Progetto di Sistemi Distribuiti e Pervasivi

Laboratorio di Sistemi Distribuiti e Pervasivi

Aprile 2016

1 Descrizione del progetto

Lo scopo del progetto è quello di realizzare un sistema di monitoraggio remoto di un ambiente sensorizzato simulato. I sensori simulati facenti parte dell'ambiente devono coordinarsi tra di loro per comunicare le loro misurazioni ad un gateway, il quale ha il compito di memorizzarle internamente. Il gateway ha anche il ruolo di offrire agli utenti del sistema un'interfaccia per effettuare svariati tipi di interrogazioni, in modo da ottenere informazioni sui sensori attualmente installati e statistiche sullo stato dell'ambiente (ad esempio la temperatura media degli ultimi n minuti). L'architettura generale del sistema è illustrata in Figura 1. I tre componenti principali da realizzare sono: Rete di sensori, Gateway e Utente. La Rete di sensori consiste in un insieme di processi che simulano i diversi tipi di sensore installati nell'ambiente. Questi processi dovranno coordinarsi tra di loro per trasmettere le varie misurazioni a Gateway. I sensori possono essere aggiunti o rimossi nella rete in maniera dinamica. Gateway è il server che ha il compito di ricevere e memorizzare i dati provenienti dai sensori. Inoltre, deve anche predisporre un sistema di monitoraggio da remoto che permetta di effettuare diversi tipi di interrogazioni sullo stato del sistema. Le interrogazioni vere e proprie vengono effettuate dagli utenti tramite l'apposito client Utente.

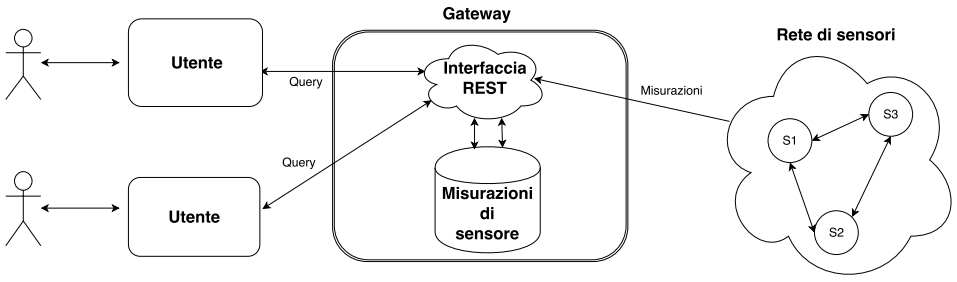

Figura 1: Architettura del sistema.

Le prossime sezioni di questo documento descriveranno in dettaglio le componenti del sistema da implementare.

2 Rete di sensori

L'ambiente simulato da monitorare è una stanza dotata di sensori di diverso tipo, che possono essere aggiunti o rimossi dinamicamente in maniera arbitraria. Vengono considerate in tutto tre diverse *tipologie* di sensore:

- Sensore di temperatura (in gradi Celsius)
- Sensore di luminosità (in lux)
- Accelerometro proveniente da smartwatch indossabile al polso da persone presenti nell'ambiente. Per semplicità considereremo solo i dati di accelerazione (in milli g, dove g = $9.81 \frac{m}{s^2}$) provenienti dall'asse X

Nel suo stato stato iniziale, la rete è priva di sensori. Successivamente, i sensori possono essere aggiunti/rimossi dinamicamente. Ognuno dei sensori presenti nella rete è simulato da un relativo **processo**, che chiameremo *nodo*. I nodi comunicano tra di loro con un opportuno protocollo tramite socket, mentre si interfacciano a Gateway tramite chiamate REST.

2.1 Misurazioni

Ogni nodo produce periodicamente delle misurazioni che sono caratterizzate da:

- Id del sensore
- Tipologia del sensore
- Valore letto
- Timestamp in millisecondi

La generazione di queste misurazioni viene svolta da opportuni simulatori, che vengono forniti già pronti per semplificare lo svolgimento del progetto. Ogni simulatore assegna come timestamp alle misurazioni il numero di secondi passati dalla mezzanotte.
Al link *http://ewserver.di.unimi.it/sdp/simulation_src_2016.zip* è quindi possibile trovare il codice necessario. Questo codice andrà aggiunto come package al progetto e NON deve essere modificato.

