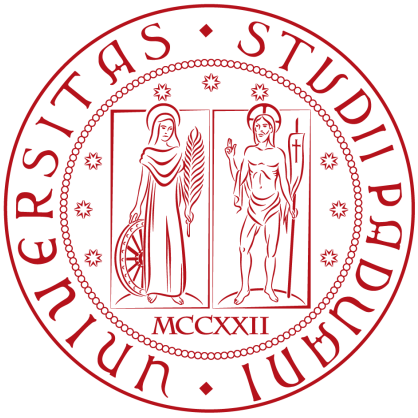

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria dell'Informazione

Modelli di Teoria dei Giochi Applicati alle Reti di Comunicazione

Laureando:
Mattia Coccato

Relatore:
Prof. Leonardo Badia

Anno accademico 2011/2012

Abstract

Scopo di questa tesi è lo sviluppo di un programma di simulazione di una rete di comunicazione in cui i vari nodi hanno facoltà decisionali, secondo le strategie derivate dagli studi della teoria dei giochi, che gli permettono di interagire con i vicini costituenti della rete, così da alterarne il comportamento globale. Con l'ausilio di tale strumento verranno discusse le principali strategie, nonché il loro impatto sulla comunicazione attraverso la rete, sfruttando il modello detto dilemma del prigioniero. Verranno altresì discussi gli effetti dei parametri modificabili, corredati dai grafici delle simulazioni effettuate.

Indice

Capitolo 1

Introduzione

Stiamo assistendo in un periodo storico in cui Internet è di vitale importanza, stiamo producendo ma soprattutto consumando sempre più informazioni, e la necessità di essere sempre raggiungibili, per lavoro o per svago, non solo mediante telefonate ed SMS, ma utilizzando anche le nuove tecnologie quali e-mail e messaggi istantanei, spingono alla proliferazione del numero di apparati connessi alla Rete. Basti pensare alla recente notizia dell'esaurimento degli indirizzi IPv4[1][12], quasi 3.5 miliardi di indirizzi univoci sono stati infatti assegnati ad altrettanti computer, smartphone, tablet, e tutta una miriade di altri dispositivi connessi ad Internet, senza contare i molti apparati presenti nelle reti di scuole ed uffici che non possiedono un proprio indirizzo univoco – accedono infatti attraverso dei router con meccanismi NAT, che permettono di condividere un unico accesso alla rete.

Avere un gran numero di dispositivi connessi pone però grossi problemi per quanto riguarda la complessità di calcolo per la distribuzione dei flussi di informazioni, che viaggiano divise in pacchetti, ma soprattutto della scelta del percorso che tali pacchetti debbano seguire. Nelle semplificazioni didattiche [5] si è soliti affermare che i nodi impieghino algoritmi di indirizzamento che smistano i pacchetti in modo da seguire la rotta con il *costo* minimo; stabilire numericamente questo costo può coinvolgere molte variabili, come la velocità di un collegamento, la saturazione di un link che comporta tempi di attesa maggiori, oppure il passaggio attraverso aree della rete indesiderate, che possono includere a titolo di esempio paesi ostili, oppure strutture vulnerabili che pongono a rischio la riservatezza dei dati in transito.

[1] Vedere anche `http://www.ripe.net/internet-coordination/ipv4-exhaustion`

Aumentando i requisiti di complessità ed entrando negli ambiti di uso reale, non tutti i nodi sono uguali agli altri, ma anzi si debbono classificare e riconoscere; di conseguenza, l'algoritmo che segue semplicemente la via di costo minimo può non essere sufficiente né desiderato. Ed è a questo punto che intervengono strategie di simulazione e interpretazione delle reti più complesse, tra cui quelle che sfruttano la *teoria dei giochi* [14]. I singoli nodi non seguono più semplicemente algoritmi statici, ma diventano capaci di *prendere decisioni*, basate sul proprio ma soprattutto sull'altrui comportamento. Si pensi a questo esempio: due nodi, A e B, devono comunicare tra loro, e per farlo hanno a disposizione due percorsi, P_1 dal costo fisso, supponiamo unitario, e P_2 il cui costo dipende dalla frazione di traffico che vi passa attraverso (ovvero, costa 0.5 se metà dei pacchetti spediti da A passano attraverso P_2) [4, pagg. 717-9]. Questo è l'esempio più semplice ove occorre *prendere una decisione*, e il costo complessivo ne è influenzato direttamente. Per di più, se le decisioni di routing vengono prese rispetto ad un singolo pacchetto – *selfish routing*[21] – si ottiene una situazione non ottimale; il singolo pacchetto cercherà sempre di seguire la via più rapida, con il risultato che *tutti* i pacchetti seguiranno l'una o l'altra via, ed il costo complessivo diventa così $C_S(A \to B) = 1$. Suddividendo in parti uguali i pacchetti si ottiene invece $C_O(A \to B) = 0.5 * 0.5 + 0.5 * 1 = 0.75$, che è inferiore al routing egoistico di ben il 33%.

La tecnica più utilizzata si basa su varianti del *dilemma del prigioniero* [1], un problema classico della teoria dei giochi, in cui i nodi scelgono nel tempo se collaborare o non collaborare con i vicini. Le ripercussioni ed i vantaggi di una scelta piuttosto dell'altra dipendono dal contesto, e possono variare dalla chiusura della connessione (es. alcuni programmi *peer-to-peer* [20]) ad un ritardo temporale.

Per permettere di simulare in maniera semplice e immediata una rete basata su nodi, ognuno dei quali gioca ripetutamente il dilemma del prigioniero, è stato realizzato un programma che, pur con delle semplificazioni rispetto ad una rete reale per ovvi motivi di complessità computazionale e di scrittura del codice, permette di osservare il comportamento di una rete, sia essa generata automaticamente dal programma stesso oppure impostata dall'utente, variandone i parametri fondamentali, quali a titolo non esclusivo:

- Il costo base di ogni arco;
- La strategia che un nodo debba seguire;

- La struttura della rete (numero di nodi e collegamenti);
- La penalizzazione o il vantaggio in caso di collaborazione/non collaborazione tra due nodi.

Alcune simulazioni ottenute variando questi aspetti sono riportate nel capitolo 4, corredate da grafici.

Capitolo 2

Il contesto

2.1 La teoria dei giochi

La teoria dei giochi è "una disciplina alquanto vasta, il cui scopo è analizzare i comportamenti strategici dei decisori (giocatori), ovvero studiare le situazioni in cui diversi giocatori interagiscono perseguendo obiettivi comuni, diversi o conflittuali" [1]. I giocatori devono essere razionali, ovvero saper ordinare le proprie preferenze possibili per massimizzare (o minimizzare, a seconda delle varie formulazioni delle situazioni) il proprio profitto da esse risultante, nonché intelligenti, il ché comporta la capacità di saper riconoscere le azioni necessarie per massimizzare il profitto (ovvero per comportarsi in maniera razionale). Unendo i giocatori, le loro strategie e una funzione che rappresenta i ricavi (o payoff) per una data n-upla di strategie otteniamo un gioco, i più semplici dei quali coinvolgono solo una coppia di giocatori, con una scelta limitata di strategie possibili; convenientemente vengono rappresentati in forma tabulare. Si assume, ma esistono una varietà di tecniche che possono essere usate anche quando questa assunzione dovesse cadere, che tutti i giocatori conoscano l'intero gioco (giochi ad informazione completa), quindi tutte le strategie possibili degli avversari, nonché la funzione ricavi associata. La tipologia di gioco più famosa, nonché quella su cui si basa il lavoro di simulazione è il cosiddetto *dilemma del prigioniero*.

