Documentazione Progetto Reti Informatiche
Ingegneria Informatica UNIPI A.A. 21/22

April 20, 2022

Guillaume Quint (577923)

Descrizione L'applicazione distribuita di messaggistica realizza tutti i requisiti funzionali descritti nella documentazione fornita dal professore. Segue il paradigma client-server per la registrazione, autenticazione, controllo della connettività, inoltro dei messaggi e salvataggio in cache per la messaggistica offline. Inoltre svolge una comunicazione di tipo peer-to-peer per la funzionalità di file-sharing e nella comunicazione tra devices nel caso in cui il server fosse irraggiungibile. Questa funzionalità di tipo inaffidabile è stata aggiunta rispetto alle specifiche originali, ed è descritta nel paragrafo del device.

Strutture dati utilizzate L'unica struttura dati utilizzata è la single-linked-list con puntatori alla testa e alla coda per facilitarne l'operazione di append. La lista viene utilizzata per la gestione della rubrica da parte dei devices e per la gestione dei profili attivi sul server.

L'utilizzo delle liste ha semplificato il lavoro in fase di sviluppo, ed è relativamente efficiente rispetto ad altre strutture dati più complesse nelle condizioni testate (non più di 10 utenti). D'altra parte, la complessità lineare di ricerca nella lista richiederebbe la sostituzione con vettori dinamici o hash-tables nel caso in cui il volume di utenza aumentasse notevolmente.

Modalità verbose Per attivare la modalità verbose è necessario cambiare nelle prime linee dei file `server.c`, `device.c`, `util/IOMultiplex.c` e/o `util/network.c` la definizione del flag per il compilatore da `DEBUG_OFF` a `DEBUG_ON`, così che venga definita la macro `DEBUG_PRINT`

Per evitare che la routine periodica di richiesta di heartbeat da parte del server "inondi" l'interfaccia dei comandi, la modalità verbose è disattivata per il modulo network. Attivandola, è possibile vedere ogni singolo pacchetto inviato e ricevuto da client e server.

Server All'avvio del server, viene mostrato il numero di porta su cui questo è in ascolto, insieme ad una breve descrizione dei comandi disponibili.

`help` mostra la lista dei comandi, `list` mostra la lista degli utenti attualmente connessi nel formato `username*timestamp_login*porta` ed `esc` chiude il server senza avvisare i devices

La registrazione e l'autenticazione degli utenti viene svolta appoggiandosi ad uno specifico file, chiamato `shadow`, con il ruolo di database. Ogni linea del file equivale ad un record per un corrispettivo utente registrato. I campi, in formato ASCII separati da spazio, sono nell'ordine `username passowrd porta timestamp_login timestamp_logout chances`

- La porta, se diversa da -1, identifica la porta sul localhost del device su cui è in ascolto l'utente che ha effettuato il login
- timestamp_login e timestamp_logout sono l'epoch time in secondi degli ultimi eventi di login e logout, e permettono di sapere (se ts_login > ts_logout) se l'utente è attualmente connesso
- Il numero di chances viene utilizzato dalla routine di controllo connettività dei devices per tenere traccia di quanto sia probabile che un utente si sia disconnesso senza avvisare. La richiesta di *heartbeat* (messaggio con codice `HRTBT`), infatti, viene svolta sul protocollo UDP per minimizzare il traffico, ma la natura inaffidabile del protocollo non assicura che tale richiesta arrivi correttamente. Per evitare che un device venga considerato offline ingiustamente, è necessario che non risponda per un numero `MAX_CHANCES` (di default, 3) di volte senza comunicare al server un messaggio con codice `ALIVE`, il quale è sufficiente a ripristinare il numero massimo di chances per quell'utente.

Il server ha il ruolo di inoltrare i messaggi ai destinatari per conto dei vari utenti. Se tutti i destinatari sono online e ricevono il messaggio, al mittente viene inviata una risposta con il codice `READ:` con il significato che il suo messaggio è stato letto da tutti. Altrimenti la risposta ha il codice `CACHE` e per ogni utente che non ha ricevuto il messaggio, il server lo salva localmente. In particolare, all'interno della cartella `hanging` e nella sottocartella chiamata come il destinatario, verrà aggiunto il contenuto del messaggio nel file chiamato come il mittente (in breve, il messaggio andrà in `./hanging/<destinatario>/<mittente>`).

Quando il server riceve un comando di `hanging` (messaggio con codice `HANG?`) da parte di un utente, gli basta rispondere con il contenuto della cartella `./hanging/<utente>/`

Il comando `show` corrisponde al trasferimento del contenuto del file `./hanging/<destinatario>/<mittente>` e successiva sua eliminazione. Inoltre, l'evento di lettura dei messaggi non recapitati viene segnalato all'utente mittente, affinché possa sincronizzare l'informazione di lettura con il suo storico di messaggi salvato localmente.

Infine il server sfrutta la funzionalità di timeout della primitiva `select` per effettuare una routine di controllo in maniera periodica. In particolare invia ad ogni device attualmente online una richiesta di heartbeat, affinché questi confermino il loro stato di connettività. All'arrivo di un segnale di alive da parte di un device, il server ripristina le chances per quell'utente.

