Tecnologie per la collaborazione sociale tra robot Mindstorm: LeJOS & ReSpecT

-relazione-

1. Obbiettivo - specifica (tesi)

   L'obbiettivo di questo lavoro è sperimentare nuove tecnologie per la collaborazione sociale basata sul linguaggio di coordinazione ReSpecT tra robot Lego-Mindstorm NXT. Si valutano i vantaggi/svantaggi dell'integrazione di tale collaborazione con LeJOS, piuttosto che con le API iCommand. Il benchmark viene realizzato confrontando le prestazioni dei vari approcci in alcuni scenari già discussi in lavori precedenti.

2. Analisi e progettazione
   a. Lavori precedenti

      Le due tesi precedenti hanno cercato di sfruttare la tecnologia ReSpecT/ReSpecT Situated per realizzare un sistema di collaborazione tra robot Mindstorm. Tale intento è stato portato a compimento con risultati soddisfacenti, sebbene non ottimali. Infatti l'approccio proposto presenta intrinsecamente dei limiti dovuti alla tecnologia scelta per l'implementazione: le API iCommand. Questo package Java consente di controllare un robot NXT attraverso una connessione Bluetooth; utilizza il firmware standard Lego NXT per trasmettere comandi scritti in Java da computer al robot.

      Inizialmente (nel 2006) iCommand è stato rilasciato per permettere agli utenti di cominciare a programmare il NXT in Java mentre LeJOS NXJ era ancora in fase di sviluppo. Ad oggi LeJOS NXJ è disponibile nella versione 0.9.1 (del 6 febbraio 2012).

      Il lavoro presente si prefigge di "trasportare" i lavori precedenti da iCommand a LeJOS NXJ indicando i vantaggi di questa "migrazione". Tra questi il più evidente vantaggio riguarda la comunicazione della variazione dei valori rilevati dai sensori del NXT. Con iCommand tali valori erano richiesti ad intervalli regolari dal coordinatore, ovviamente un polling dispendioso. LeJOS NXJ supporta il meccanismo ad eventi che garantirà che sia il NXT stesso ad inviare una lettura dei sensori qualora si verifichi un particolare evento.

   b. Lavoro attuale

      Si cerca di utilizzare sempre ReSpecT per sfruttare i vantaggi relativi alla coordinazione.

      Lo scoglio da affrontare è relativo alla scrittura del firmware per NXT. Infatti iCommand era compatibile con il firmware standard NXT, mentre la nuova soluzione implica lo sviluppo di un firmware specifico per ogni scenario analizzato in parte.

      L'architettura logica progettata contiene i seguenti componenti:

- un server su PC che gestisce i vari client NXT che si vogliono connettere. Accetta le richieste di collegamento, e in base al programma in esecuzione sul client, alloca le risorse necessarie.
- vari Tuple-Center (con engine ReSpecT) che vengono allocati per la coordinazione tra NXT che fanno parte della stessa applicazione.
- client NXT che eseguono moduli specifici

3. Implementazione (scenari supportati – codice allegato)

   Una volta finita l'iniziale analisi del progetto, il passo successivo è stato quello di scrittura del codice. Questa fase implica il maggior impegno di tempo.

   a. Il server introdotto rappresenta una specie di middleware. Accetta le richieste dei client e li associa come componenti al contesto di esecuzione di un'applicazione (scenario).
      Per esempio: il 1° scenario consiste in un robot che si muove su una griglia, mentre un altro robot lo insegue. Il server accetta la richiesta del primo NXT, chiede l'applicazione di cui fa parte e fa partire un nuovo thread che gestisca quello scenario. Quando arriva il secondo NXT, il server accetta la richiesta, e dopo aver chiesto l'applicazione di cui fa parte, lo associa al thread già in esecuzione per quello scenario.
   b. I vari scenari corrispondono ad altrettanti programmi per NXT. Questi programmi estendono una classe base che incapsula le informazioni comuni come nome del programma, identificativo del server, orientamento iniziale, etc. Poi ciascun programma in parte inizializza i sensori necessari e implementa il flusso logico per l'esecuzione dell'applicazione specifica. E' necessario specificare che per ciascuna applicazione in parte viene utilizzato un protocollo di comunicazione ad-hoc a stati. Questo è dovuto anche al fatto che LeJOS supporta l'invio su bluetooth solo di tipi base (int, string, etc.)
      Quindi, per comunicare il nome del programma il NXT invia una stringa quando si trova nello stato iniziale nel quale il server si aspetta i ricevere una stringa. D'altra parte, quando il NXT aspetta di ricevere un segnale e il server manda un intero, il NXT sa già cosa rappresenta quel intero in base allo stato dell'applicazione.
   c. La coordinazione viene garantita da centri di tuple. L'engine ReSpecT viene avviato separatamente dal server, poi ciascuna applicazione avrà il suo TC dedicato. L'applicazione inizia con la connessione dei vari client. Poi, durante l'esecuzione, il thread che gestisce un'applicazione inserisce tuple nel centro di tuple, scatenando eventuali reazioni.
   d. Ho implementato una piccola libreria di funzioni utili a snellire lo sviluppo di futuri programmi. Contiene wrapper per I/O LeJOS, segnalazione errori, logging, nonché alcune funzioni per la comunicazione bluetooth.

4. Testing

   Momentaneamente la parte di collaudo è ferma per problemi legati alla tecnologia: non riesco a fare connettere i NXT al PC. Ho comprato ben due adattatori bluetooth, ma nessuno dei due non funziona bene. Ho speso due giorni interi in laboratorio a fare prove, aggiornare driver, etc., senza venirne a capo. E' probabile che il problema sia l'architettura a 64 bit del mio portatile. Continuerò le prove anche con una macchina virtuale…

   Ovviamente il codice subirà degli adattamenti non appena riuscirò a fare qualche prova.