

Self-Organisation and Adaptation within MAS: the crowd evacuation example in Jade and TuCSoN

Alessandro BATTELLI
Michele MATTEINI
Andrea PAGLIARANI
Karin PASINI

Capitolo 1

Introduzione

Il problema della Crowd evacuation consiste nel guidare una folla lungo un percorso sicuro in caso di pericolo. In particolare, l'obiettivo di questo lavoro è quello di valutare l'adeguatezza delle tecnologie Jade e TuCSoN nella realizzazione di un sistema in grado di allontanare un gruppo di persone da un incendio scoppiato all'interno di un edificio.

1.1 Requisiti

Dato un edificio di forma qualsiasi, costituito da una configurazione variabile di stanze, allo scoppiare di un incendio, un sistema di sensori e display dovrà indicare la direzione che la folla dovrà seguire per raggiungere il punto più sicuro all'interno dell'edificio.

1.2 Analisi dei requisiti

Fondamentale è definire in modo non ambiguo il concetto di punto più sicuro all'interno dell'edificio. Per più sicuro si intende un'area che sia al tempo stesso ragionevolmente lontana in linea d'aria dall'incendio e più distante da esso dal punto di vista topologico, ovvero più distante sulla base della lunghezza del percorso seguito a partire dal punto in cui è scoppiato l'incendio.

1.3 Analisi del problema

Per guidare la folla nel punto più sicuro, il problema ci induce a utilizzare una diffusione delle informazioni relative all'incendio tramite gradienti: il gradiente assumerà valore 0 in corrispondenza dell'incendio e dovrà crescere man mano che ci si allontanerà da esso. Tali informazioni saranno diffuse in tutto il sistema grazie alla rete di sensori presente nell'edificio.

Realisticamente, i sensori sono dotati di un raggio d'azione che permette loro di interagire solamente con quelli limitrofi; pertanto, i sensori non sono onniscenti, ma hanno accesso solo ad una parte della conoscenza. Così facendo il sistema guadagna un certo grado di fault tolerance, perchè anche in caso di guasto di uno o più sensori, il sistema continuerà a svolgere il suo compito compatibilmente con i raggi d'azione dei sensori ancora funzionanti.

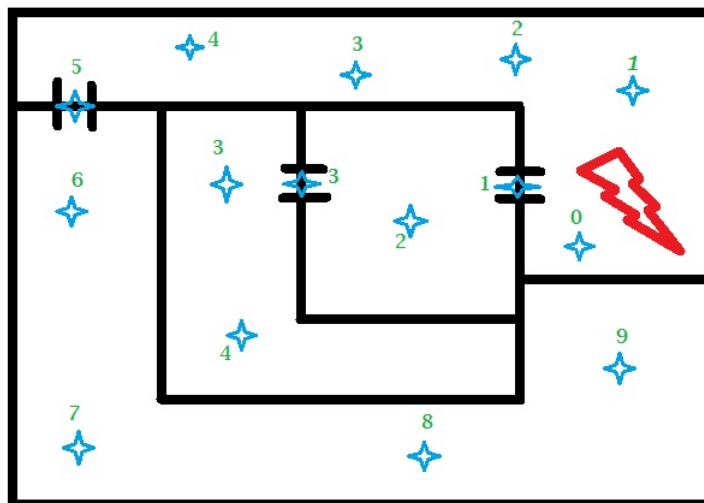
Le informazioni diffuse dai vari sensori verranno utilizzate dai display, il cui compito è quello di consigliare alla folla la direzione da seguire al fine di allontanarsi dall'incendio. Realisticamente, anche i display dovranno essere dotati di un raggio d'azione con l'obiettivo di accedere ai dati relativi solo alla determinata porzione di edificio in cui essi si trovano in un ben preciso istante.

Si noti che l'assenza del raggio d'azione (di cui in precedenza abbiamo già sottolineato l'importanza) renderebbe parecchio difficile la diffusione del gradiente, che invece così appare del tutto naturale: infatti, il gradiente verrà incrementato di un'unità ad ogni raggio d'azione.

