La prima operazione che viene fatta è quella di stabilire una ***connessione TCP*** in quanto essendo un'applicazione di instant messaging, questa non è tollerante alla perdita di dati e non è strettamente time-sensitive (cioè non è importante che il messaggio arrivi istantaneamente, anche se non deve arrivare troppo tardi). Inoltre, è ***importante che le operazioni*** di signup, login e file sharing **avvengano senza perdita di dati**.

## Rilevare utenti online e disconnessioni improvvise

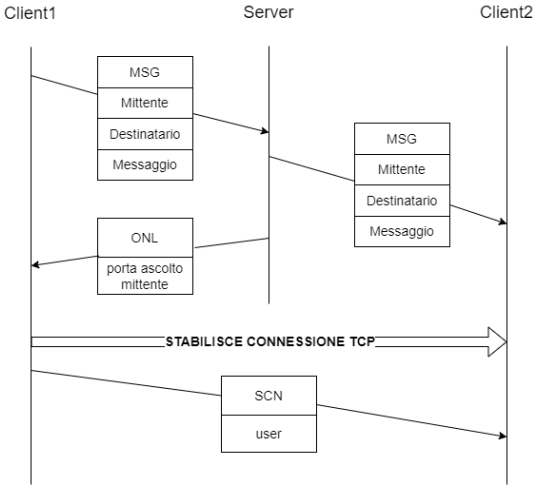
La ***connessione TCP, non viene*** chiusa dopo il login o la signup. Questo ha come ***vantaggio***:

- Tutte le **operazioni** possibili **necessitano** (almeno la prima volta) di **contattare** il **server**. Sarebbe controproducente chiudere la connessione per doverla riaprire poco dopo.
- Così facendo si **conoscono** sempre quali sono gli **utenti online** (basta memorizzare gli utenti che hanno una connessione attiva in **una struttura dati** che associa allo username la porta di ascolto e il socket descriptor) e si riescono a **rilevare le disconnessioni improvvise** degli utenti, memorizzando inoltre l'istante di disconnessione.

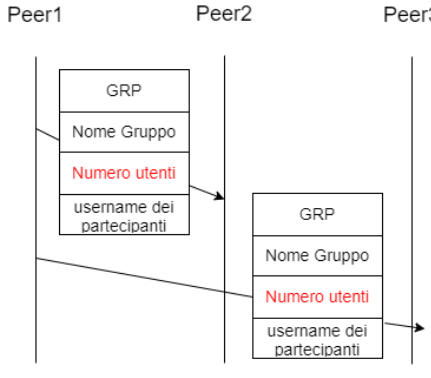
D'altro canto, però si riservano delle risorse per un socket che, dopo la fase iniziale di uso della applicazione da parte dell'utente (hanging, show, prima volta che si aprono le chat) viene utilizzata (statisticamente) direttamente alla chiusura.

Formato dei messaggi: se non espressamente indicato, per ogni campo viene mandata prima la lunghezza (essendo campi a lunghezza variabile) e poi il contenuto del campo. La lunghezza massima è di 1024 Byte.
**In ogni messaggio (eccetto la risposta del server nell'operazione di show), il primo campo è il tipo dell'operazione (una sigla che serve per identificarla).**

## Scambio di messaggi nella chat tra due utenti




Il server risponde come in figura nel caso in cui l'utente sia online, in caso contrario memorizza il messaggio e l'orario e risponde al client uno con "NON" (Not Online).
Nel caso in cui il destinatario sia online, il mittente stabilisce una connessione con il destinatario e dopo averla stabilita manda il messaggio SCN (Start Connection) seguito dal suo username: questo  per poter continuare la conversazione senza dover passare dal server ed usando lo stesso socket. Inoltre, ogni utente ha una struttura dati nella quale memorizza l'associazione socket descriptor – utente: il fatto di aver questa struttura dati ed un socket che non viene mai chiuso permette agli utenti di sapere anche quando un altro utente si disconnette (sia volontariamente attraverso il comando "out", sia improvvisamente), così da inviare eventuali nuovi messaggi per quell'utente al server. Il messaggio SCN viene mandato solo per il primo messaggio scambiato tra i due utenti.



