Documentazione Progetto Reti Informatiche

Lorenzo Grassi – A.A. 2023-2024

Server

Il server è strutturato con tecnologia concorrente: può gestire sia più partite diverse in contemporanea, sia più giocatori all'interno della stessa partita (fino a un massimo, al momento settato a 5 ma facilmente espandibile). Ogni volta che viene eseguito il comando start, che crea una sessione di gioco, viene eseguita una fork: ogni sessione contiene una partita e ha una porta diversa, i client devono connettersi alla porta giusta (la porta standard è la 4242). Il server continua ad ascoltare lo standard input e può creare nuove sessioni anche nel mentre che una partita è in corso.
Per ogni sessione vengono poi creati vari thread: in particolare oltre al thread padre, abbiamo un thread per ogni giocatore (che gestisce la connessione col rispettivo client e gestisce la comunicazione durante tutta la partita), un thread per gestire il timeout, e un thread per gestire la vittoria.
Tutti i thread condividono un'area di memoria e utilizzano diversi semafori per sincronizzarsi: questa memoria condivisa contiene tutte le informazioni della partita in corso e i dati giocatori, che sono tutti gestiti dal server.
Quando la partita comincia, il thread del timer si occupa di terminare tutti i thread dei giocatori che non si sono connessi, poi esegue una sleep, e si risveglia quando è scaduto il tempo per comunicare la fine della partita a tutti i client connessi. Il thread della vittoria viene risvegliato dal thread del giocatore che conquista l'utlimo token.
L'utilizzo di più thread permette di sfruttare meglio le potenzialità delle cpu moderne e rende il progetto più scalabile.

Client

I client sono strutturati in maniera piuttosto semplice, e utilizzano un thread aggiuntivo, oltre al principale. Dopo aver eseguito con successo la connessione al server e il login, inizia la partita e viene creato il thread aggiuntivo, mentre il principale non fa altro che aspettare un comando da standard input e inviarlo al server, per poi bloccarsi in attesa di essere risvegliato una volta che l'altro thread, che si occupa della ricezione di dati dal server, riceve la risposta a quel comando.
Il thread aggiuntivo si occupa anche degli enigmi, sfruttando l'I/O multiplexing per gestire un'eventuale ricezione di byte asincroni dal sevrer (spiegati meglio più avanti).
Questa struttura rende il client molto semplice e progettata in modo tale da richiedere poche risorse e occupare poco spazio sulla macchina.

Comunicazione

Il progetto utilizza il protocollo di livello trasporto TCP, per fare in modo di preservare l'affidabilità della comunicazione, e anche perché non è importante avere una velocità di trasferimento elevata, in quanto la quantità di dati trasferiti è molto piccola.
Il protocollo di livello applicazione, proprietario, utilizza sia messaggi di tipo binary che di tipo text.
Una volta stabilita la connessione tra un client e il server, il server aspetta che il client invii una stringa di caratteri, di lunghezza nota, contenente le informazioni di login, alla quale il server risponde con un byte che rappresenta se il login è andato a buon fine: in caso negativo il server continua ad aspettare finché il client non gli manda i dati di un'account valido.
A questo punto, il server manda un byte al client, che identifica l'id del giocatore: il client si comporta diversamente a seconda che il byte indichi giocatore 1 o no.

Il client del giocatore 1, che deve selezionare la room in cui giocare, attende un byte dal server contenente il numero delle stanze, e poi una stringa (di lunghezza nota) per ogni stanza, e risponde al server con un byte, che rappresenta la scelta del giocatore.
Una volta che il server riceve l'id della stanza, carica in memoria i dati di quella stanza e invia a tutti i client un byte di ok, e due byte per informare i client della lunghezza della partita (in secondi).
A questo punto è iniziata la partita vera e propria e il protocollo di comunicazione è il seguente: il server è sempre in ascolto per una stringa di testo, di lunghezza nota, contenente il comando, una volta ricevuta, viene analizzata e interpretata, e viene mandata una risposta.
Tutte le risposte del server durante la partita iniziano con un messaggio di un byte che indica il tipo di risposta.
Si distinguono tre tipi: i byte asincroni, che sono quelli inviati dal thread del timer o dal thread della vittoria, vengono inviati non come risposta a un comando, e quando il client li riceve la partita termina. Il client deve sempre essere in ascolto perché questi messaggi arrivano senza preavviso. Hanno codice 0xFF e 0xFE.
I messaggi di risposta normali, indicati da codice 0x01, contengono, dopo il byte di codice, un byte che indica quanti token sono stati raccolti e quanti rimanenti (entrambi su 4 bit), e poi un messaggio testuale di lunghezza nota. I messaggi di questo tipo non possono essere interrotti da byte asincroni (grazie a semafori).
I messaggi che rappresentano un enigma hanno codice 0x02, e sono seguiti da una stringa di testo, di lunghezza nota, che rappresenta la domanda dell'enigma. Una volta ricevuta la domanda, il client aspetta da standard input la risposta da inviare al server, per poi inviarla come messaggio di testo, sempre di lunghezza nota. Il server manda poi un byte che indica se la risposta è giusta o sbagliata. Quando il client è in attesa che l'utente scriva la risposta all'enigma, il sevrer potrebbe mandare un byte asincrono: come accennato sopra, questa eventualità è gestita tramite I/O multiplexing.
Il protocollo di comunicazione usato è organizzato in modo da poter aggiungere nuove feature, o nuove room, senza dover cambiare il client: i comandi sono interpretati e gestiti dal server e il client, solitamente, si limita a trasferire al server le stringhe scritte dall'utente e a mostrare le risposte sullo standard output, a eccezione del byte di codice e del byte dei token. Un aggiornamento al client è quindi necessario, ad esempio, nel caso in cui si voglia introdurre un nuovo tipo di messaggio da parte del sevrer.

Funzionalità a piacere

La funzionalità a piacere consiste nel fatto che più giocatori possono connettersi e giocare insieme nella stessa stanza: viene gestita l'interazione simultanea di più client con gli oggetti e gli enigmi presenti. Tutte le comunicazioni passano attraverso il server, che si occupa di gestire i dati e le comunicazioni con tutti i client attraverso i vari thread.