Capitolo 2

Progetto

Si ipotizza che i sensori non possano comunicare attraverso le pareti, e che quindi il gradiente possa propagarsi da una stanza all'altra solo attraverso le porte. Perchè questo sia possibile, assumiamo che sulle porte siano situati dei sensori speciali, che fungono da ponte tra una stanza e l'altra dell'edificio. In questo modo il gradiente potrà essere utilizzato per guidare la folla verso la porta (e quindi verso la stanza) più lontana.

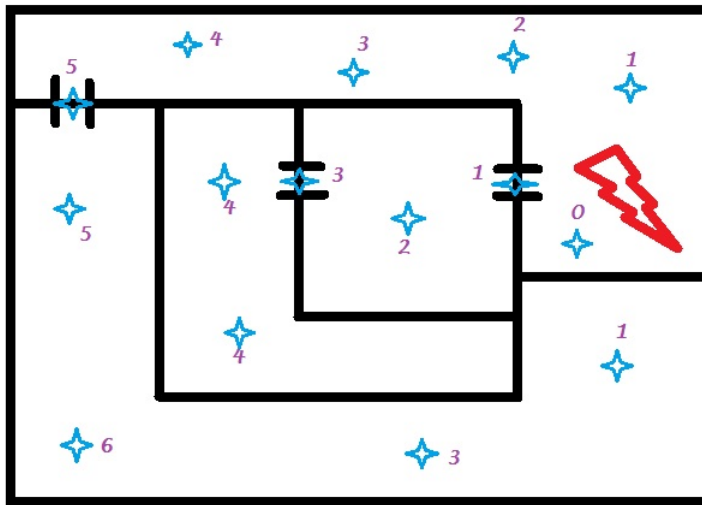


 = SENSORE

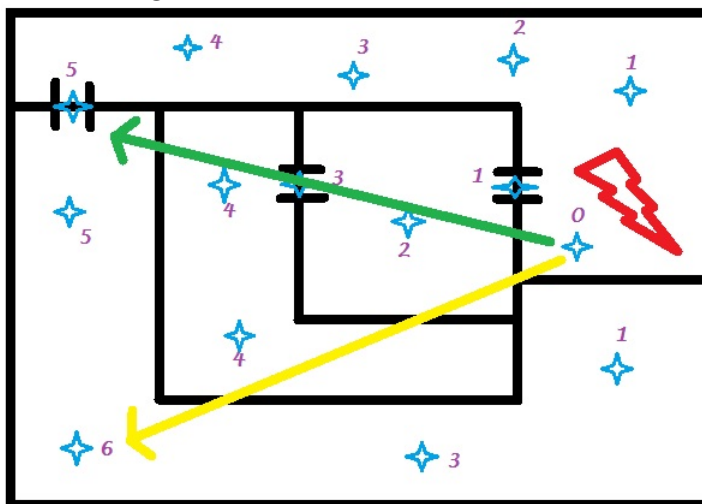
 = FUOCO

 = PORTA

Tuttavia, come si può notare dal disegno soprastante, il gradiente condurrebbe la folla verso il valore 9, che in linea d'aria è molto vicino all'incendio. Per ovviare a questo problema è necessario introdurre un secondo gradiente: questo avrà il compito di guidare la folla nel punto più lontano in linea d'aria dall'incendio. Perchè ciò sia possibile, i sensori dovranno diffondere un valore di distanza in linea d'aria, indipendentemente dalla conformazione delle varie stanze (quindi per il calcolo di questo gradiente i sensori comunicheranno anche attraverso le pareti).



Unendo le informazioni provenienti dai due gradienti, tramite un opportuno algoritmo, la folla potrà essere guidata verso l'area più sicura. In caso contrario, come si vede dalla figura successiva, basandosi unicamente su uno dei due gradienti, otterremmo direzioni consigliate errate.



Come possiamo vedere nell'immagine, la freccia verde rappresenta la direzione nel

caso si considerassero solo i sensori della stanza mentre la freccia gialla rappresenta la direzione nel caso si considerassero i sensori di tutto l'edificio. In entrambi i casi la direzione è chiaramente errata, in quanto non rispetta la topologia dell'edificio e la posizione delle porte.

