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

Luca Nardelli - luca.nardelli2@studio.unibo.it

Gruppo: Luca Nardelli, Marco Livini, Christian Pinto

Abstract

Uno dei fattori chiave che ha spinto fin dal principio lo sviluppo di sistemi distribuiti e poi di Internet è stata la necessità di condividere informazioni su larga scala. L'estrema varietà di dispositivi e delle connessioni oggi disponibili ha contribuito a marcare ancora di più la necessità di disporre di servizi che siano, oltre che sempre disponibili, "adattabili" alle diverse esigenze degli utenti. Lo scopo di questo lavoro è stato quello di sviluppare un sistema middleware di supporto per la distribuzione di notizie su larga scala, da parte di agenzie di stampa, che sia sempre disponibile, supporti diversi livelli di qualità di servizio e diverse tipologie di interazione, in modo da permettere anche a dispositivi molto leggeri (es. cellulari) di accedere al servizio. In particolare il middleware proposto è stato implementato mediante lo standard OMG DDS (Data Distribution Service). Gli esperimenti mostrano che anche in presenza di un elevato numero di fonti/fruitori, si riesce a fornire un servizio fault-tolerant che dispone di buoni livelli di scalabilità.

1. Introduzione

Distribuire notizie su vasta scala prescrive in modo implicito o esplicito una serie di requisiti. In primo luogo, il sistema di distribuzione deve essere sempre disponibile e pronto a far fronte a diversi livelli di carico. Ci si può aspettare, infatti, che la mole di notizie che il sistema dovrà smistare potrà aumentare di molto rispetto al carico medio in occasione di eventi particolari (es. giochi olimpici, eventi di cronaca, etc). In secondo luogo può risultare non realistico fare particolari assunzioni sui dispositivi che possano accedere al sistema. In un tipico scenario supponiamo che fonti e fruitori siano altamente eterogenei e che di conseguenza richiedano forme di servizio spesso differenti. Ad esempio un cellulare o PDA, rispetto ad un terminale fisso, può subire interruzioni di connessione durante la sua operatività, ma può non essere disposto a perdere delle notizie. Ultimo, ma non meno importante requisito, è l'aspetto legato alla qualità di servizio ed al contratto fra fonti/fruitori e sistema. In particolare si suppone che non tutte le notizie siano importanti in egual modo, ma che possano esserci notizie più "prioritarie" di altre. Allo stesso modo si suppone che fonti/fruitori possano richiedere al sistema diversi livelli di qualità di servizio in base a diversi criteri, ad esempio in base al costo dell'abbonamento al servizio.

Di tutte le problematiche sopra esposte in questa relazione sarà enfatizzato maggiormente l'aspetto legato all'interazione tra fonti/fruitori e sistema di consegna ed in particolare si descriverà come il sistema di consegna riesca a bilanciare il carico di notizie tra i fruitori, lo schema dei protocolli di accesso al sistema ed i criteri che portano l'architettura del sistema ad essere intrinsecamente fault-tolerant ed al tempo stesso molto scalabile ed efficiente.

Il sistema è stato implementato mediante un implementazione commerciale di DDS fornita da RTI (Real-Time Innovations). Tale scelta, seppur abbia, in parte, vincolato le scelte progettuali, ha mostrato di essere molto flessibile e di garantire ottime prestazioni. La soluzione proposta, al fine di disaccoppiare il più possibile fonti e fruitori, si discosta un po' dal classico modello di interazione peer-to-peer di DDS. Il middleware prevede, infatti, la presenza di un sistema centrale a cui fonti e fruitori fanno riferimento. Tale sistema è formato da una serie di copie attive che si bilanciano il carico e si auto-riconfigurano dinamicamente in caso di guasto.

Il resto della relazione è strutturato come segue. Il capitolo 2 descrive brevemente lo standard DDS. Il capitolo 3 descrive l'architettura base del sistema e le scelte progettuali relative al sistema centrale. Il capitolo 4 estende l'architettura precedente in ottica di aumentare la scalabilità e l'efficienza globale del sistema. Il capitolo 5 mostra i risultati sperimentali più significativi del lavoro.

2. Data Distribution Service (DDS): Standard ed implementazione RTI

Il Data Distribution Service for Real Time System è uno standard promosso dall'Object Management Group (OMG) che definisce un middleware per la distribuzione di dati in tempo reale secondo il paradigma publish/subscrive. I punti di forza di tale standard sono l'accoppiamento lasco tra le entità, l'architettura flessibile ed adattabile, l'efficienza della comunicazione, la qualità di servizio altamente parametrizzabile e la scalabilità elevata. RTI-DDS è un implementazione multi-piattaforma / multi-linguaggio abbastanza completa dello standard DDS .

