Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Laurea Magistrale in Informatica

Corso di Sistemi Middleware - Prof. Alessandro Amoroso

Progettazione e realizzazione del Texas Hold'em Poker distribuito

Presentata da:

Fulvio Marcelli

Gianmarco Santini

Natale Vadalà

Sommario

Il progetto presentato si chiama “Distributed Texas Hold’em Poker”, e consiste nella progettazione ed implementazione di un tavolo da poker, specialità Texas Hold’em, online (nella propria LAN) e distribuito, sfruttando la tecnologia Java RMI (Remote Method Invocation). La realizzazione di tale progetto è durata per circa due mesi, nei quali chi scrive ha cercato di analizzare e valutare in primis la logica del Poker, con maggiore enfasi sulla specialità stabilita. In seguito l’analisi si è spostata su 4 macro aree, che verranno trattate in questa relazione: una prima parte prettamente di logica del Texas Hold’em; un’altra di adattamento di tale logica ad un ambiente distribuito; una terza, più di sviluppo, in cui l’attenzione si è spostata sullo sviluppo di una GUI, seppur minimale, per i giocatori/clients del nostro tavolo; la quarta, in cui abbiamo studiato e implementato delle tecniche per rilevare e risolvere i crash di un nodo del sistema. Con tale progetto si vuole fornire un’applicazione che rispecchi il più fedelmente possibile il gioco del poker, che essa sia semplice, robusta ai guasti e possa essere di aiuto a chi ha il problema e/o l’esigenza di organizzare una partita possedendo solamente una rete LAN (Ethernet o Wireless) e dei computer.

Keyword: Java, RMI, distributed, poker

Indice

Abstract **i**

1 Introduzione **1**

2 Progettazione **3**

2.1 Gioco del Poker *Texas Hold'em* 3

2.1.1 Le carte 6

2.1.2 I punti 6

2.1.3 Le scommesse 8

3 Realizzazione **9**

3.1 Struttura 9

3.1.1 Il ServerRMI 9

3.1.2 L'Evaluator 16

3.1.3 La GUI 18

3.1.4 Utility 19

3.1.5 TavoloPoker 20

3.2 Diagramma delle classi 22

3.3 Scenario 24

3.4 Esecuzione 25

4 Tolleranza ai guasti **29**

Conclusioni **33**

Elenco delle figure

3.1 Server - Diagramma delle classi UML 22
3.2 Client - Diagramma delle classi UML 23
3.3 Registrazione - Vista Server - Diagramma degli stati UML . 24
3.4 Registrazione - Vista Client - Diagramma degli stati UML . 24
3.5 Esempio esecuzione 1 25
3.6 Esempio esecuzione 2 26
3.7 Screenshot esecuzione gioco 27

4.1 Esempio gioco in esecuzione 30
4.2 Esempio fault tolerance: Crash P4 31
4.3 Esempio fault tolerance: Crash P3: giocatore corrente e ripresa dell'esecuzione 32

Capitolo 1

Introduzione

Il progetto presentato si chiama "Distributed Texas Hold'em Poker", e consiste nella progettazione ed implementazione di un tavolo da poker, specialità Texas Hold'em, online (nella propria LAN) e distribuito, sfruttando la tecnologia Java RMI (Remote Method Invocation). La realizzazione di tale progetto è durata per circa due mesi, nei quali chi scrive ha cercato di analizzare e valutare in primis la logica del Poker, con maggiore enfasi sulla specialità stabilita. In seguito l'analisi si è spostata su 4 macro aree, che verranno trattate in questa relazione: una prima parte prettamente di logica del Texas Hold'em; un'altra di adattamento di tale logica ad un ambiente distribuito; una terza, più di sviluppo, in cui l'attenzione si è spostata sullo sviluppo di una GUI, seppur minimale, per i giocatori/clients del nostro tavolo; la quarta, in cui abbiamo studiato e implementato delle tecniche per rilevare e risolvere i crash di un nodo del sistema. Con tale progetto si vuole fornire un'applicazione che rispecchi il più fedelmente possibile il gioco del poker, che essa sia semplice, robusta ai guasti e possa essere di aiuto a chi ha il problema e/o l'esigenza di organizzare una partita possedendo solamente una rete LAN (Ethernet o Wireless) e dei computer.

Capitolo 2

Progettazione

La progettazione di questo progetto è stata condotta su due fronti: il primo in cui si è studiato ed analizzato il gioco del poker, le sue regole (in particolare della specialità *"texana"*) e tutte le variabili da considerare per poter imbastire un'architettura che rappresenti fedelmente ciò che le regole descrivono; il secondo fronte, invece, ha riguardato lo studio e l'adattamento degli algoritmi e strutture dati considerate in precedenza in un'ottica distribuita, al fine di avere l'astrazione di "stato del gioco" condivisa (e coerente) per ogni client.

2.1 Gioco del Poker *Texas Hold'em*

Prima di descrivere la specialità *texana*, è giusto dare un'idea di cosa sia il c.d. *poker tradizionale*.

Il poker è una famiglia di giochi di carte nella quale alcune varianti sono classificabili come gioco d'azzardo, altre come poker sportivo. Tali giochi sono caratterizzati da un sistema di combinazioni formate con le carte di ciascun giocatore (il cui confronto determina il vincitore di ogni mano) e da un meccanismo di puntate successive che offre molte possibilità tattiche e di influenza sugli altri giocatori, consentendo in particolare di

ritirarsi con perdite contenute dalle mani che non si ritiene di poter vincere.[1]
Esistono diverse varianti di tale famiglia, raccolte in 3 categorie:

1. **Draw Poker**, cioè il poker *tradizionale* o all'*italiana* (versione con il mazzo ridotto), in cui ogni giocatore ha 5 carte, che conosce solo lui e che può cambiare, e con le quali deve formare il punto;

2. **Stud Poker**, cioè una serie di varianti in cui le carte di ogni giocatore sono sia coperte che scoperte e gli vengono date une alla volta;

3. **Community-card Poker**, cioè proprio la famiglia che comprende il *Texas Hold'em*, in cui ogni giocatore possiede delle carte personali (solitamente in numero inferiore a quelle che occorrono per ottenere un punto) e delle carte comuni scoperte, utilizzabili da tutti per comporre il punto con le propri carte.

Nel poker "classico", quello della famiglia *draw*, inizia a scommettere solo chi può "aprire", cioè chi fra le proprie 5 carte ha già un qualche punto (solitamente una "coppia vestita", cioè almeno due Fanti, due Donne, due Re o due Assi), e nel caso nessuno possa aprire ci sono diverse risoluzioni: alcuni prevedono una nuova partita, altri con l'abbassamento del punto minimo di apertura.

Nel nostro caso, invece, il gioco prevede che due giocatori, ad ogni mano, paghino obbligatoriamente la quota minima di scommessa per quel giro. Tale quota si chiama **buio** (proprio perchè la si paga senza conoscere nè le proprie carte personali, nè tantomeno quelle comuni).

