UNIVERSITA' DEGLI STUDI DI BOLOGNA
II FACOLTA' DI INGEGNERIA

Corso di laurea magistrale in Ingegneria Informatica

"Multi-Agent Routing in Mobile Ad-hoc Networks"

Progetto di Sistemi Multi-Agente LM
A.A. 2010-2011

Paolo Piagnani
paolo.piagnani@studio.unibo.it
Matricola 490537

Indice

1. Introduzione

Negli attuali sistemi di comunicazione, ormai orientati verso tecnologie wireless, è necessario avere reti formate da nodi mobili, fra loro indipendenti ed in grado di coordinarsi senza bisogno di un'infrastruttura centralizzata ed organizzata.
Le reti mobili ad-hoc sono dunque il futuro di un tipo di comunicazione estremamente dinamico, basato su topologie di rete spesso non predicibili, dove i nodi possono spostarsi arbitrariamente. Ci si aspetta che le dimensioni fisiche di una struttura di questo tipo vadano oltre quelle del raggio radio di una delle interfaccie wireless coinvolte, ed è quindi necessario l'utilizzo di tecniche di routing e di gestione del traffico al fine di permettere lo scambio di informazioni fra due qualsiasi nodi della rete. A differenza delle infrastrutture cablate, le reti mobili devono garantire tempi di risposta minimi per far fronte ai continui cambi di topologia, e soprattutto devono saper ottimizzare le risorse a disposizione, ovvero la quantità limitata d'energia che alimenta i dispositivi mobili connessi, ed una banda decisamente inferiore rispetto alle reti fisse.
Le reti mobili ad-hoc sono quindi da anni materia di studio sulla quale si cerca di proporre modelli ed implementazioni partendo da approcci diversi. Uno di questi è senza dubbio un modello di sistema multi-agente (d'ora in poi MAS). In questo documento si descrive allora un modello generale per degli agenti software che, una volta implementati nell'ambiente ed attivati, siano in grado di formare tramite comunicazione autonoma con i relativi vicini, una rete mobile ad-hoc che consenta a tutti i nodi presenti e raggiungibili di esservi connessi. Lo scopo è quello di comprendere le meccaniche di questo tipo di reti, e studiarne gli aspetti fondamentali.
Gli agenti vengono definiti tramite l'estensione del linguaggio astratto Agentspeak fornita dal framework multi-agente Jason, e vengono poi implementati in un ambiente simulato e centralizzato utilizzando le librerie della stessa piattaforma, al fine di illustrarne le dinamiche di base. Agentspeak permette una definizione elegante e pratica degli agenti, e Jason ne estende le funzionalità.

2. Mobile Ad-hoc Network: MANET

2.1 Definizione

Una rete mobile ad-hoc (d'ora in poi MANET) è una rete di dispositivi mobili in grado di configurarsi autonomamente tramite collegamenti wireless, senza quindi bisogno di un'infrastruttura d'appoggio. Ogni nodo è libero di muoversi arbitrariamente in qualsiasi direzione, cambiando i propri collegamenti con i nodi vicini con frequenza elevata, e dev'essere in grado di inoltrare il traffico non destinato ad esso, fungendo da vero e proprio router. Ciascun dispositivo è quindi in grado di rilevare il segnale radio dei propri vicini e di effettuare una richiesta di connessione, al fine di formare appunto una rete mobile ad-hoc, o di unirsi ad una già esistente, e deve essere in grado di poter scambiare informazioni con qualsiasi altro nodo connesso a quella rete. Dato che una scheda radio ha un raggio di propagazione del segnale limitato, vengono implementati dei meccanismi per permettere il multi-hop. Una MANET ha le seguenti caratteristiche:

- autonoma e priva di infrastruttura;
- traffico viene inoltrato utilizzando il multi-hop;
- topologia della rete evolve dinamicamente;
- eterogeneità dei dispositivi;
- collegamenti hanno banda limitata e variabile;
- dispositivi ad energia limitata;

- scalabilità;
- autoconfigurazione, autorganizzazione, ed autoamministrazione.

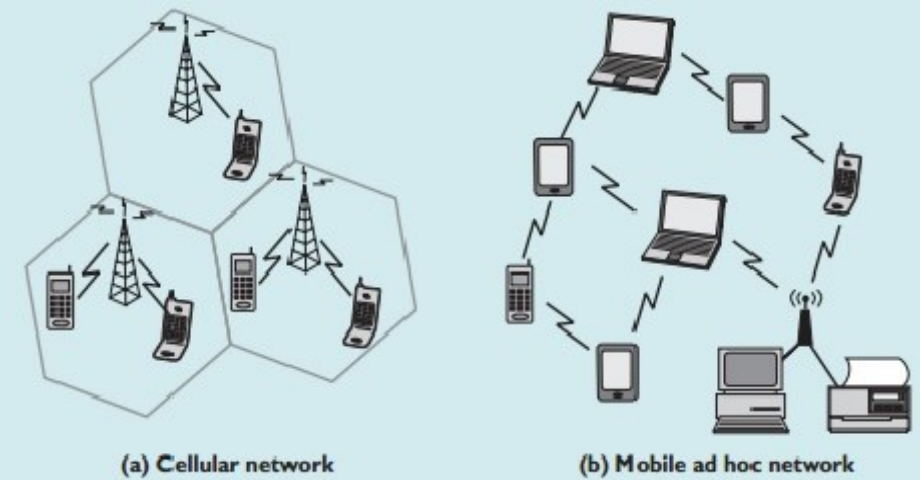

Fig.1: Reti rispettivamente con e senza infrastruttura.

Una MANET è dunque una rete estremamente dinamica, dove la comunicazione tra i nodi collegati viene mantenuta ed ottimizzata nonostante i continui cambiamenti nei costi di trasmissione.

2.2 L'Importanza delle MANETs

In un mondo orientato sempre più verso la mobilità ed il risparmio di risorse, è necessario garantire dei sistemi di comunicazione di veloce configurazione, accurati, affidabili, e soprattutto efficienti, in qualsiasi scenario possibile, anche non prevedibile.
Grazie proprio all'aumento di dispositivi portatili come notebook, tablet, smart phone (per citarne alcuni), ciascuno dei quali provvisto di interfacce wireless a corto raggio, si ha anche un conseguente incremento dello scambio dati su breve distanza, rimpiazzando i collegamenti diretti basati su cablaggi ethernet, usb, ecc. In aggiunta, questi dispositivi diventano sempre più piccoli, più economici, e più potenti. La comunicazione mobile si orienta quindi verso un modello di rete autogenerante basata su utenti tra loro indipendenti che si organizzino ed amministrino autonomamente, senza quindi necessità di comunicare direttamente con un access point.
Le reti mobili ad-hoc stanno trovando impiego in un numero crescente di campi, partendo dalle operazioni militari e di soccorso, fino ad arrivare ad applicazioni per gli utenti privati. Questo tipo di reti è adatto a tutte quelle situazioni dove le infrastrutture di rete non sono disponibili, non sono affidabili, o sono semplicemente troppo costose. Grazie alla reattività nell'autoconfigurarsi ed autogestirsi, una MANET può essere velocemente impiegata con impegno minimo da parte dell'utente, senza bisogno di una dettagliata pianificazione del cablaggio e della disposizione dei dispositivi. Inoltre, una MANET non dev'essere necessariamente fine a sé stessa, ma può essere attaccata alla rete Internet, aumentando i servizi offerti agli utenti.
Le tecnologie su cui vengono implementate le reti mobili ad-hoc sono tutte quelle prive di collegamenti materiali, quindi le schede IEEE 802.X (Bluetooth incluso) che trasmettono tramite segnale radio, oppure tecniche ad infrarossi come IrDA.

3. Sistemi Multi-Agente

3.1 Il Concetto di Agente

Dalla definizione di Russell&Norvig, un agente intelligente è un entità autonoma in grado di osservare un ambiente tramite sensori e di agire tramite attuatori, e dirige la propria attivita al fine di raggiungere un certo obiettivo. Tali agenti possono imparare ed utilizzare una base di conoscenza per raggiungere i propri obiettivi.
Dato che ciascun agente è un'entità autonoma, deve non solo incapsulare il proprio flusso di controllo, ma deve anche incapsulare un certo criterio per dirigere la propria attività. Inoltre, l'obiettivo da raggiungere non dev'essere per forza predeterminato, ma può cambiare nel tempo a seconda delle necessità dello stesso agente e dell'ambiente circostante che si modifica in continuazione.

