Assignment 3 pt2

Pepe Riccardo

October 4, 2025

1 Introduzione

La parte due del terzo assignment consiste nel realizzare una versione distribuita del gioco Agar.io partendo da una versione del gioco non distribuita. Questo dovrà avvenire sfruttando una delle seguenti tecnologie:

- Architettura distribuita ad attori utilizzando Akka
- Architettura distribuita Message-Oriented Middleware (MOM) utilizzando un middleware come RabbitMQ

1.1 Requisiti

I requisiti per il conseguimento con successo del progetto sono i seguenti:

- Il gioco deve essere sempre attivo: i giocatori possono entrare in qualsiasi momento e iniziare a giocare
- I giocatori possono trovarsi su nodi differenti e devono essere in grado di unirsi alla partita o di uscirne dinamicamente
- Quando un giocatore consuma del cibo, questo deve essere rimosso per tutti i giocatori del sistema
- Ogni giocatore deve avere una visione consistente del mondo
- Il cibo viene generato casualmente e deve essere visibile a tutti i nodi del sistema
- Il gioco si conclude quando un giocatore raggiunge una specifica massa e questa condizione deve essere comunicata a tutti i nodi del sistema

2 Architettura

La soluzione prevede l'utilizzo di una tecnologia Message-Oriented Middleware (MOM), ovvero una tecnologia che supporta una comunicazione asincrona attraverso l'utilizzo di code gestite da un middleware. La sfida consiste nel trovare il giusto equilibrio tra la minor centralizzazione possibile del gioco e garantire la coerenza di visione della partita per tutti i nodi del sistema.

Gli eventi principali del gioco sono:

- Inizio della partita
- Generazione del cibo
- Movimento dei giocatori
- Gestione della vittoria
- Giocatore che mangia un altro giocatore
- Giocatore che mangia cibo

Alcuni eventi verranno gestiti dai singoli giocatori, mentre altri verranno gestiti in modo centralizzato da un nodo definito leader. Il leader viene scelto attraverso una elezione. Ogni nodo potenzialmente può diventare leader, e nel caso in cui questo si disconnetta dal sistema e non sia più raggiungibile avviene una nuova elezione e si prende come punto di partenza l'ultima istantanea del mondo comunicata. In questo modo si ha una visione del mondo comune a tutti i giocatori, in quanto gestita in modo centralizzato, ma allo stesso tempo la morte del nodo leader non implica la fine della partita. Verranno approfonditi i suddetti eventi nelle prossime sotto-sezioni.

2.1 Inizio della partita

L'inizio della partita è una fase cruciale, in quanto il nodo che entra nella partita deve capire se è entrato all'inizio di questa oppure in una già iniziata. Il nodo inizia a pubblicare la propria posizione, se c'è già un nodo leader allora questo memorizza l'entrata di un nuovo giocatore; il nodo appena entrato così si renderà conto di essere in una partita già avviata. Nel caso in cui non riceva per una certa quantità di tempo nessuna informazione sul mondo attuale, indice una nuova elezione dove ne uscirà leader e inizierà a comunicare a tutti i nodi che entreranno il mondo da lui generato.

2.2 Generazione del cibo

La generazione del cibo avviene per mano del leader in due occasioni:

- Quando il cibo nella mappa inizia ad essere insufficiente
- Non esiste una istantanea precedente del mondo (inizio della partita)

2.3 Movimento dei giocatori

Il movimento di ogni giocatore è comunicato direttamente dal nodo stesso a tutti gli altri giocatori del sistema. Il compito del leader in questo caso è memorizzare quanto tempo prima è stata comunicata la posizione di ogni giocatore e eliminare dalla partita i nodi che non danno comunicazioni da un tempo superiore a quello considerato limite.

2.4 Gestione della vittoria

Ad ogni ciclo il leader verifica che nessun giocatore sia arrivato alla massa target, in caso contrario viene notificata la vittoria agli altri giocatori e la partita si conclude.

2.5 Giocatore che mangia un altro giocatore

Ad ogni ciclo il leader verifica se un giocatore è entrato in collisione con un altro giocatore, in caso affermativo verifica qual è il giocatore con la massa più grande e se ci sono le condizioni per mangiare, viene notificato a tutti i giocatori.

2.6 Giocatore che mangia cibo

Ad ogni ciclo il leader verifica se un giocatore è entrato in collisione con del cibo, in caso affermativo elimina il cibo dalla rappresentazione del mondo.

3 Implementazione

Di seguito vengono elencate le tecnologie usate e l'algoritmo scelto per le elezioni del leader.

3.1 Tecnologie

Si è deciso di utilizzare il middleware RabbitMQ per mettere in comunicazione i nodi. Dato che il linguaggio di programmazione scelto è Java, si è deciso di utilizzare una libreria per serializzare e deserializzare le classi in JSON: Jackson.

3.2 Algoritmo di elezione

Per la elezione del leader si è dovuto scegliere un algoritmo: la scelta è ricaduta su Bully Alghoritm, ovvero un algoritmo che elegge come leader il nodo con l'identificatore più alto. Questo algoritmo è stato scelto per la sua semplicità di implementazione; questa scelta oltre alla semplicità comporta il risvolto che scegliendo il giusto nickname è possibile aumentare le probabilità di essere o non essere il leader.

4 Ambiente di test

Il sistema è stato provato installando con Docker su un home server RabbitMQ e eseguendo il gioco su dispositivi diversi (con sistemi operativi diversi). La soluzione si è dimostrata soddisfacente, anche se l'esperienza è condizionata dal ritardo della connessione verso l'home server: con dispositivi collegati via cavo il gioco si è dimostrato molto fluido, mentre utilizzando il wifi si riscontravano ritardi nella rappresentazione della partita.

5 Possibili migliorie

- Verificare che non esistano già giocatori con un determinato nickname
- Modificare l'algoritmo di elezione, in modo tale da non far dipendere questa decisione indirettamente dall'utente