Un'ulteriore differenza sta nel fatto che nel poker tradizionale si può partecipare ad un piatto solo fino a che si è coperto tutte le scommesse fatte in precendenza (nella mano), mentre nella versione texana un giocatore

[1] https://it.wikipedia.org/wiki/Poker

può partecipare a piatti (e fare *Call* a scommesse di altri che sono più alte di quello che il giocatore può pagare) anche se non può coprire la scommessa: nasce qui l'idea di **all-in**, cioè il senso di giocare tutto quello che si ha nello *stack* (i propri soldi rimanenti) e poter vincere, ovviamente, da ognuno che perde, al massimo quello che si è inserito nel piatto.
Il poker tradizionale è molto più rigido (o tutto o niente, e tendenzialmente se c'è sproporzione economica fra i vari utenti, il più ricco è molto più avvantaggiato), mentre si potrebbe dire che la versione texana è più democratica, cosa che ci permette di affermare una massima famosa fra i giocatori di poker alla texana: *fino a che hai ancora una fiche, tutto è possibile!*

Detto questo, possiamo riconoscere 3 figure fisse in una partita di *Texas Hold'em*:

1. il **dealer**, il mazziere;
2. il **piccolo buio**, cioè quello seduto dopo il dealer, e che al buio versa metà della quota minima iniziale;
3. il **grande buio**, cioè quello seduto dopo il piccolo buio e che versa per intero la quota minima iniziale.

Dopo questa breve fase dei bui, inizia la partita vera e propria.
Ogni giocatore, al suo turno, fra un giro di scommesse e un altro, può eseguire 3 azioni (mutualmente esclusive):

- **Bet**, cioè il giocatore che apre un nuovo giro di scommesse versa una quota nel piatto;
- **Call/Raise**, dopo che qualcuno ha puntato, gli altri giocatori sono obbligati a versare nel piatto almeno quanto puntato dal primo se vogliono partecipare alla mano; **Raise** nel caso si faccia un rilancio superiore alla cifra da dover *vedere* (fare *Call*);

- **Check**, se nessuno ha puntato e si vuole rimanere in gioco "passando parola" al vicino;
- **Fold**, cioè lasciare la mano.

2.1.1 Le carte

Le carte utilizzate nel poker, in qualunque sua variante, sono solitamente le carte francesi. 52 carte, 4 semi (*Cuori*♥, *Quadri*♦, *Fiori*♣ e *Picche*♠), 13 carte per seme (dall'Asso al 10 e Fante, Donna e Re in aggiunta).

2.1.2 I punti

Prima di descrivere i punti, è importante ricordare tre cose:

1. La valutazione dell'Asso dipende dal contesto: può essere considerato inferiore al 2 se lo segue in una scala o, diversamente, maggiore del K se lo precede in una scala, o se valutato singolarmente. Es. *A*♥ 2♦ 3♠ 4♣ 5♥ , l'asso vale meno del due; *A*♥*K*♠*Q*♦*J*♣10♠, l'asso vale più del K; un Poker d'assi è maggiore di un poker di K.
2. A differenza dei *Draw poker*, nella versione texana il pareggio è consentito, mentre nelle versioni classiche esiste anche una nozione di ordinamento sui semi oltre che dei valori (Cuori>Quadri>Fiori>Picche oltre che A>K>Q>J>10>...>2)
3. Tutti i punteggi sono definiti dal *punto* e dal *valore*, es. Poker(punto) di K (valore, le 4 carte sono dei K), Colore(punto) al Q(Valore, la carta più alta è il Q), Scala(punto) al 10 (Valore, la scala discende dal 10)

I punti del Texas Hold'em sono 10:

1. Scala Reale, 5 carte consecutive in valori, il cui apice è la carta dal valore massimo (cioè l'asso) e tutte le carte sono dello stesso seme (es. Cuori).
   $A\heartsuit K\heartsuit Q\heartsuit J\heartsuit 10\heartsuit$

2. Scala a colore, 5 carte consecutive in valori le quali componenti sono dello stesso seme (es. Cuori).
   $10\heartsuit 9\heartsuit 8\heartsuit 7\heartsuit 6\heartsuit$

3. Poker, cioè 4 carte dello stesso valore (es. K).
   $K\heartsuit K\clubsuit K\diamondsuit K\spadesuit 2\heartsuit$

4. Colore, 5 carte dello stesso seme ma non consecutive in valore (es. Cuori).
   $J\heartsuit 9\heartsuit 5\heartsuit 3\heartsuit 2\heartsuit$

5. Full, composto da un tris e da una coppia. (es. K e 5).
   $K\heartsuit K\clubsuit K\diamondsuit 5\spadesuit 5\heartsuit$

6. Scala, 5 carte consecutive in valori. (es. Scala al K).
   $Q\heartsuit J\clubsuit 10\diamondsuit 9\spadesuit 8\heartsuit$

7. Tris, cioè 3 carte dello stesso valore (es. K).
   $K\heartsuit K\clubsuit K\diamondsuit 3\heartsuit 2\heartsuit$

8. Doppia Coppia, 5 carte consecutive in valori. (es. K e 5).
   $K\heartsuit K\clubsuit 5\spadesuit 5\heartsuit 2\heartsuit$

9. Coppia, cioè 2 carte dello stesso valore (es. K).
   $K\heartsuit K\clubsuit Q\diamondsuit 3\heartsuit 2\heartsuit$

10. Carta Alta, in assenza di un punteggio fra quelli sopra elencati, si premia la carta più alta (es. K).
    $K\heartsuit 10\clubsuit 7\diamondsuit 3\heartsuit 2\heartsuit$

2.1.3 Le scommesse

I giri di scommesse sono 4, alternati dall'aggiunta di carte comuni scoperte sul tavolo, un po' alla volta.

1. Prima di conoscere le carte comuni, i giocatori che seguono *il grande buio* possono fare *Call*, e mettere la quota inserita dal grande buio, o *Fold* e quindi uscire, o *Raise*, cioè vedere e rilanciare.
   In seguito si gira il **Flop**, cioè le prime 3 carte comuni scoperte.

2. Dopo il *Flop*, si procede ad un giro di scommesse che inizia dal giocatore seduto dopo il grande buio (il grande buio, nonostante il pagamento al buio di una scommessa iniziale, tradizionalmente è una posizione favorita poichè gioca per ultimo).
   Tale giro di scommesse termina con la scoperta del **Turn**, cioè la 4 carta scoperta.

3. Il penultimo giro di scommesse funziona come quello precedente, e termina con il **River**, la 5 carta comune.

4. Quest ultimo giro di scommesse avviene come quelli precedenti, seguito dallo *showdown*, la dichiarazione del/i vincitori per tale mano e la spartizione del piatto.

Capitolo 3

Realizzazione

3.1 Struttura

3.1.1 Il ServerRMI

La classe *ServerRMI.java* è la classe principale del client. In questa classe vengono gestite tutte le funzioni RMI che verranno chiamate durante l'esecuzione e vengono instanziate le classi *Gui.java, Evaluate.java e Utility.java*.