Uno dei principali punti di forza dello standard DDS è la comunicazione diretta tra fonti e fruitori in un modello peer-to-peer. Esso utilizza preferibilmente interazioni di tipo multicast molto efficiente fra le entità. Nel sistema che abbiamo sviluppato, tuttavia, è stata rilasciata l'assunzione che siano sempre disponibili interazioni di tipo multicast. Al fine di disaccoppiare le varie entità anche dal punto di vista della qualità di servizio si è scelto di inserire tra fonti e fruitori una terza entità di seguito riferita come sistema centrale. Solo per lo sviluppo del sistema centrale si è assunto la disponibilità di connessioni multicast.

Lo standard DDS supporta una serie molto ampia di accordi tra le entità circa la qualità di servizio che noi, nella nostra implementazione, abbiamo sfruttato. In particolare l'architettura supporta diversi tipi di comunicazione più o meno affidabili fra le entità ed un protocollo di negoziazione dei parametri inerenti alla qualità di servizio ben definito. Per molti di questi aspetti e per aspetti che riguardano la mobilità fra le entità si rimanda alla relazione di Pinto.

Il modello publish/subscrive di DDS è più precisamente un modello topic-based. Le informazioni sono associate a dei topic che dovranno sottoscrivere coloro che sono interessati a riceverle. Per definire il tipo di dato associato ai vari topic DDS prevede l'uso di un Interface Definition Language (IDL) standard. DDS supporta a livello di standard diversi strumenti per sottoscrivere e filtrare in modo molto granulare ed efficiente i topic. Tali strumenti sono stati, in larga parte, forniti dall'implementazione RTI e da noi utilizzati per sgravare la parte applicativa del sistema centrale di tutte le informazioni relative alle fonti/fruitori.

Per quanto riguarda il partizionamento del carico DDS supporta sia il concetto di dominio come area isolata e sia la possibilità di partizionare un dominio in sottoaree. Il sistema centrale è stato basato in modo forte sul concetto di partizioni per bilanciare il carico fra le entità e in aggiunta, per aumentare la scalabilità, si è esteso il sistema affinché supporti un architettura multi-dominio. Le politiche di partizionamento del carico, tuttavia, non sono state implementate in RTI in modo specifico per ambienti non multicast e questo ci ha costretti a riconsiderare in modo critico alcune delle scelte progettuali seppur pienamente supportate a livello di standard.

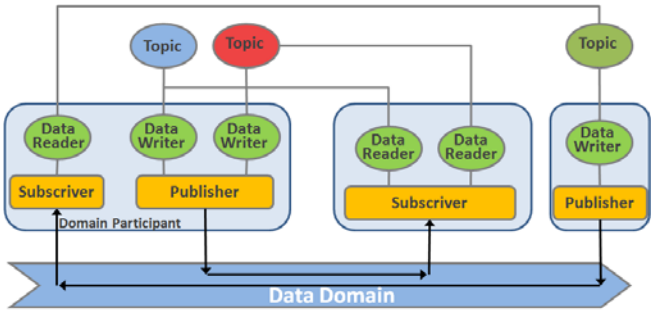

Figura 1. Entità DDS standard

In DDS le entità hanno dei nomi standard e in particolare si definisce DomainParticipant il punto di accesso alla comunicazione in uno specifico dominio, Publisher colui che pubblica e Subscriber colui che riceve i dati associati al topic che ha sottoscritto. I DataWriter e i DataReader rappresentano gli end-point di comunicazione e sono associati in modo univoco a un topic. L'architettura DDS si occupa di come propagare le informazioni di sottoscrizione dei topic mediante una procedura chiamata *discovery*. RTI, nel caso specifico, implementa la procedura di discovery mediante un protocollo RTPS (Real Time Publish

Subscribe) ad-hoc. La Figura 1 riassume le varie entità.
Una parte importante dell'architettura DDS si rivolge alla gestione dei guasti e in particolare alle politiche che permettono di monitorare il sistema e di recuperare lo stato delle entità in modo in parte trasparente. Seppur tali temi siano stati da noi ben studiati e implementati nella nostra soluzione, il resto della relazione non tratterà questi aspetti. Per la parte riguardante la gestione dei guasti, si rimanda alla relazione di Livini. Non sono stati trattati, invece, gli aspetti di DDS legati alla persistenza dei dati sul filesystem e alla sicurezza delle comunicazioni.
Nell'implementazione della soluzione si è sempre cercato di ricorrere a soluzioni supportate dallo standard DDS prima di creare dei protocolli ad hoc.

