
ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA
SEDE DI CESENA
SECONDA FACOLTÀ DI INGEGNERIA CON SEDE A
CESENA
CORSO DI LAUREA SPECIALISTICA ICT

EPIDEMIE IN COMUNITA' DI AGENTI

Sistemi Intelligenti Distribuiti LS

Prof. Andrea Omicini

Presentata da:
Alessandro
De Simone

Anno Accademico 2005-2006

Indice

| | |
|---------------------------------|----------|
| Introduzione | 1 |
| Descrizione del problema | 1 |
| Analisi | 2 |
| Progetto | 4 |
| SimpleAgent | 4 |
| GuiAgent | 6 |
| OrgAgent | 6 |
| SickAgent | 6 |
| Protocollo | 7 |
| Conclusioni | 7 |

Introduzione

Lo scopo di questo lavoro è di creare un sistema software che simuli la diffusione di una malattia contagiosa in una comunità di persone. Una volta creato questo modello, si potranno poi analizzare le dipendenze tra la diffusione del virus e i vari aspetti (parametrizzati) di comportamento sociale, come ad esempio la predisposizione alla vaccinazione preventiva o la mobilità degli individui.

Descrizione del problema

In una comunità di individui piuttosto numerosa viene inserito un individuo malato. Ogni individuo ha la necessità di muoversi, sia per lavoro che per diletto. Come noto, un individuo malato non riconosce di stare male per un certo periodo, durante il quale continua le sue normali attività (in particolare a muoversi). Visto che la malattia è contagiosa, in prossimità di un altro individuo, anche costui si ammala. Tutti gli individui, tra cui anche quelli già contagiati, possono riconoscere la malattia solo ad un livello avanzato (dopo un certo tempo): un individuo che si scopre ammalato o va in ospedale o muore entro un certo tempo, un individuo che vede da lontano un malato scappa, e comunica agli altri individui sani che incontra di vaccinarsi. In base alla quantità di vaccini disponibili ed alla possibilità di poterselo procurare, quindi, ogni individuo a conoscenza della possibilità di ammalarsi tenta di vaccinarsi. Esistono sul territorio dei centri sociali, dove gli individui si raggruppano, permangono per un pò di tempo e poi se ne vanno in cerca di un altro centro sociale.

Una volta definito questo modello del mondo, si può pensare di aggiungere dettagli che avvicinano il modello alla realtà. Si può introdurre il concetto di credito, necessario per la sopravvivenza. Il credito di ogni individuo si quantifica in distanza percorsa. Supponendo che da un centro sociale ad un altro si ha necessità di trasportare qualcosa, un individuo accumula distanza percorsa se porta a buon fine il trasporto che gli era stato affidato dal centro sociale A al centro sociale B. Un individuo spende per ogni giorno di vita un credito, ed il vaccino ha un costo in crediti proporzionale. In ospedale ogni individuo non spende niente. Se un individuo non ha crediti muore. Da qui ne diviene che l'obiettivo individuale è la sopravvivenza.

Analisi

La scelta delle tecnologie utilizzate si basa sulle conoscenze acquisite durante il corso. Dalla descrizione del problema emerge che un Multi Agent System (MAS) è idoneo alla implementazione. In breve le caratteristiche di un agente MAS sono le seguenti:

“entità autonoma situata che persegue il suo goal interagendo con l'esterno”

Per la descrizione che abbiamo acquisito a lezione di agente, è ovvio che tutte le entità presenti nel sistema sono agenti¹. Il territorio dove gli agenti si possono muovere viene discretizzato con una griglia. In particolare ogni agente ha una visione limitata del mondo, può vedere altri agenti solo se sono nelle celle adiacenti a quella che sta occupando. Ogni cella può contenere tutti gli agenti del sistema, non ha un limite superiore e quindi non ha un vero senso fisico di zona delimitata di territorio. Il territorio e i centri sociali vengono realizzati con dei centri di tupla. Al fine di poter estendere il sistema ad un numero notevole di agenti e con un territorio esteso, quindi di distribuire il sistema su più calcolatori, ho ipotizzato di suddividere il territorio in zone, composte da un certo numero di celle adiacenti. Ogni zona è sotto il controllo di un centro di tupla.

L'agente che si vuole muovere indica al centro di tupla che ha il controllo della zona in cui si trova, la direzione in cui si vuole spostare. Il centro di tupla risponde o con una risposta negativa (impossibilità di muoversi in quella direzione) o con la nuova visione del mondo dell'agente.