Ricezione

La funzione *Ricezione()* viene chiamata dal server quando il numero impostato di client vi si è connesso. In questa funzione:

- ricevo gli indirizzi IP dei client in gioco;
- ricevo il mio ID;
- viene lanciato l'aggiornamento della grafica ogni 100 millisecondi;

- viene lanciato la fault detection (la funzione *faultTolerance()*) per tutti i giocatori in gioco ogni 100 millisecondi, per rilevare eventuali crash;
- vengono impostati il dealer, piccolo Buio e grande Buio;
- inizia la partita vera e propria.

inizia

La funzione *inizia()* viene chiamata all'inizio di ogni mano dal client che per quella mano è il grande buio.

Questa funzione inizializza il mazzo di carte (una lista di interi di 52 elementi), estrae le 5 carte comuni (estrae e rimuove 5 elementi dalla lista "mazzo di carte") e chiama, attraverso una chiamata RMI, la *PescaCarte()* passandogli il mazzo di carte e le carte comuni (cosicchè tutti i client conoscono le carte comuni fin dall'inizio, per evitare errori in caso di crash).

PescaCarte

La funzione *PescaCarte()* pesca le mie carte (estrae e rimuove 2 elementi random dalla lista "mazzo di carte") e, se sono il grande Buio (l'ultimo giocatore a parlare) di quella mano, chiama la funzione *IniziaMano()* del giocatore successivo, altrimenti chiama la *PescaCarte()* del giocatore successivo.

IniziaMano

Nella *IniziaMano()* vengono messi nel piatto il piccolo Buio e il grande Buio. Se sono il grande Buio (ultimo giocatore a parlare) di quella mano chiamo la *Gioca()* del giocatore successivo, altrimenti chiamo la *IniziaMano()* del giocatore successivo.

Punta

La funzione *Punta()* è quella funzione che viene chiamata quando un giocatore mette il piccolo buio o il grande buio nel piatto o quando fa Call o Bet. Imposta tutte le variabili interessate e notifica a tutti i giocatori attraverso un broadcast l'avvenuta puntata.

Gioca

Mentre fino a questo punto tutte le operazioni citate vengono fatte in automatico senza che l'utente possa intervenire, da questo punto in poi l'utente inizia a giocare. Infatti la funzione *Gioca()* abilita i bottoni dell'interfaccia grafica che a questo punto possono essere cliccati (vengono abilitati solo i bottoni che possono essere cliccati dall'utente in quel momento della partita. Ad esempio se il giocatore precedente ha fatto una puntata sarà abilitato il bottone Call e non quello Check). Per quanto riguarda i bottoni si rimanda alla sezione 3.1.3,**La GUI**.

Flop

La fuzione *Flop()* viene chiamata dal giocatore che è l'ultimo a parlare per il primo giro di scommesse (in caso di un Raise (bet) da parte di un giocatore l'ultimo giocatore a parlare cambia: tutti i giocatori devono rispondere (fare call o fold o rilanciare a loro volta). Il cambio dell'ultimo giocatore a parlare viene gestito dalla funzione *ImpostaUltimoGiocatore()* che notifica l'ID del nuovo giocatore che deve essere l'ultimo a parlare attraverso un broadcast) sul giocatore che sarà il primo a parlare per il giro di scommesse successivo. Questa funzione mostra le prime tre carte delle CarteComuni sul tavolo e lo notifica a tutti gli altri giocatori attraverso la funzione *DisegnaCarteTavolo()*, e chiama la funzione *Gioca()* su se stesso.

Turn

La funzione *Turn()* viene chiamata dal giocatore che è l'ultimo a parlare per il secondo turno di scommesse sul giocatore che sarà il primo a parlare per il giro di scommesse successivo. Questa funzione mostra la quarta carta delle CarteComuni sul tavolo e lo notifica a tutti gli altri giocatori attraverso la funzione *DisegnaCarteTavolo()*, e chiama la funzione *Gioca()* su se stesso.

River

La funzione *River()* viene chiamata dal giocatore che è l'ultimo a parlare per il terzo turno di scommesse sul giocatore che sarà il primo a parlare per il giro di scommesse successivo. Questa funzione mostra la quinta e

ultima carta delle CarteComuni sul tavolo e lo notifica a tutti gli altri giocatori attraverso la funzione *DisegnaCarteTavolo()*, e chiama la funzione *Gioca()* su se stesso.

MostraC

La funzione *MostraC()* viene chiamata al termine di un giro di scommesse nel caso in cui tutti i giocatori ancora in gioco sono in All-in oppure lo sono tutti tranne uno (quindi nessun giocatore può più compiere azioni utili). Questa funzione viene chiamata su tutti i giocatori ancora in gioco, e mostra le mie carte a tutti i giocatori attraverso la funzione *MostraCarte()*. Inoltre quando tutti i giocatori ancora in gioco hanno mostrato le proprie carte viene chiamata la funzione *ContinuaMano()* che prosegue la mano in automatico dal punto in cui si era arrivati senza richiedere l'intervento dei giocatori.

Valuta_punteggio

La funzione *Valuta_punteggio()* viene chiamata al termine dell'ultimo giro di scommesse. Viene chiamata su tutti i giocatori ancora in gioco che aggiungono le proprie carte in un lista. Una volta che tutti i giocatori interessati hanno aggiunte le proprie carte alla lista carteValutazione, quest'ultima viene passata alla funzione *showdown()* della classe *Evaluate.java* (vedi sezione 3.1.2, **L'Evaluator**) che restituisce il/i vincitore/i della mano. I risultati di ciascun giocatore vengono mostrati sull'interfaccia grafica attraverso la funzione *MostraRisultato()*;

foldGiocatore

La funzione *foldGiocatore()* viene chiamata quando un giocatore fa fold. Questa funzione imposta tutte le variabili interessate e notifica il fold del giocatore a tutti gli altri giocatori.

TerminaMano

La funzione *TerminaMano()* viene chiamata dalla funzione *valuta_punteggio()* e attribuisce ai giocatori le vincite corrispondenti. In caso di giocatori che rimangono con 0 nello stack (non hanno più chips) li rimuove attraverso la funzione *TerminaPartita()*. Questa funzione viene eseguita su tutti i giocatori.

ReinizializzaPartita

Quando tutti i giocatori hanno eseguito la funzione *TerminaMano()* viene chiamata la funzione *ReinizializzaPartita()* che reinizializza tutte le variabili per la nuova mano e imposta i nuovi dealer, piccolo buio e grande buio. Quando tutti i giocatori hanno effettuato la *ReinizializzaPartita()* viene chiamata la funzione *Inizia()* e la nuova mano ha inizio.

TerminaPartita

La funzione *TerminaPartita()* vinen chiamata o dalla *TerminaMano()* o dalla *FaultTolerance()* (vedi sezione 3.1.4, **Utility**) e serve a rimuovere il giocatore indicato dal gioco e reimpostare tutte le variabili interessate.

MioTurno

La funzione *MioTurno()* viene chiamata appena il giocatore riceve il controllo e avvisa tutti i giocati indicando che sta giocando lui e a che punto della partita si è arrivati. Questa funzione è utile per fare riprendere la partita nel punto corrente in caso di crash.

