Trivia Quiz Multiplayer

Progetto di Reti Informatiche A.A. 2024/2025

Perroni Asia

Implementazione delle dinamiche di gioco

Trivia Quiz Multiplayer offre le funzionalità descritte nelle specifiche, perciò questa doumentazione si focalizza solo sui dettagli implementativi ritenuti più importanti.

È stata imposta una lunghezza massima in termine di numero di caratteri ai nickname (15), alle singole descrizioni dei temi (63) e alle singole domande dei quiz (255). Inoltre, poiché le domande sono pensate per avere una risposta secca e univoca, esprimibie in una sola parola o numero, questa deve avere una lunghezza massima di 63 caratteri. Se l'utente usa più parole per rispondere, solo la prima viene inviata al server per la verifica. Il terminatore di stringa è escluso dal conteggio. Si è ritenuto che questi limiti fossero più che sufficienti per lo scopo del gioco, e sono stati definiti per dimensionare adeguatamente i buffer per la lettura di input e file e quelli per invio e ricezione dei messaggi.

Si è prevista una connessione simultanea di non più di 20 giocatori. In base a ciò è stato anche dimensionato il buffer per lo scambio di messaggi, grande 1024 byte, abbastanza per far sì che tutti i diversi tipi di informazioni possano essere scambiati con un unico invio.

I principali controlli sull'input sono sul nickname e sul tema scelto. Il nickname non deve superare la lunghezza massima e non può essere vuoto, ovvero deve contenere almeno un carattere non spazio. Il tema deve corrispondere all'ID di un tema effettivamente disponibile. Questi controlli vengono fatti in primo luogo nel processo client, per non creare traffico inutile verso il server. Poi vengono ripetuti anche dal processo server, come precauzione per proteggersi da eventuali messaggi malformati inviati a causa di un comportamento imprevisto del client, che potrebbe essere dovuto ad un bug non intercettato o ad una manomissione intenzionale del software. Infatti tali eventualità potrebbero portare anche ad errori critici nell'applicazione server, come letture o scritture ad indirizzi di memoria non accessibili.

Implementazione del networking

Per gestire le richieste degli utenti, è stato implementato un server iterativo con multiplexing I/O, utilizzando la primitiva *select()*. Si è considerato che l'applicazione ha un carico computazionale limitato e interagisce con il file system solo per letture brevi. Inolre è basata su interazioni con l'utente che hanno intervalli sufficientemente grandi le une dalle altre, poiché si suppone che il giocatore impieghi qualche secondo per pensare alle risposte del quiz. Per questo motivo si è ritenuto che un server iterativo, anche se non permette di parallelizzare più operazioni come farebbe invece uno multiprocesso, fosse adeguato. Un altro vantaggio rispetto ad un server multiprocesso è non dover gestire l'accesso concorrente alle strutture dati, in particolare alla classifica, che è l'unica forma di interazione che gli utenti hanno tra di loro, ma che può venire aggiornata piuttosto frequentemente.

Lo scambio di messaggi avviene attraverso socket TCP. In primo luogo perché l'applicazione non è tollerante alla perdita di pacchetti: ogni messaggio è necessario per far proseguire lo stato del gioco; poi anche per la necessità di mantenere l'utente aggiornato in caso di eventuali problemi di connessione o dell'arresto del server, come richiesto dalle specifiche. Un altro vantaggio rispetto al protocollo UDP, dal punto di vista del server, è che, non essendoci garanzie che l'utente notifichi la chiusura della connessione tramite il comando *endquiz*, si rischia di dover mantenere in memoria strutture dati inutilizzate, in particolare tutti i record della classifica.

I socket sono impostati in modalità non bloccante in quanto può succedere in Linux che un socket venga marcato come pronto, ma poi in realtà non ci siano ancora dati da leggere, quindi il processo si bloccherebbe (https://man7.org/linux/man-pages/man2/select.2.html#BUGS).

Formato dei messaggi

I messaggi che client e server si scambiano hanno il seguente formato:

TIPO	DIMENSIONE	FLAG	PAYLOAD

- *Tipo*: unsigned integer da 1 byte. Usato per interpretare il contenuto del messaggio (*payload*), determinare le azioni da intreprendere alla ricezione e il tipo della risposta.
- *Dimensione*: unsigned integer da 2 byte. È significativo solo nell'invio della classifica, però è fondamentale visto che questa viene serializzata in record di dimensione fissa, ma numero variabile. L'applicazione client legge questo campo per sapere quanti byte del payload contengono effettivamente l'informazione.
- *Flag*: unsigned integer da 1 byte. Utilizzato solo nell'invio delle domande del quiz, si veda la tabella sotto per la spiegazione.
- *Payload*: da 0 a 1020 byte. Dati inviati.

Messaggi inviati dal server

Tipo	**Quando**	**Descrizione**
MAX_CLI_REACH_T	Appena viene stabilita la connessione	Se il limite massimo di connessioni è già stato raggiunto, il server accetta la connessione, invia un messaggio di questo tipo per avvisare l'utente e poi chiude la connessione.
CONNECT_OK_T	Appena viene stabilita la connessione	La connessione del client è stata accettata, inviato come primo messaggio in tutti i casi tranne quello visto sopra.
NICK_UNAVAIL_T	Registrazione dell'utente	Nickname non disponibile. Risposta all'invio di un nickname scelto, se questo è già preso da un altro utente.
THEME_LIST_T	Registrazione dell'utente	La lista dei temi, letta dal file *temi.txt* e copiata tale e quale nel payload, viene inviata quando l'utente sceglie un nickname valido.
QUESTION_T	Svolgimento del quiz	La domanda del quiz letta da *domande.txt* a cui il giocatore deve rispondere. Il campo *flag* è usato per segnalare l'esito della risposta precedente: • FIRST_QST: se a 1, è la prima domanda del tema e indica che il bit che segnala l'esito non è significativo; • PREV_ANS_CORRECT: a 1 se la risposta alla prececedente domanda era corretta, a 0 altrimenti; • NO_QST: se a 1, il messaggio serve solo a recapitare l'esito della risposta precedente, che era l'ultima del quiz. Il payload è vuoto.
RANK_T	Sessione di gioco	Risposta al comando *show score* dell'utente. Il payload è organizzato in record dalla struttura fissa: *<tipo><nickname><punti>*, con *tipo* e *punti* codificati come interi positivi su un byte e *nickname* come array di caratteri da 16 byte. Il client che lo riceve sa che deve stampare un numero di record pari a *dimensione / 18*.

Messaggi inviati dal client

Tipo	**Quando**	**Descrizione**
NICK_PROPOSITION_T	Registrazione dell'utente	Contiene il nickname, inviato al server per la verifica dell'unicità e, se questa va a buon fine, per la registrazione.
THEME_CHOICE_T	Prima di iniziare un quiz	ID del tema scelto, codificato come intero positivo su un byte.
ANSWER_T	Sessione di gioco	Risposta all'ultima domanda ricevuta dal server.
ENDQUIZ_T	Sessione di gioco	Quando l'applicazione client legge dall'input il comando *endquiz*, invia questo messaggio, con payload vuoto. Quando il server lo riceve, chiude la connessione.
SHOW_SCORE_T	Sessione di gioco	Quando l'applicazione client legge dall'input il comando *show score*, invia questo messaggio, con payload vuoto.