Distributed Battleship

Progetto di Sistemi Distribuiti
2025/2026

Viola Leonardo - 0000001199015

Obiettivo e Requisiti Principali

Scopo del Progetto e Vincoli di Sistema

- **Obiettivo del Progetto:** Sviluppare un'applicazione di gioco distribuita (*Battaglia Navale*) con interfaccia grafica, capace di garantire la massima continuità di servizio attraverso meccanismi trasparenti di **Fault Tolerance** e **sincronizzazione dello stato volatile**.

- **Requisiti Funzionali Principali:**
  - ***Matchmaking Automatico:*** Gestione dell'ingresso dei giocatori in stanze casuali con abbinamento automatico 1v1.
  - ***Gameplay con Timer:*** Posizionamento delle navi e selezione dei colpi vincolati da tempo limite, con posizionamento/sparo casuale di fallback in caso di inattività.
  - ***Turnazione Dinamica:*** Alternanza dei turni con prolungamento dell'attacco in caso di colpo a segno (*Hit*).

- **Requisiti Non Funzionali Principali:**
  - ***Alta Disponibilità (Availability)****:* Tolleranza al crash del primary server tramite nodi di backup e subentro dinamico.
  - ***Rilevamento delle Disconnessioni:*** Uso di *Heartbeat* e *Timeout* per intercettare il congelamento dei nodi o la caduta accidentale dei client.
  - ***Consistency dello Stato:*** Allineamento periodico dello stato volatile tra i server di replicazione.

I Nodi del Sistema

Tipologie di nodi

- **Primary Server :** È il pilastro centrale del matchmaking e della gestione del ciclo di vita del sistema. Mantiene in memoria l'elenco dei client connessi, lo stato delle stanze e l'anagrafe dei server di backup.

- **Backup Servers:** Nodi ridondanti che non interagiscono direttamente con i client durante il normale funzionamento. Rimangono in uno stato latente di ascolto, accumulando gli snapshot dello stato inviati dal Primary. Possiedono un indice di priorità nativo che stabilisce rigidamente il loro ruolo nella catena di successione.

- **Client / Peer Nodi:** Entità che integrano una doppia natura software. Si comportano da *Client* nei confronti del server per le fase di matchmaking. Si trasformano in *Peer* simmetrici durante la conduzione del match.

L'Architettura Ibrida

Il connubio tra Client-Server, Peer-to-Peer e Cluster di Replica

- **Matchmaking Centralizzato (Client-Server):** Il sistema si appoggia inizialmente su un modello *Client-Server* puro. Il Primary Server funge da da registro globale per risolvere il problema della scoperta dei nodi (*Node Discovery*). Gestisce l'ingresso dei giocatori, crea dinamicamente le stanze di gioco e accoppia i client in base alla disponibilità.

- **Gameplay Disintermediato (Peer-to-Peer):** Una volta completato il matchmaking e scambiati i rispettivi indirizzi IP e porte tramite il server, i due client aprono un canale socket TCP diretto per scambiarsi le mosse in modalità *Peer-to-Peer (P2P)*, riducendo la latenza di transito sul server.

- **Sincronizzazione del Cluster (Primary-to-Backup):** In parallelo, il Primary Server mantiene una comunicazione costante e unidirezionale con i nodi di Backup. Ad intervalli di tempo regolari, invia in modalità *push* uno snapshot aggiornato del sistema (stanze attive e utenti) per garantire l'allineamento dello stato volatile. Ogni volta che un nuovo Backup Server si registra nel sistema o si scollega, il Primary notifica immediatamente tutti i client attivi, permettendo loro di aggiornare in tempo reale la propria lista locale per il failover dinamico.

Fault Tolerance – Lato Primary

Rilevamento dei Crash dei Backup e Gestione dello Stato

- **Monitoraggio Attivo dei Replicanti:** Il Primary Server è responsabile del mantenimento dell'integrità del sistema. Non si limita a inviare passivamente i dati, ma monitora costantemente la reattività dei nodi di backup ad esso collegati.

- **Meccanismo di Push e Riscontro (ACK):** Ad intervalli regolari e definiti, il Primary invia lo snapshot aggiornato dello stato globale (stanze attive e utenti registrati) a tutti i backup. Dopo ogni invio, il Primary si aspetta una risposta di sincronizzazione avvenuta da parte di ciascun nodo di backup.