3.1.2 L'Evaluator

La classe *Evaluate.java* è una classe istanziata all'inizio del gioco, e fornisce i metodi per valutare il punteggio ottenuto da ogni giocatore alla fine della mano di poker, indicando quindi chi fra i clients ha vinto, pareggiato o perso. I metodi più importanti sono 2:

```
1 public static ArrayList < Integer > valuate(List < Integer >
    carte);
2 public static ArrayList < Integer > showdown(ArrayList <
    ArrayList < Integer >> players, ArrayList < Integer > common)
    ;
```

Il primo metodo *valuate(carte)* calcola la migliore combinazione di 5 carte su 7 da cui ottenere il punteggio massimo della mano di un giocatore: questo perchè, a differenza del poker classico (all' *italiana*) in cui ognuno ha le proprie 5 carte che compongono il punteggio, l'algoritmo deve scegliere la migliore mano di 5 carte su 7 (5 comuni e 2 personali). Tale valutazione funziona in modo abbastanza semplice:

1. Per ognuna delle 7 carte, ne ottengo valore e seme

2. Ordino le carte in base al valore (in ogni caso, a parità di punteggio *nominale* es. Coppia , vince quello con il valore più alto es.Coppia di Re vs Coppia di Donne)

3. A partire dal punteggio massimo (Scala Reale) passo il vettore delle carte (e le relative informazioni) al metodo di verifica del punteggio massimo (in questo caso, Scala Reale): in caso positivo, esco dalla verifica, altrimenti continuo con i controlli per ogni punteggio, fino ad arrivare al punteggio minimo (Carta Alta).

4. Se il punteggio ottenuto non basta per coprire un insieme di 5 carte (Carta Alta, Coppia, Doppia Coppia, Tris, Poker), allora provvedo a completare la mano con le migliori carte possibili non utilizzate per ottenere il punto, fino ad arrivare ad una mano da 5 carte.

Il secondo metodo, quello effettivamente chiamato dal *ServerRMI*, è la funzione di *showdown* (resa dei conti), e cioè la fase di una mano di poker in cui tutti i giocatori (in gioco) palesano le proprie carte personali e formano il punto con le carti comuni (e vince chi ha il punto massimo). La funzione *showdown* prende come parametri il vettore delle carte personali e il vettore di quelle comuni, dando come risultato il/i vincitore/i della mano. Il confronto conta 4 fasi:

1. Per ogni giocatore, chiamo la *valuate(carteComuni+cartePersonali)* e ottengo il suo punteggio.

2. Memorizzo i punteggi (e il valore di ogni punteggio) di tutti e ne valuto il migliore.

3. In ogni caso confronto i valori dei punti. In caso due o più giocatori abbiano lo stesso punteggio (che va da 1 a 10) e lo stesso valore (che va da 1 a 13, dall'Asso al Re), allora confronto le mani di entrambi fino ad arrivare a un discriminante che determini la mano migliore:
   se il punteggio è composto da meno di 5 carte (Carta Alta, Coppia, Doppia Coppia, Tris, Poker), confronto la/e carta/e rimanenti;
   se il punteggio è composto da 5 carte (Scala, Colore, Full, Scala a colore, Scala Reale), allora confronto le carte dalla più alta, fino ad arrivare a una differenza;
   se non vi sono differenze (es. le 5 carte comuni formano una Scala Reale a terra, quindi le carte personali sono inutilizzate per formare il punteggio), allora i giocatori pareggeranno e divideranno il piatto che gli spetta.

4. Ritorno un vettore con gli ID del/i vincitore/i

3.1.3 La GUI

La classe *Gui* è quella classe che implementa l'interfaccia grafica. In questa classe viene instanziata la classe *TavoloPoker.java* e vengono gestiti i vari bottoni:

- **Check :** attraverso il bottone *check* il giocatore passa il controllo al giocatore successivo senza puntare niente (lo può fare solo nel caso in cui nessuno dei giocatori che hanno già giocato il relativo giro di scommesse ha puntato più di lui).

- **Call :** attraverso il bottone *call* il giocatore mette nel piatto la parte di chips che mancano a coprire le puntate dei giocatori precedenti (possibile solo nel caso in cui siano state effettuate delle puntate superiori alla sua). Nel caso in cui la somma da aggiungere sia superiore alla somma delle chips disponibili quel giocatore aggiunge tutte le chips a sua disposizione e va in All-in.

- **Bet :** attraverso questo bottone il giocatore sceglie quale somma puntare. Se la somma selezionata dal giocatore è maggiore della sua disponibilità di chips, quel giocatore punta tutte le chips a sua disposizione e va in All-in. Se la somma selezionata è minore a quello che dovrebbe puntare per fare *Call* quel giocatore punta la somma mancante per fare *call* +1.

- **Fold :** attraverso questo bottone il giocatore lascia il gioco per questa mano.

Inoltre in questa classe sono presenti diverse funzione per mostrare a video le carte o i risultati che ricevo dalla classe *ServerRMI.java*.

3.1.4 Utility

La classe *Utility.java* è la classe che contiene alcune funzioni utili per il funzionamento del gioco ma che non sono funzioni RMI. La funzione principale di questa classe è la *FaultTolerance()* che serve a gestire gli eventuali crash che possono avvenire durante la partita. Ogni 100 millisecondi il client effettua una chiamata RMI su ogni altro giocatore, se un giocatore ha avuto un crash il client riceve un'eccezione che indica che quel giocatore non è più in gioco e chiama la *FaultTolerance()*. Quest'ultima rimuove il giocatore dal gioco attraverso la funzione *TerminaPartita()* della classe *ServerRMI.java* e, nel caso in cui il giocatore uscito dal gioco era colui che stava giocando, in base al punto del gioco in cui si è e a qual è l'ultimo giocatore a parlare fa riprendere il gioco in modo corretto. L'applicazione supporta fino a n-1 crash.

3.1.5 TavoloPoker

Se condideriamo la GUI come interfaccia grafica principale, la classe Tavolo Poker può essere considerata una sottoclasse della Gui. In particolare Tavolo Poker definisce un JPanel, una finestra grafica all'interno della GUI, questa classe è quella che contiene la maggior parte delle funzioni per il disegno a video usate per disegnare manipolare e colorare il tavolo da gioco, le carte, le chips, le animazioni. All'interno essa contiene:

- **Funzioni per il disegno:** comprendono la funzione loadImage per il caricamento delle immagini utilizzando un path; la funzione disegna, che date le coordinate e le dimensioni in altezza e larghezza scala l'immagine e la disegna nel punto indicato; infine la funzione DisegnaChips che si occupa di disegnare tutto ciò che è presente sul tavolo da gioco come carte, chips, valori delle chips, scritte indicative e piatto.
  Le coordinate utilizzate per il disegno degli elementi di gioco sono tutte state calcolate in relazione all'altezza e la larghezza della finestra di gioco, cioè, sarà possibile allargare, diminuire e deformare entro certi limiti la finestra e gli elementi manterrano la stesse proporzioni di altezza e larghezza.

