GABRIELE FRASSI (603011) – PROGETTO RETI INFORMATICHE

Nella realizzazione del progetto è stato adottato unicamente il protocollo TCP, in quanto l'applicazione realizzata (text messaging, e in parte file sharing) sono del tipo *loss intolerant*.

Per prima cosa l'autore vuole porre alcune riflessioni sulle strutture dati presenti nel codice.

- **Nel server**. Le informazioni relative agli utenti (username e password) sono memorizzate permanentemente nel file users.txt, curato dal server. Questo file è utilizzato solo nei seguenti casi:
    - all'avvio del server, per creare una lista di strutture informazioni_utente;
    - durante la registrazione dell'utente (si aggiunge una riga col nuovo utente);
    - durante il login (si legge il file per individuare un peer con l'username e la password indicati).

  In tutti gli altri casi il server consulta le strutture informazioni_utente e basta.
- **Nel server**. Ogni struttura informazioni_utente contiene un puntatore a una lista di mittenti (strutture messagebox_mittente), e per ogni mittente si ha un puntatore a una lista di messaggi (strutture messaggio_mittente). L'organizzazione delle strutture è pensata in modo tale da garantire un'esecuzione veloce dei comandi hanging e show.
- **Nel client**. Le connessioni TCP sono stabilite col lancio del comando chat e memorizzate in una lista di strutture connessione_peer. Sono state create funzioni di libreria che permettono di individuare un particolare oggetto della lista (una a partire dall'username, una a partire dal file descriptor del socket).
- **Nel client**. È presente una struttura chat_gruppo utilizzata per raccogliere i puntatori all'utente con cui si è andati a stabilire una chat di gruppo. Per le specifiche sulle chat di gruppo si rimanda più avanti.

Successivamente si vuole esporre gli aspetti principali dei protocolli adottati nel codice! I comandi sono trasmessi in codifica ASCII per agevolarne la lettura. In alcuni casi si è optato per una concatenazione COMANDO|<PARAMETRI> per minimizzare il numero di messaggi trasmessi.

- **Verifica dello stato del peer**
  L'utente lancia il comando chat per scrivere a un particolare peer. Si deve verificare se il peer è online o meno:
    - si lancia un comando al server indicando l'username del peer;
    - il server verifica con un PING rivolto al client se questo è online;
    - se è online restituisce il numero di porta, altrimenti segnala che è offline

  A seguito di questo passaggio viene aperta la chat. I dispositivi comunicano tra di loro senza coinvolgere ulteriormente il server. Sarà verificato nuovamente lo stato del peer solo con un nuovo lancio del comando chat.
- **Bufferizzazione dei messaggi dei peer offline**.
  Se la verifica spiegata precedentemente ha esito negativo ogni messaggio viene inviato al server. Questo memorizza i messaggi nelle liste già citate precedentemente (strutture messagebox_mittente e messaggio_mittente):
    - ricerca la struttura informazioni_utente relativa al peer offline;

    - verifica se nella lista di strutture messagebox_mittente è presente una struttura dedicata al mittente del messaggio, se non è presente la crea (timestamp e counter sono adeguatamente impostati);
    - aggiunge nella lista di messaggio_mittente il messaggio inviato.

  L'applicazione lato client gestisce i messaggi per mezzo di file aventi il seguente formato:
  inviati_<USER1>_<USER2> e pendenti_<USER1>_<USER2>
- **Recupero dei messaggi da parte di un peer precedentemente offline**.
  Il peer destinatario può ottenere l'elenco degli utenti che hanno inviato messaggi lanciando il comando hanging. Le informazioni richieste dalle specifiche (timestamp più recente) e contatore sono recuperati al volo dalle strutture messagebox_mittente. Può scaricare i messaggi di un particolare utente utilizzando il comando show. Il server:
    - crea un unico pacchetto concatenando i messaggi in attesa e invia il tutto al peer destinatario;
    - elimina la lista di messaggi in attesa e azzera il contatore posto in messagebox_mittente;
    - invia un ACK cumulativo al peer mittente.

  Il peer mittente prende atto della ricezione dei messaggi: svuota il file pendenti_<USER1>_<USER2>, spostandone il contenuto in inviati_<USER1>_<USER2>.
- **Chat di gruppo**.
  Le chat di gruppo sono gestite con approccio P2P senza coinvolgere il server:
    - col comando \a USERNAME + INVIO si indica l'username del peer che si vuole aggiungere al gruppo;
    - il peer che sta aggiungendo l'utente lo contatta;
    - l'utente risponde positivamente o negativamente.

  È già stata anticipata la presenza della struttura chat_gruppo, che contiene due puntatori. Si pongono i seguenti limiti:
    - un peer x crea il gruppo ed è l'unico a gestirlo;
    - un peer non può gestire più di un gruppo in contemporanea;
    - la chat di gruppo è vincolata al comando chat, questo significa che la chat è automaticamente eliminata se l'utente decide di uscire (\q + INVIO)
    - i peer scriveranno nel gruppo ogni volta che decidono di chattare col peer x (non devono creare un gruppo a loro volta), il peer x inoltrerà i messaggi a tutti i peer presenti nel gruppo;
    - il peer può aggiungere al gruppo un massimo di due utenti (escluso il peer e l'utente con cui chattava individualmente all'inizio);
    - si possono aggiungere al gruppo soltanto utenti con cui si è stabilita precedentemente una connessione TCP (apertura della chat individuale col comando chat);
    - non può essere aggiunto ad un gruppo un peer che si sta già occupando di un altro gruppo (questo serve per evitare flood di messaggi e per impedire un'espansione incontrollata del numero di componenti di una chat di gruppo).

I file presentanti contengono commenti sulle azioni svolte da varie righe di codice.