2.2 Il dilemma del prigioniero

Storicamente, è stata questa formulazione che ha attirato l'attenzione degli studiosi, per l'apparente semplicità nonché facilità di adattamento ad una stermi-

nata serie di ambiti più disparati, e si basa su due soli giocatori, ognuno dei quali agisce in maniera indipendente dall'altro, potendo scegliere tra due sole strategie: *confessare* o *non confessare*. La formulazione originaria risale al 1950, da parte di Merrill Flood [9, 10] e Melvin Dresher [6, 7], anche se la funzione di ricavo e il nome vengono attribuiti a Albert W. Tucker (secondo [18]). La formulazione è la seguente:

> Due persone, sospettate di aver commesso un reato sono detenute in celle separate. Ognuno può scegliere di confessare (C) oppure di non confessare (NC). La scelta di ciascuno dei due influenza anche il destino dell'altro. Infatti, se entrambi confessano, saranno condannati a 10 anni di prigione. Se solo uno dei due confessa, accusando dunque l'altro, potrà beneficiare di uno sconto di pena, e avrà quindi solo 1 anno di carcere, mentre l'altro sarà condannato a 25 anni (per l'aggravante di non aver voluto collaborare). Se nessuno dei due confessa, in mancanza di prove testimoniali, ambedue le persone saranno accusate soltanto di danni al patrimonio e condannate a 3 anni di prigione.[1]

Risulta immediatamente chiaro come la scelta di un prigioniero vada a modificare anche l'aspettativa per l'altro, ma tra i due non vi può essere dialogo e quindi accordo. Utilizzando la rappresentazione tabulare riportata nella tabella 2.1 la comprensione risulta più semplice; le righe rappresentano le strategie possibili per il primo giocatore, le colonne, dualmente, le strategie del secondo giocatore. Le singole celle della tabella sono quindi riempite con la funzione ricavo, sotto forma di coppie di valori, rispettivamente il ricavo per il primo e il secondo giocatore.

		Giocatore 2	
		C	NC
Giocatore 1	C	(10,10)	(25,1)
	NC	(1,25)	(3,3)

Tabella 2.1: La tabella della funzione payoff dell'esempio

Come è evidente, per il primo prigioniero le strategie vanno ordinate in ordine di penalizzazione – ossia di anni di carcere – nella sequenza $(C, NC) \succ (NC, NC) \succ (C, C) \succ (NC, C)$, ma non può sapere cosa sceglierà l'avversario. La strategia di confessare è in questo caso *strettamente dominante* rispetto all'alternativa di

[1] Questo esempio è tratto da [1].

non confessare (che assume quindi il ruolo di strategia *strettamente dominata*), in quanto la pena comminata è minore (10 o 1 contro 25 o 3). I giocatori razionali sono portati quindi entrambi a confessare, che però non è il miglior risultato ottenibile in assoluto, detto *ottimo di Pareto*; proprio questa apparente illogicità gli ha conferito anche il nome di paradosso del prigioniero.

2.3 Dominanza ed equilibrio di Nash

Per poter proseguire è necessario porre in forma matematica la descrizione sin ora colloquiale del gioco. Un gioco infatti è definibile come una terna

$$G = (P, S, U) \tag{2.1}$$

con rispettivamente:

- P è un insieme di giocatori, $P = \{P_1, P_2, \cdots, P_N\}$;
- S è un insieme di strategie, ciascuna delle quali è un vettore che rappresenta le strategie del singolo giocatore i-mo $S = \{S_1, S_2, \cdots, S_N\}$ con $S_i = (s_{(i,1)}, s_{(i,2)}, \cdots, s_{(i,Mi)})$ Per praticità si indicherà con $\widehat{s}_i$ la specifica strategia del nodo i-mo; chiaramente $\widehat{s}_i \in S_i$.
- U è un insieme di funzioni $u_i = U_i(s_1, s_2...., s_N)$ ognuna delle quali rappresenta il guadagno del giocatore P_i in corrispondenza delle strategie giocate dai giocatori.

Con ciò definito, una strategia $\widehat{s}_i = s_{(i,h)}$ è detta *dominante* (per il nodo i-mo ovviamente) se

$$U_i\left(s_1, s_2, ..., s_{(i,h)}, ...s_N\right) \geq U_i\left(s_1, s_2, ..., s_{(i,k)}, ...s_N\right) \quad \forall s_j, \quad j \neq i \quad e \quad h \neq k \tag{2.2}$$

ovvero, per ogni strategia scelta dagli avversari, il risultato ottenuto giocando la scelta $s_{(i,h)}$ è sempre maggiore o al più uguale a quanto ottenibile impiegando altre strategie. Qualora la disequazione valga in senso stretto, ossia

$$U_i\left(s_1, s_2, ..., s_{(i,h)}, ...s_N\right) > U_i\left(s_1, s_2, ..., s_{(i,k)}, ...s_N\right) \forall s_j, \quad j \neq i \quad e \quad h \neq k \tag{2.3}$$

si parla di strategia *strettamente dominante*. Le rimanenti strategie sono rispettivamente dette dominate e strettamente dominate da $s_{(i,h)}$. In caso il gioco verta sulla minimizzazione della funzione payoff i versi delle disequazioni vanno ovviamente considerati invertiti. In altre parole, una strategia dominante è il miglior risultato ottenibile in base alle scelte disponibili per il nodo i-esimo, eventualmente uguagliato in caso di dominanza non stretta.

L'*equilibrio di Nash* [15] è costituito da insiemi di strategie $\{s_1^e, s_2^e, ..., s_i^e, ..., s_N^e\}$ tale che

$$U_i\left(s_1^e, s_2^e, ..., s_i^e, ..., s_N^e\right) \geq U_i\left(s_1^e, s_2^e, ..., s_i, ..., s_N^e\right) \tag{2.4}$$

In altre parole, ogni insieme rappresenta una situazione dalla quale nessun giocatore trova vantaggi nel cambiare strategia, fintantoché gli altri giocatori giocano la loro strategia s_j^e [17]. Proprio John Nash, teorizzatore e formulatore del teorema che enuncia quanto sinora riportato, è arrivato a dimostrare matematicamente che ogni gioco finito ha almeno un equilibrio di Nash, al più ricorrendo a strategie miste, ovvero a distribuzioni di probabilità sulle strategie a disposizione del giocatore.

2.4 L'applicazione alle reti

L'idea di applicare lo studio della teoria dei giochi alla modellizzazione di reti non è nuova, ed è supportata anche da diverse ragioni pratiche, siano esse volte per esempio al risparmio di risorse, in termini di banda, tempo di calcolo od energia, od alla situazione socio-politica tra Nazioni. Molte tesi sono state discusse sull'argomento, anche se molte si basavano sul concetto di *small world*, come [16]. Small world, sinteticamente, indica una situazione particolare in cui tutti i nodi sono a conoscenza dell'esistenza degli altri, e vi possono collegare direttamente; nel programma ciò non è stato voluto, perché escludendo tale ipotesi si può assistere a dei cambi di percorso durante l'evoluzione del gioco. Si consideri l'ipotesi, in una rete sufficientemente densa di collegamenti, che esista un percorso P_X tra due entità X e Y; qualora due nodi intermedi E_1 e E_2 nel percorso non collaborino, in certe condizioni è possibile trovare un percorso più breve P_{New}, che non passa per il collegamento $E_1 \leftrightarrow E_2$, aggirando così l'ostacolo.

Altri studi si sono focalizzati sullo studio della topologia migliore [23], nonché sulla precisa analisi matematica dei tempi di ritardo e dei flussi dei pacchetti [2];

vi sono infatti delle situazioni in cui l'aggiunta di collegamenti a basso costo tra nodi, invece di portare benefici provoca un peggioramento delle prestazioni medie della rete; questa situazione è espressa nel *paradosso di Braess* [8] . Questo accade perché gli algoritmi di routing generalmente cercano la via migliore egoisticamente, non tenendo conto della presenza degli altri utenti nella rete né dell'effetto delle loro attività, un po' come quando in auto si cerca la via più breve, ma non si tiene conto del traffico generato dagli altri utenti – il calcolo delle penalizzazioni che questo comportamento causa costituisce il problema del *prezzo dell'anarchia* [22].