3. Il sistema centrale

L'architettura del sistema prevede che ci siano un certo numero di fonti incaricate di produrre le notizie e di consegnarle a un sistema centrale. Esso è costituito da una serie di copie attive che garantiscono la disponibilità del servizio e che instradano le notizie verso i fruitori. In questo modo fonti e fruitori sono totalmente disaccoppiati, essendoci il sistema centrale che funge da mediatore. La Figura 2 mostra l'architettura base del sistema.

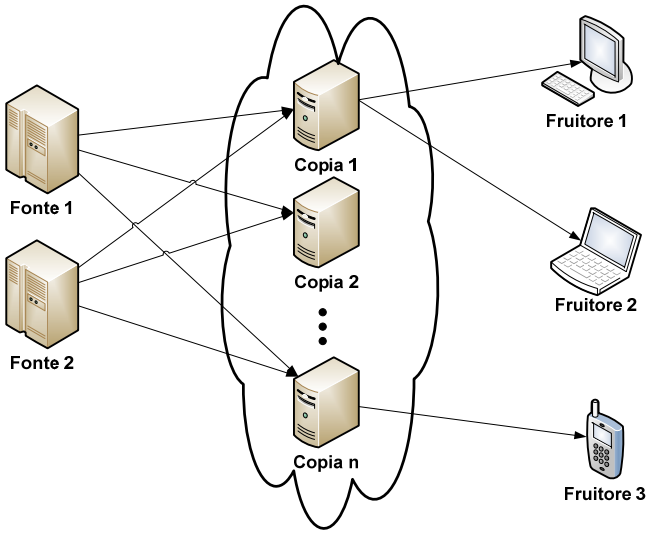

Figura 2. Architettura base del sistema

Ogni copia contiene tutto lo stato necessario a sostituire un'altra copia, se occorre. All'avvio del sistema le varie copie, in modo automatico o manuale, eleggono un master e si scambiano reciprocamente le informazioni necessarie a configurarsi per il funzionamento. Quando una fonte invia una notizia, essa è ricevuta da tutte le copie del sistema. Ad ogni copia è associata una o più partizioni del dominio su cui inviare le notizie. La cosa è necessaria poiché non essendo certi di disporre di connessioni multicast per i fruitori, il tempo necessario a consegnare una notizia a tutti i fruitori è direttamente proporzionale al numero degli stessi, se si utilizzano connessioni di tipo punto-punto. Ogni fruitore, affinché sia in grado di ricevere le notizie, deve essere associato a una partizione oltre che sottoscrivere i topic relativi alle notizie. Prima di iniziare a ricevere il fruitore invia un messaggio di JOIN al sistema e questo (in particolare una sola delle copie) gli risponde fornendogli la partizione su cui deve mettersi in ascolto. La scelta della partizione è fatta secondo il criterio di bilanciare il carico fra le varie copie. Nel caso più semplice la scelta è fatta assegnando al nuovo client la partizione che contiene il minor numero di fruitori, ma si possono pensare politiche basate sull'effettivo carico delle macchine su cui il sistema è in esecuzione. Subito dopo un JOIN, un fruitore può richiedere esplicitamente le notizie precedenti alla sua sottoscrizione ma ancora valide. Quest'architettura si presta molto bene a supportare la caduta di una delle copie, poiché nel momento in cui ci si accorge che è accaduto un guasto a un'entità è sufficiente che un'altra copia si prenda in carico temporaneamente la partizione della copia caduta. Si noti che a livello applicativo non si mantiene nessun riferimento né alle fonti né ai fruitori, ma solo la lista delle partizioni e un relativo indicatore di carico per ognuna di esse. Una volta che un fruitore è stato assegnato a una partizione, egli è in grado di sottoscrivere i topic che vuole ricevere. Lo standard DDS supporta diverse modalità di sottoscrizione per topic multipli come i MultiTopics e i ContentFilterTopics. Nella soluzione abbiamo utilizzato la politica di filtraggio dei topic basata sul contenuto che, oltre ad essere la più flessibile, è anche l'unica supportata dall'implementazione RTI. A ogni notizia sono associate delle keyword che rappresentano i topic e ogni fruitore possiede una lista di keyword che vuole sottoscrivere. Il sistema automaticamente garantisce che il client riceva solo le notizie che ha sottoscritto e una sola volta. Ad esempio, se una