L'algoritmo dei gradienti poteva essere calcolato anche dai display; tuttavia in questo modo il sistema sarebbe basato sui display mentre sarebbe opportuno che questi fossero solo un modo per rappresentare le informazioni messe a disposizione dal sistema. In questo modo i display sarebbero svincolati dal sistema ed in futuro potrebbe anche essere associato un display ad ogni persona nell'edificio, ad esempio con un'applicazione per smartphone. Se invece i display fossero stati fissi, non ci sarebbero stati problemi nel fare eseguire loro l'algoritmo, ma nel modo da noi scelto possiamo considerare un caso più generale di utilizzo del display.

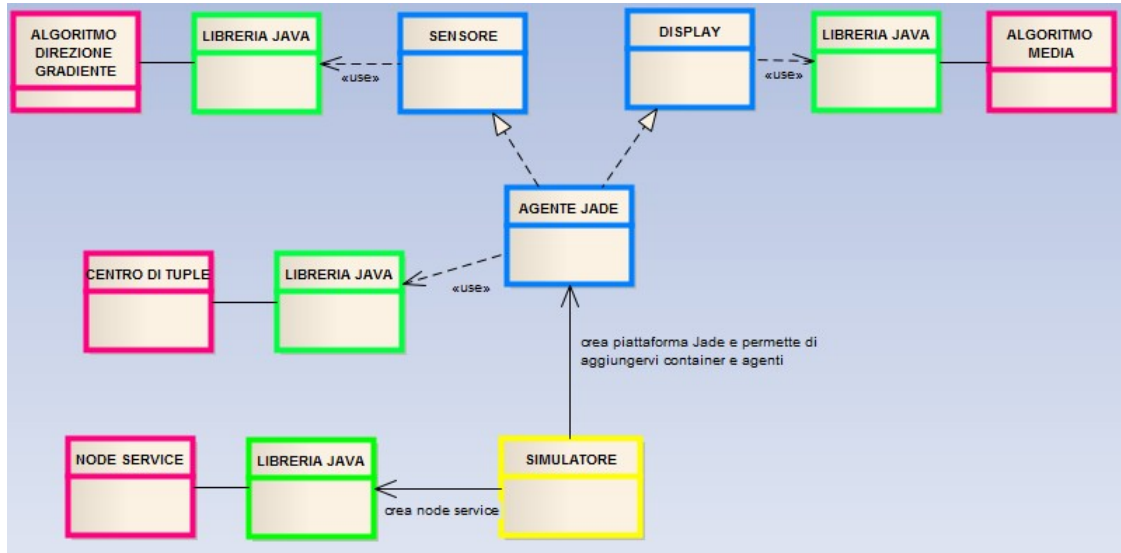
Dopo aver contribuito alla diffusione dei gradienti, i sensori dovranno (tramite l'algoritmo che elabora la direzione da seguire in base ai 2 valori di gradiente) indicare la direzione verso cui muoversi. Vi sarà poi un certo numero di display che dovranno mediare le direzioni indicate dai vari sensori al fine di mostrare concretamente alla folla la reale direzione da seguire.

Sulla base delle considerazioni appena introdotte, possiamo definire i requisiti minimi di funzionamento del sistema:

- Presenza di un sensore su ogni porta (considerando che ogni porta dovrà avere un raggio d'azione sufficiente a includere quelle a lui adiacenti).

Si noti che non è altrettanto necessario avere un display in ogni stanza, in quanto in generale i display potrebbero essere entità in possesso delle persone all'interno dell'edificio e, come tali, potrebbero anche essere assenti. Invece nel caso in cui vi sia una persona con un proprio display in una stanza, essa dovrà progressivamente spostarsi in modo da allontanarsi dall'incendio.

