il payload del messaggio. I tipi di messaggio sono:



- 0: signup
- 1: login
- 2: ack
- 3: timestamp di logout
- 4: messaggio di avvio chat
- 5: dati utenti (username e porta)
- 6: messaggio diretto cioè non passa dal server al contrario del tipo 4
- 7: hanging/show
- 8: file sharing
- 9: aggiunta ad un gruppo

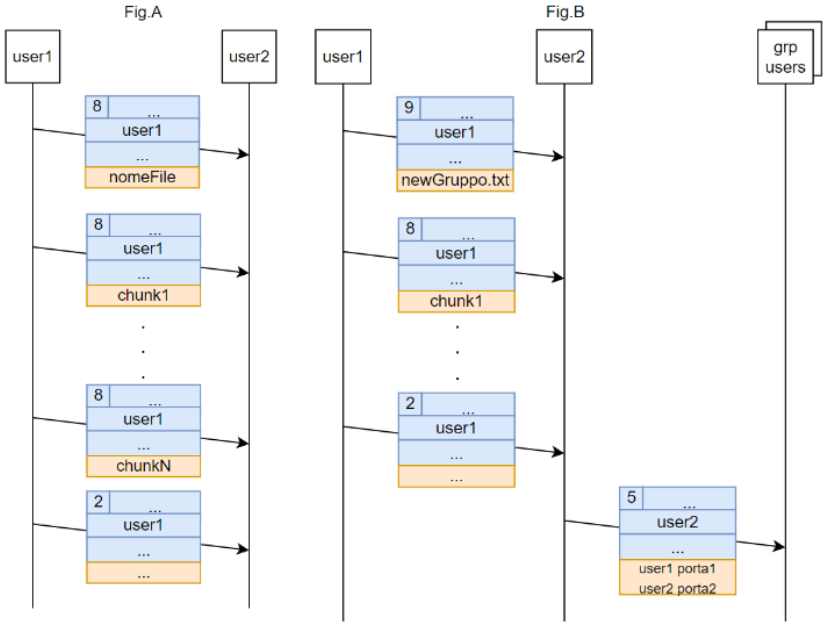
Se il server va offline i client tentano la riconnessione solamente nel momento in cui hanno bisogno di comunicare con esso, ad esempio per eseguire i comandi *hanging*, *show* e *chat* o per aggiungere un utente alla chat di gruppo. In seguito ad una disconnessione il client memorizza il timestamp attuale all'interno del file *ToSend.txt* per inviarlo al server una volta che la connessione è ristabilita così che esso completi la entry. In concomitanza della



riconnessione il client si autentica nuovamente con il server inviando username e password dell'utente così da creare una nuova entry aggiornata. Se il client viene chiuso prima di riconnettersi con il server il timestamp da inviare al server viene aggiornato. Nel caso in cui un utente *u1* stia chattando con l'utente *u2* e u1 si disconnetta, u2 passerà ad inviare messaggi di tipo 6 a messaggi di tipo 4 al server così che questo li memorizzi.

Il protocollo Peer To Peer (*Fig.A*) utilizzato per effettuare il file sharing fra i membri di un gruppo prevede che il peer che vuole condividere il file vada ad effettuare un invio del file per ogni utente all'interno del gruppo: il file è suddiviso in più chunk da al più MAXMESBUFLEN byte ciascuno che vengono inviati in ordine tramite un messaggio (tipo 8). Quando l'intero file è stato inviato, il peer invia un messaggio di ack (tipo 2).

È possibile avere solamente una chat attiva alla volta. Ogni utente possiede un file Gruppo.txt dove memorizza username e porta degli utenti facenti parte della chat attuale. Se si viene aggiunti ad una chat di gruppo (*Fig.B*) mentre si ha una qualsiasi chat aperta (singola o di gruppo) quella chat viene chiusa e si è portati forzosamente all'interno della chat di gruppo a cui si è aggiunti. L'aggiunta è fatta ottenendo prima la lista degli utenti online, questa viene fornita dal server utilizzando il protocollo di file sharing di cui sopra. Una volta scelto un utente della lista viene instaurata una connessione con esso e gli viene inviato un messaggio di tipo 9 che segnala che si è stati aggiunti ad un gruppo. Il peer che aggiunge invia al peer aggiunto il proprio file *Gruppo.txt* tramite il file sharing. Il peer aggiunto provvederà ad instaurare una connessione con ogni utente all'interno del file e ad inviare un messaggio di tipo 5 con il suo username e la porta, questi saranno aggiunti nel file di gruppo di ciascun partecipanti. Vengono inviati anche i dati del peer che ha aggiunto, questo permette agli altri peer di capire se sono stati aggiunti ad un nuovo gruppo o meno controllando se quei dati sono già presenti in *Gruppo.txt*.



Il server conserva i messaggi da inviare all'interno del file *hanging.txt* e utilizza invece il file *hanginghdr.txt* per memorizzare per ogni combinazione di sender/receiver possibile il numero di messaggi pendenti e il timestamp di quello più recente: ogni volta che viene memorizzato un messaggio in hanging sarà aggiornato hanginghdr.