

UrbaNews

- Emanuele Buccelli
- Francesco Degli Angeli
- Marco Leali

<http://apice.unibo.it/xwiki/bin/view/Courses/SdLm1213Projects-UrbaNews>

Indice

- 1) Introduzione e scopi del progetto
- 2) Agenti
- 3) Simulazione
- 4) Conclusioni

1 – Introduzione e scopi del progetto

L'idea di UrbaNews nasce con l'obiettivo di mettere in connessione il maggior numero possibile di utenti creando un servizio location-based per la diffusione di Hovering Information¹: ovvero informazioni geolocalizzate che vengono preservate e gestite dai vari nodi senza la necessità di un server centrale.

Per poter prestare maggior attenzione all'aspetto distribuito del progetto abbiamo effettuato alcune assunzioni preventive che verranno elencate di seguito:

- Il dispositivo che si connette all'applicazione deve disporre di un sistema di localizzazione geografica (quali GPS o Glonass o triangolazione wireless o un sistema ibrido) e di una connessione di rete stabile e senza interruzioni.
- Il buffer di memoria locale in cui vengono immagazzinate le informazioni relative ai nodi, è presente ma limitato.
- I dispositivi che effettuano l'accesso all'applicazione non sono necessariamente device mobili, ma possono essere anche postazioni fisse (per esempio un utente che entra da un computer fisso per vedere in quel momento le news che ci sono nella sua zona di provenienza).

Il modello **TuCSon** (tuple centres spread over the network) è stato utilizzato come piattaforma su cui basare UrbaNews, sfruttando il tuple-centre come mezzo di coordinamento fra i nodi, identificati come "Tattler" (pettegolo), ed i vari agenti, che cooperano tra loro per raggiungere gli obiettivi preposti e per il corretto funzionamento del sistema.

2 – Agenti

Gli agenti di UrbaNews ereditano la classe *AgentBase*, che mette a disposizione alcuni metodi in più rispetto alla classe *TucsonAgent* (da cui *AgentBase* deriva): il metodo *wasteRandomTime* per sospendere l'esecuzione dell'agente per un certo periodo di tempo e il metodo *getNodePosition* che restituisce le coordinate della posizione del nodo. Di seguito sono descritte brevemente le funzionalità di ogni agente.

PositionAgent. Questo agente ha i seguenti compiti: calcolare in modo casuale e con un possibile cambio di direzione il prossimo spostamento del nodo; aggiornare la sua

¹ "Hovering Information: Infrastructure-Free Self-Organising Location-Aware Information Dissemination Service" Alfredo A. Villalba Castro, Giovanna Di Marzo Serugendo, and Dimitri Konstantas

posizione, modificando la *positionTuple*; infine inserire una *movementTuple* per notificare ai nodi collegati l'avvenuto spostamento.

LinkAgent. Alla creazione all'agente, gli viene passata una lista con gli altri nodi, detti "Vicini". La prima cosa che fa l'agente è comunicare ad essi la propria presenza attraverso il metodo *comunicaNuovoTattler*, in modo da essere aggiunto alla loro lista dei Vicini. Subito dopo con il metodo *initializeVicini* aggiunge al nodo una tupla per ogni vicino, con id del nodo e posizione. Ciclicamente poi comunica a tutti i nodi collegati i propri vettori di spostamento con il metodo *comunicaSpostamenti*, in modo tale da mantenerli aggiornati sulla propria posizione. Dopo la nascita del nodo è possibile che ne vengano creati altri, per cui viene controllato il *TupleCentre* per la presenza di nuove richieste di vicinanza col metodo *ricercaNuoviTattler*. Fatto questo si procede all'analisi di tutte le tuple di movimento ricevute dagli altri nodi, aggiornando quindi la posizione dei Vicini (*aggiornaPosizioni*). A questo punto si passa al controllo della distanza con i vari nodi con il metodo *checkVicini*. Quelli troppo lontani perdono il collegamento con il nodo, mentre quelli che si avvicinano lo acquisiscono. Questo è utile per indicare a quali nodi dovranno essere replicate le HI.

SpeakAgent. Il compito di questo agente è molto semplice. Ciclicamente inserire all'interno del nodo una Hovering Information, localizzata nella posizione in cui è stata creata. Sono caratterizzate dai seguenti campi:

- **frase:** contiene il testo della HI;
- **categoria:** categoria di appartenenza dell'HI;
- **pos:** posizione dell'informazione, ottenuta dalla posizione del nodo al momento della creazione;
- **range:** distanza che indica il raggio d'azione dell'informazione;
- **rep:** valore di replica, usato come indice di quanto un'informazione stata replicata tra i nodi. Un valore di replica maggiore indica un'HI più diffusa e quindi meno critica da mantenere;
- **id:** stringa alfanumerica generata casualmente al momento della creazione. Utile soprattutto per il riconoscimento di HI duplicate;

CleanAgent. Il compito di questo agente è quello di ripulire il nodo dalle HI superflue. Ad ogni iterazione esegue i seguenti controlli:

- controlla che non ci siano HI con lo stesso ID. Nel caso siano presenti, lascia solo quella con valore di replica minore;
- controlla la distanza dell'HI con la posizione attuale del nodo. Se il valore è superiore ad una certa soglia allora la elimina;
- infine, supponendo di avere un buffer limitato per le informazioni, elimina progressivamente le HI con i valori di replica quelle in più alti basandosi sul concetto della *Generation-Based Caching*.

ReplicaAgent. Lo scopo principale dell'agente è quello di mantenere vive le HI, replicandole ai nodi più vicini. Si basa sull' *Attractor Point Algorithm* dove le repliche tendono a stare più vicine all'anchor location dell'informazione. Infatti quando un nodo si allontana dal punto di localizzazione dell'HI, l'agente la replica ai 2 nodi più vicini alla sua ancora incrementando il valore di replica (**rep**).

3 - Simulazione

Per il progetto si è scelto di sviluppare una simulazione in locale dell'esecuzione. Avremo quindi una classe *Simulator*, il cui compito sarà quello di generare dei nodi con posizione iniziale e porta casuali. Successivamente sarà compito dei vari agenti simulare i comportamenti dei Tattler e curare la preservazione delle Hovering Information.

Per far partire l'esecuzione avviare *StartGUI* ed inserire il numero di nodi che si desidera nella simulazione. E' possibile visualizzare gli spostamenti dei nodi attraverso una GUI e i loro tuple-center attraverso gli Inspector.

4 – Conclusioni

UrbaNews ci ha permesso di affrontare la complessità nel realizzare un sistema distribuito multi-agente basato sulla cooperazione e sulla condivisione di risorse. In particolare è risultato articolato ideare un metodo per ottenere una lista sempre aggiornata dei nodi vicini ai vari Tattler, sempre garantendo il corretto funzionamento del progetto. Interessante è stato applicare il concetto di Hovering Information a questo tipo di sistema evitando di centralizzare questa diffusione di notizie geolocalizzate.