In letteratura si è anche scesi a simulare più a basso livello i risultati della cooperazione tra utenti, che instaurano così percorsi preferenziali a bassa latenza modellizzando i protocolli di trasmissione e i sistemi a coda [3], nonché esprimendo matematicamente le relazioni tra i nodi tenendo conto delle attenuazioni dell'ambiente in caso di comunicazioni *wireless* e dell'efficienza spettrale [11]; la simulazione a basso livello è stata esclusa da questo lavoro per la grande complessità delle situazioni che coinvolge, anche se con un deciso lavoro di ampliamento sarebbe possibile includerla.

2.4.1 Un esempio a livello 7: BitTorrent

Finora si è discusso di problemi di routing, quindi appartenenti al livello 2 dello stack ISO/OSI, ma il dilemma del prigioniero, e gli studi della teoria dei giochi in generale, sono applicabili anche a livelli più alti, fino al livello applicazione, il più alto dei sette previsti. L'esempio forse più noto ed utilizzato dal pubblico è rappresentato da BitTorrent.

BitTorrent [19] è un noto protocollo utilizzato per lo scambio di file con la tecnica del peer-to-peer, ovvero mediante la connessione diretta tra utenti, in opposizione al modello client-server ove l'informazione è centralizzata in un singolo computer e tutti gli utenti debbono necessariamente connettersi ad esso. BitTorrent utilizza gli schemi propri del dilemma del prigioniero per gestire le connessioni in uscita, in caso di saturazione della banda disponibile, impiegando una versione leggermente modificata della strategia *tit-for-tat* [24]. Per impostazioni predefinite infatti, un utente invia i frammenti dei file richiesti in via preferenziale a quei peer da cui sta effettivamente scaricando, e "strozza" (*choke*) le connessioni agli altri; tuttavia, solitamente una porzione di banda viene riservata per riaprire, in modo casuale, alcune di queste connessioni bloccate (*unchoking*), alla ricerca di

peer potenzialmente più vantaggiosi, ma anche per favorire i nuovi utenti che, non avendo file condivisi per la comunità non avrebbero la possibilità di scaricare nulla. Il liberare alcune connessioni previene anche situazioni di potenziale stallo, per esempio tra nodi che avrebbero banda disponibile, ma non sono in grado di "prendere l'iniziativa" per connettersi.

Capitolo 3

Simulazione di reti game-teoriche

Si parta da questa ipotesi: si ha una rete costituita da sistemi embedded, alimentati a batteria oppure con un alimentatore da rete, capaci, oltre a comunicare con gli altri dispositivi, di svolgere funzioni di instradamento dei pacchetti provenienti da altri nodi della rete; ognuno di essi può dover prendere delle decisioni, poiché la comunicazione – in particolar modo su reti wireless – può rappresentare una grossa fetta dell'energia consumata, tra l'inoltrare i pacchetti oppure entrare in stati di risparmio energetico. Questo è modellizzabile come un gioco del dilemma del prigioniero, in cui ad ogni istante di tempo, ogni nodo decide se cooperare con un vicino, inoltrando il pacchetto, oppure non collaborare e far aumentare il costo della tratta, in quanto la trasmissione richiederà più tentativi e quindi più tempo; la scelta dipenderà dalla strategia adottata dal singolo nodo.

3.1 Simulatore di reti multi-hop

Partendo da questa idea base, si è costruito un programma, in linguaggio Java, costituito da oltre 3500 linee di codice, che permette la generazione e la simulazione di reti *multi-hop* (o *multi-salto*) – ovvero ove i percorsi tra sorgente e destinazione possono passare anche attraverso altri nodi, qualora non esista tra loro un collegamento diretto – per l'appunto costituite da nodi "intelligenti", che seguono le regole della teoria dei giochi. Sono state tuttavia necessarie alcune semplificazioni, per non appesantire le già molte variabili da gestire, le più importanti sono le seguenti:

- il grafo della rete è noto a tutti i nodi, che possono eseguire l'algoritmo di Dijkstra per scoprire il percorso a peso minimo;

- all'inizio della simulazione vengono stabilite delle coppie sorgente → destinazione, e rimangono fisse nel tempo;
- non sono stati considerati i singoli pacchetti che viaggiano nella rete, ma solo il costo totale del percorso tra due nodi;
- la simulazione è limitata ad un massimo di circa 50 nodi;

Per stabilire gli effetti della rete si procede a calcolare il peso di ogni percorso tra le coppie di nodi stabilite, e ne viene successivamente considerata la media. Pur essendo didatticamente valido, questo ragionamento è molto più complesso da applicarsi ai casi generali, nei quali difficilmente vi è la possibilità di eseguire il calcolo del percorso più breve mediante l'algoritmo di Dijkstra, in quanto oneroso in termini di tempo di calcolo (ha infatti complessità computazionale $O(V^2)$, ovvero proporzionale al *quadrato del numero dei nodi*) e soprattutto di conoscenza della rete e di comunicazione tra i singoli nodi. Ogni pacchetto immesso nella rete infatti provoca la variazione della traffico, della latenza e del carico di lavoro in ogni arco [25]. A rendere ancora più complesso il quadro reale, si deve tenere d'acconto la struttura propria delle reti di vaste dimensioni, solitamente divise in livelli, ove è applicato il cosiddetto *routing gerarchico* [13], in cui vi è un unico punto di accesso per livello per una determinata zona della rete; sarà questo nodo a preoccuparsi di indirizzare correttamente i pacchetti verso la destinazione specificata.

La complessità computazione dell'algoritmo di Dijkstra è inoltre alla base della scelta tecnica di limitare il numero massimo di nodi inseribili in una rete a 50, in quanto tale algoritmo deve essere invocato *ad ogni modifica* della rete – e quindi ad ogni iterazione del gioco, dato che anche modificando un solo ramo in certe situazioni si può far variare il percorso a molte coppie di nodi – per individuare il percorso a peso minimo, e calcolato *per ogni nodo*, portando la complessità complessiva ad un fattore esprimibile come $O(V^3)$.

3.2 Le classi

Il programma si compone di 15 classi, sei delle quali, raggruppate nel pacchetto `gtNetwork.ui` si occupano della gestione di tutte le finestre grafiche a disposizione dell'utente. Come descritto nel paragrafo introduttivo di questo capitolo, i nodi possono seguire diverse strategie di comportamento, che ovviamente influenzeranno quello degli altri nodi connessi.