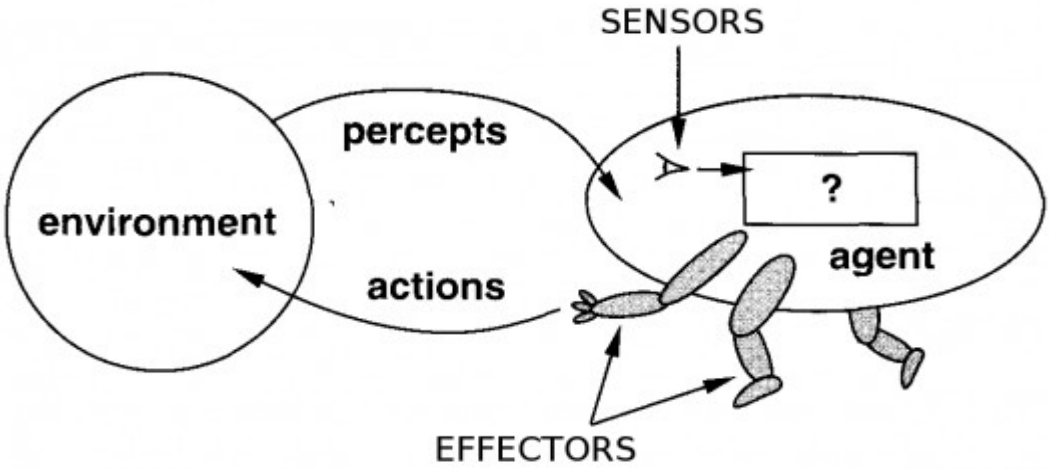

Fig.2: Il concetto di agente

Gli agenti autonomi sono proattivi. Significa che sono in grado di adattarsi all'ambiente in cui vivono: non si limitano quindi a reagire a degli stimoli, ma si pongono degli obiettivi e cercano di raggiungerli. Non essendo né onniscenti né onnipotenti, il loro comportamento e le loro reazioni si basano su percezioni dell'ambiente circostante, che possono essere più o meno accurate. In un certo senso, gli agenti autonomi cercano di prevedere un cambiamento del sistema e conseguentemente di adattarvisi, prendendo anche delle iniziative.
Un agente intelligente è dunque un'entità:

- Reattiva: reagisce ai cambiamenti dell'ambiente;
- Proattiva: interpreta i cambiamenti dell'ambiente tramite ragionamenti e si attiva eventualmente secondo un proprio criterio;
- Situazionale: è legato in maniera stretta all'ambiente in cui interagisce;
- Sociale: comunica e coopera con altri agenti situati nello stesso ambiente.

3.2 Il Modello BDI

Il modello Belief-Desire-Intention (BDI) si basa sull'idea che si possa considerare gli agenti, cioè delle entità software, come se avessero uno "stato mentale":

- Belief (credenza): è un'informazione che l'agente possiede riguardante il mondo circostante. Tale informazione non è una verità assoluta, e può dunque essere datata o non accurata.
- Desire (desiderio): è uno dei possibili stati che l'agente potrebbe voler raggiungere. Non è detto però che l'agente si attivi per raggiungerlo. E' una potenziale influenza sulle azioni che compirà l'agente, ed è quindi un'opzione da seguire, ma non un obbligo.
- Intention (intenzione): rappresenta uno stato che l'agente ha deciso di raggiungere. Tale obiettivo può essere delegato da terzi, oppure decretato autonomamente dall'agente tramite un sistema di ragionamento.

Si può considerare l'intenzione come un'azione che l'agente decide di intraprendere dopo aver scelto fra le varie opzioni a sua disposizione, cioè i propri desideri.
Quello che il modello BDI offre è una visione meno tecnica e più intuitiva di sistemi complessi. Si può pensare ad una programmazione di tipo post-dichiarativo, dove si cerca di ridurre l'enfasi sugli aspetti del controllo.

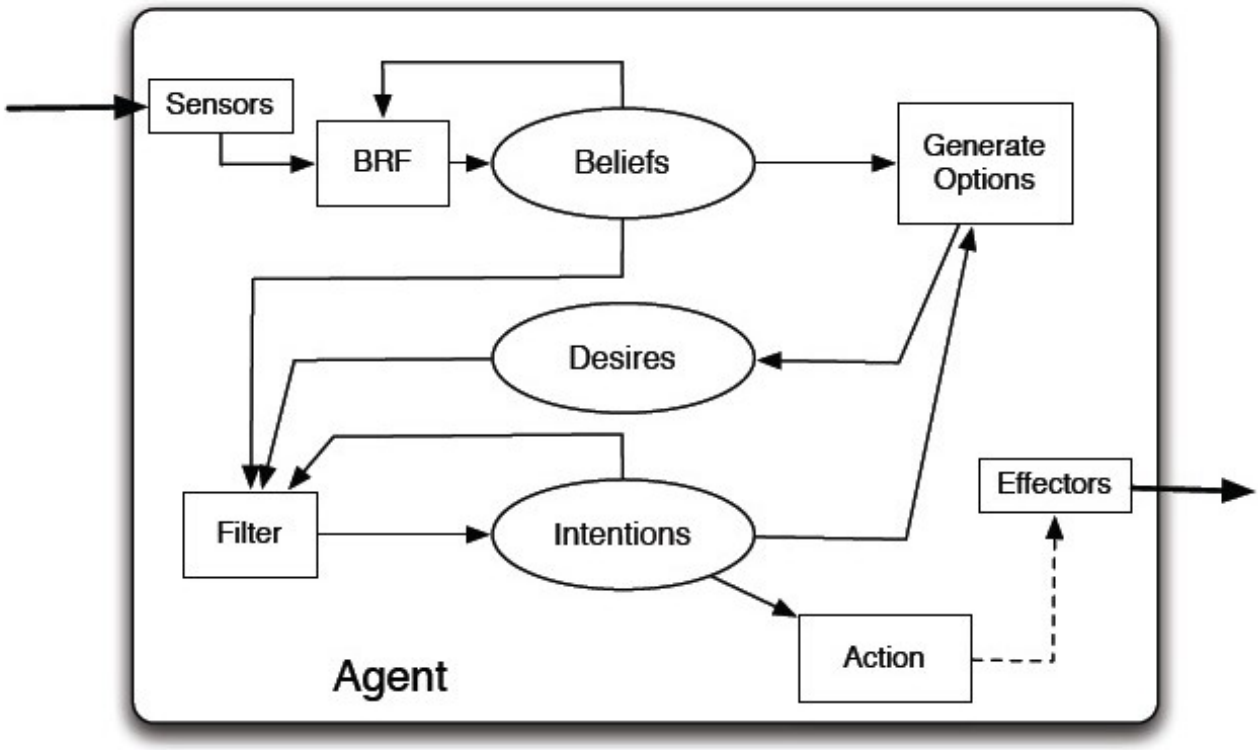

Fig.3: Architettura di base di un agente BDI [Wooldridge, 2002]

4. L'Approccio al Problema

4.1 MANET come Sistema Multi-Agente

Le reti mobili ad-hoc sono sistemi complessi, dove difficilmente il flusso di controllo può essere centralizzato, e dove quindi ogni entità partecipante deve essere indipendente e coordinarsi autonomamente al fine di inizializzare e mantenere la struttura di rete. Essendo continuamente oggetto di studio, è importante comprenderne a fondo le meccaniche, al fine di migliorare progressivamente l'efficienza del sistema e l'ottimizzazione delle risorse. Semplificare dunque le logiche di funzionamento di una MANET è un punto cruciale per capire proprio l'essenza del sistema in analisi, e per facilitarne poi la programmazione.
Pensare ad una MANET come ad un MAS viene quasi spontaneo. Si vedano le similitudini tra un nodo di tale rete ed un agente intelligente:

- Reattività: Un agente intelligente reagisce ai cambiamenti dell'ambiente. Lo stesso discorso vale per un nodo di una rete mobile, la cui scheda wireless percepisce le intensità dei segnali dei nodi vicini, e vi si collega in maniera dinamica.
- Proattività: Un agente intelligente reagisce o anticipa i cambiamenti dell'ambiente tramite dei ragionamenti. Un nodo di una MANET deve non solo reagire ai cambiamenti topologici della rete, ma deve capire e valutare le situazioni per ottimizzare le risorse disponibili. Ad esempio, se il routing da un nodo A ad un nodo B avviene tramite i nodi vicini A, C e D, A deve essere in grado di inoltrare i dati attraverso il percorso più efficiente che cambia continuamente.
- Situazione: Un agente intelligente, come un nodo di una MANET, è legato in maniera stretta all'ambiente in cui interagisce. Gli agenti devono essere eterogenei per poter organizzarsi fra loro, ed i nodi devono implementare gli stessi protocolli di scambio ed inoltro messaggi.
- Socializzazione: Le entità comunicano e cooperano fra loro per raggiungere un obiettivo comune: formare una rete mobile ad-hoc.