Ad esempio:

Agent Roma, can I move right-up?

Roma Yes, u can. You are in (1,1). Near you there are:

up Pippo, Pluto

up-left nobody

up-right nobody

down nobody

down-left Topolino

down-right Minni

¹Anche se in realtà alcune entità non hanno tutte le proprietà degli agenti, vengono modellate come tali solo per maggior chiarezza o omogeneità.

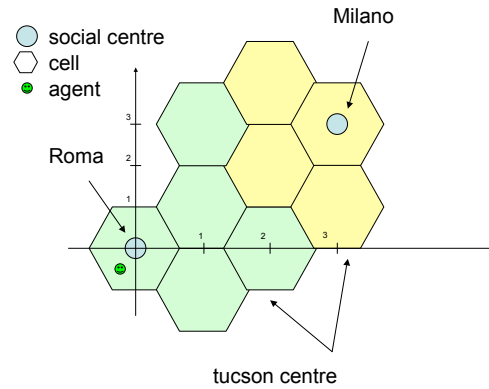


Figura 1: Suddivisione spaziale del territorio.

| Individuali | Sociali |
|--|---|
| Tempo di incubazione del virus [giorni] | Costo del vaccino [crediti] |
| Paga [crediti/cella] | Costo della vita [crediti/giorno] |
| Velocità degli agenti [celle/giorno] | Distanza a cui avviene il contagio [celle] |
| Giorni di vita rimasti a un agente malato [giorni] | Distanza a cui avviene la comunicazione [celle] |

Tabella 1: Variabili individuali e sociali.

Agent Roma, can I move right-up?

Roma No. Ask to Milano

Agent Milano, can I move right-up?

Milano Yes, u can. You are in (2,2). Near you there are ...

Ovviamente ogni zona deve sapere con chi confina. Un agente che sta in una cella di confine non è in contatto con le celle della zona confinante. Può essere visto come un modello che rappresenta l'ambientale reale (montagne, confini...)

Progetto

Dalla fase di analisi sono emersi alcuni requisiti essenziali, in particolare l'esigenza di poter distribuire l'applicazione sulla rete. Il software è basato in gran parte sulla coordinazione tra gli agenti. Serve quindi una infrastruttura che abiliti la comunicazione (in questo caso comunicazione generativa) tra gli agenti e che possa applicare una politica di coordinazione efficiente e flessibile.

E' da tenere in considerazione che l'implementazione viene pianificata a priori. Ci dovrà essere la possibilità di ispezionare il lavoro fatto ad intervalli regolari e comunque di poter avere sempre una visione di insieme che, seppur non completa, caratterizza il progetto in modo trasparente. Ho scelto di utilizzare come infrastruttura TuCSoN, in quanto soddisfa le esigenze di progetto.

Per quanto riguarda gli agenti, si deve considerare che le informazioni elaborate da ciascuno di essi sono facilmente esprimibili in un linguaggio simbolico. La scelta di usare Prolog per la loro implementazione è la più ovvia.

In figura 2 è possibile notare l'architettura logica del sistema.

SimpleAgent

Questa teoria prolog rappresenta un agente che vive nel mondo Pandemia. Riassumendo le sue proprietà, possiamo notare se appartiene effettivamente alla astrazione dei MAS.

- autonomo e pro-attivo
ogni agente-SA ha un suo flusso di controllo, condivide con le altre entità solo i dati (attraverso il protocollo specificato in seguito). E' possibile creare ogni agente-SA su un calcolatore diverso, in modo da avere la completa autonomia tra tutti gli agenti. Il task di un agente-SA è quello di viaggiare per il mondo cercando di non ammalarsi ²
- interattivo
ogni agente-SA ha una percezione limitata del mondo. In particolare può vedere solo attraverso gli artefatti di coordinazione nel contesto di

²Non è stata implementata una parte importante del progetto, cioè il comportamento di un agente-SA dal momento in cui si ammala. Ad esempio, andrebbero aggiunte alcune clausole alla teoria prolog, in modo che l'agente malato vada in un centro sociale (ospedale) e vi rimanga. Da questa fase in poi, emergerebbe anche il task sociale, ovvero la sopravvivenza di più agenti possibile.