Device All'avvio di un device viene mostrata la porta su cui questo è in ascolto e una breve lista dei comandi disponibili. Inizialmente nessun utente è connesso. E' possibile accedere o registrarsi con i rispettivi comandi `in` e `signin` specificando le proprie credenziali. Alla creazione di un nuovo profilo viene creata la propria cartella personale del device (con il nome dell'utente nel current path) e il proprio spazio dove ospitare i messaggi non recapitati quando l'utente è offline sul server (con il nome dell'utente nella cartella `./hanging/`). Attenzione: NON viene creato il file di rubrica. Inoltre l'applicazione non implementa alcuna funzione per crearla e modificarla. Pertanto il nuovo utente non sarà in grado di comunicare con nessuno, fintanto che non verrà creato il file `contacts` nella cartella personale (quindi `./<nome_utente>/contacts`) contenente il nome degli utenti con cui si vuole chattare, uno per riga. La corretta registrazione di un utente viene seguita da un login automatico.

Di default (ed il comando `make clean` riporta a questo stato) sono già presenti tre utenti. Le credenziali e le rispettive rubriche sono:

- `user1 pass` ha tra i contatti `user2`
- `user2 pass` ha tra i contatti `user1` e `user3`
- `user3 pass` ha tra i contatti `user2`

Successivamente all'accesso di un utente, il device controlla la presenza di un file di rubrica nella cartella personale e, se lo trova, ne carica il contenuto. Inoltre, se trova un file di disconnessione non notificata al server, lo legge e tenta di comunicarlo; se ci riesce, elimina il file.

Quando un utente è connesso su un device dispone di una lista di comandi aggiuntivi. `hanging` chiede al server di accedere al proprio spazio personale di messaggi in attesa di essere letti (mostra il contenuto della cartella `./hanging/<nome_utente>/`). I files che vi si trovano, chiamati come il nome dell'utente mittente, contengono i messaggi che non sono stati ancora letti. Tramite il comando `show` (invio al server di un messaggio con codice `SHOW:`) su uno di quei nomi, l'utente legge i messaggi che non gli erano stati recapitati e il device li appende allo storico dei messaggi nella chat con quell'utente.

`chat` permette di entrare nella chat con un utente presente nella rubrica, a prescindere che questo sia connesso. Viene quindi mostrato lo storico dei messaggi che coinvolgono questo utente e, dopo un separatore, quelli che gli sono stati inviati ma non recapitati, quindi che sono salvati sul server in attesa di essere letti. Quando il server notificherà della lettura di quei messaggi, verranno automaticamente spostati nello storico dei messaggi letti.

Il prompt dei comandi si modifica se si sta chattando con qualcuno per mostrare il nome degli utenti destinatari.

Tramite il comando `\q` si esce dalla chat. Il comando `\u` mostra lo status di tutti gli utenti presenti in rubrica. In particolare il device esegue una richiesta `ISONL` sul server, affinché questi gli comunichi se un utente è connesso (risposta con codice `ONLIN` contenente la porta su cui l'utente è raggiungibile), disconnesso (risposta con codice `DSCNT`) o sconosciuto, ossia non registrato sul server (risposta con codice `UNKWN`). Tramite il comando `\a` è possibile aggiungere un utente alla lista dei destinatari, solo se questo è attualmente connesso. Infine il comando `share` permette di condividere un file con tutti i destinatari. La comunicazione non coinvolge il server, ma se il device non dovesse conoscere la porta su cui raggiungere il destinatario, effettuerà prima una richiesta di risoluzione della porta sul server. Per evitare di bloccare l'input in attesa del trasferimento di files di grosse dimensioni, la condivisione viene effettuata su un nuovo processo per ogni destinatario. Alla ricezione di un file da parte di un altro device, questo viene salvato nella propria cartella con il nome `copy-<nome_originale>`

All'interno della chat è sufficiente digitare del testo seguito da invio per inviare una richiesta di inoltro del messaggio al server per tutti i destinatari

della chat. Se l'inoltro va a buon fine per tutti i destinatari, il messaggio viene aggiunto allo storico dei messaggi di ognuno dei riceventi (un file nella cartella personale con il nome del destinatario `./<nome_utente>/<destinatario_chat>`), altrimenti viene salvato in un file di cache dei messaggi inviati e non letti (file nella cartella personale nella forma `./<nome_utente>/chached-<destinatario_chat>`, in attesa della notifica di lettura da parte del server

Nel caso in cui il server non fosse raggiungibile per l'inoltro del messaggio, il device cercherà di contattare manualmente gli altri utenti tramite le informazioni che ha a disposizione. Fallisce nel caso in cui non conosca la porta del destinatario, in quanto non può fare affidamento sul server per ottenere questa informazione.

Gestione del IO Tramite la primitiva `select`, l'applicazione sfrutta il multiplexing dell'IO con possibilità di timeout per gestire i comandi provenienti da tastiera e dai sockets senza effettuare un'attesa attiva. La gestione di richieste provenienti sul protocollo TCP viene svolta su un processo separato, così che non si rischi di bloccare l'input da terminale.

Network utility L'applicazione fa affidamento sia sul protocollo TCP per la comunicazione affidabile dei messaggi, sia sul protocollo UDP per l'invio di segnali di notifiche relativamente allo stato di connessione dei devices. Tutte le comunicazioni seguono lo schema `CODICE-LUNGHEZZA-MESSAGGIO`. Viene prima di tutto inviata una stringa da 5 caratteri(+1 null byte) che definisce la natura del pacchetto, successivamente viene trasferito un valore numerico pari alla lunghezza del messaggio in byte ed infine, se la lunghezza è diversa da zero, viene inviato il messaggio vero e proprio.