Documentazione progetto Reti Informatiche

Università di Pisa – Anno accademico 23/24

Lorenzo Monaci [620826]

L'applicazione realizzata implementa il gioco dell'escape room, utilizzando l'approccio client-server.

Più client possono connettersi contemporaneamente e giocare la propria partita in maniera indipendente gli uni dagli altri.

Il progetto è stato realizzato adottando unicamente il protocollo TCP per la comunicazione, in quanto la comunicazione tra client e server è di tipo *loss intolerant*, dato che la perdita di informazioni può compromettere la partita.

Segue una breve analisi delle scelte progettuali adottate rispettivamente nel client e nel server.

Server

- Per gestire contemporaneamente l'attesa di nuove connessioni e l'input da tastiera per il comando di stop è stato usato l'**I/O multiplexing**, così da poter fermare il server in qualunque momento
- La gestione delle richieste di connessione è gestita in modo **concorrente**, creando un processo figlio che gestisce la partita di un particolare utente connesso. Questa scelta è stata fatta per sfruttare la semplicità del programma che risponde al client, evitando di occupare troppa memoria e appesantire la macchina con tanti processi molto pesanti. Il principale **difetto** di questa scelta è che per un numero grande di client connessi ci saranno molti processi che girano sulla macchina.
- Le informazioni degli utenti registrati sono mantenute in un **file di testo** *"auth.txt"* gestito dal server e modificato in fase di registrazione.
- Le **credenziali** sono salvate in una struttura dati *credenziali*.
- Le informazioni delle stanze sono contenute in una **struttura** *stanza* che contiene tutte le informazioni necessarie allo svolgimento della partita. Dopo l'inizializzazione, il server procede con l'invio dell'intera struttura al client, utilizzando un protocollo di tipo *text*.

Client

- La registrazione degli utenti **non** prevede **meccanismi di sicurezza** applicati al formato delle credenziali, le quali sono trasmesse in chiaro e non vi sono vincoli ad esempio sulla lunghezza minima della password.
- Anche il client memorizza le informazioni della stanza in una struttura di tipo *stanza*, utilizzata come riferimento e mai modificata.
- Segue un *if-else statement* nel quale il client gestisce l'invio di comandi al server, che servono solo come notifica, dato che la logica di ogni comando è implementata *client-side*.
- È stato implementato un comando extra: *"drop object"*, il quale permette di lasciare un oggetto per far si che se ne possa prendere un altro quando la capacità massima è stata raggiunta.

- I comandi sono inviati al server sotto forma numerica, come costanti definite nel file *“costanti.h”*.

Shadowman

La funzionalità a piacere implementata è l’introduzione di un terzo host al quale si connette il client, per sfidarlo tramite il comando *diceroll*. Lo shadowman lancia un dado e il client deve indovinare che numero uscirà, se ci azzecca gli verrà regalato del tempo altrimenti gli verrà sottratto.

Si comporta come un server verso il client, quindi anche per lui si utilizzano insieme **I/O multiplexing** e primitiva **fork**.

Librerie

Come accennato in precedenza, si fa uso di 3 file di libreria: *costanti.h, strutture.h* e *funzioni.h* (i nomi bastano a spiegare cosa contengono).

Traffico generato e scalabilità

La maggior parte delle comunicazioni più corpose si hanno in fase di scelta della stanza, quando si invia l’omonima struttura, poi seguono trasmissioni di interi quindi in questa fase il traffico si attenua.

La scalabilità dell’applicazione va in base alle prestazioni della macchina su cui è installato il server, dato che come accennato prima si generano molti processi per un numero elevato di utenti connessi.

Avvio

Per avviare l’applicazione si fa utilizzo del comando *make* con parametro *start*, il comando dopo aver compilato tutti e 3 i file sorgente lancia in 3 terminali diversi *(in ordine)* il server, lo shadowman e il client, quest’ultimo attende ciclicamente che il server sia fatto partire con il comando *start* tentando di connettersi ogni secondo.

Il codice sorgente contiene commenti che spiegano e delimitano le varie sezioni per una migliore leggibilità.