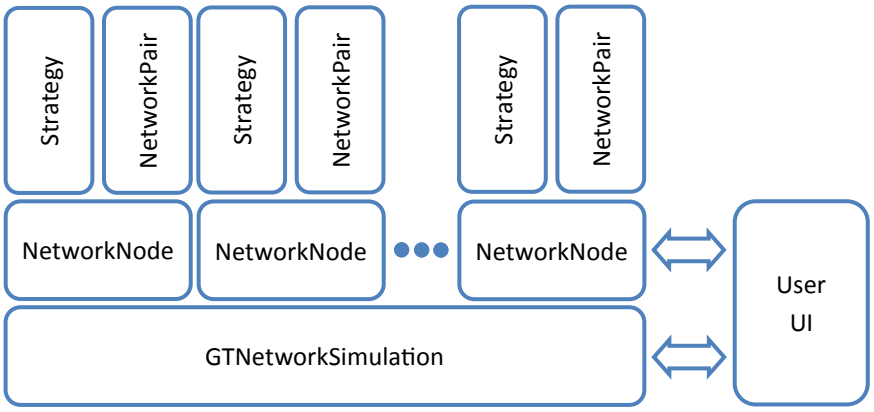

Figura 3.1: La schematizzazione assunta del programma

Il mattone fondamentale, su cui si basa tutto, è la classe `GTNetworkSimulation`, che racchiude tutti i metodi necessari per l'esecuzione della simulazione, nonché un grafo, i cui nodi sono istanze della classe `NetworkNode`. Ciò è necessario perché ogni nodo, oltre al nome, deve racchiudere altre informazioni, ossia:

- la storia delle collaborazioni con ogni nodo. Alcune strategie infatti, come la *tit-for-tat* o altre che potranno essere implementate successivamente, necessitano degli esiti passati degli scontri per poter prendere le decisioni. Per un utilizzo efficiente della memoria si è optato per assegnare gli esiti ai singoli *bit* di un `long`, quindi si salvano al massimo gli ultimi 64 esiti e si utilizzano soli 8 byte per ogni nodo connesso. Gli esiti possono semplicemente essere ricavati con operazioni di *AND bitwise*, e il salvataggio di ogni esito comporta uno scorrimento e al più un'operazione di addizione.

- l'istanza della strategia. Ogni nodo può infatti assumere una strategia diversa, un'istanza per nodo. Nulla vieta però ai metodi delle strategie implementabili di discernere a loro volta il comportamento per ogni nodo ad esso collegato; per esempio, sarebbe possibile implementare una strategia che collabori sempre con il *primo* nodo con cui gioca il dilemma del prigioniero, e negare la collaborazione a tutti i successivi nodi.

- il nodo destinazione delle comunicazioni. Dato che viene assegnato un nodo destinatario all'inizio della simulazione, un'istanza della classe `NetworkPair` si occupa della scelta del percorso, nonché del calcolo del suo costo complessivo.

Le decisioni quindi sono effettuate dai singoli nodi, utilizzando le loro strategie impostate, indipendentemente dagli altri. Decisa la collaborazione o meno da parte di ognuno dei due nodi, l'esito viene aggiunto allo storico e il peso dei rami è aggiornato.

Il peso dei rami non varia in scala lineare, ma varia secondo la seguente formula:

$$C_t = \begin{cases} C_{t-1} - \alpha(C_{t-1} - C_{MIN}) & \text{se il nodo collabora} \\ C_{t-1} + \alpha(C_{MAX} - C_{t-1}) & \text{se il nodo non collabora} \end{cases} \tag{3.1}$$

ossia, viene tolto ad ogni passaggio una parte della distanza che separa il costo attuale dall'estremo. Risulta di più semplice comprensione se l'intervallo dei valori è rappresentato in maniera grafica.

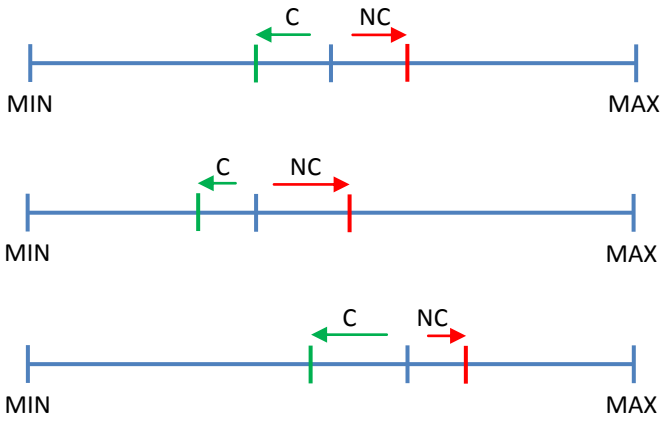

Figura 3.2: Rappresentazione grafica dell'aggiornamento dei pesi, $\alpha = 0.25$

In riferimento alla figura 3.2 si possono vedere tre situazioni:

1. Si supponga che il peso iniziale sia intermedio tra il minimo e il massimo (tratto blu); sono evidenziati i due nuovi possibili pesi, in verde in caso di collaborazione, in rosso in caso di non collaborazione. Si noti che i due intervalli sono uguali per la simmetria della figura.

2. Situazione dopo la collaborazione al passo 1: collaborando nuovamente la riduzione in termini assoluti del peso è inferiore, mentre aumenta la penalizzazione in caso di *non* collaborazione.

3. Situazione dopo la non collaborazione al passo 1; le considerazioni sono il simmetrico del punto precedente.

In aggiunta, nella codifica del programma si è prevista l'esistenza di due coefficienti diversi, uno in caso di scelte concordi tra i due nodi (α_{SAME}), l'altro in caso di scelte discordi (α_{DIFF}); ciò è utile nel caso si voglia simulare una cooperazione che *costa risorse* oltre a dare vantaggi. Chiaramente, il nodo che non coopera si avvantaggia maggiormente della collaborazione dell'altro perché non deve investire le proprie risorse.

3.2.1 Le strategie

In questa prima versione del programma si è provveduto ad inserire tre strategie fondamentali, ossia:

- `AlwaysCooperate`, che, come il nome fa intuire, *collabora sempre* con tutti i nodi che vi si collegano. Può rappresentare una valida strategia per quei nodi connessi ad una fonte di alimentazione AC.

- `NeverCooperate`, opposta alla prima, *non collabora mai* con gli altri nodi. Ritornando alla nostra rete di sensori, può essere applicabile ad un nodo "in riserva di energia", che per tentare di massimizzare la vita operativa restante taglia tutte le fonti di consumo non indispensabili.

- `TitForTat`, il cui esito della collaborazione dipende *da quello dell'avversario al "round" precedente*, ossia A collabora con X al tempo t se X ha collaborato con A al tempo $t-1$.

- `RandomStrategy`, che semplicemente ed indipendentemente dai risvolti passati sceglie se collaborare con probabilità $P(C)$ o non collaborare, ovviamente con probabilità $P(NC) = 1 - P(C)$.

Il programma tuttavia è già predisposto per la realizzazione di strategie ulteriori, anche molto complesse; queste infatti vengono caricate a *run-time*, usando il componente `Reflections`[1]. Per ampliare la gamma di ambiti simulabili è sufficiente definire nuove strategie – implementando l'interfaccia `INetworkNodeStrategy`, ed includere i file binari nella directory del programma.

[1] Vedi `http://code.google.com/p/reflections/`

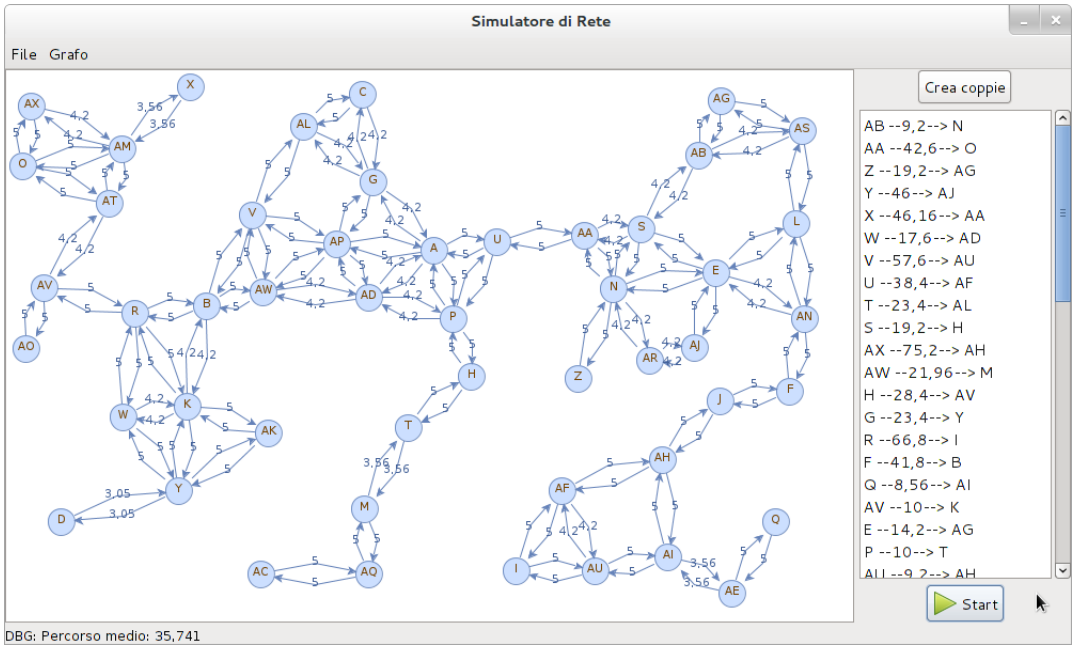

Figura 3.3: L'applicativo realizzato, mentre esegue una simulazione, su piattaforma Linux