2.1 Architettura del sistema



Come si può vedere dall'architettura sovrastante, abbiamo deciso di modellare l'edificio come un'unica piattaforma JADE, in cui ogni stanza sarà costituita da un container. All'interno di ogni stanza-container saranno presenti i vari sensori e display, modellati come agenti JADE. In questo modo, possiamo sapere in ogni momento quali agenti abbiamo in ogni stanza; inoltre, JADE permette di modellare molto bene il concetto di mobilità, fondamentale considerando che i vari display potrebbero spostarsi fisicamente da una stanza all'altra per allontanarsi dall'incendio e quindi migrare logicamente da un container all'altro. Inoltre, abbiamo ritenuto opportuno che la comunicazione tra i vari agenti fosse indiretta, mediata da un'astrazione di coordinazione: perciò, tutte le informazioni necessarie al corretto funzionamento del sistema verranno salvate all'interno di centri di tuple. Le informazioni sarebbero anche potute transitare da un agente all'altro; tuttavia (almeno in prima approssimazione) non si è scelta tale soluzione per evitare di complicare la struttura degli agenti che sarebbero dovuti essere entità intelligenti, in grado di gestire almeno parzialmente le politiche del sistema (che invece sarebbe meglio demandare al medium di coordinazione). Vedremo poi delle ulteriori motivazioni che ci hanno spinto a considerare migliore la soluzione che fa uso del medium di coordinazione.

Per semplicità, considereremo ora di avere un unico centro di tuple, in cui sono salvate le informazioni relative ai sensori dell'intero edificio.

Osservando l'immagine attentamente, si può notare che gli agenti Jade dovranno interagire con il centro di tuple ed è opportuno che non accedano direttamente ad esso ma mediante un livello di libreria. Questo perchè gli agenti hanno determinati comportamenti, indipendenti dalla sequenza di primitive Linda (di più basso livello) che, in seguito ad essi, saranno invocate sul tuple centre.

Infine, va tenuto presente che i sensori, grazie ad un algoritmo, riusciranno a produrre le informazioni a cui accederanno poi i display per consigliare, grazie ad un ulteriore algoritmo, la direzione che le persone dovranno seguire al fine di allontanarsi dall'incendio.

2.2 Algoritmi

2.2.1 Algoritmo per il calcolo di una direzione sicura

Dati i valori dei gradienti associati ad ogni sensore, occorre approssimare la direzione nella quale le persone devono muoversi per allontanarsi dell'incendio. Come spiegato sopra, sono disponibili due gradienti : il primo, che chiameremo **Air gradient** indica la distanza in linea d'aria di un sensore dall'incendio; il secondo, **Route gradient** considera la topologia dell'edificio, tracciando quindi dei percorsi validi per allontanarsi da esso. Un istanza dell'algoritmo ha come input:

- **sensorGradient** che rappresenta i valori dei gradienti nel punto in cui si vuole calcolare la direzione.
- **nearGradients** la lista dei gradienti dei sensori vicini (nel raggio d'azione del primo)

L'algoritmo funziona iterando su tutti i *route gradient* dei sensori vicini, cercando quelli che hanno un valore superiore a quello corrente: muoversi verso questi sensori significa infatti prendere un percorso che permette di allontanarsi dall'incendio. Per ognuno di questi percorsi *convenienti* viene calcolata la distanza in linea d'aria di allontanamento utilizzando l'*Air gradient* che viene quindi utilizzata per eseguire una media pesata delle direzioni valide in cui ci si può muovere, ricavate dalla posizione dei sensori vicini. I vantaggi di questo algoritmo $\tilde{A}$ che fornisce una buona approssimazione della direzione corretta in cui muoversi, utilizzando dati ricavati da una semplice propagazione di gradienti. Il problema maggiore di questo algoritmo $\tilde{A}$ dato dal fatto che si rischia di far convergere le persone verso un *vicolo ceco* che non sempre rappresenta la soluzione migliore per il fatto che non si conoscono le **potenzialità** dei percorsi, ossia la distanza massima che un percorso consente di raggiungere, per la quale sarebbe necessario un ulteriore gradiente o propagazione.

L'algoritmo $\tilde{A}$ invece robusto alla mancanza di dati e tenta comunque di calcolare una direzione nel caso in cui:

- **Mancanza del route gradient nella posizione corrente** Viene approssimato con il minimo valore del gradiente nel suo raggio d'azione tra quelli disponibili, in questo modo tutti i percorsi tranne quello che sicuramente si avvicina all'incendio, vengono marcati come convenienti.

- **Mancanza dell'air gradient nella posizione corrente** Anche questa volta si approssima con l'air minimo disponibile, questo rende comunque maggiormente prioritari i percorsi che si allontanano di più dall'incendio.
- **Mancanza del route g. in un sensore vicino** Viene considerato alla pari di quello di un percorso che ritorna verso l'incendio (quindi non conveniente).
- **Mancanza dell'air g. in un sensore vicino** Viene considerato uguale a quello in cui ci si trova, rendendo equivalenti le scelte di muoversi o meno verso l'incendio.
- **Nessun dato disponibile** L'algoritmo genera un valore di direzione speciale che indica di rimanere fermi.