- **Funzioni per l'animazione:** comprendono le funzioni per gestire l'animazione che avviene alla fine di ogni mano se vi è stata almeno una puntata. Quello che viene gestito nell'animazione sono le coordinate delle chips puntate sul tavolo dai vari giocatori: semplicemente alla fine di una mano, se ci sono state delle puntate, allora le coordinate di queste chips vengo aumentate in x e y fino a raggiungere quelle del piatto. Poi queste chips diventano non più visibili.
  Le funzioni gestite da Tavolo Poker sono quella per inizializzare il vettore delle coordinate assolute usate per far tornare le chips nelle posizioni originarie all'inizio di ogni mano, quella per far tornare il valore delle coordinate relative uguale a quello delle coordinate as-

solute e infine quella per resettare il valore del vettore che indica su quali giocatori bisogna effettuare l'animazione.

- **Funzione PaintComponent:** questa è la funzione per il disegno a video principale, all'interno di questa vengono chiamate le varie funzioni per il disegno ed essa è quella che viene richiamata ripetutamente per aggiornare gli elementi a video.

3.2 Diagramma delle classi

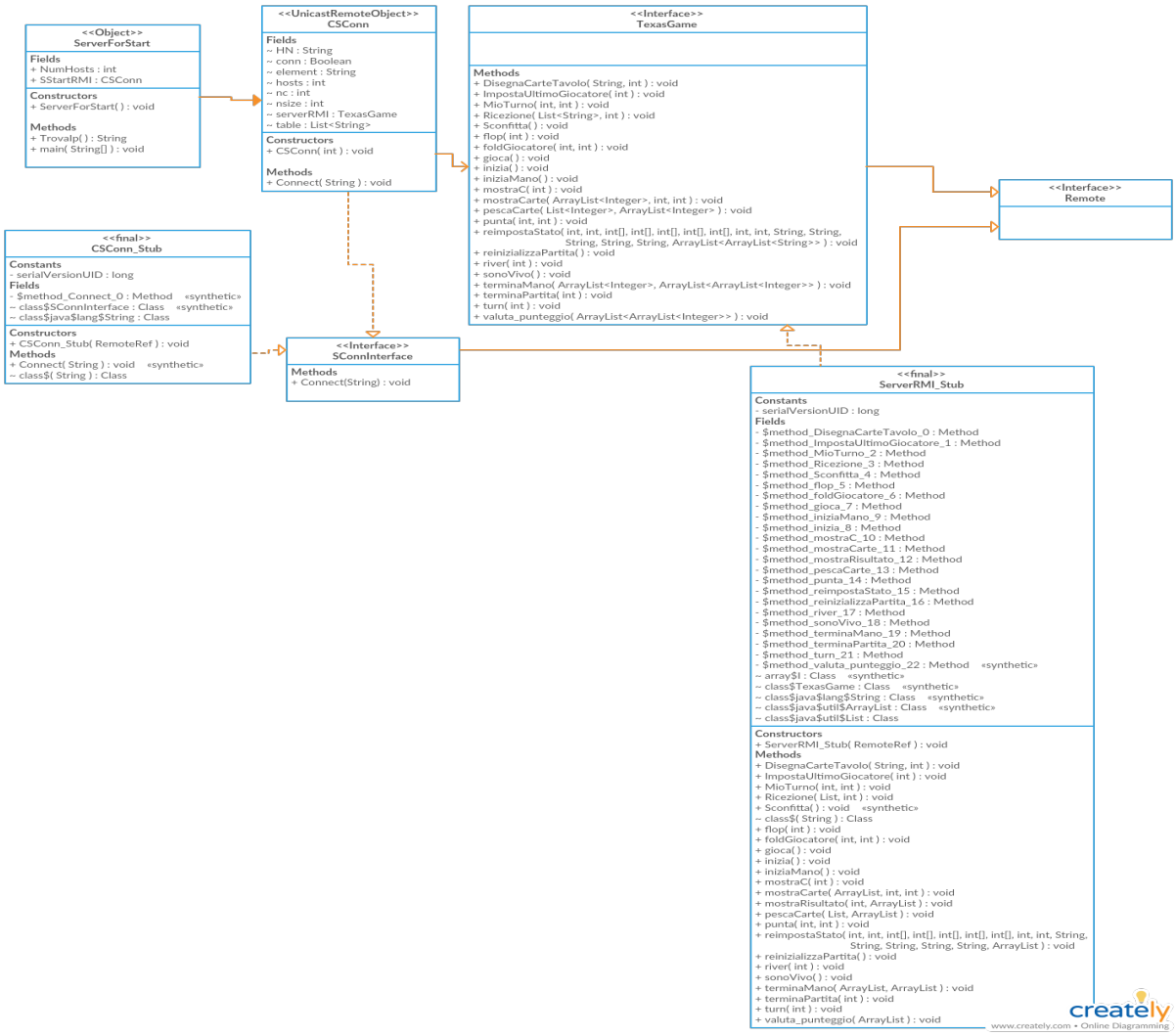

Figura 3.1: Struttura del Server - Diagramma delle classi UML

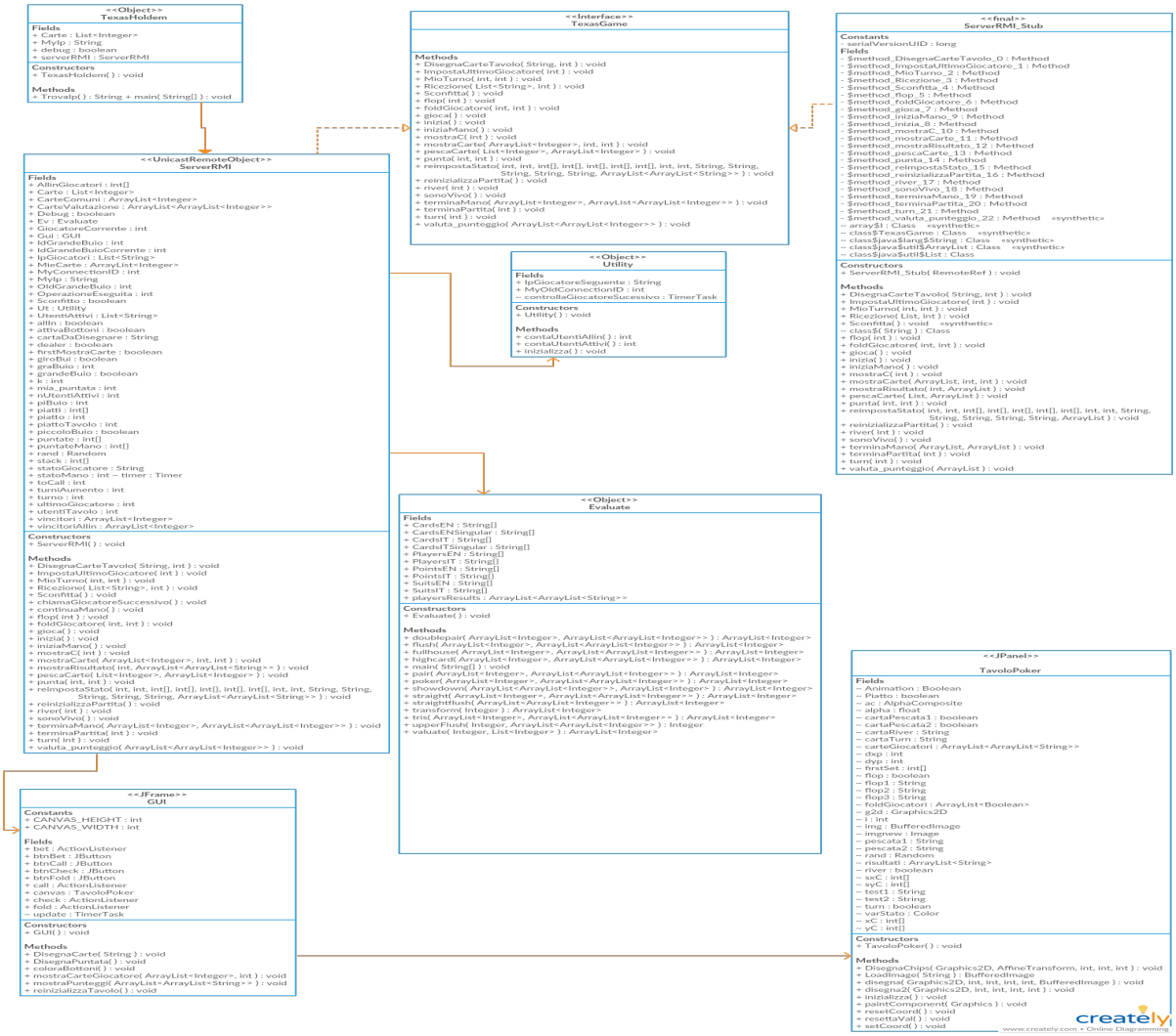

Figura 3.2: Struttura del Client - Diagramma delle classi UML

3.3 Scenario

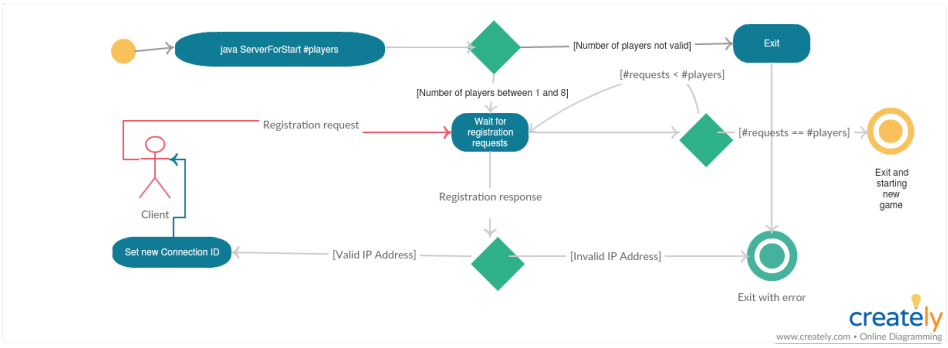

Figura 3.3: Registrazione - Vista Server - Diagramma degli stati UML

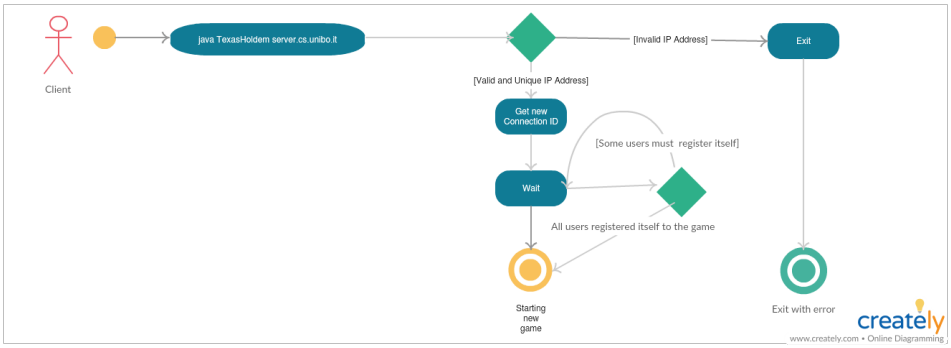

Figura 3.4: Registrazione - Vista Client - Diagramma degli stati UML

3.4 Esecuzione

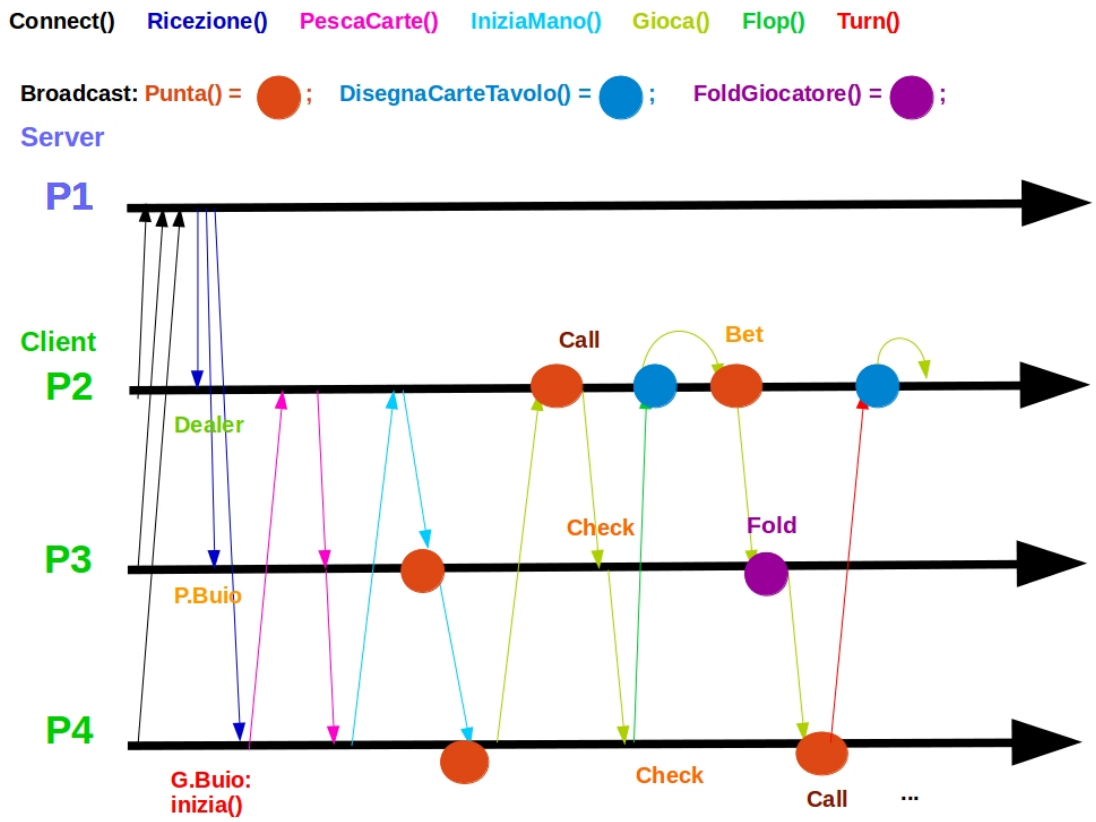

Figura 3.5: Esempio esecuzione 1

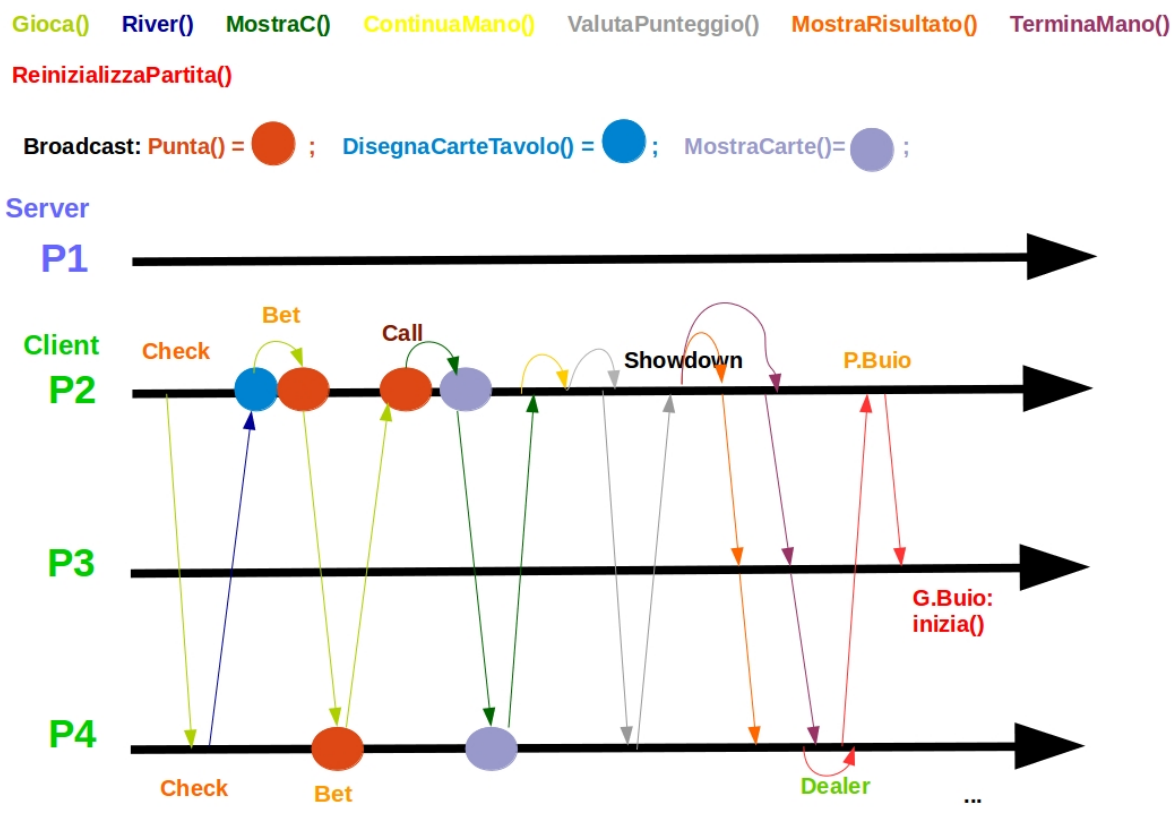

Figura 3.6: Esempio esecuzione 2

