Chat Service

Progetto Reti Informatiche Università di Pisa 2021/2022

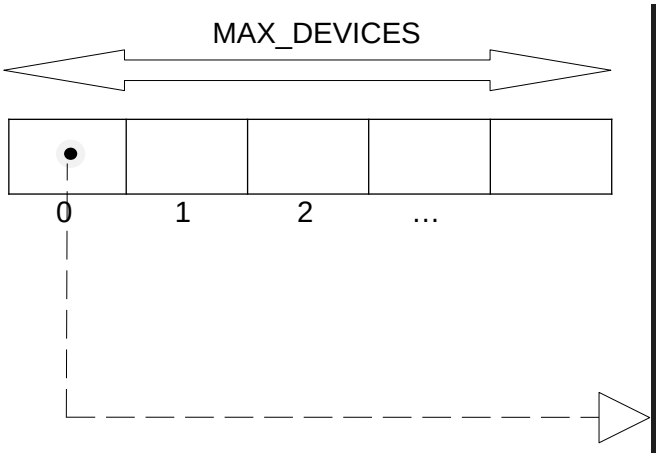
Filippo Del Ministro

Le specifiche di progetto prevedono lo sviluppo di un servizio di chat che possa connettere in chat singole (con due soli utenti) o di chat di gruppo gli utenti facenti parti della rete. Gli utenti sono gestiti in fase di registrazione e connessione da un server che tiene aggiornato lo stato della rete e gestisce i messaggi dei devices offline.

Struttura progetto

Il progetto è strutturato in tre file*: all.h, server.c, dev.c*.
Nel file all.h sono presenti alcune definizioni comuni agli altri file come costanti o funzioni di utilità (in particolare per lo scambio di messaggi).
Neil file server.c è sviluppato il codice per il funzionamento del server: questo gestirà le funzionalità del server stesso (comandi *help, list, esc*), e le richieste degli utenti.
Nel file dev.c è sviluppato il codice relativo ai dispositivi: la loro definizione, i comandi utilizzabili da utente, e le funzioni che permettono l'esecuzione dei comandi.

Per la gestione dei dispositivi, sia il server, che i devices utilizzano un array di device, ognuno identificato da un ID, assegnato dal server in fase di registrazione[1]: l'ID sarà l'indice di accesso all'array.




```
// -----------     DEVICE     -----------------
struct device{
    int port;                   // port number
    int sd;                     // TCP socket
    struct sockaddr_in addr;

    //device info
    int id;
    char* username;
    char* password;
    bool connected;             //true if connected

    //chat info
    char chat_path[15];
    bool hanging_done;
    bool busy;                  //true if already in chat
}devices[MAX_DEVICES];
```

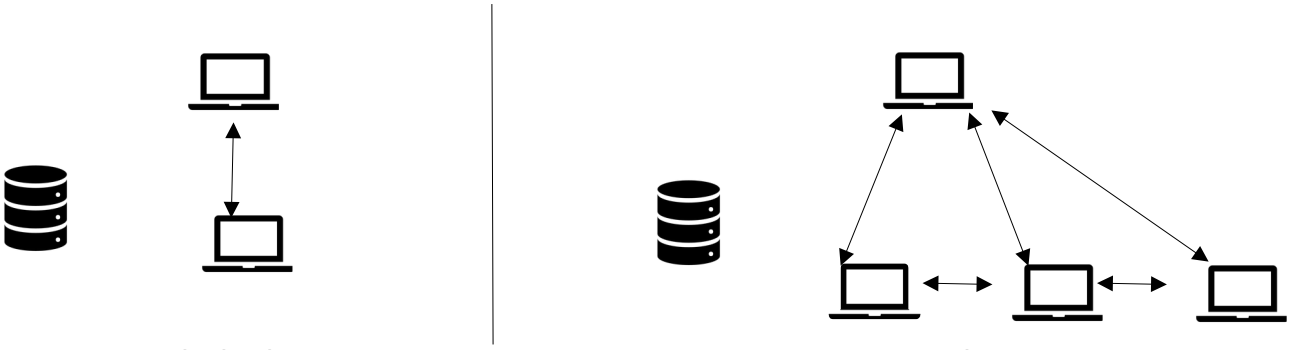
Ogni interazione con il network da parte dei dispositivi è notificata al server, il quale registra le *signup* da parte di nuovi utenti, e le operazioni di *login & logout* degli utenti già registrati: Il server è pertanto costantemente aggiornato sullo stato della rete.
I dispositivi, al contrario, si aggiornano sullo stato della rete ogni volta che ne fanno richiesta (comando *list*), oppure ogni volta che interagiscono con la rete (comandi *chat*, *hanging*, ecc.).
L' aggiornamento viene eseguito dalla funzione *update_devices()* che mette in contatto il device con il server, il quale manda al dispositivo tutte le informazione (accessibili) di ogni utente: username, numero di porta, e stato attuale (connected/busy).
Questo approccio, sebbene sia usabile solo in reti di piccole dimensioni, permette l'aggiornamento immediato di tutti i dispositivi in rete, e il disallineamento minimo delle informazioni mantenute dal server rispetto da quelle salvate dai dispositivi. Per reti di dimensioni più grandi potremmo pensare