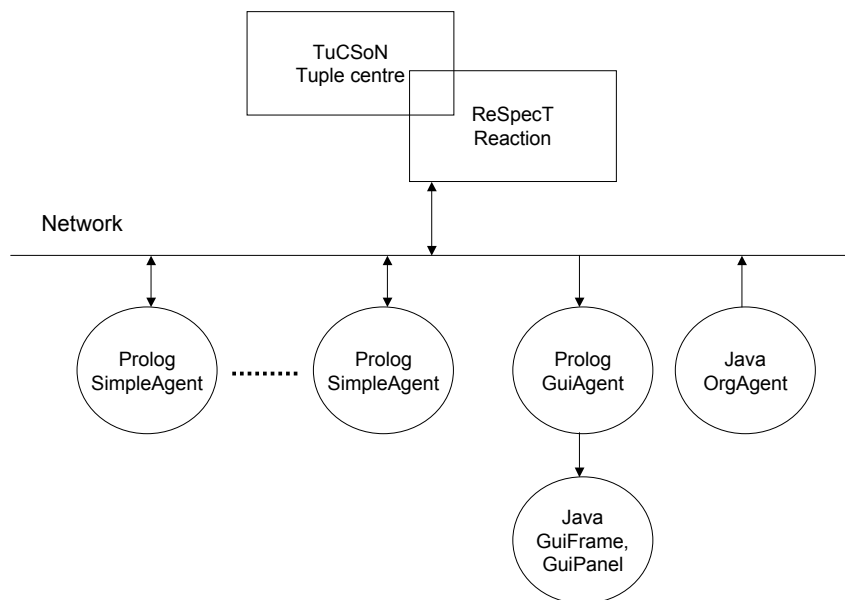


Figura 2: Architettura logica.

questo mondo (Pandemia). Al di fuori del suo mondo, un agente-SA non ha nessuna capacità

- impara
la sua conoscenza si amplia durante la sua vita. In particolare sa ricordarsi la posizione dei centri sociali e sa raggiungerli
- è intelligente
un agente-SA sceglie sempre il tragitto più breve tra se stesso e la sua meta, evitando di andare nelle celle occupate da un agente malato. Se ha ampliato le sue conoscenze della geografia del mondo, sa orientarsi
- è “distribuito”
ogni agente può essere dislocato nella rete
- gli agenti-SA agiscono sempre in buona fede
- vive in società
per il semplice motivo che il sistema con un singolo agente è inutile
- vive in un ambiente (agent environment)
un agente-SA vive nel mondo Pandemia, implementato come un insieme di nodi TuCSon sui quali vengono creati dei tuple centre che raccolgono le tuple che rispettano il protocollo.

GuiAgent

GuiAgent serve per visualizzare lo stato del mondo. Utilizza la potenzialità di tuProlog di integrarsi con l'ambiente Java. In questo linguaggio sono state implementate le classi necessarie a creare una piccola interfaccia grafica.

OrgAgent

A questo agente è delegato il compito di inizializzare e tenere organizzato il sistema.

SickAgent

Come SimpleAgent, ma inizia a vivere già ammalato.

Protocollo

Ogni agente che vuole esistere nel mondo **pandemia** deve seguire il seguente protocollo. Per eseguire un movimento, l'agente deve scrivere una tupla con la seguente struttura

```
wantToMove(ID,Feel,p(X,Y))
```

dove ID rappresenta l'identificatore dell'agente (non sono state create politiche per la gestione della omonimia perchè l'applicazione è sperimentale); **Feel** indica lo stato di salute dell'agente e può essere **good** o **sick**; **p(X,Y)** è la posizione che vuole raggiungere l'agente con la sua mossa.

L'agente deve rimanere in attesa della tupla di risposta

```
youAre(ID,Feel,p(X,Y))
```

che indica la posizione e lo stato di salute attuali.

Conclusioni

Questo progetto è servito a familiarizzare con alcuni concetti dei sistemi distribuiti ad agenti. Nel progetto sono stati risolti alcuni problemi di intelligenza artificiale e si è indagato sul concetto di intelligenza sociale e task sociale. A seguito dell'implementazione, ho acquisito familiarità sia con TuC-SoN, sia con tuProlog (ed approfondito il linguaggio prolog). Nonostante il comportamento del sistema rispetti le attese, la simulazione, a questo livello di implementazione, risulta di scarsa utilità.