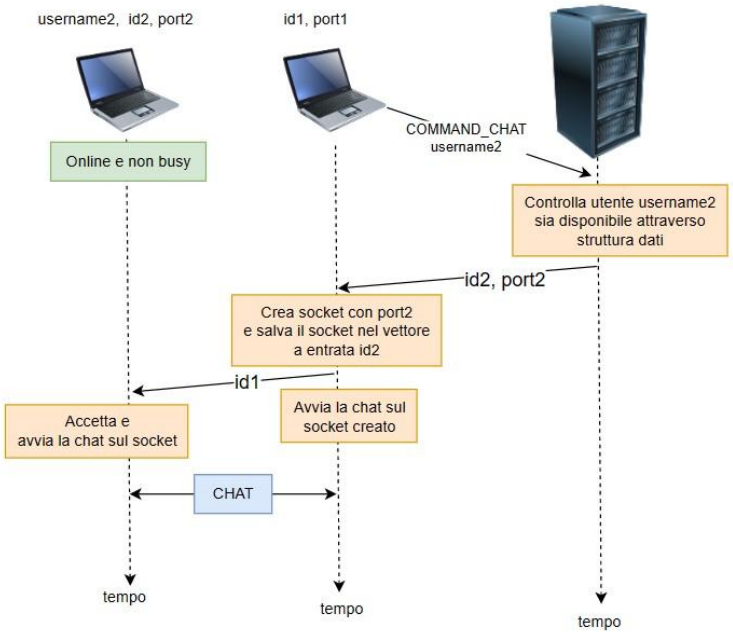
UNIVERSITÀ DI PISA

Relazione del progetto di reti informatiche 2021/2022
Servizio di instant messaging

Il progetto come da specifiche propone lo sviluppo di un'applicazione distribuita di **instant messaging** basata su un **modello ibrido** peer-to-peer e client-server.

Sia il server che i dispositivi hanno un vettore della dimensione del massimo numero di dispositivi registrabili in cui ogni entrata è un descrittore di dispositivo, l'i-esima entrata rappresenta il dispositivo con id=i.

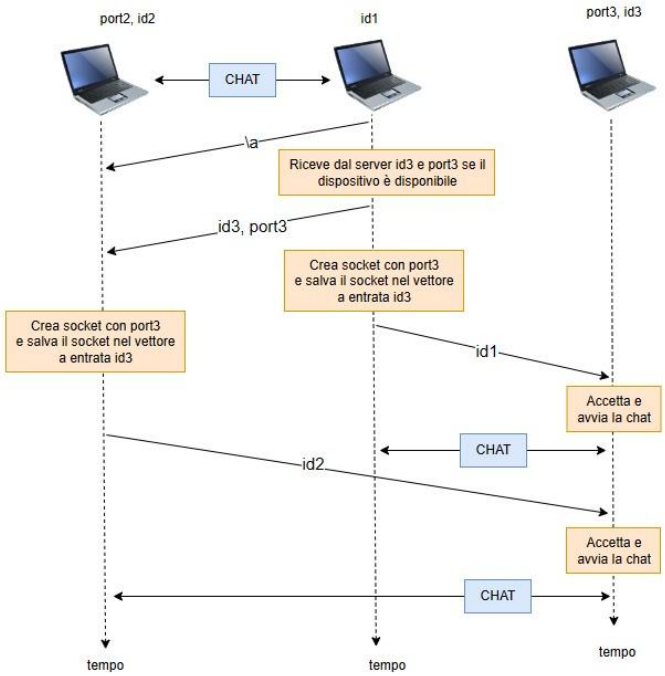
Avvio di una chat



Gli utenti una volta registrati e aver eseguito l'accesso possono **avviare chat** con gli altri utenti che hanno **in rubrica**. L'avvio è rappresentato sinteticamente in figura (caso in cui dev2 sia disponibile).

Se un utente prova ad avviare una chat con un **dispositivo offline** o che in quel momento è **occupato** in altre chat si troverà in una chat con il server che salverà i messaggi inviati e li proporrà al destinatario quando sarà disponibile; nel momento in cui il destinatario leggerà i **messaggi 'pendenti'** il mittente riceverà una **notifica di avvenuta lettura** (caso in cui dev1 sia disponibile).