notizia contiene come topic sport, calcio e derby, e un fruitore sottoscrive sport e calcio, egli riceverà la notizia una sola volta. In modo analogo è possibile filtrare la notizia in base ad altri campi come, ad esempio, la fonte di emissione o la data.
Il protocollo di accesso per i fruitori non si limita solo al messaggio di JOIN. Ogni fruitore può, infatti, richiedere una serie molto ampia di opzioni circa la qualità di servizio a seconda che sia o meno un terminale mobile o a seconda del tipo di abbonamento sottoscritto. Discorsi analoghi valgono per le fonti. Tutto il protocollo di negoziazione è supportato dallo standard DDS e si generano degli eventi se le richieste di un Subscriber non sono compatibili con le offerte di un Publisher. Tutti i messaggi di segnalazione utilizzano sia protocolli affidabili bidirezionali, che garantiscono che i messaggi non siano perduti, sia una persistenza in memoria dei messaggi che non si sono riusciti a inviare, in modo da supportare la disconnessione di un terminale (ad esempio mobile) anche durante la stessa fase di accesso al sistema. Il costo di questa scelta è relativamente basso poiché i messaggi di segnalazione si limitano a pochi saltuari messaggi come ad esempio le JOIN/LEAVE dei fruitori.
La scalabilità del sistema si ottiene aggiungendo delle copie al sistema centrale. Tuttavia quello che succede è che tutte le copie sono attive e mantengono lo stesso stato quindi, seppur sempre valida l'assunzione che il numero dei fruitori sia molto maggiore del numero delle fonti, all'aumentare del numero delle fonti il sistema non scala. Inoltre i messaggi di segnalazione devono coinvolgere tutte le copie e quindi se i fruitori sono molto dinamici (frequenti connessioni / disconnessioni) il sistema non scala. Un ulteriore motivo che ci ha spinto a estendere l'architettura è legato all'implementazione RTI che non supporta in modo estensivo le comunicazioni di tipo punto-punto creando, in certi casi, un traffico di rete superiore a quello strettamente necessario. Per superare tali limiti abbiamo sostituito il concetto di sistema centrale con un'architettura basata su più sistemi locali che si coordinano attraverso una rete di multicast e quindi sfruttando tutte le potenzialità di DDS. Ulteriori dettagli saranno forniti nel prossimo capitolo.

4. Architettura del sistema

Come accennato nel capitolo precedente un'architettura basata su un sistema centrale può rappresentare in certi casi un collo di bottiglia per il sistema e limitare la scalabilità. In aggiunta a questo esistono dei limiti legati all'efficienza. Supponiamo di porre il sistema in un ambiente reale in cui le distanze tra fonti/fruitori e sistema possano essere molto differenti. Ciò che ci si aspetta è che ci siano delle fonti/fruitori più vicine al sistema centrale che paghino poco i ritardi legati alla connessione mentre altre più lontane che dipendano fortemente dal tipo di connessione. Il caso estremo più spiacevole è quello in cui una fonte sia molto vicina ai suoi fruitori (ad esempio nel caso di un'agenzia di news locale a una regione/stato), ma lontano dal sistema centrale. Per i limiti descritti nel capitolo precedente e le considerazioni appena fatte, di seguito s'indicherà il sistema centrale come sistema locale a un dominio e si estenderà il middleware affinché supporti un'architettura multi-dominio. La Figura 3 mostra come s'intende partizionare il sistema in domini.