Per ogni tipologia di sensore, viene fornita una classe che implementa un thread di simulazione. Ogni nodo deve quindi occuparsi di far partire il thread giusto in base al ruolo che deve svolgere. Ogni thread consiste in un

loop infinito che simula periodicamente (con frequenza predefinita) le misurazioni, inserendole in una opportuna struttura dati. Di questa struttura dati viene fornita solo l'interfaccia (Buffer), la quale espone due metodi:

- $void\ add(Measurement\ m)$
- $List<Measurement>\ readAllAndClean()$

È dunque necessario creare una classe che implementi l'interfaccia. Ogni **processo** che simula un sensore avrà un suo buffer. I thread di simulazione usano il metodo *add* per riempire la struttura dati. Il metodo *readAllAndClean* deve invece essere usato per ottenere tutte le misurazioni contenute nella struttura dati. Al termine di una lettura, *readAllAndClean* deve occuparsi di svuotare il buffer in modo da fare spazio a nuove misurazioni. Il buffer deve essere popolato diversamente in base alla tipologia di sensore:

- In caso di sensore di temperatura o sensore di luminosità, le misurazioni prodotte dai simulatori vengono aggiunte normalmente nel buffer una ad una.
- Nel caso di accelerometro (il quale produce una mole molto più grossa di dati) il buffer deve essere popolato sfruttando una tecnica di tipo *sliding window* usando finestre temporali di un secondo. In pratica, invece di memorizzare ogni singola misurazione prodotta dai simulatori, si richiede di memorizzare il valore medio delle misurazioni collezionate ogni secondo.

2.2 Rete di sensori

Come è già stato introdotto, la rete di sensori è dinamica. La rete deve essere quindi capace di adattarsi dinamicamente in base ai sensori che vengono aggiunti o rimossi, e deve essere implementata con una topologia ad anello. Un problema di mutua esclusione da affrontare è quello che solo un nodo alla volta può comunicare con il gateway. Per risolvere questo problema, è necessario sfruttare la tecnica di sincronizzazione *token ring* vista a lezione.

2.3 Inizializzazione *nodo*

Ogni singolo **processo** che implementa un *nodo* necessita di diversi argomenti per essere avviato:

- Tipologia di sensore (temperatura, luminosità o accelerometro)
- Identificatore del nodo (stringa arbitraria)
- Porta di ascolto

- Indirizzo del Gateway

Una volta inizializzato, il primo compito del nodo è quello di registrarsi al Gateway. Per fare questo, sfrutta l'interfaccia REST del Gateway, comunicando le sue informazioni. Se tutto va a buon fine, il nodo riceve dal Gateway la lista dei sensori attualmente partecipanti alla rete e i relativi indirizzi, permettendo al nodo di inserirsi nella rete. Se il nodo in questione è il primo della rete, si deve anche occupare di creare il token. Altrimenti, deve inserirsi nella rete ad anello comunicando con gli altri nodi attualmente presenti. Si raccomanda di gestire anche i casi critici.

2.3.1 Gestione del Token

Nella nostra applicazione, il token deve essere utilizzato per veicolare le misurazioni di sensore prodotte da ogni nodo. Ogni qualvolta che un nodo riceve il token e il suo buffer non è vuoto, aggiunge al token le misurazioni attualmente presenti nel proprio buffer usando il metodo $readAllAndClean()$ spiegato precedentemente. Dopodichè il token viene passato al nodo successivo nella rete ad anello. In caso di buffer vuoto, il token deve invece essere passato immediatamente al nodo successivo nella rete. Il token ha una capienza massima di 15 misurazioni. Quando un nodo si accorge che il token ha raggiunto la sua capienza massima, trasmette le misurazioni in esso contenute al Gateway. Dopodichè, il sistema continua normalmente con il token "svuotato" delle sue misurazioni.

2.3.2 Uscita di un nodo dalla rete

Ogni nodo può essere rimosso in un qualsiasi momento dalla rete. Per fare ciò, è necessario inserire la stringa *exit* da linea di comando nel relativo processo. Una volta inserita questa stringa, il nodo deve occuparsi di uscire dalla rete comunicando con il Gateway e con gli altri nodi. È fondamentale che la rete continui a funzionare correttamente dopo l'uscita di un nodo (ad esempio, il token non deve essere perso). Anche in questo caso è richiesto di gestire tutti i casi critici.

3 Gateway

L'applicazione Gateway è incaricata di svolgere il ruolo di gateway per la rete di sensori, memorizzando le misurazioni man mano ricevute internamente in un'opportuna struttura dati. Inoltre, deve mantenere le informazioni sui sensori attualmente presenti nella rete e i relativi indirizzi, in modo da coordinare l'entrata e l'uscita di nodi dalla rete. Gateway espone inoltre agli utenti un'interfaccia che permette loro di realizzare le interrogazioni necessarie ad ottenere informazioni sullo stato del sistema. Queste interrogazioni

vengono quindi realizzate sulla struttura dati che contiene lo stream di misurazioni provenienti dalla rete di sensori. Infine, si occupa di inviare ad ogni utente registrato al sistema delle notifiche push riguardanti l'aggiunta e la rimozione di nodi nella rete.

