Progetto RE.VE.N.GE. DDS con Fault-tolerance del Sistema di Consegna

Progetto di: Marco Livini, Luca Nardelli, Christian Pinto

Reti di Calcolatori LS 08-09

Prof. Antonio Corradi, Ing. Luca Foschini

Relazione di Christian Pinto

1. Introduzione

Obiettivo di questo progetto è la realizzazione di un sistema di supporto per la distribuzione di notizie su larga scala. Il sistema deve essere in grado di gestire fonti e fruitori disaccoppiandoli sia in spazio che in tempo, nonché di garantire diversi livelli di qualità di servizio sia per le fonti che per i fruitori. Il sistema, inoltre, dato il forte sviluppo di internet nei dispositivi mobili sarà dotato di strumenti per consentire la fruizione delle notizie da parte di questi ultimi, affrontando problemi come la discontinuità della connessione. Il sistema è composto di tre parti principali: *Fonti*, *Fruitori* e *Sistema di Consegna*. Per i fruitori è possibile definire diversi livelli di qualità del servizio (QoS) come: affidabilità delle consegne, tempi minimi e massimi di consegna delle notizie, argomenti di interesse. Anche per le fonti, come per i fruitori, è possibile definire il livello di qualità del servizio inteso come: attendibilità delle notizie, diversa priorità delle fonti dovuta al costo delle notizie, affidabilità delle consegne. Il rispetto dei vincoli di QoS è garantito dal sistema di consegne il quale permette di ottenere disaccoppiamento spazio-temporale tra fonti e fruitori eliminando, quindi, la necessità di conoscenza reciproca e garantisce un elevata disponibilità del servizio servendosi della replicazione. Per la realizzazione del sistema di consegne si richiede l'utilizzo del Message Oriented Middleware (MOM) basato su specifiche OMG *Data Distribution Service*(DDS). In particolare è stata adottata l'implementazione RTI-DDS di Real Time Innovations. Dopo una breve introduzione del middleware, RTI-DDS, e

dell'architettura generale del Sistema passeremo a considerare le tematiche relative alla gestione della *Qualità del Servizio* ed alla gestione dei fruitori mobili. Infine verranno presentati alcuni risultati sperimentali.

2. OMG DDS

Lo standard OMG Data Distribution Service, ed in particolare l'implementazione RTI-DDS, si basa sul modello *DataCentric Publish Subscribe*(DCPS), un modello di comunicazione per applicazioni distribuite in ambienti peer-to-peer. Grazie all'utilizzo questo middleware è possibile minimizzare le interazioni tra le entità che intendono comunicare(publisher, subscriber) disaccoppiandole sia in spazio che in tempo. La gestione della comunicazione è completamente a carico del middleware, quindi eventuali ritrasmissioni o problematiche di localizzazione delle entità partecipanti sono assolutamente trasparenti all'utente(programmatore). Questo tipo di middleware permette la configurazione di diversi parametri di QoS sia lato publisher che lato subscriber che vanno a definire un contratto tra le due entità, contratto che viene stipulato solo se le politiche applicate dalle due entità sono compatibili tra di loro. Prima di partire con lo sviluppo dell'applicazione è stato necessario uno studio preliminare di questo strumento, in modo da individuarne le caratteristiche offerte che nel miglior modo si adattano alla nostra applicazione.

3. Architettura Generale

Il sistema è diviso in tre entità:

- Fonte(Source)
- Fruitore(Sink)
- Sistema di consegna

Il fulcro del sistema è il *Sistema di Consegna* che si occupa dello smistamento del traffico dalle fonti ai fruitori, della gestione della QoS e del bilanciamento del carico. Come da specifica per la realizzazione del Sistema di Consegna è stato utilizzato il middleware RTI DDS.
Volendo garantire *fault-tolerance* ed *high-availability* del servizio si è scelto di replicare il Sistema di Consegna con copie attive. L'utilizzo di copie attive per la replicazione permette di utilizzare alcune policy dello standard DDS per:

- Sincronizzazione copie
- Bilanciamento del carico

In particolare il bilanciamento del carico è realizzato utilizzando la policy PARTITION dello standard DDS. Ad ogni copia del sistema di consegna viene assegnata la gestione di una diversa partizione; le partizioni permettono una suddivisione logica dei fruitori, definendo così il sottoinsieme di fruitori che ogni copia del sistema dovrà servire(Fig. 1).