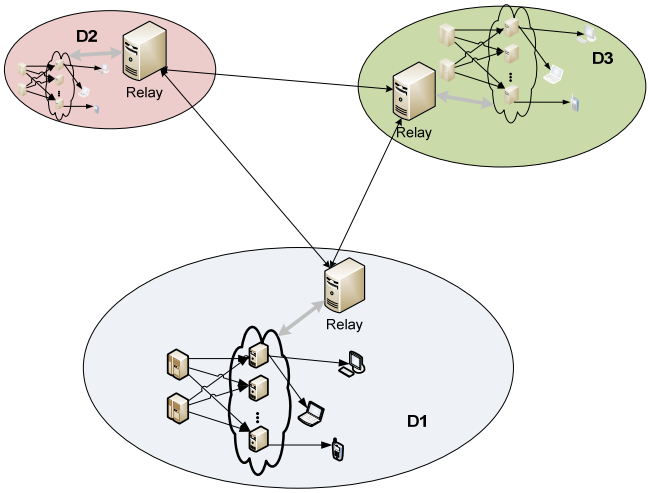

Figura 3. Architettura multi-dominio

Ogni dominio contiene le fonti/fruitori geograficamente più vicine a esso e i domini si scambiano informazioni mediante una rete di Relay peer-to-peer che si riescono a implementare in modo molto efficiente mediante lo standard DDS e l'uso di multicast. Un'architettura così fatta riduce il carico dovuto ai messaggi di segnalazione dei fruitori, confinati ora in un ambito locale al dominio, e permette di aumentare l'efficienza supponendo che le fonti e i fruitori siano vicini al

sistema. Quando una fonte invia una notizia in un dominio, essa deve comunque essere propagata in tutti gli altri domini, ma si può fare mediante dei multicast e inoltre i Relay non sono obbligati a inviare un pacchetto per ogni notizia. Al contrario il Relay raggruppa le notizie in pacchetti grandi in modo da sfruttare efficientemente la connessione di rete. Questo può comportare, purché compatibilmente con le specifiche temporali, un ritardo aggiuntivo sull'invio dei messaggi, ma ne diminuisce drasticamente il numero con un evidente guadagno dal punto di vista dell'efficienza di utilizzo del canale. In modo analogo anche le fonti/fruitori possono inviare/ricevere le notizie in blocchi, sempre rispettando i vincoli di tempo impostati come preferenze dagli utenti. Dal punto di vista di un sistema locale il Relay è una fonte come tutte le altre. Avendo esteso in questo modo l'architettura, il sistema riesce a scalare sia aggiungendo delle copie all'interno di un dominio, sia aggiungendo nuovi domini. Il Relay può essere visto anche come un'entità logica e il suo lavoro può essere anche affidato a una delle copie, infatti, il costo di inviare una notizia a tutti i domini non dipende direttamente dal numero degli stessi. Anche il Relay implementa politiche di fault-tolerance affinché un dominio non resti mai isolato. Di seguito saranno descritte alcune politiche di qualità di servizio che garantiscono al nostro sistema comportamenti adattabili in base ai diversi dispositivi connessi.

4.1 Politiche orientate alla qualità

Lo scenario di funzionamento del sistema è un ambiente in cui sono presenti diversi tipi di dispositivi. In primo luogo possiamo distinguere i fruitori a seconda che si trovino o meno in reti che supportano multicast. Il sistema è configurato per supportare sia dispositivi multicast sia dispositivi punto-punto. Al fine di testare il middleware, un fruitore può specificare esplicitamente se vuole collegarsi al sistema utilizzando solo connessioni punto-punto. È possibile specificare in modo molto granulare anche i vari protocolli utilizzati dalle varie entità. Un'entità può specificare, mediante un file di configurazione, se vuole utilizzare protocolli udp (v4 o v6) o persino tcp nel caso punto-punto, a patto che essi siano supportati anche dal sistema. In secondo luogo possiamo distinguere i dispositivi per la loro propensione alla mobilità. Il sistema supporta, anche per le news, politiche "reliable" se esplicitamente richieste dal fruitore. Un terminale mobile può, ad esempio, richiedere al sistema un collegamento che supporti brevi (arbitrariamente) disconnessioni. A livello d'infrastruttura questo si mappa con un protocollo che prevede degli acknowledge e che quindi risulta più costoso. Se un fruitore sceglie di non utilizzare dei protocolli "reliable" può, comunque, chiedere al sistema di inviare nuovamente uno o più messaggi che si accorge di aver perso. Come già accennato, al fine di disaccoppiarsi totalmente dalla fonte, all'atto della connessione, un fruitore può richiedere le vecchie notizie ancora valide presenti nel sistema utilizzando un messaggio ad-hoc. La copia del sistema responsabile della partizione cui il fruitore appartiene, in un unico invio (se le news non sono eccessive), manda un pacchetto con tutte le notizie. Sempre il fruitore, all'atto della sottoscrizione, può specificare delle politiche sui tempi minimi e massimi d'invio delle notizie. Questi sono gestiti come preferenze nel protocollo di negoziazione di DDS col sistema, nella fase di discovery.
Anche per le fonti si possono specificare opzioni più o meno affidabili in base al tipo di contratto che hanno con il sistema. In particolare, per le fonti è possibile specificare un livello di priorità delle notizie che producono. Questo a livello d'infrastruttura comporta per il sistema il dover gestire due pool di thread a differenti livelli di priorità.

