Relazione progetto reti informatiche
Fabio Piras 578207

Struttura base:

Il servizio si basa sullo scambio di codici testuali (char [4]) simil ASCII dove il receiver, in base al codice di operazione che riceve in ingresso, tratta gli ulteriori dati ricevuti in modo differente. Il vantaggio di questo sistema è che è semplice e che non bisogna inviare la lunghezza della stringa del codice in quanto è sempre la stessa; lo svantaggio, nell'attuale implementazione, è che questo sistema richiede un utilizzo di più di una send quando si vuole inviare dei dati.

Il software utilizza soltanto connessioni TCP in quanto l'applicazione è loss intolerant.

Device

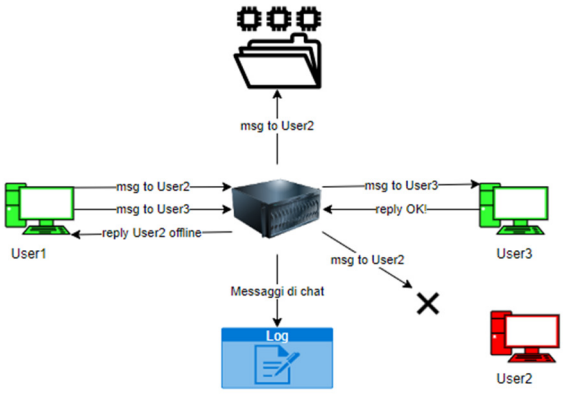
Il Device basa il suo funzionamento su una struttura dati (Device_s) che contiene diverse informazioni ad esempio: se l'utente è loggato, se sta chattando, lo username dell'utente connesso, una lista per gli utenti presenti in rubrica e una lista per gli utenti con cui sta chattando. Queste due ultime liste si basano su una ulteriore struttura dati (UserContact) che salva nome e porta di ogni contatto.

Server

Il Server basa il suo funzionamento su una struttura dati (Server_s) che tiene in memoria la porta di ascolto del server, il nome dei file di utilità e due liste che servono per tenere conto dei vari utenti iscritti al servizio e per memorizzare delle notifiche pendenti. La prima sfrutta una struttura dati (UserEntry) che a sua volta memorizza username, password, porta, timestamp_login, timestamp_logout e numero di tentativi di connessione effettuata. La seconda usa anch'essa una struttura dati (UserPending) che memorizza a chi è destinata la notifica e chi l'ha generata.

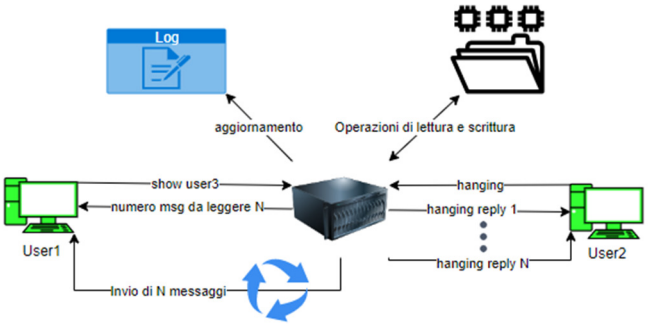
Protocolli

Messaggi di chat



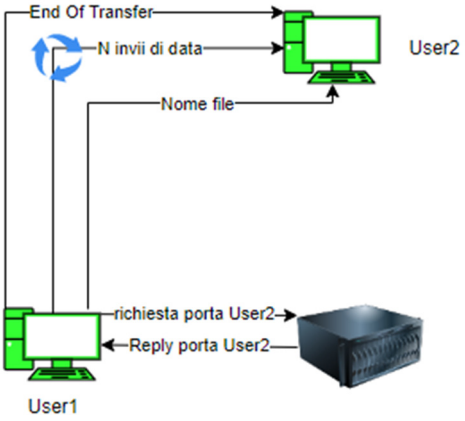
In generale quando si desidera inviare un messaggio di chat, un utente lo invia prima al server che proverà a inoltrarlo al device destinatario. Se l'operazione va a buon fine (destinatario online e in chat), il destinatario risponde con un OK! per segnalare che il messaggio è stato ricevuto e letto. Se invece il server non riesce a contattare il destinatario, il messaggio viene anzi salvato su file e viene notificato il sender che attualmente il receiver è offline; inoltre viene aggiornato il timestamp logout del destinatario se quest'ultimo non risultava offline prima del tentativo di connessione, per fare ciò il server si appoggia sulla struttura UserEntry. Se il destinatario è loggato, ma non in chat (non in figura) il device risponde segnalando che il messaggio è stato ricevuto, ma non letto. In ogni caso vengono aggiornati i file della cronologia di chat.

Hanging e Show



Per le operazioni di hanging, l'utente contatta il server che invierà N risposte generando per ciascuna una connessione diversa. Per le operazioni di show, il server invia al richiedente il numero di messaggi che deve ancora leggere, per poi procedere a creare una nuova connessione e a mandare i vari messaggi sulla stessa connessione in ordine di lettura da file, vengono inoltre aggiornati i file di log. (Non in figura) A seguito di un'operazione di show l'utente che aveva inviato i messaggi viene contattato per segnalare l'avvenuta lettura, se non è possibile contattarlo, la notifica viene salvata su file appoggiandosi alla struttura UserPending e la notifica verrà inviata quanto l'utente riavvierà la chat con l'utente che ha effettuato la show.

File transfer



Per le operazioni di trasferimento di file il sender fa prima richiesta al server di inviargli la porta su cui il destinatario è in ascolto. Dopo di che il sender invia per prima cosa il nome del file così che il receiver possa creare un nuovo file contenitore; si procede con il sender che invia ciclicamente sulla stessa connessione i "chunk" di dati e, una volta finito, invia il segnale di fine trasferimento. Il receiver dopo aver ricevuto il nome entra in un ciclo di ascolto e scrittura su file fino a che non vede arrivare il segnale di fine trasferimento.

Informazioni generali sui cicli

La scelta di inviare determinati dati tramite cicli sulla stessa connessione piuttosto che inviare i dati su connessioni differenti volta per volta è che in primis il ricevente deve creare un processo figlio per ogni nuova connessione mettendo così sotto stress la macchina, come seconda cosa vi è il rischio che i messaggi in ingresso non vengano processati nell'ordine di invio generando così confusione. Il grande svantaggio di questo sistema però è che dobbiamo inviare preventivamente il numero di send che dovremo utilizzare così che il receiver sa per quante volte deve mettersi in ascolto per ricevere dati tramite socket. Questo sistema funziona fintanto che il sender sa a priori quanti cicli di invio dovrà eseguire come nel caso della funzione di show dove si va a leggere quanti messaggi sono presenti per poi inviarli, ma questo comporta un doppio ciclo di lettura su file (uno per contare i messaggi e l'altro per copiare su buffer e inviarli) per cui potrebbe non risultare molto efficiente nel caso di documenti di grosse dimensioni.