PROGETTO – Niccolò Bedini 597488

QUADRO GENERALE e LOGICA APPLICAZIONE:

Il main del programma device consiste, dopo una prima fase dove si deve necessariamente scegliere tra l'opzione di signup e login, in un unico while gestito da una select.

Inizialmente all'interno della select saranno presenti il socket di comunicazione con il server e lo stdin, in seguito vengono aggiunti i socket relativi alle connessioni che si effettuano attraverso la chat.

Oltre al main, il cuore dell'applicativo risiede all'interno della funzione chat() invocata dal menù principale.

All' interno di questa funzione è presente una nuova select interna a quella precedente. In questo modo si emula un isolamento dal resto delle funzionalità fornite dall'applicazione. In particolare, oltre ad essere inizialmente mostrata la cronologia dei messaggi con l'utente selezionato per la chat, sarà possibile visionare esclusivamente i messaggi provenienti dall'utente selezionato, più eventuali messaggi da utenti aggiunti alla chat successivamente.

I messaggi provenienti da utenti esterni alla chat saranno visualizzati nella pagina principale solamente dopo essere usciti da quest'ultima attraverso il comando \q.

(*)

La scelta per le chat di gruppo è stata fatta in modo tale da consentire unicamente all' utente che aggiunge il nuovo membro (\u + \a) alla chat di comunicare con quest'ultimo; in questo modo non si costringe gli utenti già presenti a conversare con persone che non hanno aggiunto loro stessi.

(*)

Quando siamo dentro una chat (funzione chat()), ad ogni messaggio inviato, si itera per tutti i nomi nella lista puntata da registro (che inizialmente contiene il nome dell'utente con il quale abbiamo iniziato a chattare e poi gli eventuali utenti aggiunti in seguito con \u+\a).

Per ognuno viene chiamata la funzione INVIO();

Questa funzione gestisce tutti i casi in cui si può trovare il destinatario:

- Utente Online con connessione TCP già stabilita con il mittente.

- Utente Online senza connessione TCP.

- Utente Offline e Server Online.

- Server Offline.

(*) Eccezione fatta per il primo messaggio arrivato da un nuovo utente, quello con cui si stabilisce la connessione tcp, che viene sempre notificato.

(*) In questo modo è anche possibile effettuare il controllo dell'appartenenza del nuovo membro alla rubrica

Scelte Implementative:

1. TCP o UDP? Ho scelto di fare un'applicazione esclusivamente TCP.
   Essendo un'applicazione di messaggistica può tollerare bene un minimo ritardo (anche dell'ordine di qualche secondo) tra i messaggi scambiati tra gli utenti, e considerandone la brevità non necessito di un throughput elevato e costante.
   Ma al contrario non può tollerare perdita o errore di dati, e la scelta di implementare un canale affidabile a livello di applicazione anziché sfruttarla a livello di trasposto sarebbe stata poco pratica e efficace. Inoltre, usando TCP posso ritenermi anche più fair con il resto della rete.

2. PROTOCOLLO: il fulcro del protocollo consiste nella conversione in stringa di una struttura nella quale si indica con un intero nel campo "req" la richiesta che il server deve gestire.
   Il server è sempre in attesa del messaggio.
   A questo punto il server e il Device sono sincronizzati, e in base al tipo di richiesta vi sono una serie di SEND e RECV che servono a portare a termine l'operazione.
   Essendo che solo il primo passaggio del protocollo è sempre uguale, mentre il resto dello scambio di informazioni tra server e Device è molto eterogeneo tra una richiesta e l'altra, e volendo mantenere una uniformità globale all'interno del programma ho ritenuto il text protocol più funzionale rispetto al binary considerando anche la brevità dei messaggi scambiati.

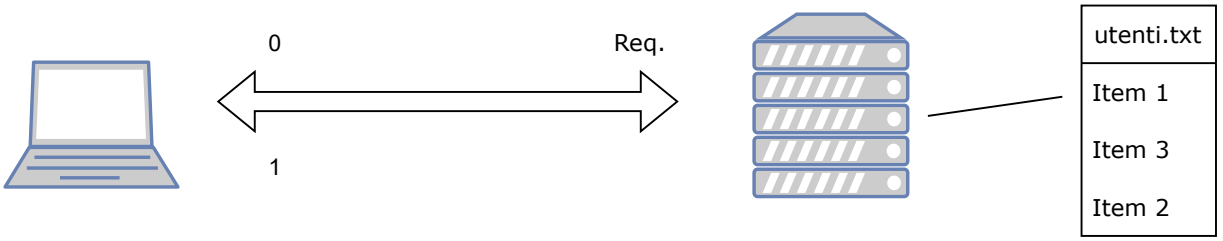
3. FILE SHARING: per quanto riguarda il file sharing (e solo per il file sharing) ho usato un binary protocol, spinto necessariamente dal fatto che i file scambiati generalmente sono di grandi dimensioni. Quindi ho utilizzato un semplice vettore di tipo opaco uint8_t, in modo tale da non avere problemi di Endianness o eventuali tipi intrinsici.
   (testato con pdf, .txt, .png, .mp3)

4. DISCONNESSIONI IMPROVVISE: all'interno delle due select, all'arrivo di un messaggio da un socket avviene sempre il controllo di disconnessione (RECV = 0), compresa quella del server. In quel caso basta togliere il socket da quelli da monitorare e aggiornare la lista delle chat attive nel Device e degli utenti loggati nel server. In questo modo tutto continua normalmente. (Se la disconnessione è del server si può continuare a messaggiare con utenti già connessi con TCP ma non iniziare conversazioni con nuovi).