Capitolo 4

Risultati

4.1 Collaborare o non collaborare

L'aspetto più importante dei modelli basati sulla reiterazione del dilemma del prigioniero è l'evoluzione temporale della rete cui i nodi appartengono. Per evidenziare la situazione si è generata una rete di 50 nodi, rappresentata in figura 4.2, e si è provveduto a far variare il comportamento dei nodi. Dopo ogni esecuzione del dilemma del prigioniero tra due nodi si provvede a calcolare il percorso medio tra le coppie, che vengono generate automaticamente e mantenute costanti durante tutta la simulazione, che consta di 1000 iterazioni del dilemma del prigioniero; l'evoluzione di tale risultato è rappresentata del grafico di figura 4.1. Si è limitato lo sviluppo della simulazione alle prime mille iterazioni, in quanto sufficienti ad individuare chiaramente l'andamento e i valori asintotici dei costi.

Come è prevedibile, la simulazione in cui tutti i nodi sono in condizione di collaborare presenta un'evoluzione del peso medio decrescente, mentre di converso, quando tutti i nodi non collaborano il peso tende ad aumentare. Quello che è meno prevedibile è invece il comportamento della rete quando intervengono i nodi con la strategia *tit-for-tat*; bastano infatti quindici nodi (il 30%) che non collaborano e trentacinque tit-for-tat per *stabilizzare* sostanzialmente il valore medio del cammino *a costo minimo* tra le coppie, proprio perché questi negano la collaborazione in tutti i rami che li collegano con nodi NeverCooperate, amplificandone l'effetto complessivo nella rete poiché penalizzano tutte le comunicazioni che si trovano costrette ad attraversare quegli archi. Espressa in percentuale, questa strategia porta un peggioramento delle prestazioni di ben il 240% rispetto alla situazione prestazionalmente ottima in cui tutti i nodi collaborano tra loro.

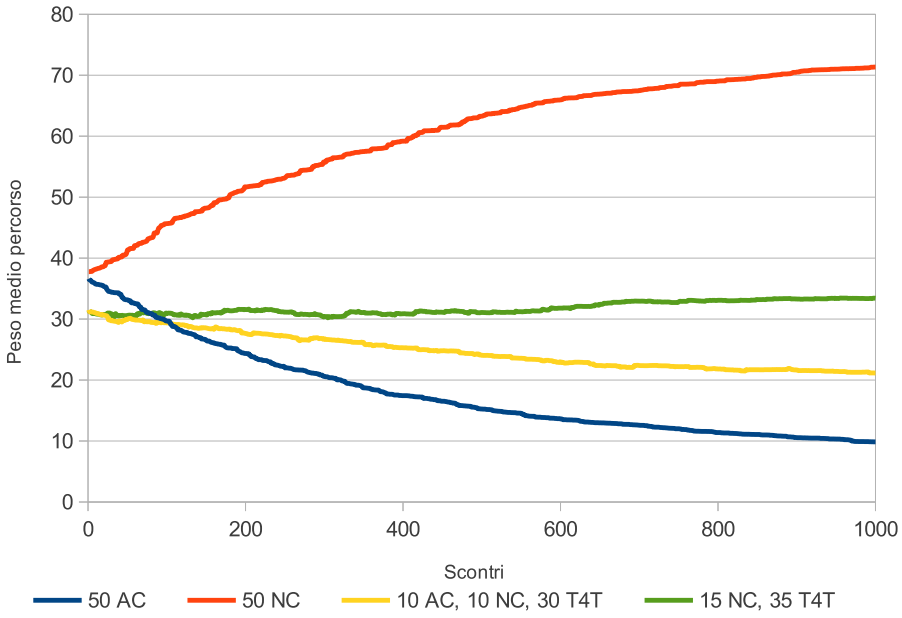

Figura 4.1: L'andamento temporale della simulazione. 50 nodi, 1000 iterazioni, $\alpha_{SAME} = 0.2$, $\alpha_{DIFF} = 0.25$

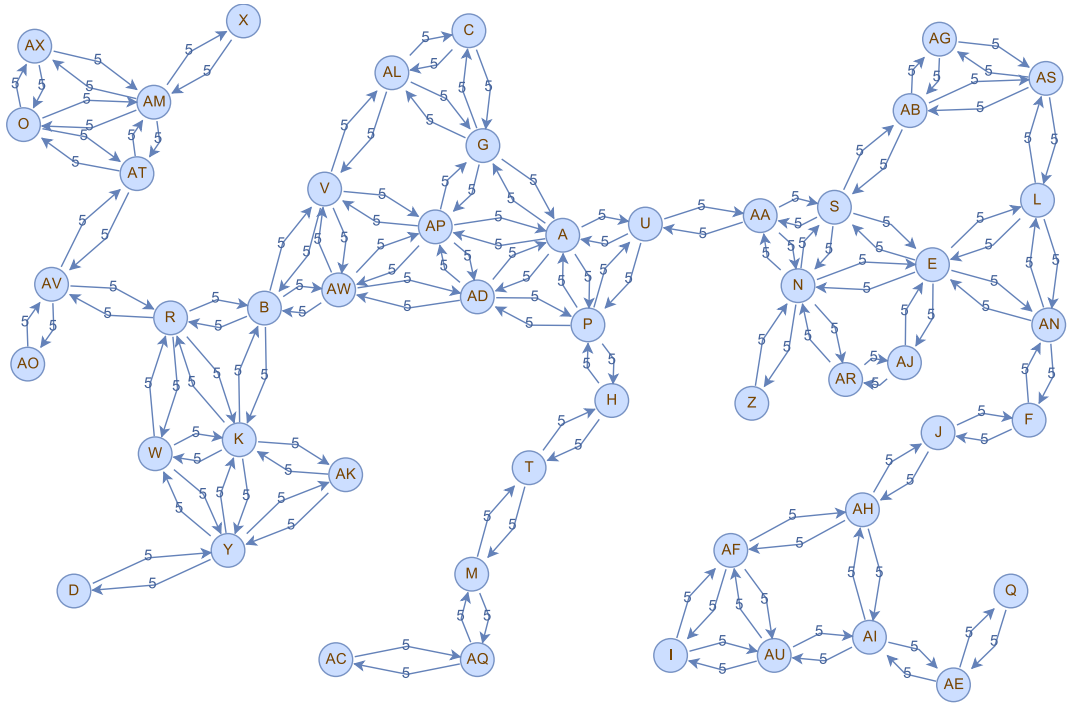

Figura 4.2: La rete della prima simulazione

La situazione intermedia (graficata in giallo) costituita da dieci nodi collaborativi, altrettanti non collaborativi e gli altri che applicano la strategia tit-for-tat, nonostante possa sembrare molto peggiorativa, in realtà, dopo un sufficiente intervallo di tempo, non si discosta molto dalla situazione ottimale, in cui tutti i nodi collaborano tra di loro, portando un peggioramento delle prestazioni di circa il 115%. Chiaramente si tratta di più del doppio del costo minimo, ma se comparato al +624% registrato nella situazione peggiore è comunque un risultato molto positivo; questa situazione potrebbe diventare interessante per la realizzazione pratica della rete, permettendo così di mantenere prestazioni comunque soddisfacenti fintantoché vi siano un numero sufficiente di nodi con le batterie cariche. Infine, è opportuno ricordare che le leggere differenze dei valori iniziali sono dovute al diverso abbinamento casuale dei nodi nelle coppie all'inizio della simulazione, ma non pregiudicano in alcun modo l'analisi asintotica delle prestazioni della rete stessa.