Utilizzando questo schema ad ogni fruitore, all'atto della connessione, viene comunicata la partizione su cui mettersi in ascolto.

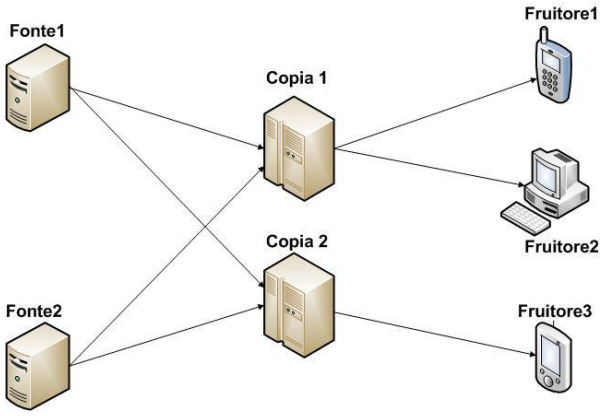

Figura 1

Infine per ottenere scalabilità più elevata è stata inserita un ulteriore entità, il relay, che permette di realizzare una suddivisione in domini. Supponendo di avere relay per ogni dominio un fruitore può connettersi al dominio al lui geograficamente più vicino. Il relay non è altro che un passacarte tra i domini, permette che una news pubblicata su un dominio sia fruibile da qualsiasi fruitore a prescindere dal dominio di sottoscrizione.
Questo schema architetturale si presta bene per un fruitore mobile, il quale spostandosi da un dominio all'altro continua a ricevere le notizie per cui manifesta interesse. Per maggiorni informazioni riguardo l'architettura del sistema e la replicazione consultare le relazioni di Luca Nardelli e Marco Livini.

4. Gestione QoS

Nella realizzazione di un applicazione distribuita prende molta importanza la gestione della *Qualità Del Servizio*(QoS), dove con QoS si intende l'insieme di vincoli in grado di modificare il flusso di informazioni che viaggiano da e verso il sistema per adattarsi meglio alle richieste o alle caratteristiche tecnologiche delle entità in gioco. Il servizio realizzato è volutamente *best-effort* in quanto le informazioni in transito all'interno del sistema hanno validità limitata ed eventuali ritrasmissioni, dovute ad un servizio *reliable*, potrebbero portare alla ricezione di notizie non più valide. Inoltre un servizio realiable prevede l'invio di un segnale di ACK per ogni news ricevuta, aumentando notevolmente il traffico in rete. Nonostante ciò è molto importante poter definire diversi livelli di qualità del servizio, dovuti ad esempio a costi di abbonamento o ad esigenze relative al numero massimo di notizie che il fruitore è in grado di elaborare nell'unità di tempo(evitando che un fruitore lento venga inondato di notizie). Particolare attenzione è da prestare ai fruitori mobili i quali solitamente, utilizzando una connessione wireless, non sono sempre presenti e necessitano di un entità che mantenga per loro le notizie fino al momento della riconnessione.
Infine volendo mettersi dalla parte del sistema di consegna possiamo considerare come qualità del servizio anche l'utilizzo

della rete, cercando di garantire un utilizzo efficiente della banda introducendo il minor overhead possibile.
Il middleware RTI-DDS offre svariati strumenti che permettono di gestire la qualità del servizio a diversi livelli di astrazione. Passiamo quindi ad una breve descrizione di quelli che sono i meccanismi utilizzati da RTI-DDS per la gestione della QoS e continuiamo poi con la descrizione delle soluzioni adottate per le diverse entità del sistema.

4.1 Gestione QoS: RTI-DDS

RTI-DDS permette di impostare svariate politiche a differenti livelli, quali:

- Topic: permette di partizionare lo spazio delle news.
- Participant: entità principale, permette di creare altre entità (publisher/subrciber, data reader/writer).
- Publisher/Subsriber: entità che si occupa effettivamente di inviare/ricevere i dati.
- Data reader/writer: endpoint di comunicazione.