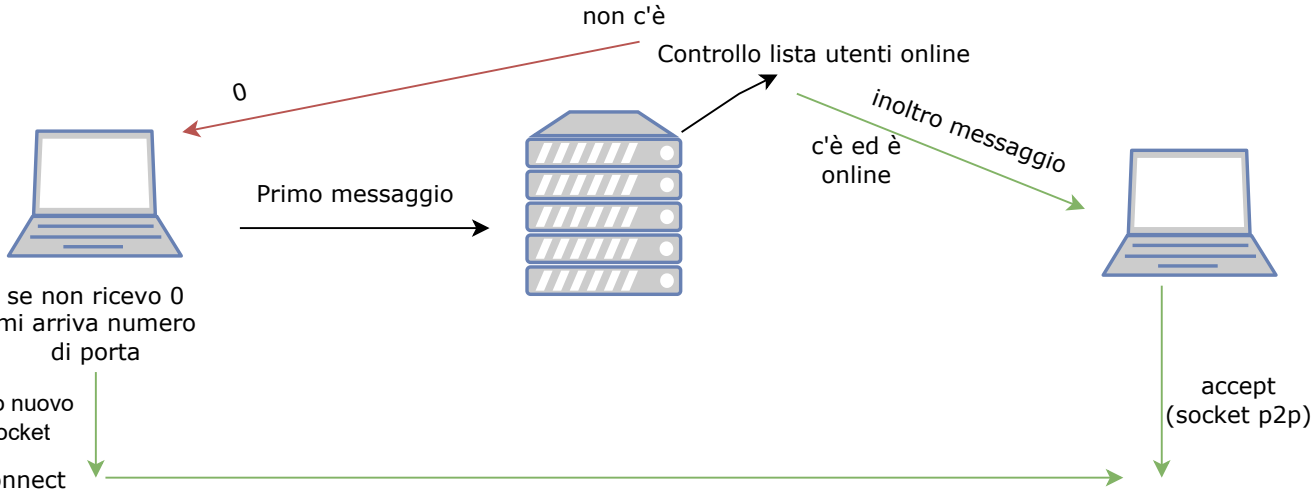
REGOLE GENERALI:

- lo username è univoco char[25],
- la chat inizia sempre con un unico utente e nel caso se ne possono aggiungere altri solo se presenti in rubrica.
- Il file di log viene azzerato al momento del logout sia che abbiamo effettuato l'hanging o meno.
- I nomi nella Rubrica vanno aggiunti manualmente.
- Ogni Device all'accensione ha 2 socket, uno che si connette al server e uno che accetta connessioni con altri Device.

LOGIN-SIGNUP:



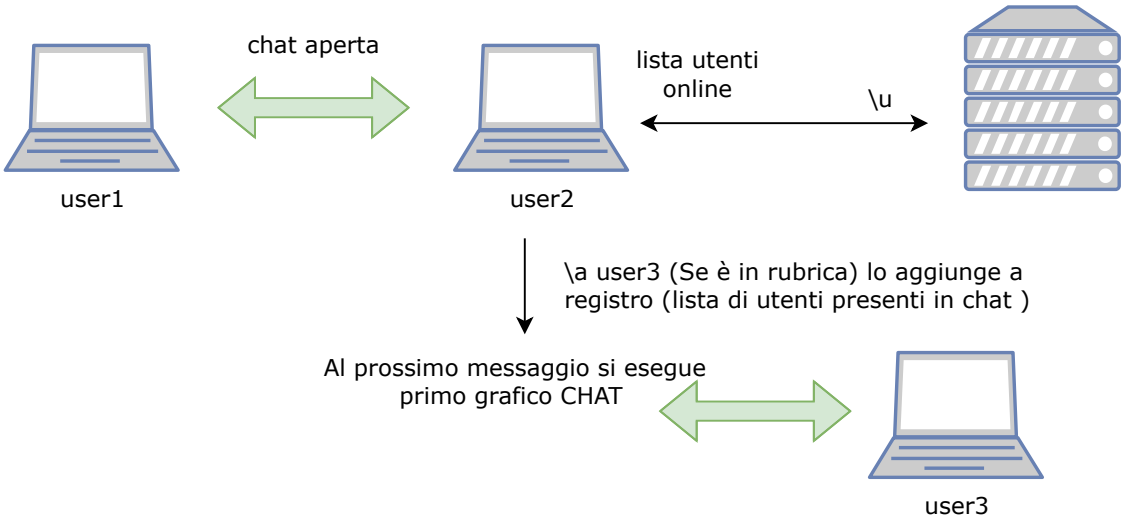
CHAT:



Se ho ricevuto 0 -> utente OFFLINE -> posso continuare a messaggiare e scrivo ogni messaggio con '*' nel file chat (*[user]:mess)
Ad ogni nuovo messaggio si ripete il ciclo che termina con la freccia rossa

Nel caso, invece, che l'utente fosse online creo connessione e invio messaggi peer-to-peer

Aggiungere membri:

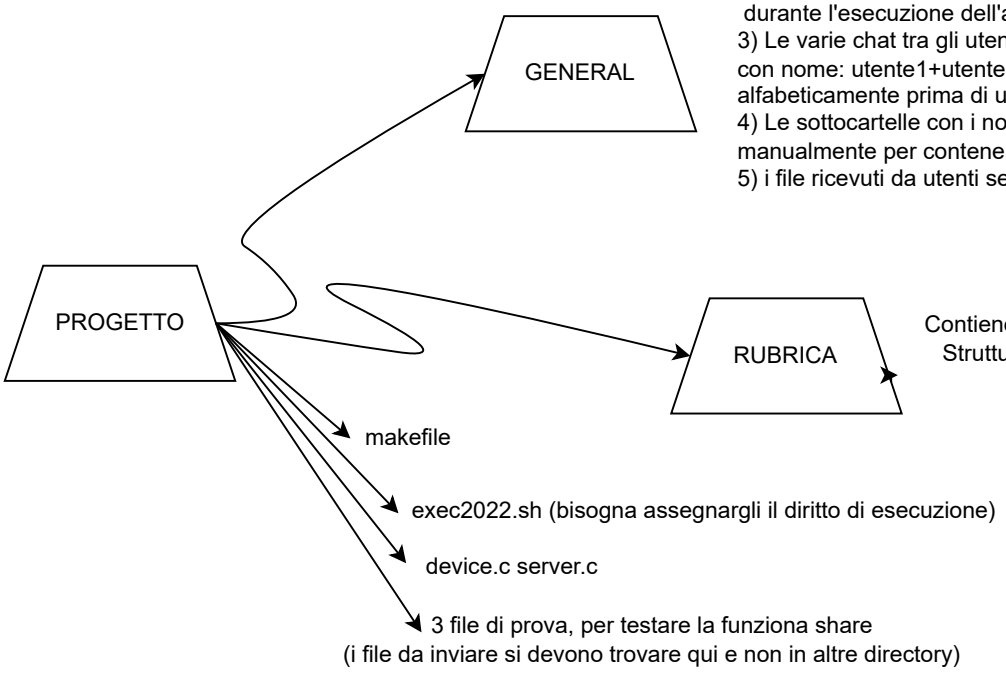


nota: solo user2 è in una chat di gruppo
user1 è in chat con user2
user3 è nel menu e riceve messaggi di user2 e entrando in chat con user2 può rispondergli senza passare dal server perchè la connessione con user2 è già stata stabilita dopo il primo messaggio.

user3 e user1 non sono in comunicazione!

DIRECTORY:

Contiene:
1) File utenti.txt che contiene la lista degli utenti iscritti all'applicazione
2) I vari file di log che vengono creati automaticamente
durante l'esecuzione dell'applicazione
3) Le varie chat tra gli utenti create automaticamente
con nome: utente1+utente2-chat.txt utente1 è
alfabeticamente prima di utente2)
4) Le sottocartelle con i nomi degli utenti da creare
manualmente per contenere i file inviati
5) i file ricevuti da utenti senza sottocartella con nome utente--nomefile



Cosa bisogna fare per avviare l'applicazione:

L'unica cosa da fare è concedere i diritti necessari a exec2022.sh

---

Di default ho aggiunto le rubriche degli utenti user1,user2 e user3 come da istruzioni, per aggiungere nuove rubriche bisogna farlo manualmente
il file utenti.txt è inizialmente vuoto, per aggiungere gli utenti user1 user2 e user3 basta fare il signup nell'applicazione.
Chiaramente i nomi nella signup devono coincidere con quelli nelle rubriche.

Per aggiungere nuovi membri a una chat digitare \u e premere invio, verrà visualizzata una lista di utenti online.
Digitare \a nomeutente e premere invio, se l'utente era online ed è nella rubrica allora verrà aggiunto alla chat.

Per inviare i file andare in una chat (3) digitare: share nomefile
Il file si deve trovare nella directory principale (dove sono già stati inseriti prova1.txt, prova2.pdf, prova3.mp3)
Il file inviato si troverà in GENERAL/nomeutente se esiste la cartella (di default ho creato solo quelle per user1,user2 e user3)
Nel caso non esista una cartella con il nome utente, il file verrà inviato semplicemente in General.