4.2 L'effetto dei nodi *tit-for-tat*

L'effetto amplificazione dei nodi *tit-for-tat* si può porre in risalto provando a simulare una rete con una percentuale variabile di nodi che non collaborano mai, e osservando l'andamento del percorso medio tra tutti i nodi, riportato nella figura 4.3. Gli altri nodi sono dapprima stati posti con strategia tit-for-tat, e successivamente come sempre collaborativi; per questa simulazione la topologia della rete è la stessa della precedente.

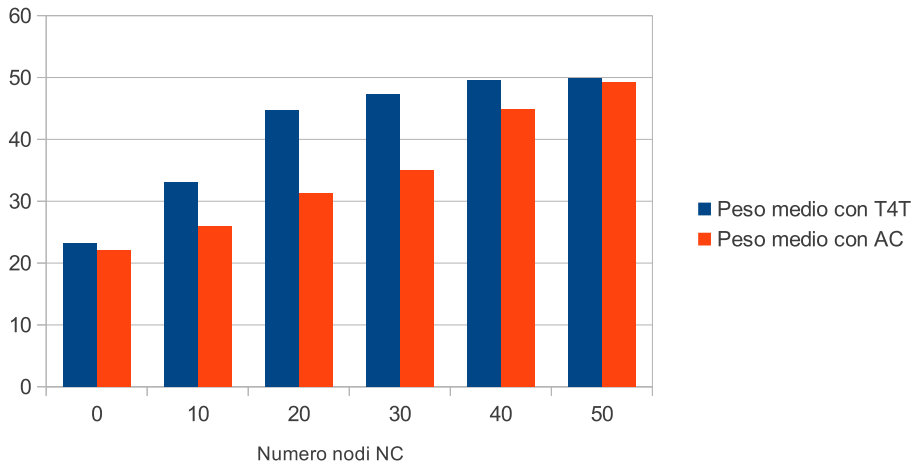

Figura 4.3: Percorso medio con percentuale variabile di nodi non collaborativi. 50 nodi, 250 scontri, $\alpha_{SAME} = 0.2$, $\alpha_{DIFF} = 0.25$

Com'è evidente, ponendo i rimanenti nodi collaborativi il costo medio cresce in maniera sostanzialmente lineare con i nodi NC; in una rete con strategia $T4T$ tuttavia, già con pochi nodi non collaborativi si ottiene una drastica penalizzazione del costo medio. Il motivo di tale penalizzazione si deve ricercare nel comportamento dei nodi $T4T$ che, per le connessioni ai nodi non collaborativi fanno sì che il peso degli archi cresca in ambo le direzioni, penalizzando così tutti i nodi che devono attraversare il link $NC \rightarrow T4T$, penalizzazione che nel caso dei nodi cooperativi non si verifica.

Ragionando come nell'esempio precedente in percentuale, la situazione in cui la differenza è più marcata si ha, per questa topologia di rete, in presenza di 20 nodi non collaborativi, dove il porre i rimanenti nodi con strategia tit-for-tat causa una penalizzazione nel calcolo del percorso a peso minimo medio di ben il 43%, quasi una volta e mezza quanto ottenibile in presenza di nodi collaborativi; più simile tra i due ambiti la situazione in cui sono presenti soli dieci oppure 30 nodi non collaborativi, che causano una differenza di costi rispettivamente del 27% e 35%.

Nodi N	Strategia C	Strategia $T4T$	Variazione
0	22.1	$23,2$	+5.11%
10	26.0	33.0	+27.10%
20	31.2	44.7	+43.02%
30	35.0	47.3	+35.31%
40	44.9	49.5	+10.26%
50	49.2	49.8	+1.20%

Tabella 4.1: Valori dei percorsi minimi, rappresentati nel grafico 4.3, e relativa variazione percentuale.

Attenzione che per questa simulazione le strategie dei nodi venivano distribuite in maniera *casuale*, in talune situazioni l'effetto potrebbe essere più marcato, oppure di contro meno incisivo. Si pensi ad esempio ad una rete con topologia *a stella*, ove il nodo centrale è connesso con tutti i nodi periferici, e questi sono a loro volta connessi solo al centro stella – non esistono collegamenti diretti detti tra *peer*; chiaramente tutti i percorsi di collegamento tra coppie di nodi coinvolgeranno sempre e solo due tratte (si esclude il caso, poco sensato nella realtà, di invio di pacchetti a se stessi, ottenibile con l'uso delle interfacce di rete virtuali dette *loopback*). Ebbene, qualora il nodo centrale non fosse collaborativo, ecco che la sua influenza all'interno della rete sarebbe massima, e si andrebbero a penalizzare

sicuramente le prime tratte dei collegamenti; qualora i nodi periferici a loro volta giochino $T4T$ ecco che la rete diventerebbe di fatto non cooperativa, facendo aumentare sensibilmente i costi dell'invio dei pacchetti. Un altro esempio potrebbe essere una rete costituita da due insiemi di nodi, sufficientemente connessi al loro interno, ma con un solo link che li collega tra loro; in questo caso, tutte le comunicazioni da una parte all'altra risentono direttamente e fortemente della situazione del link centrale, e verrebbero penalizzate da una mancanza di collaborazione.

4.3 La scelta dei coefficienti

La diversa scelta dei coefficienti di aggiornamento dei pesi α_{SAME} e α_{DIFF} ha maggior rilevanza nelle reti con nodi che prendono prevalentemente una decisione, mentre l'effetto diviene molto più erratico e imprevedibile quando si ha a che fare con nodi che giocano strategia *random*. Anche rieseguendo più volte le simulazioni, a parità di parametri, si ottengono ogni volta risultati diversi ed imprevedibili. I percorsi medi alla 250^{a} iterazione del dilemma del prigioniero tra nodi, in funzione del parametro α_{SAME} sono graficati in figura 4.4.

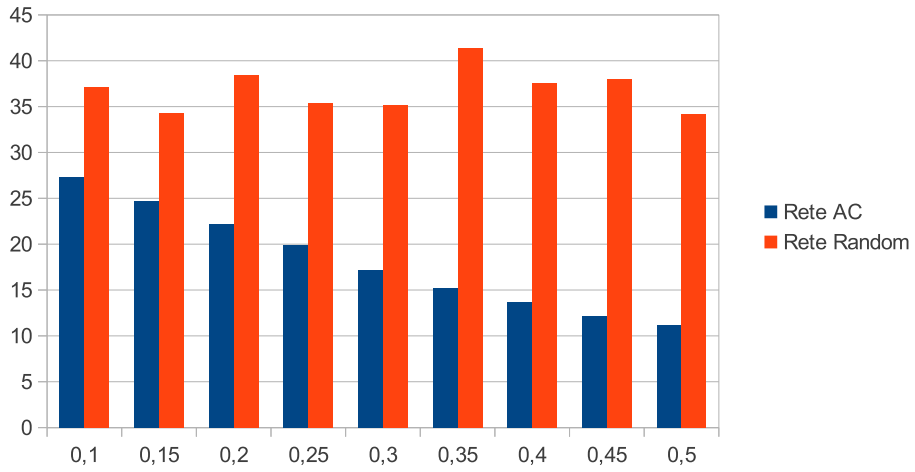

Figura 4.4: Percorso medio con α_{SAME} variabile, 50 nodi, 250 simulazioni, $\alpha_{DIFF} = \alpha_{SAME} + 0.05$ ove necessario, $P(C) = 0.5$

Come è possibile notare, e in un certo senso prevedere, i percorsi medi nel caso dei nodi collaborativi decrescono quasi linearmente, in accordo con le simulazioni matematiche della funzione di aggiornamento del peso dell'arco (figura 4.5); chiaramente, è una funzione a dominio discreto. Nel caso dei nodi con strategia

random – si è settata la probabilità di collaborazione $P(C) = 0.5$ per avere esattamente le stesse possibilità che il nodo decida di collaborare o non collaborare – l'andamento è completamente erratico e imprevedibile, nonché assolutamente non correlato alla scelta dei parametro α, poiché cambiando spesso decisione si assiste ad una sorta di *bilanciamento* tra gli aumenti e le diminuzioni del costo, che lo porta a non variare di molto.