L'applicazione di politiche su una di queste entità comporta l'applicazione delle stesse pure sulle entità sottostanti. Se ad esempio consideriamo un Subscriber al quale sono associati un gruppo di Data reader, l'applicazione di una qualsiasi politica al Subscriber si riflette su tutti i Data reader ad esso associati, purchè la politica scelta sia applicabile anche sui Data reader. Come già detto, all'atto della connessione, i terminali di comunicazione di questa architettura stipulano un contratto che permette loro la comunicazione solo ed esclusivamente se le politiche scelte da entrambe le entità sono compatibili tra di loro.
Passiamo adesso ad una descrizione di quelle che sono le politiche utilizzate nell'ambito di questa applicazione e soprattutto quali sono le politiche utilizzate per la gestione della qualità del servizio:

- RELIABILITY: permette di stabilire se la consegna delle news avviene in modo affidabile o meno. Nel caso di RELIABILITY affidabile(reliable) verrà inviato un messaggio di ACK per ogni news ricevuta, la mancata ricezione dell'ACK lato publisher provoca ritrasmissione.
- DURABILITY: definisce il livello di persistenza dei messaggi dopo l'invio, molto utile nel caso di Data writer che pubblicano dati prima che il Data reader sia creato. Viene utilizzata unitamente alla politica HISTORY.
- HISTORY: definisce il numero di news memorizzate localmente a Data writer/reader. Insieme a DURABILITY permette di stabilire

quanti messaggi “vecchi” vengono ricevuti dal Data reader all’atto della connessione.

- DEADLINE: definisce per il Data writer il periodo massimo all’interno del quale devono essere scritti i dati. Analogamente per il Data reader definisce il massimo periodo all’interno del quale ci si aspetta di ricevere dei dati.
- TIME_BASED_FILTER: permette di definire il periodo minimo di ricezione dei dati, una sola news all’interno del periodo. Deve avere un valore minore della DEADLINE.
- TRANSPORT_PRIORITY: permette di definire priorità sui Data writer. I dati scritti da un Data writer prioritario vengono trattati per primi.
- BATCH: permette di inviare messagi in Batch quando la dimensione del messaggio stesso è minore di una certo valore fissato. Garantisce un’utilizzo più efficiente della banda nel caso di messaggi piccoli.

4.1 Gestione QoS: Fonti

Per quanto riguarda le fonti le specifiche richiedono la possibilità di definire l’affidabilità delle consegne ed una priorità che la distingue dalle altre fonti.
Per l’affidabilità della consegna delle notizie si può scegliere una consegna *best-effort* oppure una consegna *reliable* mediante l’utilizzo della politica di DDS RELIABILITY(applicandola a livello di Data writer), ricordando che una RELIABILITY di tipo reliable comporta l’invio di un ACK per ogni news ricevuta. Dal lato del sistema avremo entrambi i tipi di Data reader per permettere al sistema stesso di ricevere notizie da tutti i fruitori indipendentemente dal livello di RELIABILITY scelto senza effettuare alcun tipo di registrazione presso il sistema.
Riguardo la priorità delle fonti è possibile utilizzare la politica TRANSPORT_PRIORITY con la quale i dati inviati da diversi Data writer assumono priorità diverse. Ipotizzando di definire tre livelli di priorità: Gold, Silver, Basic è possibile assegnare un diverso valore di priorità ad ogni livello ottenendo così la diversificazione in base all’abbonamento sottoscritto o al costo delle notizie.

4.2 Gestione QoS: Fruitori

Per i fruitori è necessario fare qualche considerazione in più in quanto, oltre all’affidabilità delle consegne, bisogna pure considerare numero minimo e massimo di news da ricevere nell’unità di tempo e la possibilità che il fruitore si connetta al sistema in un momento successivo rispetto a quello di connessione

