Bajnarola

Progetto di sistemi distribuiti

Davide Berardi 0000712698 Matteo Martelli 0000702472 Marco Melletti 0000699715

30 novembre 2015

Sommario

Bajnarola e' un bel gioco

1 Introduzione

2 Aspetti progettuali

Il gioco in questione e' stato realizzato nell'ottica di un sistema resistente ai guasti, come richiesto da progetto.
Tale obiettivo e' stato raggiunto utilizzando le funzioni RMI del linguaggio di programmazione Java, la comunicazione e' stata configurata in modo da risultare allo stesso tempo sia robusta sia in grado di fornire reattivita' ai client del gioco, nonostante quest'ultimo non utilizzi uno schema in tempo reale.

2.1 Il gioco

2.2 La comunicazione

3 Aspetti implementativi

3.1 Il desing pattern MVC

3.2 Implementazione dello schema di gioco

3.3 L'interfaccia grafica

3.4 La gestione e la distribuzione della rete

3.4.1 Registrazione presso la lobby

Il primo passo svolto da ogni nodo di rete e' la registrazione presso un server centralizzato comune, implementante diverse "stanze" di gioco; le cosiddette lobby.

Questo server aspettera' quindi la registrazione del numero specificato di partecipanti, inviando loro la lista dei giocatori per poi lasciare il pieno controllo ad essi, chiudendo la stanza.

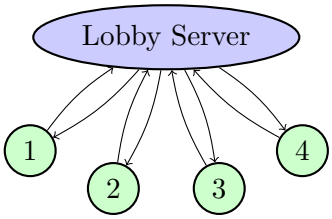


Figura 1: Registrazione presso il server lobby

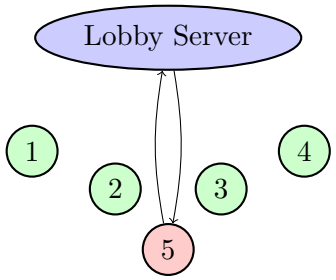
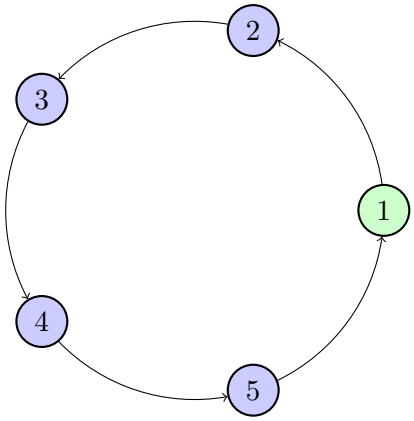
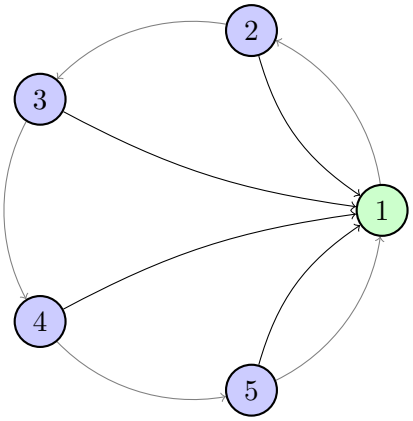


Figura 2: Eccezione del server alla richiesta di una lobby chiusa

3.4.2 Lo schema con leader dinamico.

Lo schema di distribuzione si basa sull'elezione di un leader "dinamico", questo leader e' deciso in base ad un lancio di un dado iniziale (nella realta' implementato come un random intero a 32 bit, per evitare lanci ripetuti), questo lancio viene quindi distribuito dai vari player ad ogni giocatore, creando un **ordine** di gioco, il giocatore con punteggio maggiore verra' quindi dichiarato come leader corrente, e da li a decrescere.
La topologia risulta quindi una rete monodirezionale con passaggio del testimone (token ring) per la decisione del leader corrente.





3.4.3 Aggiornamenti allo stato

3.4.4 Tolleranza ai guasti

Il sistema risulta tollerante ai guasti di tipo crash. Nel caso di un guasto di tipo crash sul nodo leader corrente, sara' il sistema ad invocare un'eccezione di tipo **RemoteException** verso i nodi interroganti, che lo elimineranno quindi dalla loro lista di interrogazione,

riconfigurando l'anello.
Nel caso di un guasto di un nodo intermedio il risultato sara' il medesimo al momento di un'interrogazione da parte dei vari nodi dell'anello.
Questo genere di configurazione mantiene coerente lo stato locale delle diverse istanze, poiche' ogni nodo aspetta la risoluzione delle varie mosse ad esso precedenti prima di essere interrogato a sua volta e poter agire.
Questo schema e' banalmente possibile utilizzando una semplice rete di tipo token ring, ma e' stato scelto di implementare il tutto come una sorta di cricca per l'aggiornamento automatico e la visualizzazione dei risultati con latenze brevi: se si fosse ponderato per una struttura completamente circolare (utilizzante lo stesso modello logico) gli aggiornamenti allo stato locale sarebbero applicati solamente dopo che il controllo (e quindi la leadership) fosse tornata al nodo richiedente, risultando in un'attesa pari ad **N-1** turni; essendo il gioco in questione un gioco di logica non propriamente reattivo e con turni di gioco potenzialmente molto lunghi e riflessivi e' stato optato per un modello piu' pesante da un punto di vista di scambio di informazioni che da un punto di vista piu' leggero come comunicazioni ma, allo stesso tempo, meno reattivo per tutti i client.

4 Valutazione

5 Conclusioni