Nel caso delle reti con nodi che mantengono le decisioni, al crescere del parametro α sono necessari sempre meno scontri per assestarsi verso i valori minimi o massimi asintotici dei pesi degli archi.

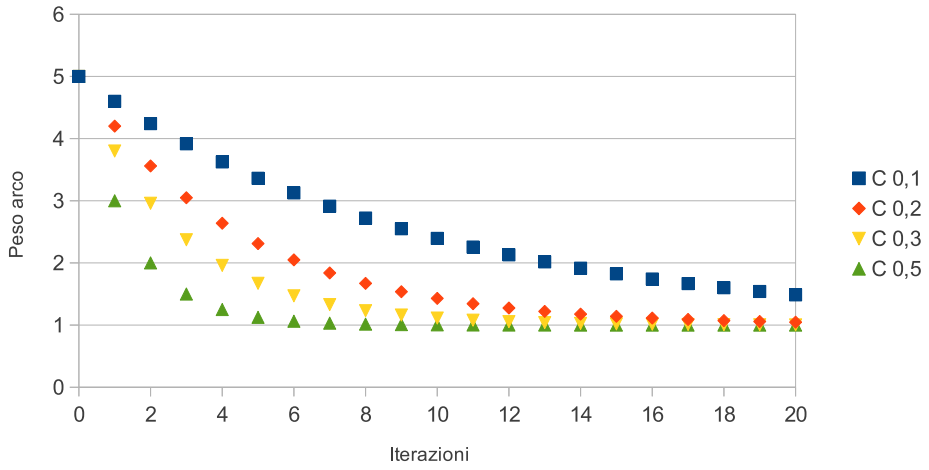

Figura 4.5: Funzione di aggiornamento del peso dell'arco

Capitolo 5

Conclusioni e sviluppi futuri

La teoria dei giochi permette di modellare una varietà di situazioni, e può coinvolgere potenzialmente un elevato numero di parametri.

Con le simulazioni e la realizzazione di questo applicativo si è voluto da una parte introdurre visioni e modellizzazioni delle reti che esulano dal programma didattico dei corsi di introduzione alle telecomunicazioni, e dall'altra dare la possibilità di simulare agevolmente l'effetto che può dare la variazione dei parametri di una rete.

É scontato ricordare come i risultati migliori si ottengano ponendo nelle reti il maggior numero possibile di nodi cooperativi, ma talvolta nella pratica ciò non è possibile oppure non è voluto. Occorre quindi progettare la rete in maniera accorta, distribuendo i nodi collaborativi in posizioni critiche, curandone se possibile il numero di collegamenti, in modo che siano facilmente raggiungibili. Inoltre, essere sempre collaborativo non è una scelta opportuna in ogni circostanza, in quanto è un comportamento prettamente altruistico [3], che porta sì a favorire gli altri nodi ma a scapito della personale efficienza. I massimi risultati si ottengono quando all'altro capo del link vi è un altro nodo disposto a sua volta a collaborare, portando così ad un basso costo di percorrenza per i pacchetti. Un compromesso tra il consumo energetico – e di risorse – e le prestazioni della rete si ottiene usando la strategia *tit-for-tat*, che ripaga i nodi collaborativi ed avversa i non collaborativi. Conseguenze diametralmente opposte si hanno quando la maggior parte dei nodi sceglie strategie egoistiche e non collabora con il resto della rete, permettendo sicuramente un risparmio di risorse ma peggiorando le performance dei collegamenti.

Come accennato nei commenti dei grafici, c'è da considerare anche la topologia della rete stessa; una rete fortemente connessa (numero di link proporzionale

a $O(V^2)$, ovvero con ogni nodo connesso a "quasi tutti" gli altri) soffre molto meno la presenza di nodi non collaborativi rispetto per esempio ad una struttura catenaria od a stella, anche se dalle simulazioni si è visto come comunque vi sia un deterioramento apprezzabile delle prestazioni qualora il numero di tali nodi sia elevato. Nelle reti a stella è importante avere la collaborazione del nodo centrale, in quanto tappa obbligata per ogni pacchetto immesso nella rete stessa; struttura questa che è la più diffusa su scala globale, dalle reti locali interne alle abitazioni, alle reti metropolitane, ai collegamenti *wireless* e su rete cellulare (GSM/UMTS).

La scelta di quanto far pesare una collaborazione assunta o mancata non ha influenze, contrariamente alle aspettative e convinzioni, nel comportamento e nelle prestazioni asintotici della rete, manifestando la propria importanza nei brevi periodi di tempo. Generalmente si tende a preferire un link dal costo stabile, piuttosto che variabile velocemente nel tempo, questo per poter anche prevedere la latenza introdotta per tutte quelle applicazioni che abbisognano di dati in tempo reale, come per esempio la telefonia via Internet (VoIP - Voice over IP), introducendo eventualmente delle scelte di indirizzamento che possono portare a percorsi più lenti ma con parametri costanti nel tempo.

5.1 Gli sviluppi futuri

Come già accennato, l'applicativo è alla sua prima stesura, e molte modifiche possono essere apportate ad esso, sia per praticità dell'interfaccia utente, ma soprattutto per quanto concerne il "motore" simulativo che pulsa sotto al mero layer utente.

5.1.1 L'aggiunta di strategie

Come detto, il caricamento delle strategie disponibili avviene a *run-time*, quindi anche non volendo ricompilare il programma è fattibile e semplice ideare nuove strategie, che possono essere modellate dalla realtà, da studi di altri esperti o Università nel settore, o perché no, ideate dalla fantasia dell'utente, e testate in una rete-tipo in poco tempo.

5.1.2 L'espansione dei limiti

Alcuni limiti possono essere facilmente superati, quale a titolo di esempio il numero di nodi, limitato in questa versione a 50, tenendo sempre in considerazione le esigenze di potenza di calcolo e memoria che la simulazione di reti di grandi dimensioni richiede. Il programma è scritto in Java, quindi le prestazioni pure sono comunque limitate rispetto ad una implementazione ad esempio in linguaggio C++ senza layer utente, ma ciò avrebbe richiesto molto più tempo per lo sviluppo nonché difficoltà da parte dell'utente per l'uso. Java è un linguaggio *interpretato*, anche se lo sviluppo dei compilatori *Just-In-Time* (*JIT*) ha di molto abbattuto l'impatto che la virtual machine ha sulle prestazioni complessive. Impostando ad esempio un ridotto intervallo ($<$ 50ms, anche se è fortemente dipendente dalla potenza di calcolo del sistema) tra le iterazioni del dilemma del prigioniero non è raro vedere il programma "accelerare" il ritmo dopo i primissimi cicli proprio per l'entrata in funzione automatica del compilatore JIT della macchina virtuale.