- **Gestione del Guasto di un Backup:** Se un Backup Server va giù e smette di rispondere ai pacchetti di stato entro la finestra temporale stabilita, il Primary ne assume il crash. Di conseguenza, lo rimuove dalla lista locale delle priorità e notifica immediatamente tutti i client attivi per aggiornare la loro mappa di failover.

Fault Tolerance – Lato Backup

Heartbeat Implicito e Subentro Automatico senza Consenso

- **Flusso di Sincronizzazione come Heartbeat:** I server di backup operano in modalità passiva ma mantengono una finestra di ascolto rigorosa. L'invio periodico dello stato da parte del Primary funge nativamente da *Heartbeat implicito*: finché il pacchetto regolare perviene, il Primary è considerato attivo.

- **Rilevamento del Crash del Primary:** Se un Backup Server non riceve lo snapshot aggiornato entro l'intervallo temporale atteso, assume immediatamente che il Primary Server sia andato in crash o sia rimasto isolato a causa di una partizione di rete.

- **Failover Deterministico a Zero Overhead:** Al rilevamento del guasto, non viene avviato alcun algoritmo di elezione o consenso distribuito, azzerando i tempi di negoziazione. Il **Backup con ordine di priorità 1** sa già in modo deterministico di essere il successore diretto e si autopromuove istantaneamente a nuovo Primary Server.

Fault Tolerance – Lato Client

Migrazione Coordinata e Rilevamento dei Crash nel P2P

- **Riconnessione Automatica al Cluster:** I client, informati in tempo reale dei cambiamenti della topologia da parte del vecchio Primary, conoscono già la gerarchia dei backup. In caso di crash del Primary, non interrogano la rete in modo casuale, ma migrano allineati verso il nuovo Primary (l'ex Backup 1).

- **Il Rilevamento del Guasto nel Canale P2P:** Durante la partita, il server è escluso dal flusso. Se un client subisce un crash improvviso o un blocco di rete, il peer avversario rischierebbe di rimanere bloccato indefinitamente in attesa della mossa o della risposta al colpo.

- **Arbitrato tramite Tempo Limite (Timeout):** Per evitare il congelamento dell'interfaccia, ogni turno di gioco è vincolato da un tempo limite massimo gestito da timer asincroni. Se un giocatore non riceve messaggi dal peer entro la scadenza del timer, assume l'indisponibilità dell'avversario ed interrompe la partita.

Logging e Checkpointing

Strategie di Tracciabilità e Gestione dello Stato Globale

- **Logging Distribuito per il Debugging:** Tutti i nodi del sistema (Primary, Backup e Client) registrano sistematicamente ogni evento significativo in locale (connessioni, cambi turno, transizioni di stato e pacchetti scambiati). Questo approccio a eventi favorisce enormemente lo sviluppo, l'osservabilità e l'identificazione di bug in un ambiente concorrente asincrono.

- **Checkpointing Asincrono sul Cluster:** Dato che il backend non utilizza storage persistente, lo stato globale viene salvato tramite *checkpointing* periodico. Il Primary Server scatta periodicamente uno snapshot della memoria (stanze attive e anagrafica nodi) e lo invia in modalità *push* ai backup per garantire la consistenza di replica.

- **Il Trade-Off della Finestra di Vulnerabilità:**
  - *Lo Svantaggio:* Il backup che subentra a seguito di un failover erediterà un checkpoint "vecchio" di alcuni secondi (l'intervallo trascorso dall'ultimo invio riuscito), rischiando di perdere le ultimissime registrazioni di client o cambi stanza avvenuti in quella finestra temporale.
  - *La Soluzione:* Questo disallineamento può essere mitigato o annullato riducendo la frequenza dell'intervallo di invio dei checkpoint, bilanciando accuratamente l'overhead di rete con il livello di consistenza desiderato.

Sicurezza Semantica – Prevenzione dei Nodi Bizantini

Scambio Preventivo delle Griglie e Validazione Locale

- **Il Trade-Off tra Fluidità dell'Interfaccia e Fiducia:** Nelle architetture tradizionali "cieche", l'esito di un colpo viene calcolato dal peer che lo subisce, il che introduce latenza di rete prima che l'attaccante veda aggiornarsi la propria Interfaccia Grafica (GUI).

