Esercitazione di Sistemi Distribuiti e Pervasivi

Suggerimenti di Sviluppo e Progettazione II

Gabriele Civitarese
gabriele.civitarese@unimi.it

EveryWare Lab
Università degli Studi di Milano
-
Docente: Claudio Bettini

Disclaimer

- Il contenuto di questi lucidi è da considerarsi puramente indicativo
  - Verranno forniti dei suggerimenti di sviluppo e di progettazione
  - Ogni studente può effettuare scelte differenti
- Le direttive su come svolgere il progetto sono riportate nel testo del progetto presente sul sito del corso.

La Rete di Case

- A tutti gli effetti è una rete P2P
  - Ogni singola applicazione *Casa* sarà sia client che server
- Ogni casa deve essere simulata da un singolo **processo** (miraccomando, non un thread!)
- Ogni casa legge con una certa frequenza il consumo medio di corrente dallo smart meter
- Le case si sincronizzano tra di loro per il calcolo di statistiche globali e per l'uso di corrente
  - La comunicazione tra nodi avviene direttamente tramite socket o grpc (a vostra scelta)
  - La comunicazione con il *server amministratore* avviene tramite librerie per chiamate REST
  - Adottate SEMPRE formati di dati standard (XML, JSON, Protocol Buffer,...)

Gestione messaggi

- All'arrivo di un qualsiasi messaggio, una casa deve esaminarlo
- Sviluppando solo server multithread, la ricezione dei messaggi è concorrente
- Un buffer condiviso di messaggi e un pool di thread per gestirli possono essere una buona soluzione
- Una buona stesura del protocollo di comunicazione può semplificare il codice che esamina i messaggi in ingresso
  - esempio di formato messaggi:
    `header content [parameters] timestamp`
- Può essere conveniente considerare i diversi *stati* dell'applicazione durante l'interpretazione dei messaggi

Dividere la progettazione della casa in stati che cambiano con il tempo

Entrata/uscita

- All'inizio del sistema il condominio è vuoto e il sever amministratore è pronto a ricevere richieste di inserimento
- Ogni volta che facciamo partire una casa, automaticamente tenta di entrare nel condominio
- Se la registrazione va a buon fine, la casa riceve la lista delle altre case nel condominio
  - Deve quindi comunicare con le altre case per entrare nella rete
- Il protocollo di entrata nella rete dipende dalla gestione della topologia
  - Come strutturiamo la rete? (rete ad anello, messaggi broadcast ...)
  - Tutti i nodi devono conoscersi tra di loro?
- Una *casa* può esplicitamente uscire dal condominio (la casa esce solo in modo esplicito)
- In questo caso la rete deve dinamicamente adattarsi al cambiamento e restare consistente

Problemi da considerare

Casi limite

- Cosa succede quando due case entrano contemporaneamente?
- Cosa succede quando una casa entra nella rete mentre qualche altra casa esce?
- Cosa succede quando una casa entra/esce durante un algoritmo di sincronizzazione distribuita?
- ...

Risulta fondamentale considerare tutti i possibili casi limite! In fase di discussione del progetto vi verranno **sicuramente** richiesti. Testateli con le *sleep()*.

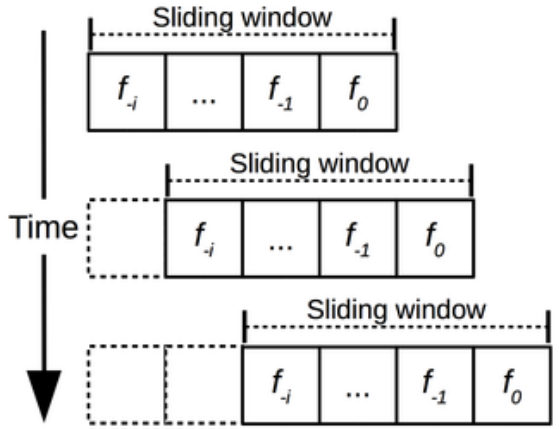
Misurazioni

- Ogni misurazione è caratterizzata da
  - Id sensore
  - Tipologia di sensore (non serve a niente)
  - Valore letto
  - Timestamp
- Ogni smart meter riversa le misurazioni in un buffer
  - Usato per calcolare periodicamente le statistiche che vengono inviate al coordinatore
  - Problemi di sincronizzazione?

Sliding window

- Tecnica standard per analisi o compressione real-time di stream di sensori
- Consiste in calcolare statistiche considerando finestre temporali di dimensione w
  - Ogni volta che vengono accumulate w misurazioni, calcoliamo le statistiche
- Utilizzo di una percentuale di overlap o per catturare le transizioni tra finestre temporali
  - La finestra temporale w_i ha una percentuale o delle misurazioni più recenti della finestra w_{i-1}

Calcolo una statistica ogni 24 misurazioni. La statistica successiva contiene il 50% delle misurazioni precedenti.



Statistiche locali e globali

Calcolo statistiche

- Ogni *casa* calcola periodicamente statistiche locali (ovvero media su sliding window)
- Ogni statistica locale deve essere condivisa con le altre case
  - Come scambiamo i messaggi? Dipende dall'architettura scelta!
- Come coordinarsi per comunicare la statistica globale al server amministratore?
  - Algoritmo di mutua esclusione distribuita?
  - Elezione di un coordinatore?

quando una casa calcola la statistica globale chiede il lock per inviare la statistica (complesso)

- Il server amministratore deve quindi conoscere statistiche locali e globali
- Risulta fondamentale capire come assegnare timestamp (o intervalli temporali?) alle statistiche

Uso di corrente extra

Non c'è una vera coda per le case in attesa di corrente extra, perché non c'è una casa che memorizza la struttura dati. Probabilmente vanno usati wait e notify.

Mutua esclusione distribuita

- Solo due case del condomino contemporaneamente possono chiamare il metodo *boost*
- La mutua esclusione distribuita serve solo per ottenere il consenso
- Protocollo ad altissimo livello:
  - Casa richiede di poter utilizzare boost
  - Le altre danno l'ok solo se non ci sono già due case che l'hanno richiesto. Altrimenti la casa richiedente "aspetta"
  - Una volta che una casa riceve l'ok dal condominio, può chiamare *boost*
  - Alla fine del metodo, il condominio deve essere avvisato sulla conclusione dell'uso extra da parte della casa
- La scelta dell'algoritmo di mutua esclusione da usare dipende dalla topologia della rete!

Pattern Singleton

- In diversi problemi di sincronizzazione è necessario un oggetto di coordinazione
- In un sistema con un dispatcher e molti thread, il problema è riferirsi alla stessa istanza
- Possibilità:
  - Il dispatcher crea l'oggetto.
  - I working thread ricevono i riferimenti degli oggetti condivisi tramite il costruttore
  - Problema: all'aumentare del numero di riferimenti la gestione degli oggetti condivisi si complica.
- Soluzione:
  - Ogni thread si "prende" da solo l'oggetto.
  - Uso del pattern Singleton per accedere all'unica istanza di una determinata classe

Pattern Singleton (2)

```
public class Singleton {
        private static Singleton instance = null;

        public synchronized static Singleton getInstance() {

                if (instance==null)
                        instance = new Singleton();

                return instance;
        }


        //Il costruttore private impedisce la creazione di istanze da parte di classi esterne
        private Singleton() {/*...some code...*/}


        ...
}
```

Buon lavoro!