3.1 Interfaccia con gli utenti

L'applicazione Gateway mette a disposizione degli utenti che intendono monitorare l'ambiente sensorizzato un servizio REST per effettuare le interrogazioni sui dati collezionati dai sensori. Devono quindi essere disponibili diversi tipi di servizi:

- Log in di un utente
- Interrogazioni sullo stato della rete e statistiche sui dati raccolti
- Log out di un utente

Per effettuare il log in, un utente deve semplicemente inviare il proprio nome utente che deve essere univoco all'interno del sistema. Nel caso che il nome trasmesso da un utente sia già stato usato, deve essere trasmesso un opportuno messaggio di errore.

3.2 Interfaccia con i sensori

Gateway predispone un'interfaccia REST anche per la rete di sensori. Un primo compito di questa interfaccia è di gestire la registrazione e la rimozione dei nodi dalla rete. In caso di registrazione, Gateway si assicura che non sia registrato nel sistema un altro nodo con lo stesso identificatore. Se il controllo ha buon fine, aggiunge il nodo alla rete (tenendo traccia dell'indirizzo di rete) e trasmette ad esso la lista degli altri nodi attualmente registrati con i relativi indirizzi. Per ogni sensore registrato, Gateway deve memorizzare il timestamp di aggiunta del sensore alla rete, in modo tale da poter calcolare, quando richiesto, il tempo di uptime (ovvero "da quanto tempo il nodo fa parte della rete"). In caso di rimozione, Gateway si occupa solamente di cancellare il nodo dalla lista dei nodi facenti parte della rete, dando un feedback positivo ad operazione completata correttamente. L'altro importante compito di Gateway è quello di ricevere le misurazioni raccolte dalla rete di sensori, che vengono salvate in un'opportuna struttura dati (che per semplicità può essere in memoria centrale).

3.3 Interrogazioni

Il sistema deve essere predisposto a gestire diversi tipi di interrogazione:

- Ottenere la lista dei sensori attualmente presenti nella rete con relative informazioni (identificatore, tipo) e il tempo di uptime

- Ottenere la misurazione più recente e il relativo timestamp di uno specifico sensore
- Ottenere la media, massimo e minimo delle misurazioni dal tempo t_1 al tempo t_2 per uno specifico sensore
- Ottenere la media, massimo e minimo delle misurazioni dal tempo t_1 al tempo t_2 tra tutti i sensori di uno stesso tipo

Nel caso di interrogazioni che agiscono su intervalli, deve essere restituito un opportuno messaggio di errore se non sono presenti dati nell'intervallo specificato. Allo stesso modo, nel caso di interrogazioni che richiedono misurazione più recente, deve essere restituito un messaggio di errore se quel tipo di misurazioni non è ancora stato collezionato.

3.4 Notifiche push

Ogni utente registrato nel sistema si rende disponibile a ricevere notifiche push: ogni volta che un nodo viene aggiunto o rimosso nella rete, Gateway si deve occupare di mandare in broadcast a tutti gli utenti che hanno effettuato il log in una notifica dell'avvenuto evento. Ovviamente, quando un utente effettua il log out, deve essere rimosso dalla lista di broadcast. Si pensi ad una soluzione architetturale appropriata per introdurre questa feature al sistema.

4 Utente

L'applicazione Utente è un client (per semplicità a linea di comando) utilizzato dagli utenti che intendono monitorare il sistema. In fase iniziale vengono richiesti all'utente il nickname che si intende utilizzare e l'indirizzo (ip e porta) del sistema da monitorare. Con queste informazioni, il programma si deve occupare di registrare l'utente al sistema. Se viene ricevuto un messaggio dal sistema di nome utente già utilizzato, deve essere chiesto all'utente di inserirlo nuovamente. A registrazione completata correttamente, il programma propone in loop all'utente una scelta:

- Effettuare una delle interrogazioni specificate nella sezione precedente
- Terminare il programma (in questo caso il programma dovrà segnalare il log out dell'utente al sistema)

L'applicazione Utente, inoltre, deve restare in ascolto di comunicazioni delle notifiche push riguardanti l'aggiunta o la rimozione di nodi nel sistema. Quando una notifica viene ricevuta, deve essere comunicata tempestivamente all'utente.

5 Semplificazioni e limitazioni

Si ricorda che lo scopo del progetto è dimostrare la capacità di progettare e realizzare un'applicazione distribuita e pervasiva. Pertanto gli aspetti non riguardanti il protocollo di comunicazione, la concorrenza e la gestione dei dati di sensori sono considerati secondari. Inoltre è possibile assumere che :

- nessun nodo si comporti in maniera maliziosa,
- nessun nodo termini in maniera incontrollata

Si gestiscano invece i possibili errori di inserimento dati da parte dell'utente.