5. Risultati dei test

La soluzione da noi sviluppata è stata testata sia dal punto di vista del bilanciamento del carico, sia nel caso di guasti alle varie entità che in presenza di terminali mobili. In questa sezione mostrerò solo gli esperimenti più rilevanti relativi alla parte di carico del sistema e della rete per valutarne efficienza e scalabilità. Rimando alle rispettive relazioni per i risultati degli altri test.
I vari test sono stati effettuati con diversi elaboratori presenti nel laboratorio di facoltà più due portatili d'ultima generazione. I computer erano collegati in rete mediante una connessione a 100Mbps che si è rivelata essere il collo di bottiglia nei nostri esperimenti. Quello che si è riscontrato è che le

entità iniziano a perdere dei pacchetti quando la rete è utilizzata per più del 60-70% e soprattutto in presenza di nuove richieste di segnalazione, a causa dei picchi di traffico. Ogni copia del sistema e dei relay è stata fatta eseguire su un elaboratore completamente a essa dedicata. Più copie di fonti/fruitori sono state lanciate su singole macchine. Le fonti generavano notizie secondo processi di Poisson di valor medio configurabile. Utilizzando notizie di dimensione intorno a 1,5 KB e frequenza d'invio per le fonti di circa 20 news/sec, quello che si è riscontrato è che una singola scheda di rete, e quindi una singola macchina, non supportava più di una ventina di fonti/fruitori senza perdita di pacchetti. Non è stata riscontrata, invece, nessuna situazione di carico eccessivo per la memoria e il processore delle varie macchine. I sistemi sono stati configurati per fornire dei dati quantitativi sull'effettivo rate di ricezione delle news, mentre i fruitori sono stati configurati per segnalare la perdita di ogni singolo pacchetto. I dati sono stati raccolti utilizzando strumenti di monitoring delle interfacce di rete forniti da terze parti.

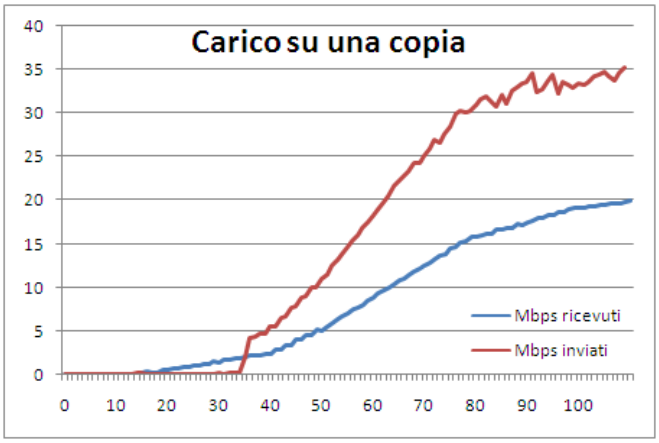

Figura 4. Carico news su una copia

Un primo test significativo è stato fatto per valutare il carico massimo di notizie che una copia del sistema è in grado di supportare senza iniziare a perdere pacchetti. In presenza di solo due fruitori e con un centinaio di fonti distribuite sulle varie macchine, i risultati mostrano che una singola copia è in grado di gestire correttamente fino a 1500-2000 news/sec che corrispondono a un traffico per la scheda di rete del solo elaboratore di circa 18 Mbps di dati. In Figura 4 è mostrato l'aumentare dei dati sulla scheda di rete avviando gradualmente le fonti fino a quando i fruitori iniziano a perdere dei pacchetti. Si vede dal grafico che quando le notizie arrivano a circa 18 Mbps, il sistema non riesce più a garantire un traffico in uscita di 36 Mbps stabilmente. Si suppone qui, come anche in altri test, che i due (macro)fruitori siano interessati a tutte le news in modo da simulare che esistano in media due fruitori interessati a ogni notizia.

