Progetto di Sistemi Distribuiti e Pervasivi

Laboratorio di Sistemi Distribuiti e Pervasivi

Maggio 2020

1 Descrizione del progetto

Lo scopo del progetto è quello di realizzare un sistema di monitoraggio distribuito del livello di inquinamento atmosferico di un quartiere di una smart city. Nel quartiere sono presenti diversi nodi ai quali sono collegati dei sensori per il rilevamento di PM10. Per motivi di privacy, questi nodi necessitano di comunicare e coordinarsi tra di loro per inviare ad un gateway della smart city dati aggregati sul livello di inquinamento del quartiere. Questo gateway ha il compito di memorizzare questi dati aggregati e renderli accessibili ad analisti. Questi ultimi devono poter effettuare interrogazioni al gateway per ottenere le statistiche sul livello di inquinamento. L'architettura generale del sistema è illustrata in Figura 1. I tre componenti principali da realizzare sono: Nodo, Gateway e Client Analista. La rete di nodi consiste in un insieme di processi che simulano i nodi del quartiere al quale sono associati i sensori di PM10. Questi processi dovranno coordinarsi tra di loro per trasmettere le varie misurazioni a Gateway. I nodi 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 nodi. 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 analisti tramite l'apposito client Analista.

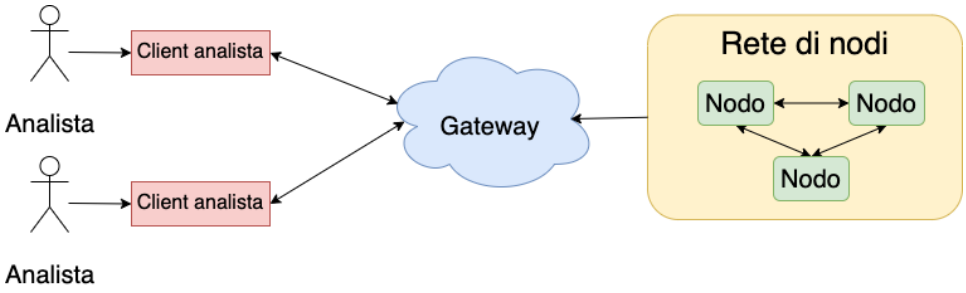

Figura 1: Architettura del sistema.

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

2 Nodo

Il quartiere della città è composto da una rete peer to peer di nodi computazionali. Ognuno di questi nodi è associato ad un sensore che misura periodicamente il livello di polveri sottili nell'aria (PM10) in $\mu g/m^3$. Nel suo stato stato iniziale, la rete è priva di nodi. Successivamente, i nodi possono aggiungersi e rimuoversi dinamicamente. Ognuno dei nodi presenti nella rete è simulato da un relativo **processo**, che chiameremo *Nodo*.

I nodi devono coordinarsi per inviare periodicamente al gateway un'unica statistica riassuntiva del quartiere. La rete deve deve essere implementata con una topologia ad anello e solo un nodo alla volta può comunicare la statistica al gateway. Per risolvere questo problema di mutua esclusione, è necessario sfruttare la tecnica di sincronizzazione *token ring* vista a lezione.

I nodi comunicano tra di loro con un opportuno protocollo tramite *grpc*, mentre si interfacciano a Gateway tramite chiamate REST.

2.1 Inizializzazione

Un *nodo* deve essere inizializzato specificando il suo identificatore, il numero di porta di ascolto per la comunicazione tra *nodi* e infine l'indirizzo IP e il numero di porta del *gateway*. Una volta avviato, il processo *nodo* deve avviare il suo simulatore di sensore PM10 (vedi Sezione 2.2) e registrarsi al quartiere tramite il *gateway*. Se l'inserimento va a buon fine (ovvero, non esistono altri nodi con lo stesso identificatore), il *nodo* riceve dal server l'elenco di *nodi* presenti nella rete peer-to-peer, in modo tale da poter presentarsi per entrare in maniera decentralizzata nella rete. Quindi, la rete peer-to-peer deve costruirsi senza alcun ausilio da parte del *Gateway*, il cui unico compito è quello di fornire l'elenco di nodi già presenti nel quartiere.