2.2.2 Algoritmo per l'approssimazione della direzione del display

Il singolo display riceve più direzioni dai vari sensori nel suo raggio, deve quindi effettuare una di **media** per approssimare la giusta direzione nella posizione in cui si trova. L'unica eccezione a questo comportamento é quando un sensore segnala al display una posizione sicura (la direzione che indica si rimanere fermi), in questo caso l'algoritmo ignora le direzioni e indica di muoversi verso quel punto: nel caso reale questo punto rappresenterebbe un *punto sicuro di raccolta*.

2.3 La coordinazione del sistema in TuCSoN

Come già accennato la coordinazione del sistema è delegata ad un tuple centre, con l'obiettivo di sgravare gli agenti Jade da questo compito, non di loro competenza e al tempo stesso di incapsulare i dati di interesse.

Sarebbe possibile adottare diverse soluzioni a questo scopo:

- Utilizzare il medium di coordinazione solamente come un repository in cui salvare le informazioni utili all'intero sistema.
- Sfruttarlo anche per governare direttamente il sistema, limitando l'insieme delle operazioni ammissibili.

Infatti la prima alternativa non permette all'astrazione di coordinazione di introdurre politiche all'interno del sistema. Pertanto, in tal caso tale incombenza dovrebbe essere gestita dagli agenti TuCSoN che sono quelle entità che hanno il compito di interagire con il tuple centre attraverso primitive Linda. Mostriamo ora un esempio che chiarirà la natura della suddetta interazione.

Teniamo presente che i sensori devono registrarsi al tuple centre per poter far parte del sistema, inviando ad esso le informazioni relative al loro codice identificativo, alla loro posizione assoluta all'interno dell'edificio, alla stanza di cui fanno parte e alla portata del loro raggio d'azione. Considerando realisticamente che non possono esserci più sensori nella stessa posizione o con lo stesso identificativo, gli agenti TuCSoN per registrarsi dovrebbero agire in questo modo:

- Controllare che non esistano nel tuple centre sensori aventi la medesima posizione.
rdp(sensor(uuid(A), position(x(valX), y(valY)), room(H), range(S)))
- Se esistono, non possono registrarsi. In caso contrario devono controllare che non esistano nel tuple centre sensori aventi lo stesso identificativo.
rdp(sensor(uuid(valUuid), position(x(X), y(Y)), room(H), range(S)))
- Se esistono, devono eliminare la tupla esistente.
in(sensor(uuid(valUuid), position(x(X), y(Y)), room(H), range(S)))
- A questo punto possono registrarsi.
out(sensor(uuid(valUuid), position(x(valX), y(valY)), room(valRoom), range(valRange)))

Tuttavia agli agenti TuCSoN interessa solamente poter registrare i sensori Jade che hanno fatto loro la richiesta (tramite le funzioni di libreria precedentemente citate), quindi questa prima soluzione, oltre a non permettere al centro di tuple di governare il sistema, non è neppure completamente corretta.

Pertanto alla fine abbiamo adottato la seconda alternativa, possibile grazie a reazioni Respect, che permettono di introdurre i comportamenti coordinativi desiderati. Con reazioni del tipo:

```
reaction(
  out(sensor(uuid(Z), position(x(X), y(Y)), room(K), range(R))),
  invocation,
  (
    rd(sensor(uuid(A), position(x(X), y(Y)), room(H), range(S))),
    out(eliminaSensor)
  )
).
```

```
reaction(
  out(sensor(uuid(Z), position(x(X), y(Y)), room(K), range(R))),
  (completion, success),
  (
    in(eliminaSensor),
```

```

in(sensor(uuid(Z),position(x(X),y(Y)),room(K),range(R)))
)
).

```

```

reaction(
out(sensor(uuid(Z),position(x(X),y(Y)),room(K),range(R))),
invocation,
in(sensor(uuid(Z),position(x(F),y(G)),room(H),range(S)))
).

