Relazione Progetto – Trivia Quiz con Architettura Client/Server

Studente: Alessio de Dato – Matricola: 635256
Corso: Reti Informatiche – A.A. 2024/2025

1. Introduzione

Il progetto sviluppato implementa un sistema di *Trivia Quiz* multiutente basato su architettura client/server, con comunicazione affidabile su socket TCP. Gli utenti, attraverso il client, possono collegarsi al server, selezionare un tema di domande e partecipare a una classifica globale costantemente aggiornata.

La finalità principale è la realizzazione di un sistema robusto e modulare, in grado di gestire in parallelo più connessioni concorrenti, garantendo coerenza delle strutture dati condivise e un'esperienza di gioco interattiva e reattiva.

2. Comunicazione Client–Server

È stato adottato il protocollo TCP con socket bloccanti, privilegiando l'affidabilità della trasmissione rispetto all'efficienza.

- Pregio: il TCP garantisce consegna ordinata e integrale dei pacchetti, consentendo inoltre di rilevare in maniera tempestiva la chiusura della connessione da parte di un interlocutore.
- Difetto: l'uso di buffer a lunghezza fissa (per nickname, domande e risposte) introduce inevitabili sprechi di byte, risultando meno efficiente rispetto a un protocollo con header e payload variabile.

I comandi speciali Mostra Punteggio e Fine Quiz sono sempre disponibili, anche durante una sessione di domande. Questa scelta aumenta la flessibilità dell'applicazione, ma comporta maggiore complessità nella gestione del protocollo.

3. Struttura del Server

Il server (server.c) è implementato secondo un modello multithread, con un massimo di 8 client simultanei.

3.1 Caricamento dei temi

All'avvio, il server scansiona la cartella qa/ e carica dinamicamente i file .txt disponibili. Ogni file rappresenta un tema di quiz, costituito da 5 coppie domanda–risposta. Questo approccio semplifica la manutenzione: per aggiungere un nuovo tema è sufficiente inserire un file nella cartella, senza modificare il codice.

3.2 Gestione dei client

Il server mantiene un array di slot giocatori (GiocatoreStato), ognuno associato a un thread. Ogni slot conserva il nickname e il tema attualmente in corso. L'accesso concorrente all'array è protetto tramite mutex.

3.3 Classifiche

Le classifiche sono implementate come liste doppiamente concatenate (NodoPunteggio) e mantenute aggiornate dinamicamente: i record vengono ordinati per punteggio e, in caso di parità, per tempo di completamento.

- Pregio: l'aggiornamento incrementale consente una classifica sempre coerente e immediatamente consultabile.
- Difetto: la gestione delle liste è più complessa rispetto a strutture array-based, e l'uso intensivo di mutex può introdurre overhead crescente con l'aumentare del numero di client.

3.4 Shutdown controllato

Lo spegnimento del server avviene in maniera pulita premendo Q da console. Vengono chiuse tutte le connessioni attive, terminati i thread e rilasciate le risorse, garantendo la consistenza delle strutture dati.

4. Struttura del Client

Il client (client.c) costituisce l'interfaccia utente testuale.

4.1 Login e nickname

L'utente inserisce un nickname, verificato dal server per garantirne l'univocità. Il nickname viene memorizzato localmente e può essere riutilizzato con invio a vuoto.

4.2 Menu e selezione temi

Il client riceve l'elenco dei temi disponibili dal server, ma visualizza solo quelli non ancora completati, sfruttando una bitmask (mask_done).

4.3 Gestione input

L'input è gestito da funzioni dedicate, con utilizzo di select() per intercettare sia l'input utente sia eventuali shutdown del server. L'utente può in qualsiasi momento richiamare la classifica o terminare la sessione.

4.4 Storico locale

Il client mantiene uno storico locale delle categorie completate nella sessione, permettendo all'utente di visualizzare i punteggi conseguiti.

- Pregio: interfaccia robusta e resiliente a disconnessioni.
- Difetto: i dati non sono persistenti, poiché non vengono salvati su file.

5. Analisi critica complessiva

Le scelte progettuali risultano coerenti con gli obiettivi formativi del corso:

- Il modello multithread garantisce semplicità e parallelismo, ma non scala bene su scenari reali con centinaia di client; un approccio event-driven o basato su pool sarebbe più efficiente.
- I buffer a lunghezza fissa semplificano il protocollo, riducendo la complessità implementativa, ma sono poco efficienti e non adatti a sistemi in produzione.
- Le classifiche in memoria consentono aggiornamenti immediati, ma la mancanza di persistenza limita l'utilità pratica.
- L'interfaccia testuale soddisfa i requisiti minimi e garantisce stabilità, ma la mancanza di una GUI limita la fruibilità per utenti non tecnici.

6. Conclusioni

Il progetto implementa un sistema distribuito conforme alle specifiche: gestione concorrente, classifiche dinamiche, comunicazione affidabile. Si distingue per la robustezza del protocollo e per la modularità delle componenti.

I limiti principali riguardano scalabilità, persistenza dei dati ed efficienza della comunicazione. In prospettiva, il sistema potrebbe essere ampliato con:

- persistenza delle classifiche su file o database,
- protocollo ottimizzato con pacchetti a lunghezza variabile,
- interfaccia grafica o web-based.