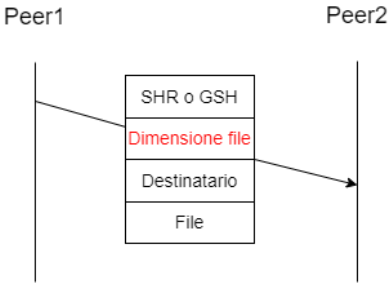

## Creazione e abbandono di un gruppo

Quando un peer riceve il messaggio di creazione di un gruppo, controlla se per ogni utente possiede già un socket sul quale scambiare messaggi, se così non fosse si deve occupare di creare una connessione con questo utente (e mandare il messaggio SCN visto in precedenza). Una volta creato il gruppo, dunque, sappiamo che tutti gli utenti appartenenti sono online e che sono raggiungibili grazie alla creazione dei socket. Può capitare alcune volte (e solo nel caso in cui il gruppo venga creato quando nessuno utente ha mai comunicato con nessun altro) che due peer provino a connettersi contemporaneamente tra loro, creando di fatti due socket

di comunicazione monodirezionali, vanificando così lo sforzo di avere un unico canale di comunicazione per ogni coppia di utenti.
La figura mostra il comportamento dell'applicazione nel caso in cui il Peer 1 crei un gruppo e si suppone che Peer2 e Peer3 siano online.
*Il campo numero utenti è l'unico per il quale prima non viene inviata la dimensione del campo.*
È possibile inoltre per un utente **abbandonare** un **gruppo** attraverso il **comando \l,** in quel caso verrà mandato un messaggio con solo il tipo "LEV".
Il gruppo verrà abbandonato **automaticamente in caso di disconnessione dell'utente**.

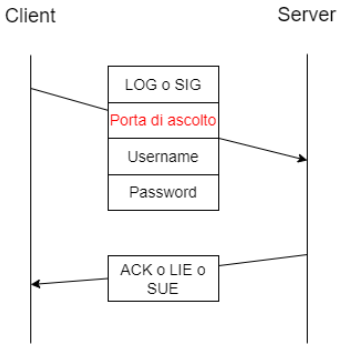
Scambio di messaggi nella chat di gruppo: il formato dei messaggi è identico al precedente con l'aggiunta, prima del campo mittente, del **campo "nome gruppo".**

## Condivisione dei file



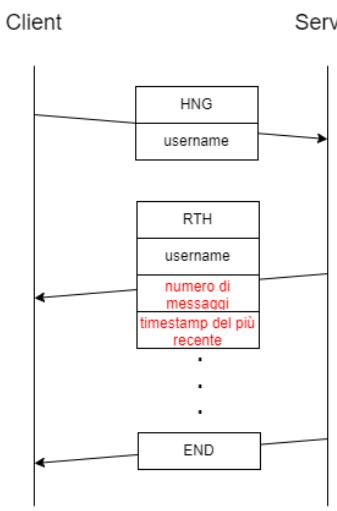
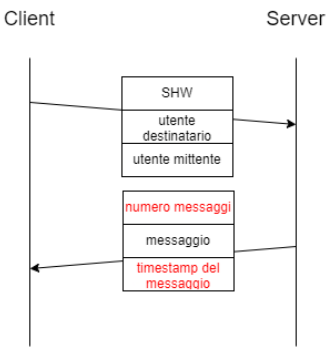
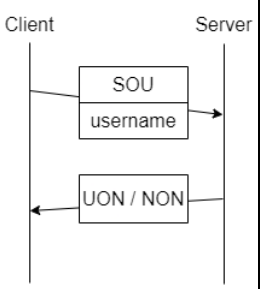
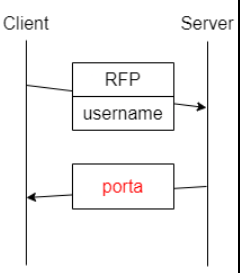
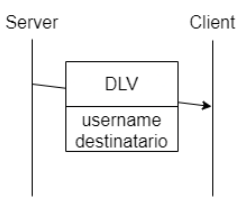

**È possibile avviare il comando \s nomeFile solo all'interno di una chat.** Il tipo può essere SHR (condivisione di un file in una chat singola) o GSH (Group Share), ma il formato del messaggio rimane uguale. *Il campo dimensione file è l'unico per il quale prima non viene inviata la dimensione del campo*. È possibile che un utente debba passare dal server prima di condividere il file, ma solo per poter stabilire la connessione con l'/gli altro/i utente/i (quindi per chiedere se l'utente è online e su quale porta ascolta).
Il file viene letto ed inviato per un max di 1024 byte per volta, cosicché se il file fosse molto grande non verrebbe copiato interamente in memoria centrale ed inoltre, la dimensione massima dei campi è appunto di 1024 byte.

## Login e Signup




In **entrambi i casi** l'utente dovrà inserire "**NUMERO PORTA SERVER** username password".
Le possibili risposte del server significano:
ACK = operazione andata a buone fine; LIE = LogIn Error; SUE : SignUp Error. *Il campo porta di ascolto è l'unico per il quale prima non viene inviata la dimensione del campo.*
Nel caso del login, se nell'ultima sessione l'utente si era disconnesso mentre il server era offline, manderà un messaggio del tipo "OUT" + timestamp dell'istante di uscita (non si specifica la dimensione del campo)

## Restanti tipi di messaggi scambiati:

| Hanging | Show message | Utenti online | Richiesta porta | Notifica consegna |
|---|---|---|---|---|
|  |  |  |  |  |











I campi scritti in rosso sono i campi di cui prima non si è mandata la dimensione.