Chat di gruppo



I dispositivi in chat possono poi **aggiungere altri utenti** in modo da creare una **chat di gruppo**. In questo caso ogni coppia di dispositivi sarà connessa da un socket di comunicazione: questo metodo permette ai dispositivi di poter uscire dalla chat indipendentemente dagli altri, ma **non scala** con l'aumentare dei dispositivi in chat ([n*(n-1)]/2 connessioni con n numero di dispositivi in chat), per risolvere tale problema si dovrebbe ripensare la struttura delle connessioni facendo sì che tutti i dispositivi inviino i messaggi a un unico dispositivo in chat e questo dispositivo si occupasse poi di inoltrare i messaggi agli altri dispositivi, avremmo così solo n

connessioni. Si avrebbe però un ritardo doppio ad ogni invio e bisognerebbe gestire il caso in cui il dispositivo gestore andasse offline.

L'aggiunta di un utente a una chat che passa da 2 a 3 utenti è rappresentata in figura, è omessa la richiesta di dati che il dispositivo 1 fa al server riguardo il dispositivo 3.

Gestione dei comandi in chat

Quando un dispositivo scrive in chat viene prima controllato se esso sia un comando (\u per la lista di utenti online, \a per aggiungere utenti, \s per inviare file), in tal caso si invia e si svolgono le operazioni necessarie. In caso in cui si scriva un semplice messaggio viene prima inviato agli altri dispositivi un OK_SIGNAL per indicare che sta per essere ricevuto un messaggio e viene poi aggiunto al messaggio il nome del mittente ed il timestamp di invio in modo che il ricevitore possa stamparlo e visualizzare direttamente le informazioni.

Gestione di interruzioni improvvise

Nel caso in cui un dispositivo si disconnetta senza effettuare la *out* non provoca problemi agli altri dispositivi in quanto tutte le *recv* sono gestite in maniera da controllare se venga ricevuto un ERROR_SIGNAL così che si possa continuare le proprie operazioni senza interruzioni.
Il dispositivo viene però ancora considerato online dal server quindi se un dispositivo chiedesse di iniziare una chat con esso non riceverebbe messaggi di errore dal server; nel momento in cui però il dispositivo avviasse il socket con il dev offline riceverebbe un errore sulla *connect()*, grazie ad esso riconoscerebbe che è offline ed interromperebbe il processo di chat e segnalerebbe al server che il dev non risponde ed è dunque considerabile offline, il server aggiornerebbe quindi lo stato del dev nel suo descrittore.

Se un dispositivo si interrompesse improvvisamente durante una chat continuerebbe a risultare busy e quindi se un altro provasse ad avviarvi una chat riceverebbe dal server un messaggio che indica che il dispositivo è occupato; la situazione non è comunque un problema poiché al dispositivo che avvia la chat sapere che l'altro dispositivo sia offline o busy non fa differenza.

File e directory

dev.c e serv.c: file in cui è scritto il codice dei dispositivi e del server.
utility.c: file contente funzioni di supporto agli altri due file.
rubric: cartella contente le rubriche di ogni dispositivo, ogni file nella cartella rappresenta la rubrica del dispositivo, i nomi contenuti sono quelli nella rubrica.
chat_device_i: contiene le history delle chat che il dispositivo i ha avuto con gli altri dispositivi, i file all'interno hanno l'id del dispositivo con cui si ha avuto la chat.
log_register: contiene il log degli accessi di ogni dispositivo, nei file la prima colonna indica la porta su cui si ha avuto l'accesso, la seconda colonna il timestamp di login, la terza il timestamp di logout.
pending_messages: cartella contenente i messaggi pendenti che verranno inviati dalla show, dopo la show i messaggi vengono cancellati.
pending_messages.txt: file contenente una struttura a 3 dimensioni, va vista come una matrice 10*10 (10 è il numero impostato come numero massimo di dispositivi registrabili), in cui ogni elemento della matrice è composto da una coppia di valori. Esempio: il terzo e il quarto numero rappresentano il secondo elemento della matrice: indicano i messaggi pendenti che il dev 1 ha inviato al dev 2, il primo valore è il numero, il secondo il timestamp dell'ultimo messaggio inviato.
restoreserver.txt: file scritto al momento della esc dal server, contiene info utili per il ripristino.

Luca Cappabianca