- **Scambio Iniziale delle Mappe:** Al termine della fase di posizionamento e prima dell'inizio della partita vera e propria, i due client si scambiano l'intera disposizione delle rispettive flotte. La mappa avversaria viene memorizzata in locale, rimanendo rigorosamente invisibile all'utente.

- **Aggiornamento Istantaneo della GUI:** Quando un giocatore seleziona una cella da colpire, il client locale calcola l'esito (*Acqua* o *Colpito*) attingendo direttamente alla memoria locale. La GUI si aggiorna istantaneamente all'atto del click, azzerando i tempi di attesa del round-trip della rete.

- **Mitigazione del Guasto Bizantino (Anti-Cheating):** Questo approccio neutralizza la minaccia di un avversario *Bizantino* (un nodo malizioso con codice modificato) che, pur di non perdere, mentirebbe sistematicamente a ogni attacco dichiarando il falso (es. rispondendo sempre "acqua" anche se colpito). Avendo la griglia avversaria pre-validata all'inizio, il client locale controlla autonomamente l'integrità semantica del match, impedendo qualsiasi alterazione opportunistica del punteggio.

Testing e Validazione del Sistema

Verifica della Robustezza e della Correttezza Distribuita

- **Strategia di Testing Ibrida:** Per garantire la stabilità del sistema, la fase di validazione ha combinato **test automatici di unità e integrazione** (via JUnit) con **test manuali guidati da scenari**, coprendo sia la correttezza logica dei singoli componenti che la resilienza alle anomalie di rete.
- **Test Automatici (JUnit):**
  - ***Logica di Dominio:*** Verifica deterministica delle regole di gioco (posizionamento valido delle navi, sovrapposizioni, calcolo dei colpi e condizioni di vittoria).
  - ***Protocollo di Comunicazione:*** Test unitari sulla corretta serializzazione e deserializzazione dei messaggi JSON scambiati tra i nodi.
  - ***Gestione dei Timer:*** Validazione dei thread asincroni per l'assegnazione dei turni e lo scatto automatico delle mosse di fallback.
- **Test Manuali per Interazioni "Critiche":**
  - ***Simulazione di Crash Improvvisi:*** Spegnimento forzato del Primary Server durante un matchmaking per verificare l'autopromozione del Backup 1 e la riconnessione dei client.
  - ***Partizioni e Disconnessioni P2P:*** Chiusura repentina del client o disattivazione della scheda di rete per testare l'intervento dei timeout e l'arbitrato del server.

Analisi delle Proprietà del Sistema Distribuito

Disponibilità, Integrità ed Evolvabilità

- **Disponibilità (Availability):** Massimizzata sul piano infrastrutturale. Il tempo di indisponibilità (*downtime*) del backend durante un failover tende a zero grazie all'assenza di negoziazione tra i backup. Lato client, la disponibilità del servizio di gioco è protetta dai meccanismi di timeout che evitano il congelamento dell'applicazione.

- **Integrità dello Stato:** Preservata sia in modo orizzontale (attraverso la sincronizzazione atomica e continua tra Primary e Backup che previene la divergenza dei dati delle stanze) sia sul piano verticale del gameplay (grazie allo scambio preliminare delle griglie che sigilla le regole semantiche del match).

- **Eterogeneità ed Evolvabilità (Modellazione Software):** L'utilizzo di messaggi serializzati in JSON standard via UTF-8 disaccoppia completamente la logica applicativa dal mezzo fisico. La netta separazione interna tra i *Connectivity Services* (gestione socket) e i *Controller* della logica di dominio permette di evolvere il sistema con facilità, rendendo possibile la sostituzione dell'interfaccia Java Swing con un'applicazione Web o Mobile senza dover riscrivere il protocollo o i server.

Sviluppi Futuri

- ***Canale di Comunicazione Accessorio:*** Introduzione di un sistema di chat testuale asincrono e parallelo sfruttando lo stesso socket TCP del P2P.

- ***Matchmaking Personalizzato (Private Rooms):*** Estensione del protocollo per supportare, oltre al join casuale, la creazione di stanze private protette da un token/codice univoco condivisibile, permettendo sfide dirette tra amici ed eliminando la totale casualità del matchmaking attuale.