Figura 3.7: Screenshot esecuzione gioco

Capitolo 4

Tolleranza ai guasti

Il Sistema tollera fino a n-1 fault. Vengono gestiti solo fault di tipo persistente. Durante la sua esecuzione il gioco può essere visto come un anello nel quale ciascuno nodo verifica ogni 100 millisecodi se gli altri giocatori sono ancora attivi oppure hanno subito un eventuale fault.
Per fare questa verifica ogni nodo esegue una chiamata RMI alla funzione *SonoVivo()* di tutti gli altri nodi. Se la chiamata solleva un'eccezione il nodo interessato non è più attivo, altrimenti è ancora in gioco.

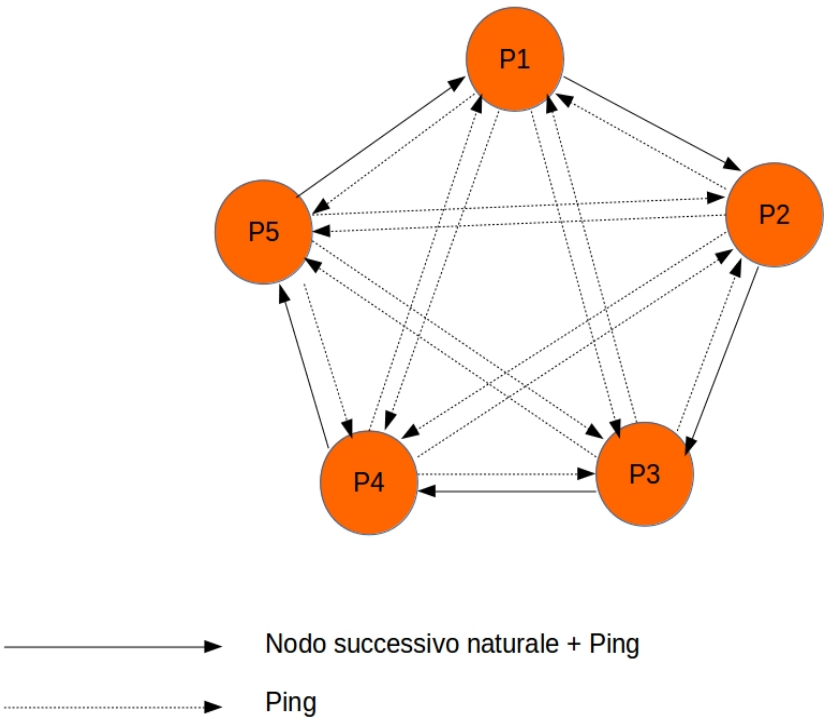

Figura 4.1: Esempio gioco in esecuzione

In questo modo tutti i giocatori rilevano un eventuale nodo in crash e lo rimuovono dalla partita per mezza della funzione *TerminaPartita()* della classe *ServerRMI.java* (vedi capitolo 3.1.1: *Il ServerRMI*).