Un secondo test rilevante riguarda come più copie del sistema riescano a bilanciarsi i fruitori a essi connessi. In questo scenario il numero di notizie che ogni copia deve elaborare è di circa 350 news/sec. In tali condizioni si è visto che ogni copia del sistema era in grado di gestire correttamente una decina di (macro)fruitori e che si riesce a scalare in modo quasi lineare aggiungendo delle copie. Anche qui i limiti sono dovuti al collo di bottiglia della scheda di rete. Se vogliamo quantificare il risultato ottenuto possiamo considerare che 350 news/sec corrispondono a circa 4 Mbps in ingresso all'elaboratore e con 10 (macro)fruitori punto-punto a 40 Mbps in uscita da essa. Con una rete da 100Mbps, essere riusciti a gestire circa 25 fruitori con 3 sistemi copia è dovuto solo alle forti ottimizzazioni di RTI a livello d'infrastruttura. In questa condizione limite non è comunque per nulla garantito che ogni fruitore riceva i pacchetti tutti e in ordine, poiché ogni singolo messaggio ulteriore sulla rete comporta una perdita di qualche pacchetto. Quello che si è riscontrato per avere un funzionamento ottimale è quello di mantenere la rete occupata per non più del 60-70% come già accennato.

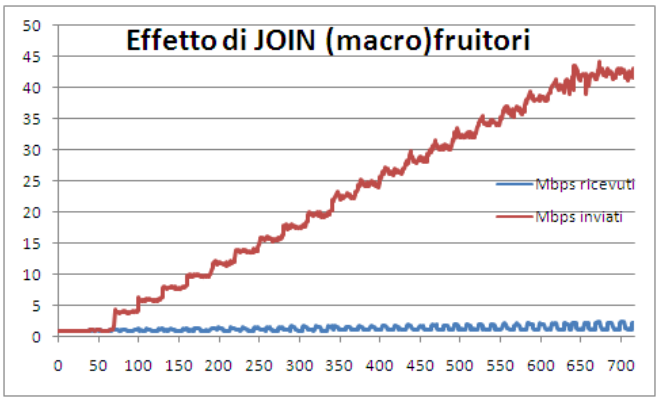

Figura 5 Effetto di JOIN (macro)fruitori

Nel grafico in Figura 5 è mostrato lo stesso esperimento con una mole di notizie pari a circa 1 Mbps dal punto di vista di una copia. Si vede che la

copia riceve una quantità pressoché costante di pacchetti in ingresso con lievi aumenti dovuti alle JOIN dei fruitori, mentre invia in uscita una quantità di dati proporzionale al numero di fruitori. Ogni gradino nel grafico corrisponde all'ingresso di un nuovo fruitore nel sistema. La scalabilità è limitata poiché tutti i fruitori considerati nell'esperimento sono di tipo punto-punto. Il sistema copia, come si può notare, garantisce un corretto invio di notizie solo fino a circa 45 Mbps su una rete da 100 Mbps, oltre tale soglia i fruitori iniziano a perdere dei pacchetti. Non sono stati riscontrati problemi, invece, né dal punto di vista dell'occupazione di memoria (il processo arriva a scarsi 300 MB), né dal punto di vista del processore che lavorava intorno al 20%.
Per quando riguarda i Relay, l'aver a disposizione solo un'interfaccia di rete ha limitato i benefici di avere più domini. Aggiungere domini aumenta in modo lineare il numero di fruitori che si riescono a gestire senza l'eccessivo overhead aggiuntivo dovuto ai messaggi di segnalazione.