```

l'agente TuCSoN potrà limitarsi ad eseguire la registrazione, con una semplice:

```

out(sensor(uuid(valUuid),position(x(valX),y(valY)),room(valRoom),range(valRange)))

```

2.3.1 Librerie per l'interazione col centro di tuple

Come già anticipato gli agenti Jade interagiranno con il tuple centre tramite un livello di librerie, al fine di disaccoppiare logicamente l'evoluzione dei loro comportamenti da quella degli agenti TuCSoN; quindi ogni agente Jade (sensore, porta o display) comunicherà con un proprio agente TuCSoN. In particolare i sensori e le porte (che ricordiamo essere esse stesse dei sensori speciali) disporranno di funzioni in grado di:

- Registrarsi al tuple centre
- Verificare la presenza di un incendio
- Segnalare la presenza di un incendio
- Diffondere l'AirGradient (ovvero il gradiente che esprime la lontananza dall'incendio in linea d'aria)
- Diffondere il RouteGradient (ovvero il gradiente che esprime la distanza dall'incendio in termini di percorso)
- Impostare l'informazione relativa alla direzione
- Recuperare l'elenco dei gradienti relativi ai sensori (o porte) limitrofi
- Segnalare lo spegnimento dell'incendio

I display invece avranno un'unica funzione a loro disposizione, con l'obiettivo di:

- Recuperare l'elenco delle informazioni relative alle direzioni impostate dai sensori (o porte) limitrofi

2.4 Il comportamento dei coordinabili in Jade

All'interno del nostro sistema la piattaforma Jade risulta particolarmente utile per vari motivi:

- Modellare la topologia delle stanze dell'edificio
- Definire il comportamento delle entità coordinabili
- Avere un modo naturale per gestire la mobilità dei display all'interno dell'edificio

Ogni stanza può essere modellata tramite un container Jade, che includerà al suo interno i vari agenti che in un dato istante di tempo si trovano in quella determinata stanza. In questo contesto si potrebbe decidere a livello teorico di avere container locali alla piattaforma Jade oppure remoti.

Considerando il caso di container locali si può subito notare la presenza di numerosi aspetti che ne sconsigliano l'uso:

1. Tutte le stanze-container dell'edificio dovrebbero essere poste su un unico nodo, con ovvi problemi di fault tolerance.
2. La mobilità dei display da una stanza-container ad un'altra sarebbe fittizia in quanto, se dal punto di vista logico migrano da un container ad un altro, dal punto di vista fisico afferiscono sempre allo stesso nodo.

L'insieme di questi aspetti rende di difficile applicabilità nel mondo reale questa soluzione. Molto più naturale risulta invece distribuire i container, che è ciò che abbiamo effettivamente implementato.

Di base esisteranno la piattaforma Jade con il relativo main container, che include già l'Agent Management System e il Directory Facilitator; a questi dovranno essere aggiunti altri due agenti: uno che attende le richieste di creazione di nuovi container, l'altro che crea un agente (sensore, porta o display) su un container remoto precedentemente creato.

Una volta creati i container e registrati i vari agenti (sensore, porta o display) al loro interno, questi ultimi eseguiranno comportamenti prefissati.