Appendice A

Diagramma UML delle classi

Figura A.1: La prima parte del diagramma UML

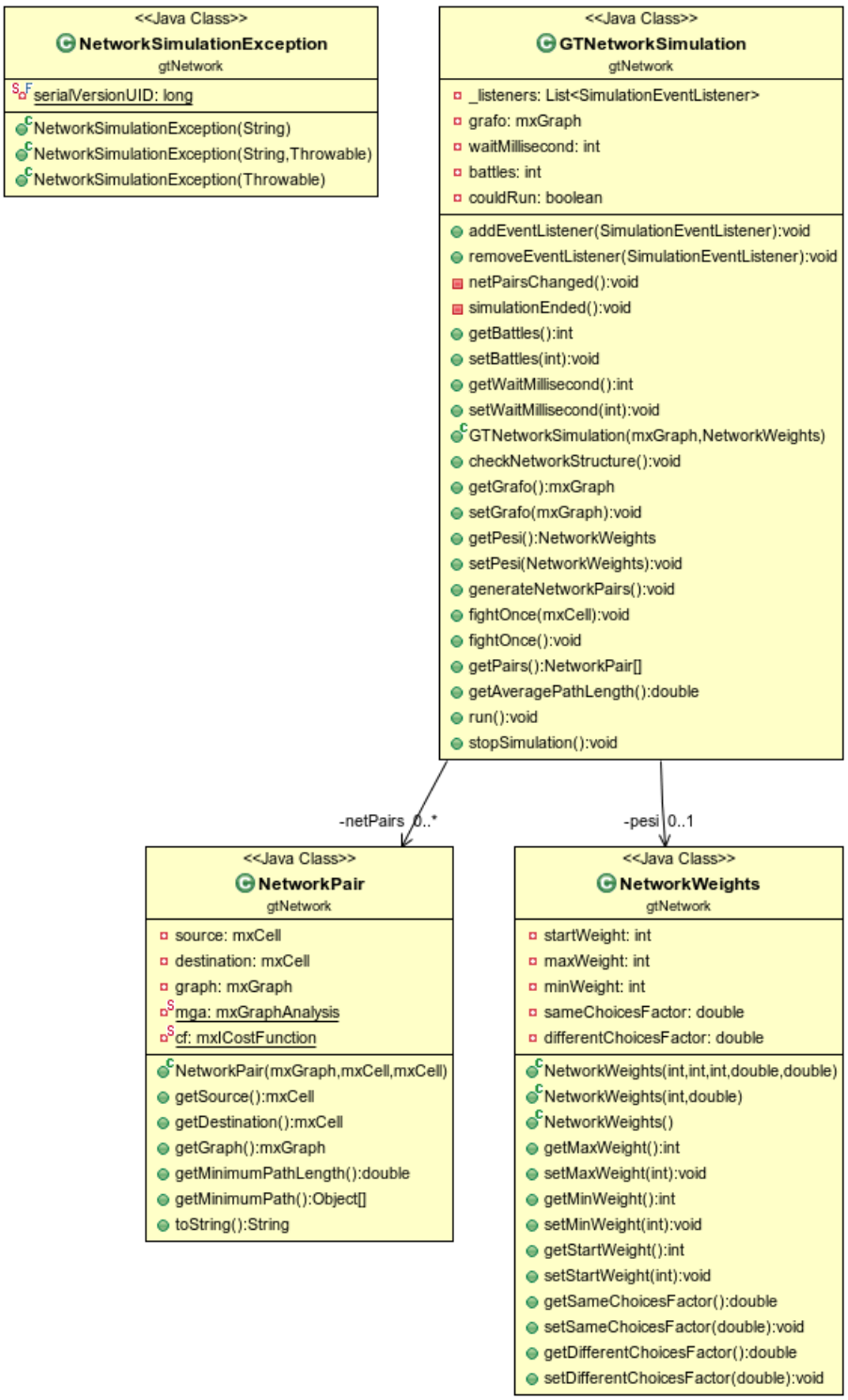

Figura A.2: La seconda parte del diagramma UML

Bibliografia

[1] Alessandro Agnetis. *Introduzione alla Teoria dei Giochi*. `http://www.dii.unisi.it/~agnetis/introgiochi.pdf`.

[2] Rudolf Ahlswede, Ning Cai, Shuo yen Robert Li, and Raymond W. Yeung. Network information flow. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 46(4):1204–1216, 2000.

[3] Leonardo Badia, Marco Levorato, Federico Librino, and Michele Zorzi. Cooperation techniques for wireless systems from a networking perspective. *IEEE Wireless Communications*, 2(17):89–96, April 2010.

[4] Nevio Benvenuto and Michele Zorzi, editors. *Principles of Communications, Networks and Systems*. Wiley, 2011.

[5] D.P. Bertsekas and R.G. Gallager. *Data networks*. Prentice-Hall international editions. Prentice Hall, 1992.

[6] Melvin Dresher. *Games of Strategy: Theory and Applications*. Prentice-Hall applied mathematics series. Prentice-Hall, 1961.

[7] Melvin Dresher. *The Mathematics of Games of Strategy*. Dover Books on Mathematics Series. Dover Publications, 1981.

[8] David Easley and Jon Kleinberg. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge University Press, 2010.

[9] Merrill Meeks Flood. *A preference experiment*. Rand Corporation, 1952.

[10] Merrill Meeks Flood. *Some experimental games*. Project Rand research memorandum. Rand Corporation, June 1952.

[11] Piyush Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 46(2):388–404, 2000.

[12] ICANN. Available pool of unallocated ipv4 internet addresses now completely emptied. `http://www.icann.org/en/news/releases/release-03feb11-en.pdf`, 2011.

[13] Leonard Kleinrock and Farouk Kamoun. Hierarchical routing for large network - performance evluation and optimization. *Computer networks: The International journal of distributed informatique*, 1(3), 1977.

[14] Kevin Leyton-Brown and Yoav Shoham. *Essentials of Game Theory: A Concise, Multidisciplinary Introduction*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2008.

[15] John F. Jr. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America*, 36(1):48–49, 1950.

[16] Masahiro Ono and Mitsuru Ishizuka. Prisoner's dilemma game on network. In *Proceedings of the Eighth Pacific-Rim International Workshop on Multi-Agents, PRIMA2005, Kuala lumpur*, pages 9–22, 2005.

[17] Martin J. Osborne. *An Introduction to Game Theory*. Oxford University Press, USA, 2003.

[18] William Poundstone. *Prisoner's Dilemma*. Anchor Books, 1992.

[19] J. A. Pouwelse, P. Garbacki, D.H.J. Epema, and H. J. Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In *4TH International Workshop on Peer-to-Peer Systems (IPTPS)*, 2005.

[20] Kavitha Ranganathan, Matei Ripeanu, Ankur Sarin, and Ian Foster. To share or not to share, an analysis of incentives to contribute in collaborative file sharing environments. `http://www2.sims.berkeley.edu/research/conferences/p2pecon/papers/s1-ranganathan.pdf`.

[21] Tim Roughgarden. *Selfish Routing*. PhD thesis, Cornell University, `http://theory.stanford.edu/~tim/papers/thesis.pdf`, 2002.

[22] Tim Roughgarden. *Selfish Routing and the Price of Anarchy*. The MIT Press, 2005.

[23] Leila Shayanpour. Game theory in network design. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.132.6863`, 2008.

[24] Karthik Tamilmani. *Robustness of Bittorrent Protocol: A study on enhancing the Bittorrent Protocol*. CreateSpace Independent Publishing Platform, 2011.

[25] Zheng Wang and Jon Crowcroft. Analysis of shortest-path routing algorithms in a dynamic network environment. *ACM SIGCOMM Computer Communication Review*, 22(2):63–71, April 1992.