[1] Il server manda ERR_CODE nel caso in cui l'username del dispositivo che fa signup sia già stato assegnato o se ci sono già MAX_DEVICES nel network: in tal caso si interrompe la fase di registrazione.

di modificare la *update_devices()*, con una *update_device(int ID)* che aggiorni le informazioni di un singolo device, e non dell'intero network.

Chat

Il servizio di chat è sviluppato in modo che ogni utente crei una connessione diretta con gli altri utenti direttamente interessati; quando l'utente user0 esegue `chat user1`[2], si stabilisce una connessione tra i due utenti, preceduta da un breve handshake in cui user1 riceve il dev_ID di user0. Il server non viene coinvolto: soprattutto per una questione di privacy, si è cercato di limitare al minino il flusso di informazioni da devices a server, che non è infatti in grado di sapere tra quali nodi è stabilita una connessione (chat), ma solamente quali nodi sono "*busy*" e quali no.
Questo approccio viene ripetuto anche per la gestione delle chat di gruppo: non si informa mai il server delle intenzioni dei dispositivi, ma semplicemente si fa una richiesta di update_devices() (per esser aggiornati sullo stato della rete) per poi contattare direttamente i dispositivi da aggiungere alla chat, stabilendo una connessione diretta con un socket dedicato.
Questo permette inoltre la disconnessione asincrona dei dispositivi, che possono uscire dalla chat in ogni momento, in modo del tutto indipendente dagli altri device in chat o dal server.



Single chat | Group chat

Comunicazione

L'invio di messaggi è stato interamente gestito mediante l'utilizzo di funzioni che, come prima cosa, permettono il coordinamento tra sender e receiver sulla dimensione del pacchetto:

- *send_int*: invia un tipo timbrato[3] e lo converte in intero.
- *send_msg*: utilizza send_int per informare sulla dimensione del messaggio da inviare.
- *send_file*: scorre un file, leggendo una dimensione accordata a priori, e la invia al ricevitore

Le rispettive funzioni in ricezione sono *recv_int, recv_msg, recv_file*.

Messaggi pendenti

Ogni qualvolta un device provi a mandare messaggi ad un device offline, i messaggi vengono intercettati dal server che li salva in un file testuale[4]; quando il destinatario torna online e esegue il comando *hanging*, il server invierà l'intero file al destinatario.
Per gestire messaggi pendenti in modo completo è stata utilizzata una matrice che salva, per ogni utente, il numero di messaggi pendenti ricevuti da ogni altro utente, e il timestamp dell'ultimo messaggio (*server.c* [riga 48] per i dettagli implementativi).

Sicurezza

Ogni comando che non sia signup o login, è protetto da una fase di autenticazione (trasparente all'utente) in cui il device, dopo essersi identificato mediante dev_ID, comunica al server username & pswd. Questo evita l'attacco di script esterni che, mandando una serie di OPCODE e dev_ID validi, potrebbero forzare alcuni comandi illeciti su utenti iscritti alla rete. (funzione *authentication()* per dettagli).

---

[2] Per poter procedere, user1 deve esistere nella rete, esser online, e non già impegnato in una chat.

[3] uint16_t

[4] Un file per ogni coppia sender/receiver.