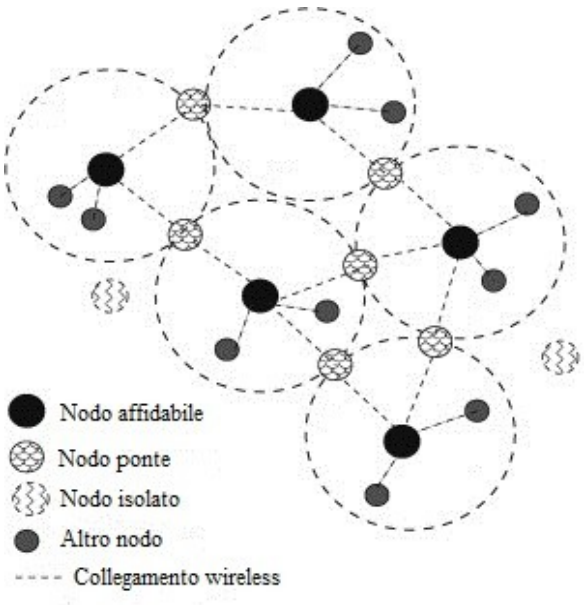

Fig.4: Un esempio di MANET. Visto come MAS, ogni nodo di rete è un agente.

Estendendo le similitudini tra MAS e MANET, si evidenzia il fatto che in una rete mobile esiste la possibilità concreta che un nodo sparisca dalla rete: definitivamente, cioè per esaurimento della batteria di un dispositivo, o temporaneamente per una perdita di segnale. Tale occorrenza può risultare in un calo di efficienza della rete, ma non per questo si deve bloccare l'intero sistema. Un agente che scompare dall'ambiente può quindi essere dimenticato, ma il resto dei nodi deve prenderne atto e reagire di conseguenza.
Si propone quindi una visione di una rete mobile ad-hoc dove ciascun nodo sia equivalente ad un agente intelligente. Tali agenti sono in grado di autogestirsi e coordinarsi al fine di inizializzare e mantenere una rete senza bisogno di un controllo centralizzato, ma solo tramite percezioni ambientali e scambio di informazioni tra loro.
Utilizzando poi il modello BDI, si pensi alle percezioni dei segnali wireless come alle percezioni dell'agente dall'ambiente, che vanno ad aggiornare la sua *belie base,* ovver la propria base di credenze. La rete viene formata tramite scambio di informazioni riguardanti le distanze tra un nodo ed i propri vicini, e sulla base di tali informazioni ciascun nodo costruisce una propria tabella di routing. Tramite una procedura standard, ciascun nodo ha l'intenzione di mantenersi connesso.

4.2 Routing e Costi di Rete

Una volta inizializzata, la MANET dev'essere in grado di automantenersi, cioè i nodi che vi appartengono devono, nei limiti fisici della percezione del segnale radio, garantire connettività ed efficienza. Ed è proprio quando si parla di ottimizzazione dei costi e risparmio di risorse che diventa fondamentale un approccio ad agenti, perchè le entità in gioco devono ragionare per prendere la scelta migliore, senza essere influenzate esclusivamente dalle percezioni dell'ambiente.
Si consideri ad esempio uno scenario standard in cui un nodo funzioni come router durante l'inoltro di messaggi fra due nodi qualsiasi della rete. Tale nodo si troverà sul percorso a costo minimo. L'invio di dati avviene via radio utilizzando la scheda wireless del dispositivo, e la trasmissione del segnale necessario comporta un certo consumo della batteria a disposizione, avente energia limitata. Dato che uno degli obiettivi di una MANET dev'essere quello di mantenere la connessione tra i nodi il più a lungo possibile, minore sarà il livello energetico della batteria di un dispositivo, maggiore diventerà il costo per inoltrare i pacchetti tramite quest'ultimo, finchè non diventerà più efficiente per la rete il routing attraverso un nodo differente.
Un discorso analogo può essere fatto per gestire la banda disponibile. Come già detto, i dispositivi wireless dispongono di schede di rete con capacità spesso inferiori rispetto ai dispositivi fissi e di collegamenti meno affidabili, e quindi di una banda relativamente limitata e fortemente variabile. Dato che il controllo del traffico non è centralizzato, deve essere gestito autonomamente da ciascun agente, onde evitare possibili congestioni della rete e conseguente perdita di dati e necessità di ritrasmissione.
E' utile definire ad alto livello dei criteri per ottimizzare il routing, pensando ad uno scambio di messaggi fra agenti. In questo modo, si può astrarre completamente dalle tecnologie utilizzate e delineare dei comportamenti standard per ciascuna entità connessa ad una MANET. Ciascun agente si occuperà di gestire l'inoltro dei messaggi prendendo atto dei percorsi meno costosi basandosi sulle proprie credenze, indotte da quelle dei nodi vicini. Dovrà ragionare in modo tale da reagire il più velocemente possibile ai cambiamenti di costo dei vari percorsi al fine di scegliere sempre quello meno costoso, cercando allo stesso tempo di preservare le proprie risorse (un dispositivo senza energia equivale ad un agente morto).

5. Il Progetto

5.1 Tecnologie Utilizzate: Jason

Jason è una piattaforma per lo sviluppo di sistemi multi-agente. Basato su Java, è un interprete per una forma estesa del linguaggio di programmazione per agenti, AgentSpeak(L). Jason ne implementa le semantiche operazionali ed aggiunge un insieme di meccaniche per migliorare le abilità degli agenti, propronendo una programmazione più intuitiva seguendo il modello BDI. AgentSpeak(L) si basa sul paradigma di programmazione logica per architetture ad agenti BDI, fornendo un framework elegante per programmare agenti intelligenti. Un programma sarà dunque scritto utilizzando fatti e regole, usando la logica per rappresentare la semantica del linguaggio e il metodo di risoluzione per raggiungere i goal.
Un agente Agentspeak è creato specificando una *belief base* (un insieme di conoscenze/credenze di base), ed un insieme di *plan* (piani). I plan vengono attuati per raggiungere dei goal (obiettivi), che si dividono in *achievement goals* e *test goals*, che rispettivamente indicano che l'agente vuole ragggiungere uno stato del mondo in cui il predicato associato sia vero, oppure controllare se il predicato associato unifichi con quello delle conoscenze dell'agente.
Per avviare un plan, l'agente reagisce ad un certo *triggering event,* che può essere interno (un *subgoal* dev'essere raggiunto) oppure esterno (un aggiornamento di un belief o di un goal dovuto ad una percezione dell'ambiente). Un plan si riferisce ad un'azione che l'agente è in grado di compiere sull'ambiente in cui vive. Il corso di azioni che un agente si impegna ad eseguire per gestire un determinato evento viene detto *Intention,* che altro non è che una serie ordinata di *subplan.*
Il ciclo con cui un agente Jason ragiona (seguendo il modello BDI) è il seguente:

1. Percepisce l'ambiente circostante
2. Aggiorna la propria *belief base*
3. Riceve eventuali comunicazioni da parte di altri agenti
4. Sceglie quei messaggi "socialmente accettabili"
5. Sceglie un evento da gestire
6. Raccoglie tutti i *plan* rilevanti
7. Decide quali tra questi sono applicabili
8. Sceglie un *plan* da applicare
9. Sceglie una *Intention* da eseguire in seguito
10. Esegue uno step di tale *Intention*.

Jason fornisce dunque gli strumenti necessari per poter definire un agente basandosi sul modello BDI ed un linguaggio di programmazione logica, e mette a disposizione ulteriori librerie per definire il tipo di ambiente in cui gli agenti andranno ad operare: ad architettura centralizzata, oppure ad architettura distribuita utilizzando i middleware SACI o JADE.
Come si evince dalla Fig. 3, il sistema da realizzare è definito su tre strati:

1. Sistema Multi-Agente: è il livello più astratto. La struttura, il comportamento, e la comunicazione di livello più alto fra gli agenti vengono definite in Jason.
2. Strato di Servizio Middleware: fornisce le librerie necessarie per collegare il sistema multi-agente allo strato di comunicazione vero e proprio, dove sono in esecuzione i protocolli di comunicazione di livello più basso, affinchè i dispositivi su cui girano gli agenti possano fisicamente comunicare.
3. Strato di Rete: permette la comunicazione di basso livello tra i dispositivi fisici della rete.

