Mapping SAPERE on TuCSoN

Marco Piraccini

October 22, 2012

Abstract

Questa relazione vuole mettere in luce gli aspetti chiave del progetto SAPERE e le caratteristiche fondamentali della tecnologia TuCSoN, definendo successivamente un possibile mapping per realizzare le astrazioni di SAPERE attraverso gli strumenti messi a disposizione da TuCSoN soddisfacendo le proprietà emergenti dello scenario tecnologico pervasivo attuale.

1 Introduzione

Lo scenario tecnologico moderno sempre più orientato alla computazione pervasiva impone lo studio di opportuni modelli di coordinazione, che permettano di abilitare e regolare l'interazione tra servizi pervasivi, soddisfacendo proprietà chiave quali *situatedness*, *adaptivity*, *diversity*.

Il progetto SAPERE intende affrontare questo problema dalle fondamenta rivolgendo l'attenzione allo studio di sistemi naturali, in particolare bio-chimici, i quali sotto molti punti di vista possono essere assimilabili a sistemi pervasivi, condividendone le proprietà base e offrendo pattern comportamentali molto interessanti. Da questa analisi è possibile ricavare un opportuno insieme di leggi che hanno il ruolo di modellare lo spazio di coordinazione e quindi governare la dinamica del sistema nel suo complesso.

Ma in quale modo realizzare un modello di coordinazione che sfrutti questi pattern? Attraverso quali astrazioni?

Un possibile approccio potrebbe essere rivolgersi a tecnologie a spazi di tuple (ad esempio *Linda*), estendendo però lo spazio di coordinazione con politiche interne che regolino il comportamento del sistema, in altre parole permettere l'ingegnerizzazione, la programmazione dello spazio di tuple. Questo è il principio alla base della tecnologia TuCSoN(Tuple Centres Spread over the Network), nel quale lo spazio di tuple è regolato attraverso un insieme di leggi dette *reactions* (un centro di tuple TuCSoN quindi non è altro che uno spazio di tuple programmabile).

La tecnologia TuCSoN per le sue proprietà (descritte nel seguito) risulta essere sicuramente un valido strumento per reificare un'infrastruttura di coordinazione per le enitità di un sistema distribuito pervasivo. Tuttavia, per poter

modellare adeguatamente i pattern bio-chimici individuati, necessita di alcune estensioni fondamentali:

- estendere i centri di tuple con funzionalità di **match semantico**
- definire un insieme di *reactions* tali da realizzare una sorta di **simulatore chimico online**; cioè determinare come selezionare le reazioni da eseguire ad ogni step temporale e l'intervallo tra due step.

2 SAPERE

Il progetto SAPERE vuole, come detto, coordinare un insieme di servizi pervasivi distribuiti in modo da realizzare un sistema dinamico e flessibile. Lo scenario cui si riferisce è il seguente: un elevato numero di sensori, display personali o pubblici e device mobili innescano dinamicamente un insieme di attività che devono essere coordinate in modo da sfruttare al meglio le informazioni collezionate e offrire agli utenti servizi e notizie quanto più aggiornate e utili.

Le caratteristiche del sistema di pervasività e distribuzione impongono di considerare aspetti quali:

- **situatedness**: le informazioni da presentare all'utente devono essere selezionate sulla base del contesto corrente, considerando lo stato dell'ambiente e sociale circostante (es. preferenze dell'utente);
- **self-***: il sistema deve essere in grado di adattarsi automaticamente ai cambiamenti ambientali, deve offrire in sostanza caratteristiche di autogestione e auto-organizzazione;
- **diversity** : il sistema deve poter gestire in modo adeguato entità, servizi, informazioni di ogni tipo, permettendo di accogliere sempre nuovi servizi per migliorare le funzionalità del sistema e la sua qualità.

Una caratteristica molto interessante è che queste proprietà costituiscono anche le fondamenta della maggior parte degli ecosistemi naturali. Adottando un tipo di ricerca inter-disciplinare è quindi possibile studiare sistemi naturali, per derivare pattern e regole base da applicare poi ai nostri sistemi artificiali. In pratica quello che vogliamo realizzare è un insieme di entità situate che interagiscono in accordo ad un ben definito insieme di regole naturali, forzate dall'ambiente in cui sono situate.

Le regole naturali sono inglobate in *eco-laws* rappresentate come reazioni chimiche ognuna con un certo rate (probabilità che ha la reazione di scattare) mentre le varie entità del sistema(servizi, display, device) sono viste come specifiche tipologie di reagenti con associato un valore di concentrazione (che ne indica la relativa attività/rilevanza in un certo contesto). Lo stato del sistema evolve attraverso l'esecuzione delle reazioni nel tempo: ogni reazione consuma un certo insieme di *reagenti* per produrre un certo insieme di *prodotti* modificando lo stato del sistema.

