LABORATORIO DI INGEGNERIA DEI SISTEMI SOFTWARE

Introduction

Un caso di studio con cui iniziare ad affrontare l'analisi e la progettazione di sistemi distribuiti è un sistema formato da a N Produttori che inviano informazione a 1 Consumatore.

Requirements

Costruire un sistema software distribuito costituio da N (con N >= 1) Produttori che inviano informazione a 1 Consumatore, il quale deve elaborare tale informazione.
a dislocazione dei componenti sui nodi di elaborazione può essere:

- OneNode: tutti i componenti operano nello stesso nodo;
- TwoNodes: gli N Produttori operano in uno stesso nodo, mentre il Consumatore opera in un diverso nodo;
- ManyNodes: il Consumatore opera in suo proprio nodo, mentre i Produttori operano su K nodi diversi (con K maggiore di 1, e al più uguale a N).

Requirement analysis

- Produttore e Consumatore rappresentano gli unici due tipi di entità interagenti necessari all'interno del sistema.
- I requisiti non stabiliscono come l'informazione debba essere trasmessa e ricevuta.
- Le modalità di scambio di informazioni fra Produttori e Consumatore non sono stabilite: in particolare, non si precisa se il Produttore debba ricevere una risposta dal Consumatore.

Problem analysis

Architettura logica

Stando all'analisi dei requisiti, l'architettura dovrà necessariamente prevedere una modalità di invio di messaggi dai Produttori al Consumatore. Sulla base di questo, l'architettura logica del sistema può essere organizzata in diversi modi, ad esempio:

- **Modello Client-Server**: l'architettura prevede diversi client che effettuano richieste verso un server. In questo caso, i client sono rappresentati dai Produttori, mentre il server è dato dal Consumatore. Stando ai requisiti, l'interazione può essere asincrona non bloccante (le richieste dei client possono essere "fire & forget"). In questo modello, i Produttori devono conoscere il Consumatore, mentre non è vero il viceversa.
- **Publisher-Subscriber (Pub/Sub)**: In questo caso, è presente un gestore della comunicazione (broker), a cui i publisher (i Produttori) e il subscriber (il Consumatore) sono iscritti; quest'ultimo riceverà i messaggi dal broker, che li avrà ricevuti precedentemente dai publisher. In questo modello, i Produttori e il Consumatore conoscono il broker.
- **Modello ad Eventi**: qui, i Produttori sono visti come puri emettitori di eventi; l'informazione viene emessa, e il Consumatore potrà ascoltarla. In questo modello, vi è totale disaccoppiamento delle parti.

In ogni caso, per comunicare, le due parti dovranno concordare un coerente formato dei messaggi.

Ulteriori considerazioni

Come già specificato, è necessario che si preveda una modalità di invio delle richieste dai Produttori al Consumatore, mentre non è necessario il viceversa. Qualora non si intenda sviluppare una modalità di risposta al Produttore (Fire & Forget), quest'ultimo non avrebbe modo di verificare la corretta ricezione della richiesta.

Test plans

I piani di test prevedono la verifica di una richiesta, proveniente da uno dei nodi Produttore per il nodo Consumatore, e sarà necessario verificare:

- il corretto invio dei messaggi da parte dei Produttori.

- la corretta ricezione dei messaggi da parte del Consumatore.
- il corretto formato dei messaggi.

Qualora si intenda sviluppare una modalità di risposta al Produttore, diventa necessario prevedere le analoghe verifiche anche per tale messaggio.

Project

La scelta di progettazione ricade sul modello Client-Server visto nella fase di Analisi, implementato sul protocollo TCP. Come specificato nella fase precedente, l'architettura prevede i Produttori come clienti e il Consumatore come server; inoltre, i client e il server non risiedono necessariamente sullo stesso nodo. Sulla base di queste informazioni:

- I Produttori devono conoscere il Consumatore per potergli inviare una richiesta: ciascuno dei Produttori dovrà quindi conoscere indirizzo IP e porta su cui il Consumatore riceve le richieste.
- Si sceglie di prevedere due modalità di invio delle richieste: Request e Forward.
  - Per un messaggio Request, il Produttore resterà in attesa di una risposta (Sincrono) da parte del Consumatore: si prevede pertanto l'invio di un messaggio di ACK (acknowledgement) dal Consumatore per comunicare la ricezione della richiesta al Produttore.
  - Per un messaggio Forward, il Produttore non si aspetta una risposta (Asincrono), e non è prevista quindi nessuna ulteriore gestione della comunicazione in tale scenario (Fire & Forget).
- Il Consumatore gestirà le richieste in modo First In First Out (FIFO): sarà necessario prevedere una coda in cui inserire le richieste da gestire.
- Il formato dei messaggi scambiati sarà quello espresso e definito nella interfaccia IApplMessage.

Per abbattere il costo di progetto, la progettazione avviene sulla base delle librerie *unibo.basicomm23* già fornite. Attraverso di esse:

- realizzeremo il Produttore come un client abilitato alla Trasmissione di messaggi: troviamo già implementata la logica per l'instaurazione della connessione con il Server, attraverso la classe *ActorNaiveCaller*; ci limiteremo quindi a scrivere la *business logic* della gestione dei messaggi.

- realizzeremo il Consumatore come un server abilitato alla Ricezione di messaggi: anche in questo caso, la logica per l'instaurazione della connessione bidirezionale è già implementata; la classe ServerFactory si occuperà di restituire un opportuno oggetto Server in grado di gestire la connessione. Ci limiteremo a implementare la gestione dei messaggi a livello applicativo, basandoci sull'interfaccia *IApplMsgHandler*.

Autovalutazione

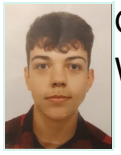
Voto: B
L'architettura e la progettazione inviati erano sostanzialmente corrispondenti con quelli presentati in aula durante la scorsa lezione (versione con Interaction). Non avevo previsto Unit Test (visti in JUnit) per la verifica del formato dei messaggi, né il logging di dispatch, ma unicamente stampe a video.

Testing

Deployment

Maintenance

By Tommaso Sgreccia - Email Istituzionale: tommaso.sgreccia@studio.unibo.it



GIT repo: https://github.com/WheelyMcBones/ISS2024