Relazione Progetto: P2P Kakuro

Corso di Sistemi Distribuiti

Introduzione

Questo progetto consiste nella realizzazione di un gioco multiplayer del Kakuro, un puzzle logico-matematico simile al Sudoku. La particolarità di questa implementazione risiede nell'architettura scelta: invece di utilizzare un tradizionale modello client-server con un server centrale che gestisce tutte le partite, ho sviluppato un sistema completamente decentralizzato basato su rete peer-to-peer e DHT (Distributed Hash Table).

Il Kakuro è un gioco dove il giocatore deve inserire numeri da 1 a 9 nelle celle bianche, facendo in modo che la somma dei numeri in ogni gruppo orizzontale o verticale corrisponda all'indizio presente nella cella nera adiacente. La sfida aggiuntiva è che non si possono ripetere numeri all'interno dello stesso gruppo.

Architettura Peer-to-Peer

Nei sistemi tradizionali, quando più utenti vogliono giocare insieme, si connettono tutti a un server centrale che coordina la partita. Questo approccio, sebbene semplice da implementare, presenta diversi problemi: se il server si guasta, nessuno può più giocare; il server deve essere mantenuto e pagato; e con molti utenti il server può diventare un collo di bottiglia.

Con l'architettura peer-to-peer che ho implementato nella classe P2PNetworkManager, questi problemi vengono eliminati. Ogni giocatore che avvia l'applicazione diventa un nodo della rete, capace di comunicare direttamente con gli altri giocatori senza passare attraverso un intermediario. I dati delle partite non sono salvati su un server, ma sono distribuiti tra tutti i giocatori connessi alla rete tramite la libreria TomP2P. Questo significa che se un giocatore si disconnette, gli altri possono continuare a giocare senza problemi. Non esiste un singolo punto di fallimento che possa bloccare tutto il sistema.

Distributed Hash Table

Il cuore del sistema è la DHT (Distributed Hash Table), gestita dalla classe DHTOperations. Questa struttura dati permette di salvare e recuperare informazioni in modo distribuito. Funziona così: quando voglio salvare i dati di una partita chiamata "Partita1", il sistema calcola un hash usando il metodo Number160.createHash() che implementa l'algoritmo SHA-1. Questo hash determina su quale nodo della rete verranno salvati i dati.

Quando un altro giocatore vuole unirsi a "Partita1", calcola lo stesso hash tramite il metodo createChallengeKey() e sa esattamente dove trovare i dati, senza dover chiedere a un server centrale. La bellezza di questo sistema è che i dati sono automaticamente distribuiti in modo uniforme tra tutti i nodi della rete. Nel mio progetto, la DHT memorizza tre tipi di informazioni tramite chiavi specifiche: la lista dei giocatori attualmente connessi (LOGGED_PLAYERS_KEY), la lista delle sfide pubbliche disponibili (PUBLIC_CHALLENGES_KEY), e i dati di ogni singola partita in corso.

Gestione della Consistenza

Un problema tipico dei sistemi distribuiti è la consistenza dei dati. Immaginiamo che due giocatori inseriscano un numero nella stessa cella nello stesso istante: chi vince? Senza un meccanismo di controllo, uno dei due aggiornamenti potrebbe andare perso.

Per risolvere questo problema ho implementato un sistema chiamato vDHT (versioned DHT) nella classe DHTOperations. Ogni oggetto GameSession ha un campo version che viene incrementato ad ogni modifica tramite il metodo incrementVersion(). Quando un giocatore vuole aggiornare i dati della partita, il metodo putWithVersion() prima controlla se la versione locale corrisponde all'ultima versione salvata nella rete. Se qualcun altro ha già fatto una modifica, il sistema rileva il conflitto, ricarica i dati aggiornati e riprova l'operazione.

Questo meccanismo, chiamato controllo ottimistico della concorrenza, garantisce che nessun dato venga mai perso, anche quando più giocatori agiscono contemporaneamente. Il metodo placeNumber() nel P2PNetworkManager utilizza un ciclo di retry che tenta fino a 3 volte in caso di conflitti di versione.

Funzionamento del Gioco

All'avvio dell'applicazione nella classe KakuroApp, il giocatore inserisce un nickname per identificarsi. Il sistema verifica che il nickname non sia già utilizzato controllando la lista dei giocatori nella DHT. Successivamente accede alla lobby, dove può vedere tutte le sfide pubbliche disponibili oppure crearne una nuova con tre livelli di difficoltà: EASY (6x6), MEDIUM (8x8) o HARD (10x10).

Le sfide sono gestite dalla classe GameSession che può assumere tre stati: WAITING (in attesa), RUNNING (in corso), FINISHED (terminata). Il creatore di una sfida ne diventa il proprietario (ownerNickname) e ha il compito di avviare la partita quando ci sono almeno due giocatori. Durante la partita, ogni giocatore ha la propria copia del tabellone (playerBoards) e vede solo i numeri che ha inserito lui stesso. Il sistema di punteggio è competitivo: chi trova per primo un numero corretto guadagna un punto. Il set foundCells tiene traccia delle celle già scoperte, garantendo che inserimenti successivi dello stesso numero corretto non assegnino punti. La partita termina quando un giocatore completa tutto il tabellone.

Tecnologie Utilizzate

Per la realizzazione del progetto ho utilizzato Java 11 come linguaggio di programmazione. La rete peer-to-peer è gestita dalla libreria TomP2P, una delle più complete implementazioni di DHT disponibili per Java che fornisce le classi PeerDHT, PeerBuilder e le operazioni di put/get. L'interfaccia grafica è stata sviluppata con Java Swing, utilizzando il tema FlatLaf (FlatDarkLaf) per ottenere un aspetto moderno e scuro. Per la gestione dei parametri da linea di comando (-ma, -mp, -lp) ho usato Args4j, mentre il sistema di logging è basato su SLF4J e Logback. Il progetto è gestito con Maven, che si occupa della compilazione e della gestione di tutte le dipendenze esterne definite nel pom.xml.

Conclusioni

Questo progetto dimostra come sia possibile realizzare un'applicazione multiplayer completamente decentralizzata, applicando concretamente i concetti di sistemi distribuiti. L'architettura peer-to-peer elimina la necessità di un server centrale, rendendo il sistema più resiliente, scalabile e economico da mantenere. La sfida principale è stata gestire la consistenza dei dati in un ambiente distribuito, problema che ho risolto implementando un sistema di versioning (vDHT) che garantisce l'integrità delle informazioni anche in presenza di operazioni concorrenti. Il risultato finale è un gioco funzionante che permette a più persone di sfidarsi su puzzle Kakuro, comunicando direttamente tra loro attraverso una rete decentralizzata, senza dipendere da alcun server centrale.

Autore: IVANA TOLVA Matricola: 338676

Anno Accademico: 2025/2026