Questo tipo di modello di coordinazione nature-inspired può essere conseguito in modo abbastanza semplice attraverso un modello a spazio di tuple "chimico", ovvero una normale infrastruttura a spazi di tuple con in più le funzionalità di match semantico e motore chimico.

I reagenti sono mappati in tuple logiche che contengono informazioni riguardo al tipo di reagente e la sua quantità presente in quello spazio, le leggi del sistema in forma di reazioni dovranno essere inserite come logica di funzionamento del sistema, programmando in qualche modo gli spazi di tuple. Ogni entità del sistema è inoltre caratterizzata da una certa rappresentazione semantica all'interno di un determinato spazio di tuple.

Il sistema dal punto di vista architetturale è composto da una serie di nodi distribuiti nella rete, ognuno contenente uno o più spazi di tuple. Gli spazi sono collegati attraverso la relazione di prossimità fisica/logica, in modo da poter trasferire tuple tra spazi vicini.

Le proprietà prima espresse sono garantite in modo pressoché automatico:

- **situatedness**: lo stato corrente locale è definito dalle tuple presenti nello spazio;
- **self-***: le proprietà di auto-organizzazione e adattività del sistema sono soddisfate selezionando un insieme opportuno di regole, ricavate da pattern naturali (es. *prey-predator system*) e messe in esecuzione dal simulatore chimico.
- **diversity**: la funzionalità di match semantico permette di definire regole generali da applicare in casi specifici e potenzialmente nuovi.

Risulta quindi evidente che gli elementi chiave per il corretto funzionamento del sistema sono:

Match semantico Definire una specifica ontologia per ogni spazio di tuple (ad esempio usando Description Logic), definire tuple semantiche e una opportuna funzione di matching che restituisca un valore compreso tra 0 e 1, detto grado di match;

Simulatore chimico online La selezione di una regola dipende dall'implementazione del cosiddetto motore chimico, cioè dall'insieme di regole che determinano quali reazioni eseguire e quando. In generale si considera un approccio non deterministico seguendo l'algoritmo di Gillespie, estensione delle catene di Markov tempo discrete, in cui una reazione ha una certa probabilità di essere selezionata (*reaction rate*) pari al suo rate intrinseco moltiplicato per la concentrazione dei suoi reagenti e per il valore ottenuto dalla funzione di matching tra template dei reagenti e reagenti presenti nello spazio delle tuple. Le tre operazioni fondamentali del motore sono: computare le *reaction rates*, selezionare in modo non deterministico le reazioni da eseguire e calcolare l'intervallo temporale tra uno step e il successivo.

Il caso di studio concreto preso in considerazione per l'implementazione del sistema è costituito da un'infrastruttura di display distribuiti in un terminale di un aeroporto; loro funzione è mostrare le informazioni di maggiore rilievo per gli utenti che si trovando di fronte ad essi, sulla base delle preferenze espresse (contenute nei device delle persone o nelle loro carte d'imbarco), del contesto ambientale e sociale corrente, delle priorità dei servizi, della localizzazione dei display etc.

Usando dinamiche *prey-predator* si vuole verificare che servizi che non interessano agli utenti gradualmente svaniscano lasciando invece emergere quelli più rilevanti. Servizi nuovi e/o aggiornati dovranno competere con quelli più vecchi a livello locale e spaziale per determinare quali debbano sopravvivere, seguendo un pattern analogo all'evoluzione naturale in cui gli individui più forti dominano. Si intende poi sfruttare meccanismi basati sul gradiente computazionale per visualizzare in un certo display i servizi e le informazioni che interessano all'utente prospiciente ad esso, sulla base delle sue richieste/preferenze e della localizzazione del display.

Queste dinamiche sono ottenute attraverso opportune reazioni chimiche, selezionate e messe in esecuzione dal motore chimico.

3 TuCSoN

TuCSoN è un modello di coordinazione per agenti distribuiti e potenzialmente autonomi, intelligenti e mobili.

Gli agenti (coordinabili) sono coordinati attraverso i centri di tuple ognuno situato in uno specifico nodo di uno specifico device di rete. Ogni device è individuato dal suo indirizzo IP e può contenere uno o più nodi che offrono i loro servizi su specifiche porte (nodo di default alla porta 20504). A sua volta, ogni nodo può ospitare uno o più centri di tuple specificati dal loro identificativo univoco (se non specificato ci si riferisce al centro di tuple di default determinato in fase di configurazione). Agenti e device possono essere distribuiti in rete (e di conseguenza anche i centri di tuple associati ai nodi dei device).

Le entità base del sistema sono quindi due: agenti e centri di tuple. I primi hanno un ruolo pro-attivo, coordinandosi attraverso operazioni sugli spazi di tuple quali inserimento, rimozione e lettura di tuple dallo spazio, che sono primitive messe a disposizione dal linguaggio di coordinazione TuCSoN. I centri di tuple hanno invece un ruolo reattivo e sono costituiti da due componenti principali: uno spazio condiviso per la comunicazione basata su tuple (spazio di tuple) e uno spazio con comportamento programmabile (spazio di specifica).