della fonte. Dobbiamo quindi considerare la possibilità di inviare, qualora richiesto, alcune news precedenti alla connessione del fruitore.
Le considerazioni fatte per l'affidabilità delle consegne sono analoghe a quelle fatte per le fonti, quindi si utilizza anche in questo la politica RELIABILITY.
I fruitori utilizzano pure dei messaggi di segnalazione utilizzati per sottoscrivere/lasciare un sistema o per, come vedremo dopo, chiedere eventuali ritrasmissioni. Per questo tipo di messaggi abbiamo scelto una politica relativa all'affidabilità delle consegne di tipo reliable in modo che nessuno di questi segnali venga perso.
Per monitorare numero massimo e minimo di pacchetti ricevuti nell'unità di tempo si è scelto di utilizzare le politiche DEADLINE e TIME_BASED_FILTER. La politica DEADLINE permette di impostare il massimo periodo all'interno del quale ci si aspetta di ricevere i dati e nel caso in cui la deadline risulti violata viene scatenato un particolare evento, *on_requested_deadline_miss*, che va ad invocare un listener dal quale è possibile notificare, ad esempio, l'avvenimento in modo da poter prendere le relative contromisure. Per il numero massimo di news nell'unità di tempo viene in nostro aiuto la politica TIME_BASED_FILTER la quale notifica, lato subscriber, la ricezione di più di una news all'interno dell'intervallo specificato. Anche in questo caso una violazione scatena un evento, *on_sample_lost*, al verificarsi del quale si può agire opportunamente, ad esempio chiedendo la ritrasmissione delle news dato che una violazione di questa politica porta a scartare tutte le news ricevute all'interno dello stesso intervallo. La Figura 2 aiuta meglio a capire il funzionamento congiunto di queste due politiche.

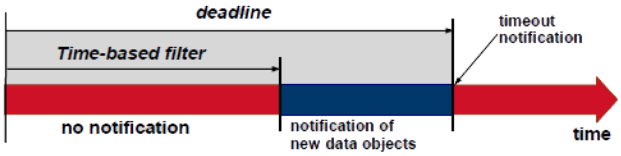

Figura 2

Così facendo, preso il secondo come unità di tempo, il fruitore si aspetta di ricevere un minimo di 1sec/deadline news ed un massimo di 1sec /time_based_filter news.
La scelta è caduta su una gestione lato fruitore in quanto volendo trasferire il controllo di questi tempi sul sistema di consegna saremmo andati incontro ad una architettura client-server, avendo dovuto prevedere un canale per ogni singolo fruitore andando così a snaturare il modello publisher-subscriber. Nonostante ciò il carico introdotto sul fruitore per la gestione di questi tempi è minimo, poiché contenuto nel carico del middleware.
Per terminare questa trattazione ci resta da considerare fruitori di tipo "late joiner", cioè quei fruitori che si connettono al sistema in un momento successivo rispetto a quello di connessione del/dei publisher che pubblicano notizie di interesse per il

fruitore stesso. RTI-DDS permette, attraverso l'utilizzo di alcune politiche quali: DURABILITY ed HISTORY, la ricezione dei vecchi messaggi, in numero pari alla profondità impostata per HISTORY(il cui parametro kind va impostato a KEEP_LAST), settando la DURABILITY a TRANSIENT_LOCAL (i messaggi vengono inviati fin quando è presente il relativo Data writer). Questa tecnica, richiedendo una RELIABILITY *reliable*, è in contrasto con il nostro sistema che come scelta progettuale predilige consegne *best-effort*, mentre considera come scelta opzionale la possibilità di consegne *reliable.* E' stato perciò necessario implementare un modulo che permette di ricevere le vecchie notizie mediante l'invio di un messaggio di segnalazione (interazione di tipo pull) da parte del fruitore. Le notizie così richieste vengono inviate in forma esclusiva al solo fruitore richiedente.

4.2.1 Gestione QoS: Fruitori mobili

Le applicazioni distribuite cominciano ad affacciarsi anche nel mondo mobile grazie all'enorme sviluppo di dispositivi portatili quali PDA, telefoni cellulari dotati di elevata potenza di calcolo nonché di connessione wireless. Dispositivi del genere hanno solitamente connessioni sporadiche ed effettuano accesso al sistema saltuariamente da punti geograficamente diversi. Il sistema deve dare supporto a questi dispositivi mantenendo per essi i messaggi nei momenti di disconnessione e prevedendo la possibilità di roaming.
Grazie all'inserimento dell'entità Relay il roaming da un dominio ad un altro risulta molto semplice da realizzare, in quanto è sufficiente chiedere una de-registrazione(LEAVE) dal dominio corrente per poi registrarsi(JOIN) presso il dominio, attualmente, geograficamente più vicino.
Il mantenimento dei messaggi nei periodi di disconnessione è realizzato grazie alla politica HISTORY il cui parametro kind va settato a KEEP_ALL(vengono mantenuti tutti i messaggi), così facendo all'atto della connessione il fruitore mobile riceve tutti i messaggi dal momento della sua disconnessione.