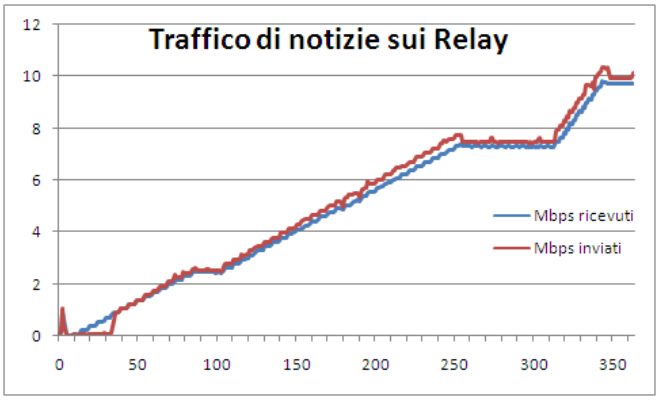

Figura 6. Traffico di notizie sui Relay

La Figura 6 mostra il traffico di notizie su di un Relay che viene caricato da una successione di 4 blocchi da 15 sorgenti l'uno. Non si è riscontrato nessun tipo di problema nello smistamento delle notizie fino a saturazione di tutta la banda disponibile. Il Relay è arrivato a smistare fino a 10 Mbps di notizie senza particolari perdite di pacchetti da parte dei fruitori, ma sulla stessa rete erano presenti contemporaneamente tutte le fonti, sistemi e fruitori. In un deploy reale si suppone che i Relay abbiano a disposizione reti dedicate di grandi capacità. In ottica di ottimizzazione, i Relay sono capaci di ricevere / inviare sia singole notizie sia pacchetti batch di dimensione limitata. Quello che si è riscontrato è che seppur la dimensione massima di un pacchetto UDP sia di 64 KB, l'infrastruttura RTI si rifiuta di arrivare a tali dimensioni ed ha limitato i blocchi di notizie a poche decine di unità (con notizie da 1,5 KB). La Figura 7 riassume complessivamente il traffico di rete all'aumentare della frequenza d'invio di notizie (valor medio della distribuzione di Poisson aggiornato nel tempo) e all'aumentare del numero di fruitori dal punto di vista dei Relay e del sistema.

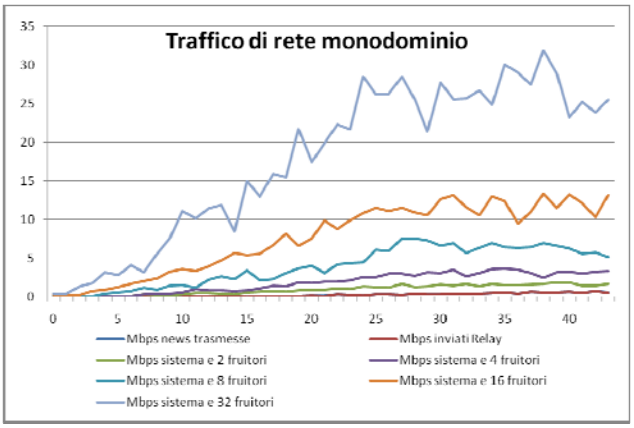

Figura 7. Traffico di rete monodominio

5.1 Considerazioni su RTI

A seguito degli esperimenti e monitorando la rete possiamo confermare alcune importanti funzionalità dell'implementazione RTI utilizzata, ma anche criticarne delle altre. Un'importante funzione riscontrata è che l'architettura RTI si configura in modo molto semplice su più interfacce di rete o protocolli o modalità di funzionamento, molto spesso senza l'intervento dell'utente o con la semplice predisposizione di file di configurazione. Inoltre, l'implementazione RTI utilizza la rete in modo molto efficiente raggruppando i pacchetti in base alla configurazione dell'utente. Quest'ultima funzionalità è stata comparata col nostro modulo applicativo di raggruppamento delle news e si è rivelata leggermente più efficiente, stabile, ma soprattutto molto più adattabile ai diversi protocolli e quindi indubbiamente da preferire.
A sfavore dell'implementazione RTI è da sottolineare la chiusura del middleware che comporta una difficile diagnosi degli spesso non chiari errori d'infrastruttura a runtime e soprattutto una non sempre corretta gestione delle politiche di filtraggio Publisher-side dei topic. Quest'ultima

cosa, seppur ancora non matura poiché introdotta solo con l'ultima versione a oggi disponibile dell'infrastruttura, comporta un traffico sulla rete ulteriore assolutamente non necessario.

6. Conclusioni

L'idea di sviluppare un sistema di diffusione news su larga scala contemplando la presenza di un sistema centrale e di terminali eterogenei con requisiti molto differenziati si discosta un po' dai modelli classici per cui l'implementazione RTI, ma in generale il paradigma publish\subscribe è stato pensato. La nostra implementazione e soprattutto gli esperimenti hanno dimostrato che lavorando in modo mirato sull'architettura sia possibile ottenere comunque buoni risultati. Seppur i test non siano riusciti ad evidenziare appieno i limiti dell'architettura, sono stati sicuramente significativi a chiarire alcuni limiti di un architettura così pensata indipendentemente dall'implementazione. In particolare è stato cruciale il trade-off tra efficienza e affidabilità del servizio: il modello publish\subscribe è essenzialmente best-effort e quindi bisogna considerare la possibile perdita d'informazioni. Lo standard DDS ha aggiunto a tale modello gli strumenti necessari a rendere affidabile il servizio, ma usare tali strumenti in modo estensivo limita di molto la scalabilità del sistema. L'implementazione RTI si è rivelata molto potente e flessibile per ottenere il giusto compromesso tra affidabilità e scalabilità, essendo molto configurabile. Nel complesso ci riteniamo soddisfatti del lavoro svolto e dei risultati ottenuti che mostrano una minima intrusione delle scelte da noi operate nell'architettura utilizzata (perlomeno su singole reti non a elevate prestazioni).