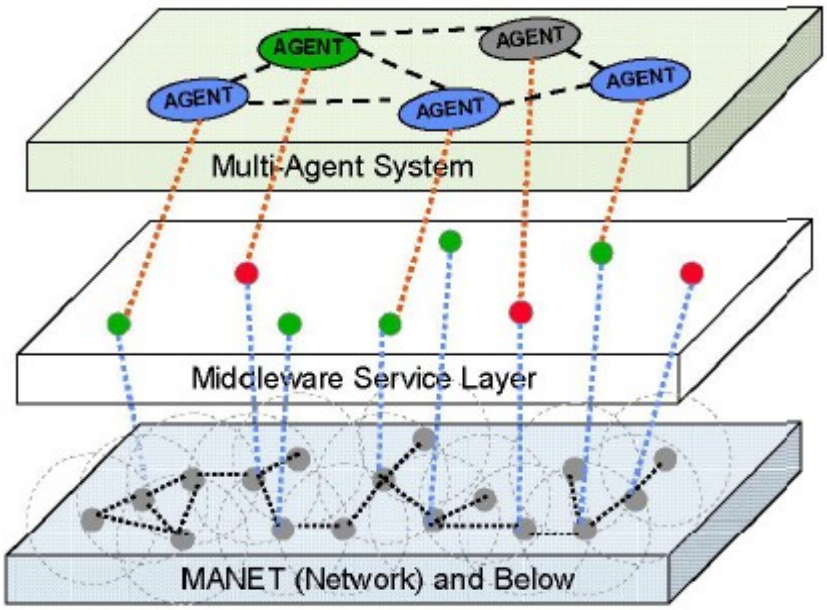

Fig. 5 Gli strati di una MANET basata su MAS.

5.2 Il Modello Proposto

Nel modello che si propone, si vuole presentare un'idea di MANET dove ogni nodo di tale rete sia un agente intelligente.
Gli agenti sono definiti in Jason, utilizzando quindi la semantica dei simboli e delle azioni Jason, ed il linguaggio Java quando necessario per definirne delle azioni non presenti già nelle librerie Jason o per estenderne delle esistenti. Viene definito un unico modello di agente Jason per garantire che tutte le entità coinvolte siano fra loro eterogenee.
Viene usata un'architettura centralizzata, dove l'ambiente viene simulato estendendo le librerie fornite da Jason. Il middleware e lo strato di rete sono dunque semplicemente simulati in Java, al fine di concentrare l'attenzione sul sistema multi-agente e semplificare l'interazione fra gli agenti Jason.
Il sistema presenta un numero N di agenti collocati nell'ambiente. Una volta inizializzati, gli agenti effettuano le seguenti azioni:

1. Simulano l'accensione della propria scheda di rete wireless: inviano in broadcast all'ambiente il proprio segnale radio
2. Ricevono dall'ambiente il segnale di eventuali nodi vicini: a seconda dell'intensità del segnale, l'agente vi assegna un costo ed inserisce tali nodi nella propria Neighbor Table, registrando l'istante di ricezione
3. Aggiungono alla propria Routing Table una entry per ciascun vicino trovato
4. Inviano un messaggio con la propria Routing Table a tutti i vicini

La tabella di routing di ciascun nodo contiene, per ogni route, il *next hop*, il nodo di destinazione, e il costo della route passando dal next hop indicato. Una volta ricevuto un messaggio di routing, l'agente si comporta nel seguente modo:

1. Estrae la tabella dal messaggio e ne confronta ciascuna entry con quella corrispondente nella propria tabella di routing
2. Aggiunge la entry nella propria tabella, annotando l'istante di inserimento, se
   a) la route non è ancora presente
   b) la route verso il nodo destinazione è presente, ma tramite un diverso next hop
   c) la route è presente, ma quella corrente ha costo maggiore rispetto alla nuova
   d) la route è presente, quella corrente ha costo minore o pari alla nuova, ma è passato un tempo T > X, dove T è la differenza fra l'istante corrente ed il vecchio istante di inserimento della entry corrente. T è un valore arbitrario e determinante nella reattività della rete ai cambiamenti topologici.

Dato che il goal primario di ogni agente è quello di rimanere connesso alla rete mobile, esso ripeterà in continuazione il ciclo iniziale (salvo attivare la propria scheda che è già attiva). La rete quindi è in grado di configurarsi in autonomia, senza bisogno di un sistema di coordinazione centralizzato, nè di una supervisione esterna. Gli agenti partecipanti si applicano proattivamente a cercare i propri vicini ed inviare in broadcast la propria tabella di route periodicamente. Così facendo, le route migliori vengono sempre rilevate, e quelle vecchie vengono cancellate ed aggiornate a quelle più recenti.
Tramite le percezioni ambientali, un agente può monitorare lo stato dei propri vicini valutandone i costi dei collegamenti diretti a seconda dell'intensità del segnale. E' bene evidenziare il fatto che un agente non conosce la posizione esatta dei vicini in relazione al sistema di coordinate dell'ambiente, ma si rende solo conto di avere eventualmente dei vicini ad una certa distanza a seconda dell'intensità con cui riceve il loro segnale. Inoltre, un agente non conosce l'intera topologia della rete: conosce tutti i nodi che ne fanno parte, ma sa come raggiungere i nodi più lontani solo tramite il next hop. Un agente non è tenuto a conoscere ad esempio i vicini di un nodo che non può raggiungere direttamente, nè a conoscere gli hop della route (eccetto il next hop) per inviare o inoltrare dei dati a destinazione.
Ciascun agente calcola i costi dei collegamenti coi nodi vicini tenendo conto dei seguenti fattori:

- l'intensità del segnale *I* (più alta è, minore è il costo);
- il livello di batteria rimasto *BV* (maggiore la carica rimasta, minore il costo. Un dispositivo collegato ad una fonte di alimentazione avrebbe un ulteriore bonus);
- il livello di affidabilità RV (un nodo con mobilità elevata, oppure con visibilità variabile nella rete risulta meno affidabile di un nodo più stazionario, quindi i collegamenti verso di esso risultano più costosi);
- il livello di carico LV (i collegamenti con un nodo rallentato da una congestione del traffico della scheda di rete sono più costosi).

Sulla base di tali caratteristiche, si propone una formula per modellare il costo *C* assegnato da un nodo *i* ad un qualsiasi collegamento tra sè stesso ed un nodo vicino *j*:

$$C_{ij} = \frac{I_{ij}}{(BV_i * BC_i) + (RV_{ij} * RC_i) + (LV_i * LC_i)}$$

$$\text{con } BC_i + RC_i + LC_i = 1$$

I_{ij} è l'intensità del segnale del nodo *j* ricevuto da *i*. BV_i , RV_{ij} , LV_i sono espressi come valori percentuale, e BC_i , RC_i , LC_i sono i relativi coefficienti di importanza (normalizzati). Si è deciso di uniformare il valore di uno stesso coefficiente per ciascun agente per mantenere il modello d'agente del sistema il più omogeneo possibile, ma la possibilità di attribuirvi numeri differenti è

sicuramente un approfondimento da considerare. Si consideri poi il fatto che il valore dei coefficienti, e dunque l'importanza di ciascun fattore, è situazionale. Ad esempio, in una rete dove la maggior parte dei dispositivi è penalizzata da un'autonomia limitata, il coefficiente relativo al livello di batteria dev'essere più rilevante rispetto ad uno scenario con nodi più longevi.
Bilanciando correttamente i coefficienti in gioco si ottiene una stima pesata, e di conseguenza più realistica, dei costi dei collegamenti della rete. E' preferibile tuttavia non assegnarvi dei valori direttamente nel modello, in quanto essi sono strettamente correlati alle tecnologie in gioco, e la buona ingegneria del software favorisce la maggior indipendenza possibile fra modello e tecnologia.

5.3 Implementazione degli Agenti

E' stato creato un unico modello per definire una qualsiasi delle entità del sistema, in modo da garantire l'omogeneità degli agenti, dove ogni nodo della rete corrisponde ad un agente intelligente. Nel costruire l'agente Jason, si ricordano le proprietà fondamentali di un nodo di una MANET:

- Il raggio broadcast dei dispositivi è limitato
- La comunicazione dipende dal routing dei nodi intermedi
- Ciascun nodo deve poter fungere da router

L'agente Jason viene definito con gli strumenti necessari per partecipare proattivamente nella comunicazione con i propri peer:

- Percezioni e broadcast periodici
- Protocollo *table-driven*
- Tabelle di routing sempre aggiornate

L'agente Jason ha le seguenti proprietà, definite nel *set of initial beliefs:*

Initial Beliefs	
Nome	**Descrizione**
battery(X).	Indica il livello corrente di batteria del dispositivo, in percentuale.
battery_coefficient(C).	Coefficiente di importanza del livello di batteria. Valore compreso fra 0 e 1.
load(X).	Indica la percentuale del carico sulla scheda di rete. Minore il valore di X, maggiore il traffico da gestire.
load_coefficient(C).	Coefficiente di carico. Valore compreso fra 0 e 1.
reliability(X,Y).	Indica l'affidabilità X, in percentuale, del collegamento con il nodo vicino Y.
reliability_coefficient(C).	Coefficiente di affidabilità. Indica l'importanza per il nodo corrente che i collegamenti verso i propri vicini siano affidabili.

wcard(X).	Indica se la scheda di rete è attiva o spenta (X ha valore binario). Condizione necessaria affinchè l'agente possa connettersi è che la scheda wireless sia accesa.
signal_strength(S,N).	Indica l'intensità N in scala numerica N del segnale e la corrispondente stringa S in linguaggio naturale (es. "very low", "low", "average", "high").

I goal iniziali di ogni agente sono, in ordine, quello di attivare la propria scheda di rete e di connettersi ad una rete o ad un nodo vicino. Il simbolo '!' denota un *achievement goal,* mentre un '?' indica un *test goal* :

Initial Goals	
Nome	**Descrizione**
!start.	Genera l'evento di "start". Eseguito una sola volta all'inizializzazione dell'agente.
!connect.	Genera l'evento di "connect". L'agente deve costantemente mantenere la connessione alla rete, ed è quindi ciecamente dedicato a perseguire questo goal in maniera ciclica.

Ciascun agente ha poi il proprio *set of plans*. Quando succede un certo evento, definito *triggering event,* l'agente deve considerare che cosa fare per gestirlo. Controlla quindi se ha un piano appropriato che soddisfi un certo *context,* ovvero un insieme di condizioni: se le condizioni sono soddisfatte, l'agente attiva il piano relativo. Si ricorda che ci sono due tipi di *triggering event* in Jason: quelli dovuti all'aggiunta o alla rimozione di un *belief* (preceduti rispettivamente dai simboli '+' o '-'). Nella tabella seguente vengono nominati e descritti solamente i piani più rilevanti per la logica del modello ad agenti proposto. Per semplicità, sono state escluse le variabili di ciascun piano, il contesto viene descritto in maniera non formale, e le *intention* contengono solamente i *subplan* più rilevanti.

Relevant Plans			
Triggering Event	**Context**	**Intention**	**Descrizione**
+!start		+wcard.	Attiva la scheda wireless.
+!connect	Scheda wireless attiva.	!find_neighbors; !broadcast_table; !check_buffer; !connect.	Inizializza e mantiene la connessione dell'agente alla rete.
+!find_neighbors		scan; !update_neighbors_table.	Percepisce dall'ambiente la presenza di agenti vicini, e controlla lo stato della Neighbor Table.

+! broadcast_table		send_agent.	Invia la propria Routing Table a tutti gli agenti vicini.
+!update_neighbors_table	Esiste una entry relativa ad un vicino *i* nella Neighbor Table dal quale inserimento è passato almeno un tempo *T*.	-neighbor; -route; !broadcast_bye.	Rimuove il vicino *i* dalla Neighbor Table e tutte le entry nella Routing Table che hanno l'agente *i* come destinazione o come next hop, .
+!broadcast_bye		send_agent.	Invia un messaggio a tutti i nodi vicini informando la mancata presenza del nodo *i* dalla rete.
+!add_neighbor_route.	Non esiste già una route per il vicino.	+route.	Aggiunge la entry relativa al vicino alla Routing Table.
+!send_data_packet		!find_min_route; send_agent.	Trova nella Routing Table il percorso a costo minimo verso il nodo *i* tramite il next hop *j*, ed invia quindi al vicino *j* il messaggio.
+!find_min_route			Trova la route minima, tramite il next hop *j*, per arrivare al nodo *i*.
+!evaluate_cost		?signal_strength; ?load; ?load_coefficient; ?battery_coefficient; ?battery; ?reliability_coefficient; ?reliability; .evaluate.	Pesca dalla *belief base* tutte le informazioni necessarie a calcolare il costo del collegamento in esame, ed esegue il calcolo.
+!send_table.		send_agent.	Invia la propria Routing Table a tutti i nodi vicini.
+!compare_route.	Non esiste ancora una route verso il nodo destinazione.	+route.	Confronta la propria Routing Table con quella ricevuta, ed inserisce la entry.

+!compare_route.	Non esiste ancora una route verso il nodo destinazione tramite il next hop specificato .	+route.	Confronta la propria Routing Table con quella ricevuta, ed inserisce la entry.
+!compare_route.	Esiste già la route, ma la nuova route è più economica.	+route.	Confronta la propria Routing Table con quella ricevuta, ed inserisce la entry.
+!compare_route.	Esiste già la route, ma quella corrente non è stata aggiornata per un certo periodo di tempo *T.*	+route.	Confronta la propria Routing Table con quella ricevuta, ed inserisce la entry.
+!check_buffer.			Controlla se nella belief base ci sono ancora pacchetti da inoltrare per i quali non è stato ricevuto ancora nessun acknowledgement.

Jason mette a disposizione una serie di librerie dove vengono definite le azioni interne (*Internal Actions,* nella sintassi precedute da '.') che un agente può compiere. Tali azioni riguardano sia logiche interne al comportamento di un agente (ad es. l'annotazione dell'istante temporale corrente, oppure la cancellazione di una parte della propria *belief base,* ecc.), sia operazioni di comunicazione fra pià agenti (ad es. invio di messaggi). Per un approfondimento, si consiglia la visione del manuale di Jason.
L'unica *internal action* che è stata aggiunta dal modello proposto riguarda il calcolo del costo di un collegamento.

Internal Actions	
Nome	**Descrizione**
.evaluate	Utilizzando i parametri della formula proposta nel modello per il calcolo del costo di un collegamento fra due agenti, ne restituisce il risultato.

Ad ogni ciclo di mantenimento della connessione, ovvero l'esecuzione del piano *connect,* viene sottratta una quantità X di carica della batteria per simulare il calo periodico di energia, e a ciascuna operazione di invio di messaggi *send_agent* corrisponde un ulteriore decremento dovuto all'aumento di potenza necessario per effettuare la trasmissione. L'agente interessato aggiornerà il proprio *belief* relativo. Più un nodo trasmette quindi, maggiore diverrà il costo dei collegamenti verso esso allo scopo di disincentivare il routing tramite tale nodo e preservare le risorse a disposizione.

Ogniqualvolta l'agente riceve una percezione dall'ambiente riguardante un vicino (un evento *signal*, spiegato nel punto seguente), ne aggiunge la relativa entry alla propria Neighbor Table, indicandone l'identificativo e l'intensità del segnale relativo, ed annotando l'istante di ricezione e la variazione dell'intensità del segnale nel tempo. Il controllo di tale variazione serve ad aggiornare il livello di affidabilità relativo all'agente vicino, aggiornando il *belief* corrispondente *reliability*. Maggiore la variabilità del segnale, maggiore l'incertezza di disponibilità della largehzza di banda del collegamento, che risulta dunque meno affidabile per trasmettere dati.
Infine, dev'essere gestito anche il problema della congestione sui collegamenti di rete. Un agente che non riesce a smaltire il traffico che passa tramite esso deve aumentare il costo dei propri collegamenti in modo tale da poter svuotare almeno parzialmente il buffer della propria scheda di rete. E' importante quindi che un agente intervenga prima che si verifichi un problema di congestione e la conseguente perdita di pacchetti che dovranno essere poi ritrasmessi. Il *belief* relativo allora, *load,* verrà aggiornato decrementandone il valore all'aumentare del traffico da smaltire, e viceversa. Tale aggiornamento avviene, nel modello qui discusso, ad ogni ricezione di un pacchetto di dati (l'evento *data_packet,* discusso nel punto seguente), e successivamente quando il pacchetto viene memorizzato (significa che l'agente ne era il destinatario) oppure inoltrato.