4.2 Gestione QoS: Rete

Oltre che limitarci all'osservazione della qualità del servizio ai morsetti del sistema, cioè lato Fonte e Fruitore, è stato interessante considerare anche gli aspetti di rete dell'applicazione, specialmente per quanto riguarda sfruttamento efficiente della banda e comunicazioni unicast.
Per sfruttare in modo efficiente la banda è bastato utilizzare un ulteriore politica di RTI-DDS, la politica BATCH. Questa politica permette di differire l'invio di una news per inviarla in batch insieme ad altre. Le notizie vengono inviate in batch, di

dimensione massima 64 KB , solo se la loro dimensione è minore di un certo valore prefissato, in questo modo riusciamo a diminuire il traffico in rete a parità di notizie inviate e nel caso di notizie di piccole dimensioni.
Molto più interessante è stata l'analisi per comunicazioni di tipo unicast. RTI-DDS è pensato per ambienti "chiusi", reti dedicate nelle quali solitamente è possibile effettuare comunicazioni multicast. La nostra applicazione è pensata per servire fruitori che si connettono al sistema dalla rete internet nella quale il multicast è difficile da realizzare. Supponiamo di trovarci in una situazione in cui sia possibile comunicare con multicast, il numero di news(pacchetti) inviate in rete è indipendente dal numero fruitori connessi. E' quindi molto più semplice inviare una notizia a tutti i fruitori a prescindere dal suo contenuto o dalla partizione di ascolto e delegare questi ultimi del filtraggio delle news a loro dirette, anziché caricare il sistema del filtraggio delle news prima dell'invio. Se spostassimo questo schema di funzionamento, chiamato filtraggio *Consumer Based*, in una situazione in cui la maggior parte dei fruitori può comunicare solo in modalità unicast andremmo ad introdurre un elevato overhead di comunicazione proporzionale al numero di fruitori e dovuto all'invio di ogni notizia un numero di volte pari al numero di fruitori stessi. Nella sua ultima implementazione RTI-DDS ci viene incontro, anche se non in modo eccessivamente efficiente, considerando la possibilità di filtraggio *Producer Based* delle news inviate, registrando(a carico del fruitore) un *Content Filter*. Il Content Filter è un "Oggetto" che permette al middleware di filtrare i messaggi di interesse per il subscriber che lo ha registrato. Sarà poi il middleware a discriminare fruitori unicast da fruitori multicast e ad attivare la politica di filtraggio opportuna.

5. Test

L'applicazione è stata testata per valutare occupazione di banda, carico del processore, ed occupazione di memoria ottenendo risultati soddisfacenti pur non essendo riusciti a raggiungere i limiti reali dell'applicazione. In questa relazione verranno presentati solo i risultati strettamente legati alla relazione stessa, cioè qualche considerazione sui tempi medi di roaming tra due domini per un fruitore mobile. Gli altri test sono rimandati alle relazioni di Marco Livini e Luca Nardelli.
Nella simulazione del roaming un fruitore connesso ad un dominio effettua uno spostamento ad un altro a lui più vicino. Inizialmente il fruitore invia un messaggio di *LEAVE* al sistema presso il quale è connesso ed effettua una join sul nuovo dominio. Il tempo di roaming medio registrato, considerato a partire da quando il fruitore non riceve più news dal dominio

corrente a quando comincia a ricevere news dal nuovo dominio, è di 10 secondi circa. Il numero di news perse è minimo in quanto il fruitore al momento della connessione sul nuovo dominio riceve le news prodotte prima della sua sottoscrizione.

6. Conclusioni

Nella realizzazione di questo sistema il middleware RTI-DDS, anche se non utilizzato per le applicazioni per cui è stato penstato, è stato indispensabile in quanto permette, con molta facilità, la configurazione di diversi parametri di QoS e facilita la comunicazione lato publisher/subscriber grazie ad un meccanismo completamente automatico di discovery delle entità. Nonostante ciò il suo funzionamento black-box non permette di avere feedback completi sullo stato delle entità e su eventuali errori ottenuti a runtime.
L'applicazione realizzata presenta una buona scalabilità soprattutto in presenza di relay garantendo la minima intrusione, in termini di cpu e memoria, nei sistemi sui cui è stata installata.