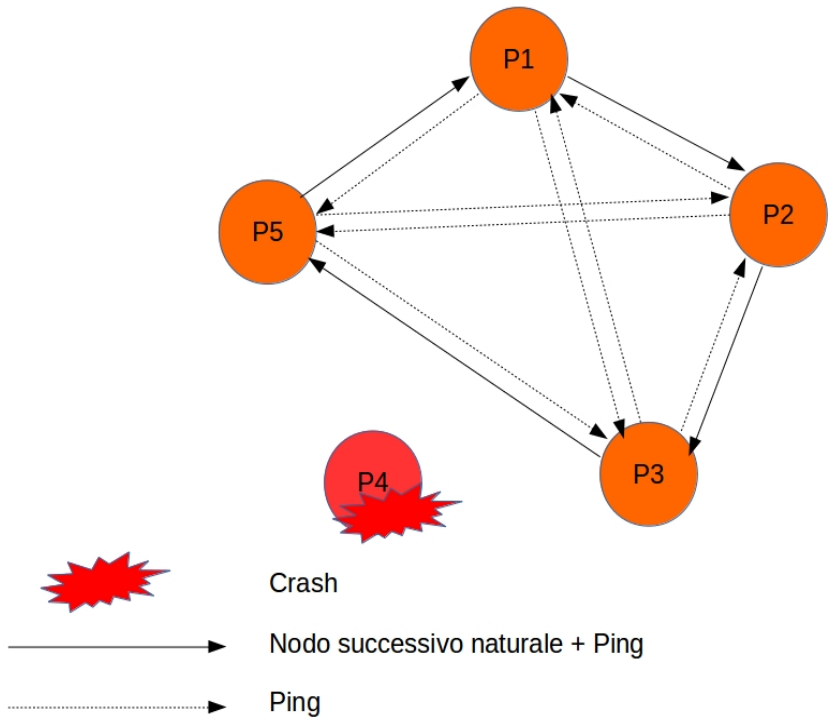

Figura 4.2: Esempio fault tolerance: Crash P4

Durante l'esecuzione ogni nodo appena riceve il controllo esegue un broadcast indicando a tutti gli altri giocatori il giocatore corrente (cioè se stesso) e l'operazione che si appresta ad eseguire. Per fare ciò viene utilizzata la funzione *MioTurno()* della classe *ServerRMI.java*, (vedi capitolo 3.1.1: *Il ServerRMI*).
Questo è fondamentale nel caso in cui il nodo che ha il controllo del gioco

in quel momento subisce un crash.
In questo caso il nodo che precede il nodo fallito deve fare ripartire il gioco chiamando il giocatore corretto e indicandogli a che punto dell'esecuzione si è arrivati, e riesce ad estrapolare queste informazioni grazie agli ultimi dati ricevuti dalla *MioTurno()* ed ad altre variabili interne.
Questo meccanismo funziona anche in caso di crash simultanei.

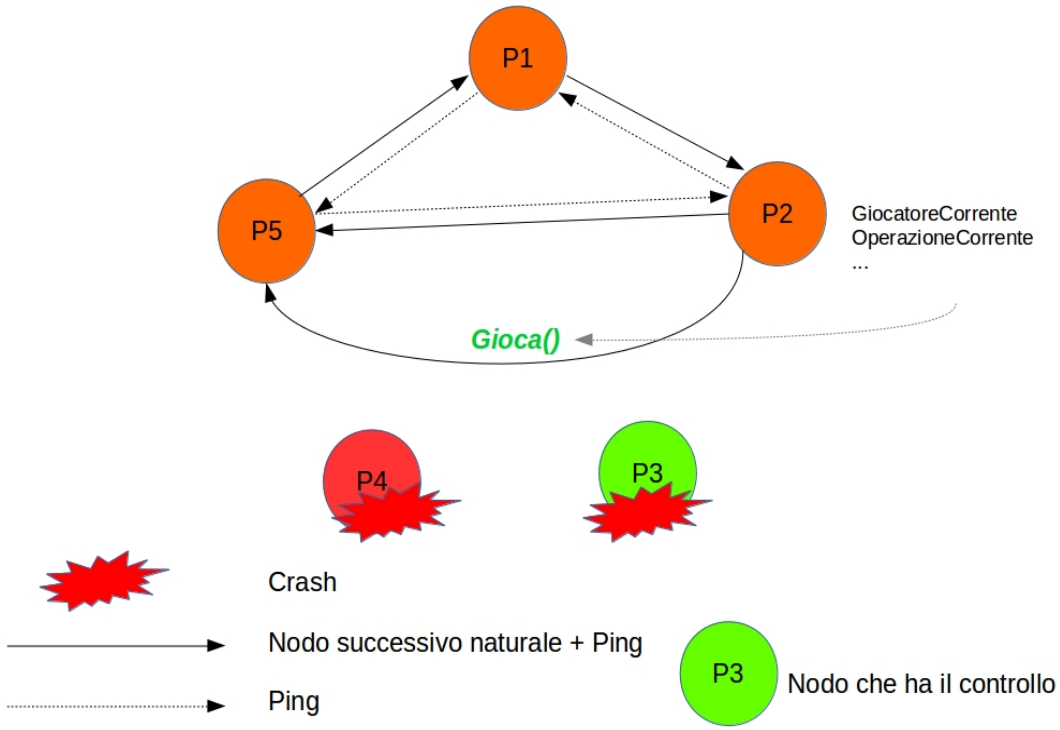

Figura 4.3: Esempio fault tolerance: Crash P3: giocatore corrente e ripresa dell'esecuzione

Conclusioni

Problematiche riscontrate

Dopo aver studiato e realizzato un sistema distribuito per il gioco del Poker, le problematiche in cui ci siamo imbattuti sono molteplici. Sicuramente non è banale implementare un gioco distribuito, ancor di più pensare a una fault tolerance pari a n-1 in concorrenza con Java RMI. I problemi riscontrati sono stati di natura implementativa (la libreria Graphics2D di Java usa metodi bloccanti e la nostra necessità era che venissero utilizzati all'aggiornamento dello stato del gioco) e di natura prettamente legata al linguaggio e alle Remote Method Invocation, visto e considerato che non esistono costrutti per fare Broadcast (l'abbiam simulato con dei cicli di Unicast) e che la concorrenza non è gestita di default, ragion per cui abbiamo dovuto assegnare un metodo per verificare il fault per ogni giocatore (SonoVivo1(), SonoVivo2(), ..., SonoVivoN()) .

Possibili sviluppi

- **Timeout :** aggiungere un timeout entro il quale ogni giocatore deve scegliere la mossa da effettuare. Se entro questo tempo non compie nessuna azione il giocatore esegue *fold*.
- **Numero utenti :** aumentare il numero degli utenti supportato dal gioco.
- **Quantità numero utenti dinamica :** prevedere che i giocatori che parteciperanno alla partita siano quelli che si registrano al server entro un determinato tempo.
- **Espansione del gioco :** permettere agli utenti di scegliere tra più giochi, ad esempio poker all'italiana, texas hold'em, ecc.

La nostra esperienza

In conclusione, portare a compimento tale progetto è stata una sfida entusiasmante per chi scrive, che ha messo alla prova le nostre abilità di programmazione e quelle di confrontarsi con un'architettura del tutto diversa da quella classica a cui si era abituati (sviluppo web, mobile app, in genere approccio client/server) . Inoltre è stato molto interessante formalizzare la fault tolerance e condurre la fase di testing del gioco con i crash, in cui si è constatato quanto fosse facile non considerare alcuni casi di errore durante i fault. In generale , siamo soddisfatti del nostro lavoro, sebbene sia ancora migliorabile e espandibile, cosa che ci auguriamo.