Sebbene le librerie di Java forniscano molteplici classi per la gestione di situazioni di concorrenza, per fini didattici gli studenti sono invitati a fare esclusivamente uso di metodi e di classi spiegati durante il corso di laboratorio. Pertanto, eventuali strutture dati di sincronizzazione necessarie (come ad esempio lock, semafori o buffer condivisi) dovranno essere implementate da zero e saranno discusse durante la presentazione del progetto.

Nonostante alcune problematiche di sincronizzazione possano essere risolte tramite l'implementazione di server iterativi, per fini didattici si richiede di utilizzare server multithread. Inoltre, per le comunicazioni via socket, è richiesto di usare formati standard (ad esempio JSON, XML) per lo scambio di dati.

6 Presentazione del progetto

Il progetto è da svolgere individualmente. Durante la valutazione del progetto verrà richiesto di mostrare alcune parti del programma, verrà verificata la padronanza del codice presentato, verrà verificato il corretto funzionamento del programma e verranno inoltre poste alcune domande di carattere teorico inerenti gli argomenti trattati nel corso (parte di laboratorio). E' necessario presentare il progetto sul proprio computer.

Il codice sorgente dovrà essere consegnato al docente prima della discussione del progetto. Per la consegna, è sufficiente archiviare il codice in un file zip, rinominato con il proprio numero di matricola (es. 760936.zip) ed effettuare l'upload dello stesso tramite il sito http://upload.di.unimi.it . Sarà possibile effettuare la consegna a partire da una settimana prima della data di ogni appello. La consegna deve essere tassativamente effettuata entro le 23:59 del secondo giorno precedente quello della discussione (es. esame il 13 mattina, consegna entro le 23.59 dell'11).

Si invitano inoltre gli studenti ad utilizzare, durante la presentazione, le istruzioni `Thread.sleep()` al fine di mostrare la correttezza della sincronizzazione del proprio programma. Si consiglia vivamente di analizzare con

attenzione tutte le problematiche di sincronizzazione e di rappresentare lo schema di comunicazione fra le componenti. Questo schema deve rappresentare il formato dei messaggi e la sequenza delle comunicazioni che avvengono tra le componenti in occasione delle varie operazioni che possono essere svolte. Tale schema sarà di grande aiuto, in fase di presentazione del progetto, per verificare la correttezza della parte di sincronizzazione distribuita.

Nel caso in cui il docente si accorga che una parte significativa del progetto consegnato non è stata sviluppata dallo studente titolare dello stesso, lo studente in causa: a) dovrà superare nuovamente tutte le prove che compongono l'esame del corso. In altre parole, se l'esame è composto di più parti (teoria e laboratorio), lo studente dovrà sostenere nuovamente tutte le prove in forma di esame orale (se la prova di teoria fosse già stata superata, questa verrà annullata e lo studente in merito dovrà risostenerla). b) non potrà sostenere nessuna delle prove per i 2 appelli successivi. Esempio: supponiamo che gli appelli siano a Febbraio, Giugno, Luglio e Settembre, e che lo studente venga riconosciuto a copiare all'appello di Febbraio. Lo studente non potrà presentarsi agli appelli di Giugno e Luglio, ma potrà sostenere nuovamente le prove dall'appello di Settembre in poi. Il docente si riserva la possibilità di assegnare allo studente in causa lo svolgimento di una parte integrativa o di un nuovo progetto.

7 Parti facoltative

7.1 Prima parte

Estendere il sistema in maniera tale che l'aggiunta e la rimozione di sensori nella rete sia resa possibile agli utenti attraverso l'applicazione Utente. Gestire i possibili nuovi casi di concorrenza.

7.2 Seconda parte

In questa seconda parte facoltativa, l'assunzione che nessun nodo termini in maniera incontrollata non è più vera per la rete di sensori. Si gestisca quindi il caso in cui un nodo della rete di sensori crolli senza preavviso. In particolare, è necessario gestire in maniera appropriata la possibile perdita del token e la riorganizzazione della rete, che deve scoprire automaticamente quando un nodo è crollato in maniera incontrollata.

8 Aggiornamenti

Qualora fosse necessario, il testo dell'attuale progetto verrà aggiornato al fine di renderne più semplice l'interpretazione. Le nuove versioni del progetto verranno pubblicate sul sito del corso. Si consiglia agli studenti di controllare regolarmente il sito.

Al fine di incentivare la presentazione del progetto nei primi appelli disponibili, lo svolgimento della parte facoltativa 1 diventa obbligatoria a partire dall'appello di Settembre (incluso), mentre entrambe le parti facoltative (1 e 2) saranno obbligatorie negli appelli successivi.