2.2 Sensore di PM10

Il sensore di PM10 produce periodicamente delle misurazioni che sono caratterizzate da:

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

All'interno del processo *Nodo*, il sensore di PM10 viene simulato con un opportuno thread. Questo simulatore viene fornito già pronto per semplificare lo svolgimento del progetto. Il codice necessario alla simulazione può essere trovato sul sito del corso. Questo codice andrà aggiunto come package al progetto e NON deve essere modificato.

In fase di inizializzazione, ogni nodo dovrà quindi occuparsi di far partire il suo simulatore per generare misurazioni. Il simulatore è un thread che consiste in un loop infinito che simula periodicamente (con frequenza predefinita) le misurazioni, aggiungendole ad una struttura dati. Di questa struttura dati viene fornita solo l'interfaccia (Buffer), la quale espone il seguente metodo:

- $void\ addMeasurement(Measurement\ m)$

Risulta dunque necessario creare una classe che implementi l'interfaccia. Il thread del simulatore usa il metodo $addMeasurement$ per riempire il buffer.

2.3 Statistica locale

Ogni nodo deve produrre periodicamente una statistica relativa al livello di inquinamento misurato. Per questo motivo, deve effettuare periodicamente una media dei dati nel buffer. In particolare, è necessario implementare la tecnica della *sliding window* presentata durante le lezioni di teoria, considerando una dimensione del buffer di 12 misurazioni e un overlap del 50%

2.4 Sincronizzazione distribuita

2.4.1 Dinamicità della rete

La rete ad anello deve adattarsi dinamicamente quando dei nodi vengono inseriti e rimossi. Risulta necessario garantire che la rete ad anello resti sempre consistente a fronte di casi limite (ad esempio, quando più nodi entrano e/o escono contemporaneamente).

2.4.2 Calcolo e invio statistica del quartiere

Come spiegato in Sezione 2.3, ogni nodo produce periodicamente una media locale del livello di inquinamento. Quando ogni nodo della rete ha prodotto una media locale, la rete deve calcolare una media di queste statistiche e mandarle al server. Solo un nodo della rete può essere incaricato di questo compito. Per risolvere questo problema di mutua esclusione distribuita è richiesto di utilizzare l'algoritmo token ring.

Il token deve veicolare le informazioni relative alle medie locali calcolate da ogni nodo. Quando un nodo riceve il token ed ha prodotto una nuova

media locale, deve inserirla nel token prima di passarlo al nodo successivo nell'anello. Il primo nodo che si accorge che tutti i nodi della rete hanno prodotto una nuova media locale trattiene il token per calcolare la media delle statistiche nel token e inviarla al gateway. Sucessivamente all'invio della statistica globale, il token deve essere "svuotato" dalle medie locali prima di poter passare al nodo successivo.

2.5 Uscita dalla rete

Ogni nodo può decidere esplicitamente di uscire dalla rete peer-to-peer in qualsiasi momento. L'applicazione *Nodo* deve quindi predisporre un'interfaccia a linea di comando per permettere di inserire un comando relativo all'uscita dalla rete. Quando questo comando viene inserito, prima di terminare, il nodo deve comunicare al *Gateway* e agli altri nodi della sua uscita. Si assume che un nodo non possa uscire dalla rete in modo incontrollato. È fondamentale che la rete continui a funzionare correttamente dopo l'uscita di un nodo (ad esempio, il token non deve essere perso).