Il linguaggio di coordinazione di TuCSoN mette a disposizione 9 primitive di tipo

- *out(Tuple)* : inserimento della tupla Tuple nello spazio di tuple.
- *in(TupleTemplate)* : viene rimossa una tupla scelta in modo casuale tra quelle che fanno match con *TupleTemplate* e quindi restituita; primitiva bloccante finchè non trova una tupla che fa match con il template. La sua

variante non bloccante è *inp* (restituisce la tupla in caso di successo, *false* altrimenti).

- *rd(TupleTemplate)* : viene letta una tupla scelta in modo casuale tra quelle che fanno match con *TupleTemplate* e quindi restituita; primitiva bloccante finchè non trova una tupla che fa match con il template. La sua variante non bloccante è *rdp* (restituisce la tupla in caso di successo, *false* altrimenti).

- *no(TupleTemplate)* : termina con successo nel caso in cui non esista alcuna tupla che fa match con *TupleTemplate*, restituendo il template; altrimenti si blocca in attesa che la condizione precedente sia verificata. Variante non bloccante *nop*, restituisce true/false nel caso in cui non esista/esista una tupla nello spazio che fa match con il template.

- *get* : resitutisce tutte le tuple nello spazio; se non ne è presente nessuna restituisce la lista vuota;

- *set(Tuples)* : riscrive lo spazio di tuple con la lista di tuple *Tuples* restituendola.

Questo set di primitive base è stato esteso con una variante *bulk*(*in_all*, *rd_all*, etc), cioè primitive di massa analoghe alle precedenti ma che restituiscono la lista completa delle tuple che fanno match con il template e una variante *uniforme* (*uin*, *urd*) che selezionano la tupla da restituire tra quella che fanno match, non in modo casuale, ma seguendo una distribuzione di probabilità uniforme.

Lo spazio di specifica è invece programmato tramite il linguaggio di specifica ReSpecT e le primitive di meta-coordinazione. ReSpecT si basa su tuple del tipo *reaction(Event,Guards,Effects)* il cui significato è: quando occorrono gli eventi *Events*(una qualsisasi primitiva di coordinazione o di meta-coordinazione), applica allo spazio di tuple gli effetti specificati in *Effects*(un qualsisasi insieme di primitive di coordinazione o di meta-coordinazione) se le guardie *Guards* (*request/response, endo/exo*) sono verificate. Le primitive di meta-coordinazione sono del tutto analoghe a quelle di coordinazione, con l'unica differenza che agiscono su tuple del linguaggio ReSpecT, permettendo quindi di inserire nello spazio la sua specifica di comportamento.

4 Mapping concettuale SAPERE on TuCSoN

Si analizza ora come poter realizzare in modo concreto le caratteristiche del progetto SAPERE sfruttando l'infrastruttura tecnologica TuCSoN.

Il modello a spazio di tuple chimico può essere implementato con TuCSoN considerando le due estensioni precedentemente evidenziate, ovvero match semantico e motore chimico.

Per quanto riguarda la funzione di match semantico, l'idea è quella di associare ad ogni centro di tuple una ontologia usando la *Descrition Logic*, in cui

descrivere concetti e ruoli (relazioni tra individui). Questo porta alla definizione di tuple semantiche che mantengono informazioni riguardo nome, concetto a cui appartiene e individui a cui sono collegati tramite specifici ruoli. Per recuperare le tuple è necessario definire un template semantico ognuno associato con un concetto dell'ontologia. La funzione di match restituisce quindi il grado di rilevanza di una certa tupla rispetto un certo template.

Per quanto riguarda la seconda fondamentale estensione, TuCSoN offre, attraverso il linguaggio di specifica ReSpecT, la possibilità di programmare il comportamento dello spazio di tuple e quindi la realizzazione di un simulatore chimico online (come descritto in precedenza) come un insieme di primitive di meta-coordinazione che agiscono nello spazio di specifica inserendo/rimuovendo/leggendo reazioni rappresentate nel linguaggio ReSpecT.

Appare quindi evidente che TuCSoN permetta di modellare bene una infrastruttura distribuita per coordinare complessi sistemi pervasivi. Da un punto di vista topologico attraverso la realizzazione di diversi centri di tuple locali distribuiti sui vari nodi della rete, ognuno dei quali mantiene le informazioni necessarie per comunicare con i vicini (indirizzo Ip del device, porta del nodo, identificatico centro di tuple). Dalla prospettiva temporale, disponendo di reazioni che scattano in background a istanti di tempo prefissati. Dal punto di vista probabilistico, attraverso il linguaggio di specifica ReSpecT, permettendo di definire un simulatore chimico che computi adeguatamente la reaction rates, selezioni in modo non deterministico le leggi da eseguire e determini l'intervallo tra step consecutivi seguendo la dinamica di sistemi di Markov tempo discreti.