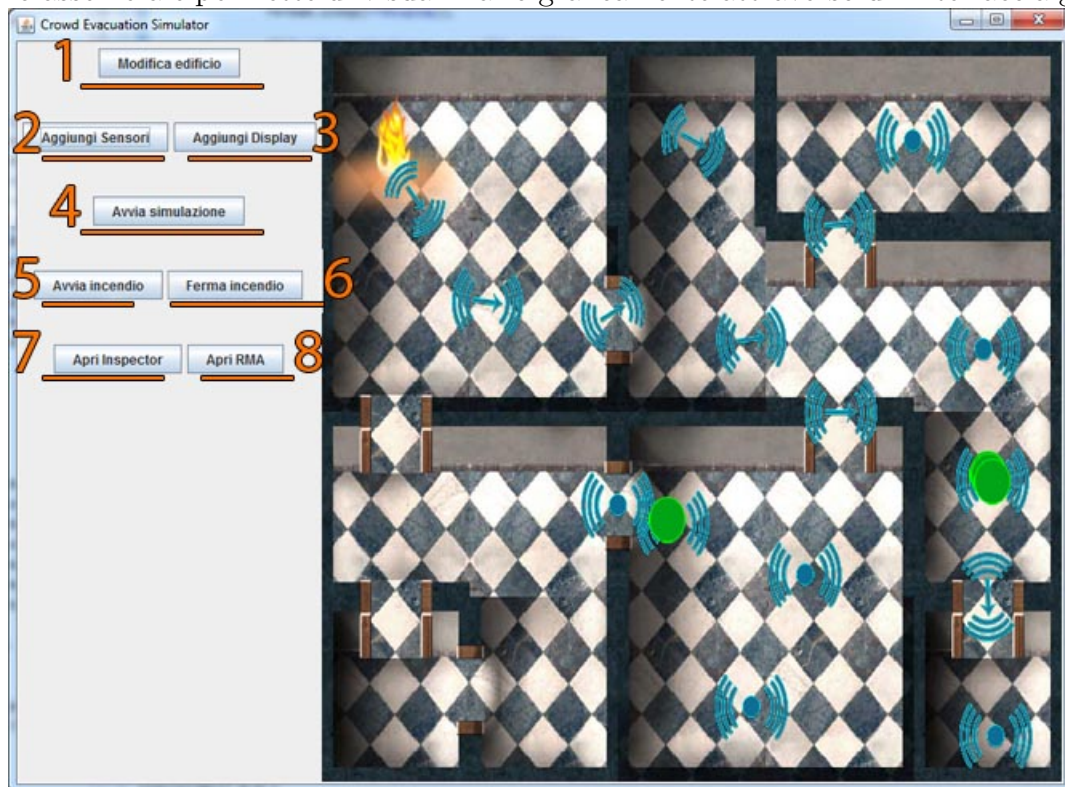
I sensori e le porte, dopo aver effettuato la registrazione al tuple centre, verificano periodicamente la presenza di un incendio e, se esso è presente, diffondono nel sistema i gradienti (AirGradient e RouteGradient) e al tempo stesso recuperano i gradienti dei sensori limitrofi per poi calcolare, tramite l'opportuno algoritmo, la direzione consigliata e aggiungerla al tuple centre.

I display periodicamente recuperano le informazioni relative all'elenco delle direzioni impostate dai sensori a loro limitrofi, per poi, tramite l'opportuno algoritmo, decidere la direzione effettivamente da seguire al fine di allontanarsi dall'incendio. Nel momento

in cui essi capiscono in che direzione andare, hanno bisogno di comunicarlo al simulatore (che ovviamente potrà monitorarne lo spostamento); perchè ciò sia possibile lato simulatore, ovvero nel main container, dovrà essere presente anche un'opportuno agente che resti in attesa dei valori provenienti dai vari sensori. Da parte sua, il simulatore oltre ad aggiornare graficamente la posizione dei vari display all'interno dell'edificio sulla base dell'elenco delle direzioni recuperate dal suddetto agente, dovrà anche provvedere a comunicare ai display stessi le loro nuove coordinate, eventualmente informandoli nel caso di cambio di stanza. I display, ricevute le nuove informazioni, aggiornano la loro posizione e, se è cambiata anche la loro stanza, dovranno anche provvedere a migrare nel container relativo alla stanza appena raggiunta.

2.5 Simulatore

Il simulatore é un applicazione java costruita sopra il sistema distribuito descritto, che lo assembla e permette di visualizzarlo graficamente attraverso un'interfaccia grafica. :



Questo permette all'utente di personalizzare la simulazione e controllarla, attraverso i comandi disponibili da interfaccia:

1. bfModifica edificio Permette di disegnare una piantina personalizzata su cui effettuare la simulazione.

2. bfAggiungi sensori Aggiunge alla piantina i sensori che segnaleranno la direzione da seguire.
3. bfAggiungi display Aggiunge alla piantina i display, nella loro posizione iniziale.
4. bfAvvia simulazione Avvia la simulazione, avviando la piattaforma jade e creando agenti e centri di tuple necessari.
5. bfAvvia incendio Permette di posizionare un incendio, in una determinata posizione dell'edificio.
6. bfFerma incendio
7. bfApri Inspector
8. bfApri RMA

Capitolo 3

Aspetti avanzati

Terminato il progetto con un unico nodo TuCSoN possiamo considerare come il nostro sistema possa essere esteso introducendo più nodi con relativi tuple centre. Questa soluzione rappresenta meglio il caso reale, in cui occorre avere una maggiore fault tolerance e in cui la diffusione delle informazioni tra le varie parti del sistema è non solo logica ma anche fisica.