3 Gateway

Il *Gateway* si occupa di ricevere dal quartiere le statistiche che verranno poi interrogate dagli analisti. Inoltre, si occupa di gestire l'ingresso e l'uscita di nodi dalla rete peer-to-peer. Deve quindi offrire diverse interfacce REST per: a) gestire la rete di nodi, b) ricevere le statistiche sull'inquinamento e c) permettere agli analisti di effettuare interrogazioni. Il *gateway* non deve in alcun modo coordinare la topologia di comunicazione della rete peer-to-peer dei nodi e non deve occuparsi di risolvere problemi di mutua esclusione distribuita.

3.1 Interfaccia per i nodi

3.1.1 Inserimento

Quando vuole inserirsi nel sistema, un *nodo* deve comunicare al server:

- Identificatore
- Indirizzo IP
- Numero di porta sul quale è disponibile per comunicare con gli altri nodi

Il server aggiunge un nodo al suo elenco di nodi presenti nel quartiere solo se non esiste un altro nodo con lo stesso identificatore. Se l'inserimento va a buon fine, il *gateway* restituisce al nodo la lista di nodi già presenti nella rete, specificando per ognuno indirizzo IP e numero di porta per la comunicazione.

3.1.2 Rimozione

Un *nodo* può richiedere esplicitamente di rimuoversi dalla rete. In questo caso, il gateway deve semplicemente rimuoverlo dall'elenco di nodi del quartiere.

3.1.3 Statistiche

Il *gateway* deve predisporre un'interfaccia per ricevere dalla rete le statistiche riguardanti il livello di inquinamento. È sufficiente memorizzare queste statistiche in una struttura dati che permetta successivamente l'analisi da parte degli analisti.

3.2 Interfaccia per gli analisti

Il *gateway* deve fornire dei metodi per ottenere le seguenti informazioni:

- Numero di nodi presenti nella rete
- Ultime n statistiche (con timestamp) di quartiere
- Deviazione standard e media delle ultime n statistiche prodotte dal quartiere

4 Client Analista

L'applicazione *Client Analista* è un'interfaccia a linea di comando che si occupa di interagire con l'interfaccia REST precedentemente presentata in Sezione 3. L'applicazione deve quindi semplicemente mostrare all'analista un menu per scegliere uno dei servizi di analisi offerto dal server, con la possibilità di inserire eventuali parametri che sono richiesti.

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 i sono considerati secondari. Inoltre è possibile assumere che:

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

Si gestiscano invece i possibili errori di inserimento dati da parte dell'utente. Inoltre, il codice deve essere robusto: tutte le possibili eccezioni devono essere gestite correttamente.

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**, escludendo quindi la parte di puntatori a strumenti avanzati di sincronizzazione. 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. Si richiede di utilizzare il framework *grpc* per la comunicazione tra processi e NON le socket. In fase di presentazione del progetto, lo studente è comunque tenuto a conoscere le socket che, essendo parte del programma di laboratorio, possono essere oggetto di domande.

Qualora fossero previste comunicazioni in broadcast, queste devono essere effettuate in parallelo e non sequenzialmente.

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 l'esecuzione progetto sul proprio computer.

Il codice sorgente dovrà essere consegnato al docente prima della discussione del progetto. Per la consegna, è sufficiente archiviare **esclusivamente il codice sorgente** 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 svol-

te. 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

In questa parte facoltativa, si vuole estendere il sistema con un sistema di notifiche push per gli analisti. È necessario realizzare un'architettura che permetta agli analisti di ricevere una notifica quando:

- Risulta disponibile una nuova statistica sul quartiere
- Un nuovo nodo entra nella rete
- Un nodo esce dalla rete

Quando l'applicazione Client Analista riceve una notifica push, deve semplicemente stamparla su schermo. La soluzione proposta deve evitare il polling da parte degli analisti.

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 nodi. Si gestisca quindi il caso in cui un nodo vada in 'crash' senza preavviso. In particolare, è necessario gestire correttamente i nuovi casi limite che si possono verificare, tra i quali la possibile perdita del token.

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.