5.4 Simulazione dell'Ambiente

L'ambiente nel quale interagiscono gli agenti è una simulazione realizzata estendendo le classi Java già fornite da Jason. E' stato utilizzato il pattern architetturale *Model-View-Controller,* in modo da separare i compiti fra i componenti software che interpretano i tre ruoli principali:

- il *Model* fornisce i metodi per accedere ai dati utili all'applicazione;
- il *View* visualizza i dati contenuti nel model;
- il Controller riceve i comandi dell'utente (in genere attraverso il View) e li attua modificando lo stato degli altri due componenti

Questo schema, fra l'altro, implica anche la tradizionale separazione fra la logica applicativa (in questo contesto spesso chiamata "logica di business"), a carico del *controller* e del *model*, e l'interfaccia utente a carico del *view.*
La simulazione prevede l'utilizzo di un ambiente in due dimensioni, ad architettura centralizzata, totalmente definito in Java. Ciascun agente avrà dunque una propria posizione in riferimento al sistema di coordinate usato per l'ambiente, con il vincolo che due o più agenti non possono occupare la stessa posizione.
Sono state modellate alcune *environmental action* per permettere all'agente di interagire con l'ambiente.

Environmental Actions	
Nome	**Descrizione**
scan.	L'agente riceve una percezione dall'ambiente per ogni vicino in quell'istante, e la relativa intensità.
move.	Simula lo spostamento nell'ambiente di un agente.
on.	Simula la ricomparsa dell'agente nell'ambiente.

off.	Simula la scomparsa dalla visibilità di rete di un agente (ad es. lo spegnimento della scheda di rete del dispositivo associato).
send_agent(D,T,M).	Viene usata dall'agente per inviare un messaggio M di tipo T ad un agente vicino M (rimpiazzando l'azione standard fornita da Jason). Visto che i messaggi vengono inviati da un nodo all'altro tramite l'ambiente, è proprio quest'ultimo che deve propagare il messaggio al destinatario, simulando possibili ritardi o perdite del collegamento.

L'agente riceve dall'ambiente dei *percept,* eventi esterni che esso può scegliere se e come gestire. Si è scelto di implementare le due percezioni più rilevanti: quella di ricezione del segnale di un agente vicino, e di ricezione di un messaggio.

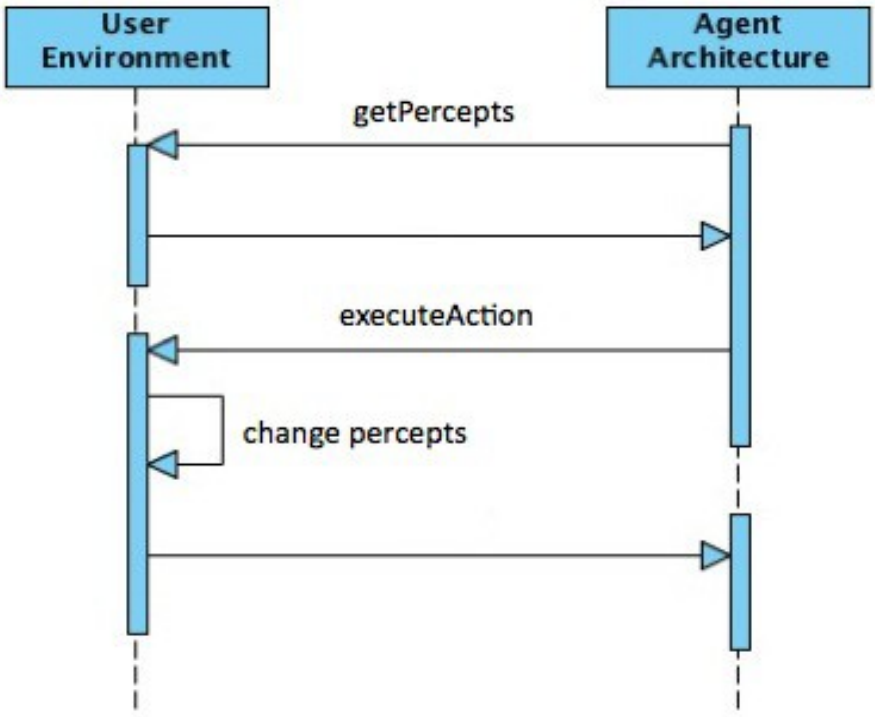

Fig. 6 L'interazione fra un agente Jason e l'ambiente circostante.

Percepts	
+signal	L'agente riceve la percezione del segnale diffuso da un vicino, la relativa intensità, e ne aggiunge la entry nella propria Neighbor Table.
+message	Ricezione di un pacchetto dati, ovvero un messaggio. Se l'agente è il destinatario, ne elabora il contenuto ed invia un messaggio di acknowledgement al mittente, altrimenti provvede ad inoltrarlo verso la destinazione.

Le azioni che vengono eseguite durante l'esecuzione del sistema multi-agente vengono stampate sulla *MAS Console* di Jason, rendendo più comprensibile l'evoluzione dello stesso sistema e più semplice il debug in caso di incongruenze.

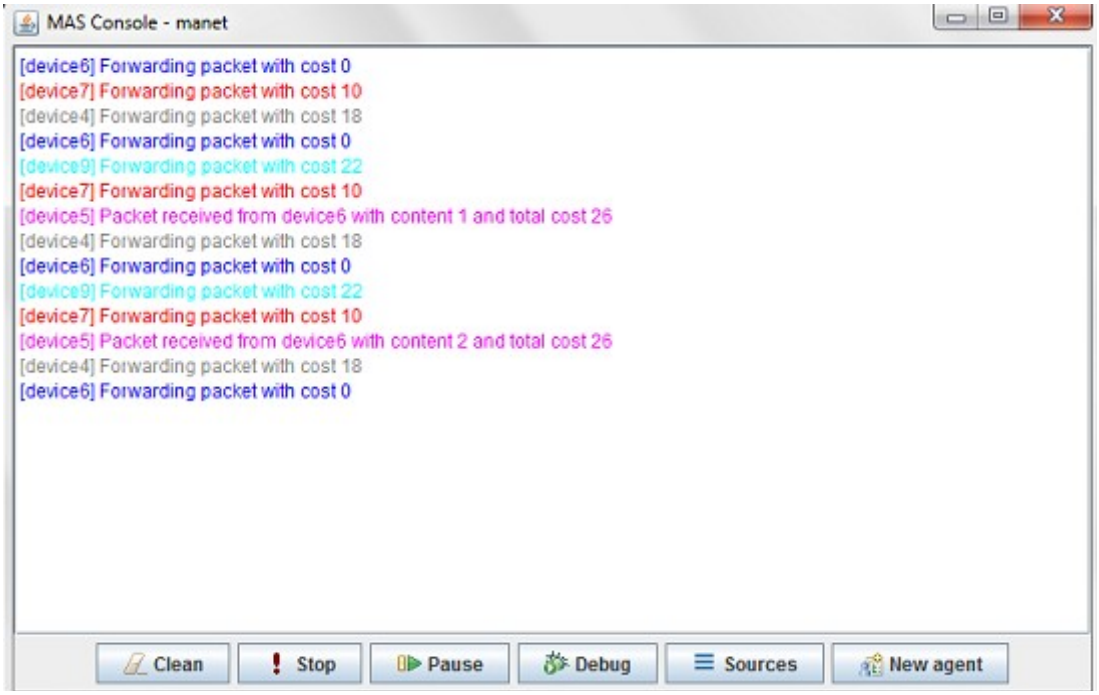

Fig. 7: La MAS console di Jason

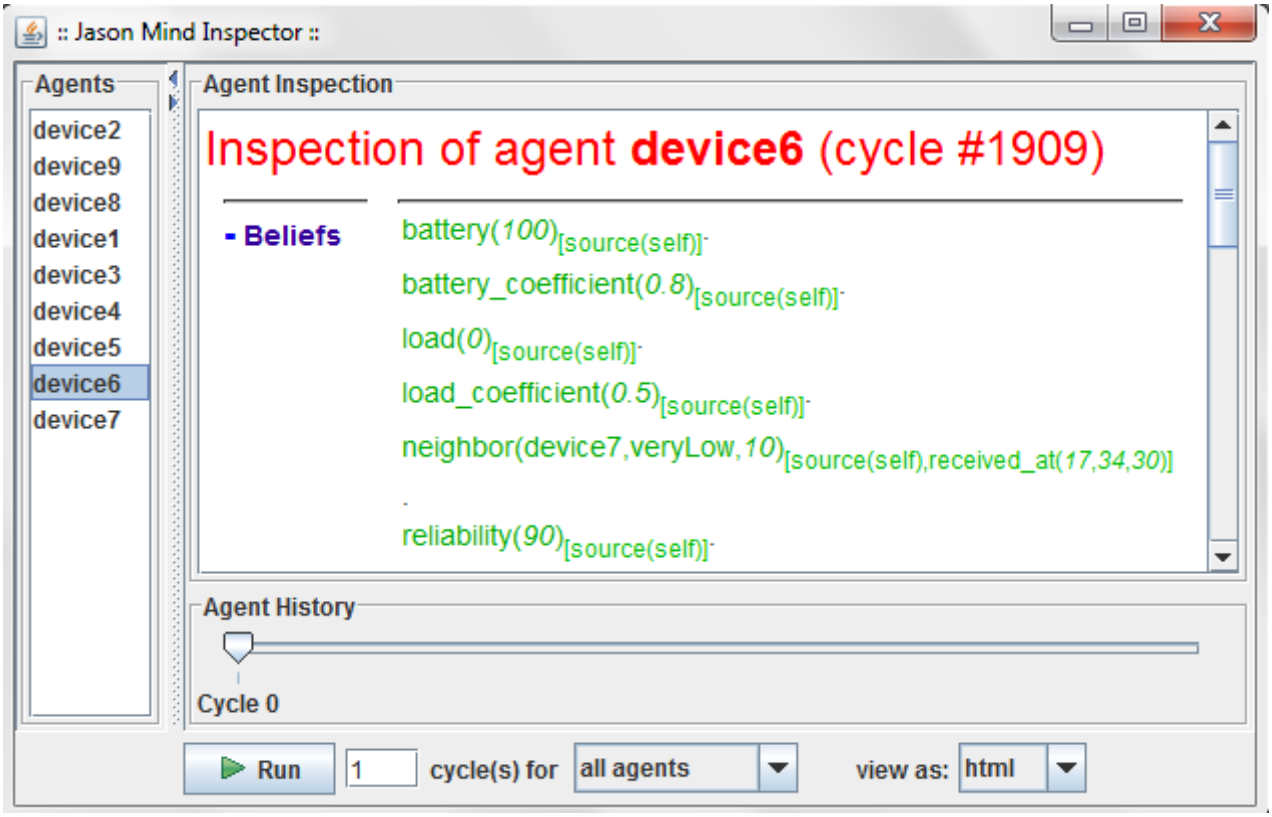

Fig. 8: Il mind inspector fornito da Jason

Il framework Jason è dotato oltretutto di un comodo strumento, chiamato *Mind Inspector,* specializzato nel fornire al programmatore una visione dello stato mentale corrente di ciascun agente attivo nel sistema, in un qualsiasi istante dell'esecuzione. Per ogni entità, è dunque possibile visualizzarne la belief base e l'intera *stack of subplans* (la pila dei sotto-piani), cioè l'intention che l'agente sta eseguendo durante un determinato *reasoning cycle* (ciclo di ragionamento). Per facilitare il debug dell'applicazione, può essere eseguito un reasoning cycle alla volta, ed è possibile tornare anche a degli step precedenti scorrendo la *Agent History.* Dal mind inspector infine è possibile scegliere se fermare l'esecuzione di uno o più agenti, quando ripristinarla, o se "ucciderli" definitivamente (per l'esecuzione corrente).

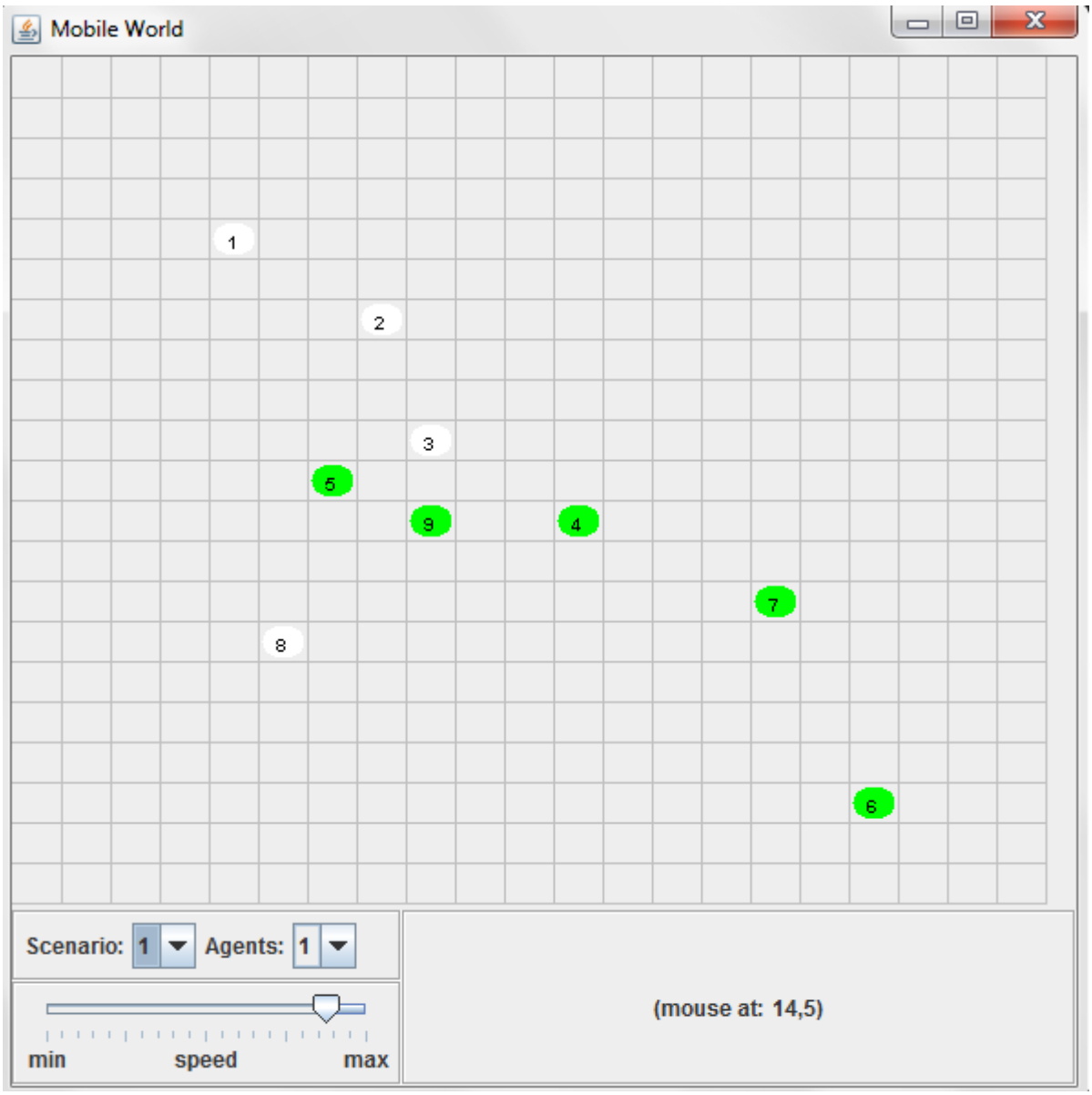

Fig 9 Il routing dei pacchetti tra l'agente Device6 e l'agente Device5.

La simulazione mostra semplicemente il routing di dati (messaggi) tra una coppia di agenti distanti ciascuno diversi hop dall'altro. All'avvio dell'esecuzione, ciascun agente si attiva e provvede in autonomia a trovare i propri vicini e a creare la propria Routing Table. Il percorso per inoltrare i messaggi è quello a costo minimo conosciuto da ciascun agente, e varia nel tempo a seconda dei cambiamenti di posizione degli agenti, e dei loro livelli di batteria, affidabilità, e traffico.

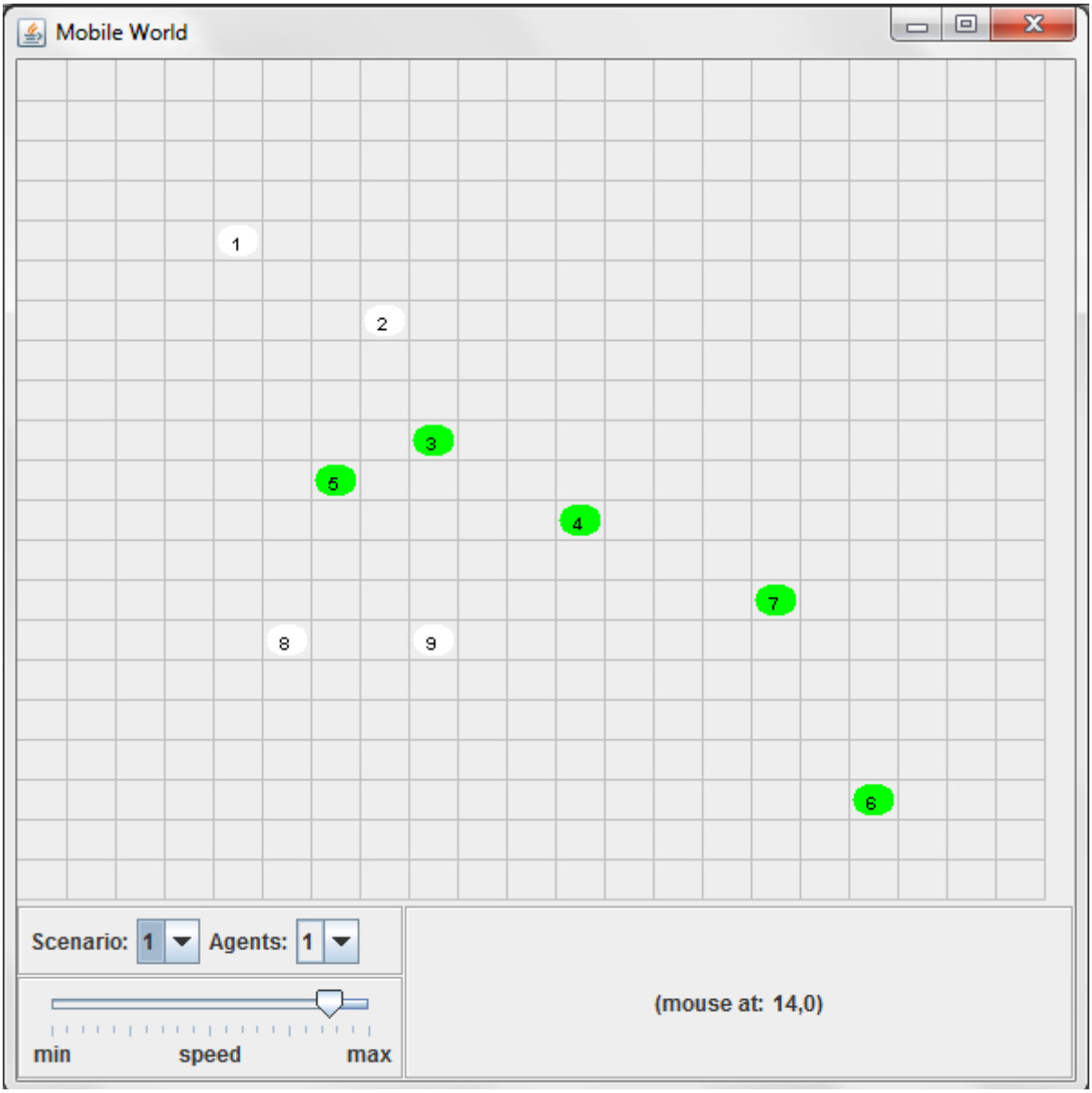

Fig. 10: Con lo spostamento dell'agente Device9, il routing dei messaggi passa attraverso il Device3, seguendo il percorso meno costoso.

6. Conclusioni

6.1 Considerazioni

Il modello discusso propone il paradigma ad agenti come approccio per gestire il problema di inizializzare e mantenere una rete mobile ad-hoc in modo efficiente. L'utilizzo del framework Jason, basandosi sul linguaggio di programmazione Agentspeak, permette di definire una MANET come un sistema multi-agente specificando un unico modello di agente eterogeneo dove ciascun nodo della rete è rappresentato ad alto livello come un'entità intelligente. La simulazione è realizzata in un ambiente centralizzato, definendo quelle che sono le forme d'interazione esterne che ciascun agente può effettuare. Si è scelto Jason come framework in quanto consente di definire e rappresentare il sistema multi-agente e le singole entità in modo semplice ed elegante.
Gli obiettivi di una MANET devono essere la velocità di comunicazione e la preservazione delle risorse a disposizione, fattori che sono fra loro strettamente correlati. Ottimizzare la comunicazione significa garantire reattività a cambiamenti frequenti e non predicibili, e quindi un maggiore impiego di risorse (più elaborazioni per la CPU e scheda di rete, più overhead per la rete e più banda utilizzata). E' importante l'utilizzo di un buon modello e di un alto livello di astrazione per comprendere i comportamenti delle entità coinvolte e le loro caratteristiche, al fine di approfondire quei fattori che determinano l'ottimizzazione dell'intero sistema. Un protocollo di routing efficiente può aumentare le perfomance della comunicazione e consentire un dispendio energetico più economico e più omogeneo, a patto però che i costi della rete siano valutati in maniera idonea.
Col modello ricavato dallo studio, si è proposta dunque un'analisi focalizzata sull'importanza di una mentalità proattiva da parte degli agenti della rete, atta a prevenire certi comportamenti del sistema e a reagire tramite decisioni ponderate. E per tali decisioni si incentra l'importanza sui costi dei collegamenti, la cui corretta assegnazione, tenendo conto dei parametri più rilevanti, serve proprio a rendere le scelte degli agenti veramente significative. Un'ottima qualità del segnale (molto difficile da pretendere costantemente in una rete mobile) non è sufficiente a giustificare un basso costo di un collegamento fra due nodi: tale collegamento potrebbe infatti divenire presto inaccessibile a causa della scomparsa di uno dei due nodi dovuta all'esaurimento della batteria del proprio dispositivo, riducendo le prestazioni dell'intera rete. Un discorso analogo lo si può fare anche per una congestione di un nodo, che porterebbe ad un rallentamento dell'inoltro dei messaggi, e quindi dell'intero sistema. Infine, la tramissione dei dati su un collegamento ad intensità di segnale non costante non garantisce una minore perdita di informazioni ed un flusso più omogeneo tra due nodi, dato che la larghezza di banda a disposizione è molto variabile e di conseguenza meno predicibile.
La visione astratta con cui si è approcciato lo studio delle MANET, cioè usando un approccio a sistema multi-agente, e la conseguente implementazione in Jason, ha facilitato la comprensione dei fattori chiave coinvolti nella prestazione di queste reti, e ne ha analizzato soprattutto l'aspetto collegato ai costi dei collegamenti tra i nodi, rappresentati come agenti intelligenti, e quello di una forma d'interazione e comunicazione di alto livello.

6.2 Sviluppi Futuri

L'implementazione del modello proposto può essere arricchita utilizzando protocolli di routing multi-agente efficienti, scegliendo tra quelli già proposti negli ultimi anni da diversi team di ricercatori nel campo delle MANET. Si potrebbero considerare ulteriori meccaniche, come ad esempio l'esclusione volontaria di un nodo dalla rete, oppure la divisione in sottoreti. Infine, si potrebbero effettuare delle simulazioni per validare il modello proposto, cercando di valutare un peso più accurato dei fattori di costo, utilizzando uno o più algoritmi di routing ad agenti.

E' logico pensare, vista la natura distribuita del sistema modellato, anche all'implementazione in un ambiente non centralizzato, mantenendo la definizione dell'agente in Jason, ed utilizzando un middleware come JADE (Java Agent DEvelopment framework), oppure degli spazi di tuple quali TuCSoN (Tuple Centres Spread over the Network) o TOTA (Tuples on The Air).

7. Bibliografia

[1] J.P. Macker, W. Chao, R. Mittu, M. Abramson: "Multi-Agent Systems in Mobile Ad hoc Networks". Military Communications Conference, October 2005. MILCOLM 2005. IEEE.

[2] J.P. Macker, W. Chao, J.W. Dean: "A Study of Multiagent System Operation Within Dynamic Ad hoc Networks". Military Communications Conference, November 2008. MILCOLM 2008. IEEE.

[3] J. Hoebeke, I. Moerman, B. Dhoedt, P. Demeester: "An Overview of Mobile Ad hoc Networks: Applications and Challenges". Journal of the Communications Network, Vol.3, July 2004.

[4] C. S. Raghavendra, J. Stepanek: "Power-Aware Broadcasting in Mobile Ad Hoc Networks". The Aerospace Corporation, May 1999.

[5] Y. Qiu, P. Marbach: "Bandwidth Allocation in Ad Hoc Networks: A Price-Based Approach". Department of Computer Science, University of Toronto, 2002.

[6] M. Abdullah, H. Bakhsh: "Agent-based Dynamic Routing System for MANETs". Collage of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia, September 2008.

[7] R. Bindhu: "Mobile Agent Based Routing Protocol with Security for MANET". International Journal of Applied Engineering Research, Dindingul, Volume 1, November 2010.

[8] J.P. Macker, W. Chao, M. Abramson, I. Downard: "Cooperative Multi-Agent Systems in Mobile Ad hoc Networks". Military Communications Conference, October 2006. MILCOLM 2006. IEEE.

[9] R.H. Bordini, J.F. Hubner: "Programming Multi-Agent Systems in Agentspeak using Jason". Wiley, 2007.