Abbiamo ragionevolmente assunto che i container Jade possano essere collocati solamente in corrispondenza dei punti remoti in cui sono attivi nodi TuCSoN. A parte l'aggiunta di più nodi TuCSoN, l'architettura rimane sostanzialmente invariata, con un livello di coordinazione del sistema affidato ai tuple centre, un livello di entità coordinabili che, mediante opportuni algoritmi e propri comportamenti, entrano in contatto con esso, e un livello applicativo che permette la simulazione dei precedenti.

Analizziamo ora brevemente alcuni cambiamenti riguardanti le singole parti del sistema.

3.1 Algoritmi

Gli algoritmi per il calcolo della direzione approssimata a partire dei gradienti limitrofi e per il calcolo della direzione consigliata a partire da quelle approssimate limitrofe restano invariati.

3.2 La coordinazione del sistema in TuCSoN mediante librerie

Le reazioni Respect che permettono di governare il sistema dovranno riflettere la natura dello stesso, consentendo la coordinazione tra i vari nodi distribuiti. Di conse-

guenza cambiano anche lievemente le librerie utili per interagire con i tuple centre. In particolare, i sensori disporranno di funzioni in grado di:

- Registrarsi al tuple centre
- Verificare la presenza di un incendio
- Segnalare la presenza di un incendio
- Diffondere nelle stanze adiacenti non ancora notificate l'informazione relativa alla presenza di un incendio
- Diffondere l'AirGradient (sia all'interno della propria stanza che in quelle adiacenti)
- Diffondere il RouteGradient (solo all'interno della propria stanza)
- Impostare l'informazione relativa alla direzione
- Recuperare l'elenco dei gradienti relativi ai sensori (o porte) limitrofi
- Segnalare lo spegnimento dell'incendio
- Diffondere nelle stanze adiacenti non ancora notificate l'informazione relative allo spegnimento dell'incendio

Le porte avranno a disposizione delle operazioni simili a quelle dei sensori, ma bisogna tener presente che, essendo il punto di collegamento tra due stanze adiacenti, dovranno consentire il passaggio delle informazioni da una all'altra. In particolare, una porta potrà:

- Registrarsi ai tuple centre di entrambe le stanze che essa collega
- Verificare la presenza di un incendio (in una stanza per volta)
- Segnalare la presenza di un incendio (in una stanza per volta)
- Diffondere nelle stanze adiacenti non ancora notificate l'informazione relativa alla presenza di un incendio
- Diffondere l'AirGradient (sia all'interno delle proprie stanze che in quelle adiacenti)
- Diffondere il RouteGradient (all'interno di entrambe le proprie stanze)
- Impostare l'informazione relativa alla direzione (in una stanza per volta)

- Recuperare l'elenco dei gradienti relativi ai sensori (o porte) limitrofi (in una stanza per volta)
- Segnalare lo spegnimento dell'incendio (in una stanza per volta)
- Diffondere nelle stanze adiacenti non ancora notificate l'informazione relative allo spegnimento dell'incendio (in una stanza per volta)

I display invece mantengono inalterate le loro funzionalità.

3.3 Il comportamento dei coordinabili in Jade

Anche in questo caso le modifiche sono molto limitate. Infatti gli agenti Jade di sensori e porte, a differenza della soluzione con un unico tuple centre, oltre a segnalare l'incendio e lo spegnimento dello stesso all'interno della propria stanza, dovranno anche provvedere a diffonderne le informazioni nelle stanze adiacenti. I display invece nel caso in cui debbano migrare in un'altra stanza, dovranno avvalersi di un nuovo agente TuCSoN necessario per l'interazione con il nuovo tuple centre.

3.4 Simulatore

Anche il simulatore risente di questi cambiamenti e deve quindi essere esteso per supportare nodi remoti, i cui indirizzi possono essere specificati da interfaccia:

